1. Introduction

Donorschoose.org is a US-based non-profit organization that allows individuals to donate directly to public school classroom projects.Founded in 2000 by former public school teacher Charles Best, DonorsChoose.org was among the first civic crowdfunding platforms of its kind. The organization has been given Charity Navigator's highest rating every year since 2005.In January 2018, they announced that 1 million projects had been funded.To get students what they need to learn, the team at DonorsChoose.org needs to be able to connect donors with the projects that most inspire them.

Problem Statement

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the assignment is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

2. Importing Libraries

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
```

```
C:\Users\aksha\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, op
ening SSH/SCP/SFTP paths will be disabled.  `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install
paramiko` to suppress')
C:\Users\aksha\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; al
iasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

3.Directory List

In [2]:

```
import os
os.chdir("D:\\applied AI\\Donorchoose")
```

4. About the dataset

The train_data.csv is the dataset provided by the DonorsChoose containin features as follows :-

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted.**Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project.**Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher.**Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_4:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

5. Reading the data

In [3]:

```
train_data=pd.read_csv("train_data.csv")
res_data=pd.read_csv("resources.csv")
```

In [4]:

```
print("number of datapoints=",train_data.shape) #shape will tell us the number of projects we have
which is 109248
```

```
print("columns/atrributes name=",train_data.columns)
print(train_data.head(3))
```

```
number of datapoints= (109248, 17)
columns/atrributes name= Index(['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'],
      dtype='object')
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.
2       21895  p182444  3465aaf82da834c0582ebd0ef8040ca0            Ms.

  school_state project_submitted_datetime project_grade_category  \
0           IN        2016-12-05 13:43:57          Grades PreK-2
1           FL        2016-10-25 09:22:10             Grades 6-8
2           AZ        2016-08-31 12:03:56             Grades 6-8

          project_subject_categories     project_subject_subcategories  \
0                Literacy & Language                     ESL, Literacy
1  History & Civics, Health & Sports  Civics & Government, Team Sports
2                    Health & Sports    Health & Wellness, Team Sports

                                       project_title  \
0   Educational Support for English Learners at Home
1              Wanted: Projector for Hungry Learners
2  Soccer Equipment for AWESOME Middle School Stu...

                                     project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...
2  \r\n\"True champions aren't always the ones th...

                                     project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...             NaN
1  The projector we need for our school is very c...             NaN
2  The students on the campus come to school know...             NaN

  project_essay_4                           project_resource_summary  \
0             NaN  My students need opportunities to practice beg...
1             NaN  My students need a projector to help with view...
2             NaN  My students need shine guards, athletic socks,...

   teacher_number_of_previously_posted_projects  project_is_approved
0                                             0                    0
1                                             7                    1
2                                             1                    0
```

In [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
# Replacing datetime columns to date column
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(train_data.columns)] #if x encounters column name project_submitted_datetime it will replace by date
#so a new column Date is created

#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40-84039
train_data['Date'] = pd.to_datetime(train_data['project_submitted_datetime']) #pd.to_datetime converts argument to datetime
train_data.drop('project_submitted_datetime', axis=1, inplace=True) #dropping the column project_submitted_date
train_data.sort_values(by=['Date'], inplace=True)#sorting the dataframe by date


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
train_data = train_data[cols] #adding the new column


train_data.head(2) #displaying the dataframe
```

Out[5]:

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	

In [6]:

```
print("datapoints in resources=",res_data.shape)
print("attributes of resources=",res_data.columns)
print(res_data.head(3))
```

```
datapoints in resources= (1541272, 4)
attributes of resources= Index(['id', 'description', 'quantity', 'price'], dtype='object')
        id                                        description  quantity  \
0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
1  p069063        Bouncy Bands for Desks (Blue support pipes)         3
2  p069063  Cory Stories: A Kid's Book About Living With Adhd         1

    price
0  149.00
1   14.95
2    8.45
```

By looking at the shape of train_data we can see that we have around 109k projects

and resources.shape tells us that we have around 15mn resources,resources can be greater than project because for each project we can have more than resources needed

In [7]:

```
#Refer-> https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/

price_data = res_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index() #grouping is done on the basis of ids and agggreating the sum of price and quantity column

#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.merge.html?highlight=merge#pandas.merge
train_data = train_data.merge(price_data, on='id', how='left')
print(train_data.head(1))
```

```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0        8393  p205479  2bf07ba08945e5d8b2a3f269b2b3cfe5           Mrs.

  school_state                 Date project_grade_category  \
0           CA  2016-04-27 00:27:36          Grades PreK-2

  project_subject_categories            project_subject_subcategories  \
0             Math & Science  Applied Sciences, Health & Life Science

                                 project_title  \
0  Engineering STEAM into the Primary Classroom

                                     project_essay_1  \
0  I have been fortunate enough to use the Fairy ...

                                     project_essay_2  \
0  My students come from a variety of backgrounds...

                                     project_essay_3  \
0  Each month I try to do several science or STEM...

                                     project_essay_4  \
0  It is challenging to develop high quality scie...
```

```
                             project_resource_summary  \
0  My students need STEM kits to learn critical s...

   teacher_number_of_previously_posted_projects  project_is_approved   price  \
0                                            53                    1  725.05

   quantity
0         4
```

**Counting number of projects approved and not approved

In [8]:

```
#Refer for documentation: https://www.geeksforgeeks.org/python-pandas-index-value_counts/
approved_not_approved=train_data['project_is_approved'].value_counts()
print(approved_not_approved)
print("*"*50)
approved_not_approved1=train_data['project_is_approved'].value_counts(normalize=True)
print("in percentage=",approved_not_approved1)
```

```
1    92706
0    16542
Name: project_is_approved, dtype: int64
**************************************************
in percentage= 1    0.848583
0    0.151417
Name: project_is_approved, dtype: float64
```

Imbalanced Dataset where class-label 1 is 85% and 0 is 15%

Feature Preprocessing

Preprocessing of project_subject_categories

In [9]:

```
#Refer ->https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
#Refer for documentation ->https://www.programiz.com/python-programming/methods/string/strip
categories = list(train_data['project_subject_categories'].values) #creating a list of  all the va
lues in project subject categories
clean_cat=[]
for i in categories: #taking each category at a time
    temp="" #creating a empty string
    for j in i.split(","): # splitting each word separated by a comma
        if 'The' in j.split():
            j=j.replace('The',"") #replacing the every occurence of "The" with ""
        j=j.replace(" ","") #replacing every white space with ""
        temp+=j.strip()+" " #removing all leading and trailing whitespaces and then adding a white
space at the end
        temp = temp.replace('&','') #replacing & with "_"
        temp=temp.lower()
    clean_cat.append(temp.strip())
    #showing the result
print(clean_cat[23])
```

```
mathscience
```

In [10]:

```
train_data['clean_categories']=clean_cat #creating a new column as clean_categories
train_data.drop(['project_subject_categories'], axis=1,inplace=True) #dropping the subject categor
y
```

In [11]:

```
# Counting number of words in a corpus/clean_categories
#Refer ->https://stackoverflow.com/questions/8139239/how-to-count-words-in-a-corpus-document
```

```
#Refer ->https://stackoverflow.com/questions/8139239/how-to-count-words-in-a-corpus-document
from collections import Counter
my_counter = Counter()
for word in train_data['clean_categories'].values:
    my_counter.update(word.split())

print(dict(my_counter)) #printing the dictionary
sortd=sorted(my_counter.items()) #with sorted function on dictionary it sorts in aplhabetical
order of value
print("="*50)
print(sortd)

# Refer -> sorting dictionary in python by value : https://www.geeksforgeeks.org/python-sort-pytho
n-dictionaries-by-key-or-value/
#https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-lambda-function/
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
```

```
{'mathscience': 41421, 'specialneeds': 13642, 'literacylanguage': 52239, 'appliedlearning': 12135,
'historycivics': 5914, 'musicarts': 10293, 'healthsports': 14223, 'warmth': 1388, 'carehunger':
1388}
==================================================
[('appliedlearning', 12135), ('carehunger', 1388), ('healthsports', 14223), ('historycivics', 5914
), ('literacylanguage', 52239), ('mathscience', 41421), ('musicarts', 10293), ('specialneeds',
13642), ('warmth', 1388)]
```

Preprocessing of project_subject_subcategories

In [12]:

```
#Refer ->https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
#Refer for documentation ->https://www.programiz.com/python-programming/methods/string/strip
subcategories = list(train_data['project_subject_subcategories'].values) #creating a list of  all
the values in project subject categories
clean_subcat=[]
for i in subcategories: #taking each category at a time
    temp="" #creating a empty string
    for j in i.split(","): # splitting each word separated by a comma
        if 'The' in j.split():
            j=j.replace('The',"") #replacing the every occurence of "The" with ""
        j=j.replace(" ","") #replacing every white space with ""
        temp+=j.strip()+" " #removing all leading and trailing whitespaces and then adding a white
space at the end
        temp = temp.replace('&','') #replacing & with "_"
        temp=temp.lower()
    clean_subcat.append(temp.strip())
    #showing the result
print(clean_subcat[24])
```

```
specialneeds
```

In [13]:

```
train_data['clean_subcategories']=clean_subcat #creating a new column as clean_categories
train_data.drop(['project_subject_subcategories'], axis=1,inplace=True) #dropping the subject cate
gory
```

In [14]:

```
# Counting number of words in a corpus/clean_categories
#Refer ->https://stackoverflow.com/questions/8139239/how-to-count-words-in-a-corpus-document
from collections import Counter
my_counter1 = Counter()
for word in train_data['clean_subcategories'].values:
    my_counter1.update(word.split())

print(dict(my_counter1)) #printing the dictionary
sortd1=sorted(my_counter1.items()) #with sorted function on dictionary it sorts in aplhabetical
order of value
print("="*50)
print(sortd1)
```

```
# Refer -> sorting dictionary in python by value : https://www.geeksforgeeks.org/python-sort-pytho
n-dictionaries-by-key-or-value/
#https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-lambda-function/
subcat_dict = dict(my_counter1)
sorted_subcat_dict = dict(sorted(subcat_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
```

```
{'appliedsciences': 10816, 'healthlifescience': 4235, 'specialneeds': 13642, 'literacy': 33700, 'e
arlydevelopment': 4254, 'mathematics': 28074, 'socialsciences': 1920, 'historygeography': 3171, 'e
sl': 4367, 'extracurricular': 810, 'visualarts': 6278, 'environmentalscience': 5591,
'literaturewriting': 22179, 'gymfitness': 4509, 'music': 3145, 'teamsports': 2192,
'performingarts': 1961, 'collegecareerprep': 2568, 'other': 2372, 'charactereducation': 2065,
'foreignlanguages': 890, 'healthwellness': 10234, 'civicsgovernment': 815, 'economics': 269,
'communityservice': 441, 'financialliteracy': 568, 'nutritioneducation': 1355,
'parentinvolvement': 677, 'warmth': 1388, 'carehunger': 1388}
==================================================
[('appliedsciences', 10816), ('carehunger', 1388), ('charactereducation', 2065),
('civicsgovernment', 815), ('collegecareerprep', 2568), ('communityservice', 441),
('earlydevelopment', 4254), ('economics', 269), ('environmentalscience', 5591), ('esl', 4367), ('e
xtracurricular', 810), ('financialliteracy', 568), ('foreignlanguages', 890), ('gymfitness',
4509), ('healthlifescience', 4235), ('healthwellness', 10234), ('historygeography', 3171),
('literacy', 33700), ('literaturewriting', 22179), ('mathematics', 28074), ('music', 3145),
('nutritioneducation', 1355), ('other', 2372), ('parentinvolvement', 677), ('performingarts', 1961
), ('socialsciences', 1920), ('specialneeds', 13642), ('teamsports', 2192), ('visualarts', 6278),
('warmth', 1388)]
```

Text Preprocessing

**First we have to merge all the essay columns into a single column and then count the number of words in essay's of approved projects and essay's of rejected projects

In [15]:

```
# merge two column text dataframe: https://stackoverflow.com/questions/19377969/combine-two-column
s-of-text-in-dataframe-in-pandas-python
train_data["project_essay"] = train_data["project_essay_1"].map(str) +train_data["project_essay_2"]
.map(str)+train_data["project_essay_3"].map(str) +  train_data["project_essay_4"].map(str)
            #Here the .map(str) converts string to all the coulmns in project_eassy_1/2/3/4
print(train_data['project_essay'].head(3))
```

```
0    I have been fortunate enough to use the Fairy ...
1    Imagine being 8-9 years old. You're in your th...
2    Having a class of 24 students comes with diver...
Name: project_essay, dtype: object
```

Essay Text

In [16]:

```
# printing some random essays.
print(train_data['project_essay'].values[10])
print("="*50)
print(train_data['project_essay'].values[20000])
print("="*50)
print(train_data['project_essay'].values[942])
print("="*50)
print(train_data['project_essay'].values[451])
print("="*50)
print(train_data['project_essay'].values[99])
print("="*50)
```

```
My students yearn for a classroom environment that matches their desire to learn. With education c
hanging daily, we need a classroom that can meet the needs of all of my first graders.I have the p
rivilege of teaching an incredible group of six and seven year olds who absolutely LOVE to learn.
I am completely blown away by their love for learning. Each day is a new adventure as they enjoy l
earning from nonfiction text and hands on activities. Many of my students are very active learners
who benefit from kinesthetic activities. Sometimes learning, while sitting in a seat, is
difficult. I want every child the opportunity to focus their energy in order to do their best in
school!Ideally, I would love to delve right into \"flexible seating\" where students are provided
many different seating options (chairs, hokki stools, on mats on the ground, etc.) and they have t
```

```
he freedom to choose which ever seat they feel they need. My student would be able to choose which
seating option will best help them learn. In addition, a pencil sharpener, mobile easel, magnetic
strips and mounting tape will help make our classroom better suited for 6 and 7 year olds.This pro
ject will be so beneficial for my students in that they will be able to better focus their energy.
Something so small, choosing their own seat, will help encourage a positive learning environment t
hat promotes learning for all students. The easel will help make our classroom more mobile, becaus
e it is both dry erase and on wheels. Magnetic strips, mounting tape and a pencil sharpener will a
llow for more resources for the students during the school day.
==================================================
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the bi
ggest enthusiasm for learning. My students learn in many different ways using all of our senses an
d multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nSt
udents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it's healthy for their bodies. This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroo
m garden in the spring. We will also create our own cookbooks to be printed and shared with famili
es. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
==================================================
Can you imagine sitting still for hours on end? I can't do that as an adult and I certainly don't
expect my students to be able to either!I teach at a school with a very diverse population. We hav
e students from every many ethnicity and backgrounds. Our school is between 2 major cities. Many s
tudents receive free or reduced lunches and we have a good size military population. \r\nI love my
class but they are very bouncy and love to move!I want to offer my students the choice to sit in t
he seats they want! They currently sit in hard plastic chairs that are NOT comfortable! I want the
m to be comfortable and be able to wiggle around and use energy, which promotes brain power! Each
morning they will have the chance to pick their seat so they can start the day of right!This proje
ct will make a difference because research has shown that the more kids move - the more they learn
! By giving them as many opportunities as possible toe move (even when in their seats) I can help
them live up to their full potential!
==================================================
\"If kids come to us from strong, healthy functioning families, it makes our job easier. If they d
o not come to us from strong, healthy, functioning families, it makes our job more
important.\"~Barbara Colorose.My students are housed in a Life Skills Unit, which is considered th
e most restricted due to their behaviors and/or disabilities.  We are a public high school located
in a high-poverty area. We are avid participants in Special Olympics and Community Based
Instruction.Many students at our school come hungry and our resources are limited. I would be able
to provide a healthy snack to those in need. I would also use as positive motivators throughout
the day. I would use many of the snacks as counting items in order to engage my students with extr
a needs.  The trail mix is great for sorting, classifying and graphing.This project will improve m
y classroom because I cannot always afford to buy the snacks I would like to have as motivators.
Sometimes, a little snack is all that is needed to get them back on track and ready to learn.
==================================================
A typical lesson in my school starts with a read aloud from a picture book to introduce the
reading or writing tasks students are learning.  These read-alouds serve as mentors in the
learning process.Units of study in Reading and Writing are the curricular guides at my project-bas
ed, Reggio-inspired elementary school.  Students are eager to learn a new teaching point each day,
which is usually inspired by the context of the daily read-aloud.  The texts allow us to talk
about our shared reading experience, since the students love to chatter!When the students have acc
ess to quality read-alouds that strongly relate to our daily teaching point, they are able to expe
rience the academic standard in the realistic context of literature.  For example, literacy expert
Katie Wood Ray advises using the book Beekeepers as an example that exhibits what writers do when
they share a slice of their life.  These books and guides offer unlimited lessons about what good
readers and writers do.Your donation will allow students to live in the worlds of these books!  Th
ey will be able to participate in memorable lessons that engage their minds.  Read-alouds can be t
he key to hooking them into learning about reading and writing.
==================================================
```

In [17]:

```python
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    # 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 [18]:

```
test = decontracted(train_data['project_essay'].values[20000])
print(test)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the b
iggest enthusiasm for learning. My students learn in many different ways using all of our senses a
nd multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nS
tudents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies. \r\nStudents will gain math and literature skills as well as a life long enjoyment for health
y cooking.nannan
==================================================
```

In [19]:

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

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smallest students with the big
gest enthusiasm for learning. My students learn in many different ways using all of our senses and
multiple intelligences. I use a wide range of techniques to help all my students succeed.
Students in my class come from a variety of different backgrounds which makes for wonderful
sharing of experiences and cultures, including Native Americans.  Our school is a caring community
of successful learners which can be seen through collaborative student project based learning in a
nd out of the classroom. Kindergarteners in my class love to work with hands-on materials and have
many different opportunities to practice a skill before it is mastered. Having the social skills t
o work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is
the perfect place to learn about agriculture and nutrition. My students love to role play in our p
retend kitchen in the early childhood classroom. I have had several kids ask me,  Can we try cooki
ng with REAL food?  I will take their idea and create  Common Core Cooking Lessons  where we learn
important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies.   Students will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
```

In [20]:

```
#remove special character: https://stackoverflow.com/a/5843547/4084039
test = re.sub('[^A-Za-z0-9]+', ' ', test) #square bracket creates either or set; + signifes 1 or m
ore character
print(test)
```

```
 A person is a person no matter how small Dr Seuss I teach the smallest students with the biggest
enthusiasm for learning My students learn in many different ways using all of our senses and multi
ple intelligences I use a wide range of techniques to help all my students succeed Students in my
class come from a variety of different backgrounds which makes for wonderful sharing of
experiences and cultures including Native Americans Our school is a caring community of successful
learners which can be seen through collaborative student project based learning in and out of the
classroom Kindergarteners in my class love to work with hands on materials and have many different
opportunities to practice a skill before it is mastered Having the social skills to work
cooperatively with friends is a crucial aspect of the kindergarten curriculum Montana is the
perfect place to learn about agriculture and nutrition My students love to role play in our
pretend kitchen in the early childhood classroom I have had several kids ask me Can we try cooking
with REAL food I will take their idea and create Common Core Cooking Lessons where we learn
important math and writing concepts while cooking delicious healthy food for snack time My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce make our own bread and mix up healthy plants from our classroom
garden in the spring We will also create our own cookbooks to be printed and shared with families
Students will gain math and literature skills as well as a life long enjoyment for healthy cooking
nannan
```

In [21]:

```
s=set(stopwords.words('english'))
print(s)
```

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

In [22]:

```
#Combining all the above statments to transform our text in a clean text
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(train_data['project_essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in s)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:15<00:00, 7261.94it/s]
```

In [23]:

```
#printing the text after preprocessing
preprocessed_essays[0]
```

Out[23]:

```
'i fortunate enough use fairy tale stem kits classroom well stem journals students really enjoyed
i would love implement lakeshore stem kits classroom next school year provide excellent engaging s
tem lessons my students come variety backgrounds including language socioeconomic status many lot
experience science engineering kits give materials provide exciting opportunities students each mo
nth i try several science stem steam projects i would use kits robot help guide science
instruction engaging meaningful ways i adapt kits current language arts pacing guide already teach
material kits like tall tales paul bunyan johnny appleseed the following units taught next school
year i implement kits magnets motion sink vs float robots i often get units know if i teaching rig
ht way using right materials the kits give additional ideas strategies lessons prepare students sc
ience it challenging develop high quality science activities these kits give materials i need
provide students science activities go along curriculum classroom although i things like magnets c
lassroom i know use effectively the kits provide right amount materials show use appropriate way'
```

In [24]:

```
train_data['preprocessed_essays']=preprocessed_essays
train_data.drop(['project_essay'], axis=1,inplace=True)
```

Project title text

In [25]:

```
# Printing some random project title
# printing some random essays.
print(train_data['project_title'].values[7])
print("="*50)
print(train_data['project_title'].values[9])
print("="*50)
print(train_data['project_title'].values[16])
print("="*50)
print(train_data['project_title'].values[23])
print("="*50)
```

```
21st Century Learning with Multimedia
==================================================
Dash and Dot Robotic Duo Needed
==================================================
Help us travel the world...VIRTUALLY!
==================================================
Techies in Training
==================================================
```

In [26]:

```
#1.Decontraction
test1 = decontracted(train_data['project_title'].values[7])
print(test1)
print("="*50)
```

```
21st Century Learning with Multimedia
==================================================
```

In [27]:

```
#2. Removing newline breakline etc
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
test1 = test1.replace('\\r', ' ')
test1= test1.replace('\\"', ' ')
test1= test1.replace('\\n', ' ')
print(test1)
```

```
21st Century Learning with Multimedia
```

In [28]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
```

```
test1 = re.sub('[^A-Za-z0-9]+', ' ', test1) #square bracket creates either or set; + signifes 1 or
more character
print(test1)
```

```
21st Century Learning with Multimedia
```

In [29]:

```
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for title in tqdm(train_data['project_title'].values):
    test1 = decontracted(title)
    test1 = test1.replace('\\r', ' ')
    test1 = test1.replace('\\"', ' ')
    test1 = test1.replace('\\n', ' ')
    test1 = re.sub('[^A-Za-z0-9]+', ' ', test1)
    # https://gist.github.com/sebleier/554280
    test1 = ' '.join(e for e in test1.split() if e not in s)
    preprocessed_title.append(test1.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248
[00:01<00:00, 75909.56it/s]
```

In [30]:

```
preprocessed_title[0]
```

Out[30]:

```
'engineering steam primary classroom'
```

In [31]:

```
train_data['preprocessed_title']=preprocessed_title
train_data.drop(['project_title'], axis=1,inplace=True)
```

Teacher Prefix

In [32]:

```
train_data['teacher_prefix'].head(5) #printing the first 5 values to see what preprocessing should
be made
```

Out[32]:

```
0    Mrs.
1     Ms.
2    Mrs.
3    Mrs.
4    Mrs.
Name: teacher_prefix, dtype: object
```

Need to convert it into lowercase as well as remove the punctuation at the last

In [33]:

```
from tqdm import tqdm
import string
preprocessed_prefix=[]
for prefix in tqdm(train_data['teacher_prefix'].values):
    test=str(prefix).strip(".")
    test=test.lower()
    preprocessed_prefix.append(test)
```

```
100%|██████████████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 1564069.97it/s]
```

In [34]:

```
preprocessed_prefix[3]
```

Out[34]:

```
'mrs'
```

In [35]:

```
train_data['preprocessed_prefix']=preprocessed_prefix
#train_data.drop(['teacher_prefix'], axis=1,inplace=True)
```

Grade Category

In [36]:

```
train_data['project_grade_category'].head(5) #printing the first 5 values to see what
preprocessing should be made
```

Out[36]:

```
0    Grades PreK-2
1       Grades 3-5
2    Grades PreK-2
3    Grades PreK-2
4       Grades 3-5
Name: project_grade_category, dtype: object
```

In [37]:

```
train_data['project_grade_category'].value_counts()
```

Out[37]:

```
Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
```

In [38]:

```
preprocessed_grade=[]
for grade in tqdm(train_data['project_grade_category'].values):
    grade=grade.strip(" ")
    grade=grade.replace(" ", "_")
    grade=grade.replace("-","_")
    preprocessed_grade.append(grade)
```

```
100%|██████████████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 1403720.60it/s]
```

In [39]:

```
preprocessed_grade[0:5]
```

Out[39]:

```
['Grades_PreK_2', 'Grades_3_5', 'Grades_PreK_2', 'Grades_PreK_2', 'Grades_3_5']
```

In [40]:

```
train_data['preprocessed_grade']=preprocessed_grade
train_data.drop(['project_grade_category'], axis=1,inplace=True)
```

project_resource_summary

In [41]:

```
train_data['project_resource_summary'].head(5)
```

Out[41]:

```
0    My students need STEM kits to learn critical s...
1    My students need Boogie Boards for quiet senso...
2    My students need a mobile listening center to ...
3    My students need flexible seating in the class...
4    My students need copies of the New York Times ...
Name: project_resource_summary, dtype: object
```

In [42]:

```
from tqdm import tqdm
preprocessed_resource = []
# tqdm is for printing the status bar
for resource in tqdm(train_data['project_resource_summary'].values):
    sent = decontracted(resource)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in s)
    preprocessed_resource.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248
[00:02<00:00, 49677.41it/s]
```

In [43]:

```
preprocessed_resource[0:5]
```

Out[43]:

```
['my students need stem kits learn critical science engineering skills the kits focus important sc
ience concepts the robot works engineering skills',
 'my students need boogie boards quiet sensory breaks putty sensory input focus',
 'my students need mobile listening center able enhance learning',
 'my students need flexible seating classroom choose comfortable learn best',
 'my students need copies new york times best seller wonder book okay now think deeply compare con
trast structures']
```

In [44]:

```
train_data['preprocessed_resource']=preprocessed_resource
train_data.drop(['project_resource_summary'], axis=1,inplace=True)
```

KNN(K-Nearest Neighbor)

Preparing Data For Splitting

In [45]:

```
#how to drop a column in pandas-> https://pandas.pydata.org/pandas-
docs/stable/reference/api/pandas.DataFrame.drop.html
print(train_data.head(3))
```

```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0        8393  p205479  2bf07ba08945e5d8b2a3f269b2b3cfe5           Mrs.
1       37728  p043609  3f60494c61921b3b43ab61bdde2904df            Ms.
```

```
2       74477  p189804  4a97f3a390bfe21b99cf5e2b81981c73           Mrs.

  school_state                Date  \
0           CA 2016-04-27 00:27:36
1           UT 2016-04-27 00:31:25
2           CA 2016-04-27 00:46:53

                                     project_essay_1  \
0  I have been fortunate enough to use the Fairy ...
1  Imagine being 8-9 years old. You're in your th...
2  Having a class of 24 students comes with diver...

                                     project_essay_2  \
0  My students come from a variety of backgrounds...
1  Most of my students have autism, anxiety, anot...
2  I have a class of twenty-four kindergarten stu...

                                     project_essay_3  \
0  Each month I try to do several science or STEM...
1  It is tough to do more than one thing at a tim...
2  By having a mobile listening and storage cente...

                                     project_essay_4  ...  \
0  It is challenging to develop high quality scie...  ...
1  When my students are able to calm themselves d...  ...
2  A mobile listening center will help keep equip...  ...

   project_is_approved   price  quantity  clean_categories  \
0                    1  725.05         4       mathscience
1                    1  213.03         8      specialneeds
2                    1  329.00         1  literacylanguage

                clean_subcategories  \
0  appliedsciences healthlifescience
1                      specialneeds
2                          literacy

                                 preprocessed_essays  \
0  i fortunate enough use fairy tale stem kits cl...
1  imagine 8 9 years old you third grade classroo...
2  having class 24 students comes diverse learner...

                        preprocessed_title preprocessed_prefix  \
0      engineering steam primary classroom                 mrs
1                      sensory tools focus                  ms
2  mobile learning mobile listening center                 mrs

  preprocessed_grade                              preprocessed_resource
0      Grades_PreK_2  my students need stem kits learn critical scie...
1         Grades_3_5  my students need boogie boards quiet sensory b...
2      Grades_PreK_2  my students need mobile listening center able ...

[3 rows x 21 columns]
```

In [46]:

```
x=train_data.drop(columns=['id',"teacher_id","Date",'project_essay_1','project_essay_2','project_essay_3','project_essay_4'])
```

In [47]:

```
print(x.head(3))
```

```
   Unnamed: 0 teacher_prefix school_state  \
0        8393           Mrs.           CA
1       37728            Ms.           UT
2       74477           Mrs.           CA

   teacher_number_of_previously_posted_projects  project_is_approved   price  \
0                                            53                    1  725.05
1                                             4                    1  213.03
2                                            10                    1  329.00

   quantity  clean_categories                clean_subcategories  \
0         4       mathscience  appliedsciences healthlifescience
```

```
1          8      specialneeds                              specialneeds
2          1  literacylanguage                                  literacy

                                  preprocessed_essays  \
0  i fortunate enough use fairy tale stem kits cl...
1  imagine 8 9 years old you third grade classroo...
2  having class 24 students comes diverse learner...

                         preprocessed_title preprocessed_prefix  \
0      engineering steam primary classroom                 mrs
1                       sensory tools focus                  ms
2  mobile learning mobile listening center                 mrs

  preprocessed_grade                              preprocessed_resource
0      Grades_PreK_2  my students need stem kits learn critical scie...
1         Grades_3_5  my students need boogie boards quiet sensory b...
2      Grades_PreK_2  my students need mobile listening center able ...
```

In [48]:

```
sample_data_1=x.sample(frac=.90)
```

In [49]:

```
print(sample_data_1.head(3))
print("="*50)
print(sample_data_1.shape)
```

```
        Unnamed: 0 teacher_prefix school_state  \
2989        172858            Ms.           NV
101704      116753            Mr.           CA
70112        45151           Mrs.           KY

        teacher_number_of_previously_posted_projects  project_is_approved  \
2989                                               3                    1
101704                                             2                    1
70112                                              0                    1

         price  quantity              clean_categories  \
2989    262.78         6                   mathscience
101704  340.61         3  literacylanguage mathscience
70112   159.97        12                   mathscience

                          clean_subcategories  \
2989    environmentalscience healthlifescience
101704           literaturewriting mathematics
70112   environmentalscience healthlifescience

                                      preprocessed_essays  \
2989    did know state nevada spends least education f...
101704  we 100 free lunch program improvement title i ...
70112   my school located central kentucky quite large...

                           preprocessed_title preprocessed_prefix  \
2989          hands stem science microscopes                  ms
101704  wowing our presentations with colors                  mr
70112   flexible minds need flexible seating                 mrs

       preprocessed_grade                              preprocessed_resource
2989           Grades_6_8  my students need 5 frey scientific student mic...
101704         Grades_3_5  my students need color printer copier ink cart...
70112          Grades_6_8  my students need 5 ellipticals 5 stability bal...
==================================================
(98323, 14)
```

In [50]:

```
y1=sample_data_1['project_is_approved']
```

In [51]:

```
sample_data_1=sample_data_1.drop(columns='project_is_approved')
```

In [52]:

```
print(sample_data_1.head(3))
```

```
        Unnamed: 0 teacher_prefix school_state  \
2989        172858            Ms.           NV
101704      116753            Mr.           CA
70112        45151           Mrs.           KY

        teacher_number_of_previously_posted_projects   price  quantity  \
2989                                               3  262.78         6
101704                                             2  340.61         3
70112                                              0  159.97        12

                    clean_categories                     clean_subcategories  \
2989                     mathscience  environmentalscience healthlifescience
101704  literacylanguage mathscience           literaturewriting mathematics
70112                    mathscience  environmentalscience healthlifescience

                                      preprocessed_essays  \
2989    did know state nevada spends least education f...
101704  we 100 free lunch program improvement title i ...
70112   my school located central kentucky quite large...

                          preprocessed_title preprocessed_prefix  \
2989          hands stem science microscopes                  ms
101704  wowing our presentations with colors                  mr
70112   flexible minds need flexible seating                 mrs

       preprocessed_grade                              preprocessed_resource
2989           Grades_6_8  my students need 5 frey scientific student mic...
101704         Grades_3_5  my students need color printer copier ink cart...
70112          Grades_6_8  my students need 5 ellipticals 5 stability bal...
```

In [53]:

```
y1.value_counts(normalize=True)
```

Out[53]:

```
1    0.848703
0    0.151297
Name: project_is_approved, dtype: float64
```

In [54]:

```
#sample 2 for avg w2v and tfidf w2v
sample_data_2=x.sample(frac=.40)
```

In [55]:

```
print(sample_data_2.head(3))
```

```
        Unnamed: 0 teacher_prefix school_state  \
90859        94690           Mrs.           TX
103455      139845            Ms.           CA
83219       103768           Mrs.           KY

        teacher_number_of_previously_posted_projects  project_is_approved  \
90859                                              5                    1
103455                                            51                    1
83219                                             70                    1

         price  quantity            clean_categories  \
90859    14.85        96            literacylanguage
103455  133.49         5  historycivics specialneeds
83219   153.87        26             appliedlearning

                   clean_subcategories  \
90859                         literacy
103455  financialliteracy specialneeds
```

```
103455  financialliteracy specialneeds
83219                parentinvolvement

                                      preprocessed_essays  \
90859   my students brilliant opinionated loving stude...
103455  my students come various backgrounds their soc...
83219   tell i forget teach i may remember involve i l...

                                       preprocessed_title preprocessed_prefix  \
90859                    first grade begins novel studies                 mrs
103455  learning financial literacy working pizza rest...                  ms
83219                                  parent involvement                 mrs

       preprocessed_grade                              preprocessed_resource
90859       Grades_PreK_2  my students need popular classic beginner nove...
103455         Grades_6_8  my students need osmo kits pizza co add hands ...
83219       Grades_PreK_2  my students need storage boxes storage bench c...
```

In [56]:

```
y2=sample_data_2['project_is_approved']
```

In [57]:

```
sample_data_2=sample_data_2.drop(columns='project_is_approved')
```

In [58]:

```
print(sample_data_2.head(3))
```

```
        Unnamed: 0 teacher_prefix school_state  \
90859        94690            Mrs.           TX
103455      139845             Ms.           CA
83219       103768            Mrs.           KY

        teacher_number_of_previously_posted_projects   price  quantity  \
90859                                              5   14.85        96
103455                                            51  133.49         5
83219                                             70  153.87        26

                  clean_categories             clean_subcategories  \
90859             literacylanguage                        literacy
103455  historycivics specialneeds  financialliteracy specialneeds
83219              appliedlearning               parentinvolvement

                                      preprocessed_essays  \
90859   my students brilliant opinionated loving stude...
103455  my students come various backgrounds their soc...
83219   tell i forget teach i may remember involve i l...

                                       preprocessed_title preprocessed_prefix  \
90859                    first grade begins novel studies                 mrs
103455  learning financial literacy working pizza rest...                  ms
83219                                  parent involvement                 mrs

       preprocessed_grade                              preprocessed_resource
90859       Grades_PreK_2  my students need popular classic beginner nove...
103455         Grades_6_8  my students need osmo kits pizza co add hands ...
83219       Grades_PreK_2  my students need storage boxes storage bench c...
```

In [59]:

```
print(sample_data_1.shape)
print(y1.shape)
print("="*50)
print(sample_data_2.shape)
print(y2.shape)
```

```
(98323, 13)
(98323,)
==================================================
(43699, 13)
```

```
(43699,)
```

In [60]:

```
# ============================== loading libraries =========================================
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection
# ==========================================================================================
```

In [61]:

```
# split the data set into train and test
#how to stratify using knn->https://stackoverflow.com/questions/34842405/parameter-stratify-from-m
ethod-train-test-split-scikit-learn
X_1, X_test, y_1, y_test =model_selection.train_test_split(sample_data_1,y1, test_size=0.40, random
_state=5,stratify= y1)#random spliiting of data into test and train
```

In [62]:

```
X_train, X_cv, y_train, y_cv = train_test_split(X_1, y_1, test_size=0.40,random_state=5,stratify= y
_1) # this is random splitting of train data into train anc cross-validation
```

Use of Stratification->
https://www.researchgate.net/publication/324527882_Acceleration_Algorithm_for_k_Nearest_Neighbor_Classification_Based

In [63]:

```
print(X_train.head(3))
```

```
       Unnamed: 0 teacher_prefix school_state  \
85920       86146           Mrs.           FL   
82825       15596           Mrs.           IL   
24030       91384            Ms.           LA   

       teacher_number_of_previously_posted_projects   price  quantity  \
85920                                           125  217.43         7   
82825                                             0  226.29         9   
24030                                             1   27.99         5   

            clean_categories              clean_subcategories  \
85920  mathscience musicarts  environmentalscience visualarts   
82825           specialneeds                     specialneeds   
24030            mathscience                      mathematics   

                                     preprocessed_essays  \
85920  we school located outside tampa florida high p...   
82825  i work students disabilities high poverty area...   
24030  my students inner city african american kids e...   

                                      preprocessed_title preprocessed_prefix  \
85920  exploring groundwater pollution an earth day i...                 mrs   
82825                         sitting still while moving                 mrs   
24030                                        math stools                  ms   

      preprocessed_grade                              preprocessed_resource  
85920         Grades_6_8  my students need 2 experiment kits watercolor ...  
82825         Grades_3_5  my students need opportunity move sit without ...  
24030         Grades_6_8  my students need technology center classroom f...  
```

In [64]:

```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(35395, 13) (35395,)
(23598, 13) (23598,)
(39330, 13) (39330,)
====================================================================================================
```

Difference between fit, transform and fit_transform

Reference: https://datascience.stackexchange.com/questions/12321/difference-between-fit-and-fit-transform-in-scikit-learn-models

Reference: https://stackoverflow.com/questions/45704226/what-does-fit-method-in-scikit-learn-do

These methods are used for dataset transformations in scikit-learn:

Let us take an example for Scaling values in a dataset:

Fit : fit method, when applied to the training dataset,learns the model parameters (for example, mean and standard deviation). We then need to apply the;

Transform : transform method on the training dataset to get the transformed (scaled) training dataset. We could also perform both of this steps in one step by applying fit_transform on the training dataset.

Then why do we need 2 separate methods - fit and transform ?

In practice we need to have a separate training and testing dataset and that is where having a separate fit and transform method helps. We apply fit on the training dataset and use the transform method on both - the training dataset and the test dataset. Thus the training as well as the test dataset are then transformed(scaled) using the model parameters that were learnt on applying the fit method the training dataset.

In case of algorithms like KNN or logistic Regression etc the fit model learns the best function in the training data and then estimated method is then applied using transform on the test data to calculate the class-label for the test data

Vectorization

One Hot Encoding of Categorical Data

Category feature

In [65]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
) #creating vocabulary
vectorizer.fit(X_train['clean_categories'].values) #learning from the train data
print(vectorizer.get_feature_names())
print('='*50)
categories_ohe_train=vectorizer.transform(X_train['clean_categories'].values)#applying learned par
ameters to train,test and cv values
print("Shape of train data after one hot encoding",categories_ohe_train.shape)
print("train data after one hot encoding",categories_ohe_train[0:5, :])
categories_ohe_cv=vectorizer.transform(X_cv['clean_categories'].values)
print('='*50)
print("Shape of CV data after one hot encoding",categories_ohe_cv.shape)
print("CV data after one hot encoding",categories_ohe_cv[0:5, :])
categories_ohe_test=vectorizer.transform(X_test['clean_categories'].values)
print('='*50)
print("Shape of test data after one hot encoding",categories_ohe_test.shape)
print("test data after one hot encoding",categories_ohe_test[0:5, :])
```

```
['carehunger', 'warmth', 'historycivics', 'musicarts', 'appliedlearning', 'specialneeds',
'healthsports', 'mathscience', 'literacylanguage']
==================================================
Shape of train data after one hot encoding (35395, 9)
train data after one hot encoding   (0, 3)	1
  (0, 7)	1
  (1, 5)	1
  (2, 7)	1
  (3, 5)	1
  (4, 7)	1
==================================================
Shape of CV data after one hot encoding (23598, 9)
CV data after one hot encoding   (0, 7)	1
  (1, 5)	1
  (1, 7)	1
  (2, 8)	1
  (3, 7)	1
  (4, 5)	1
==================================================
Shape of test data after one hot encoding (39330, 9)
test data after one hot encoding   (0, 7)	1
  (1, 7)	1
  (2, 6)	1
  (3, 7)	1
  (3, 8)	1
  (4, 7)	1
```

Sub-Category feature

In [66]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_subcat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print('='*50)
subcategories_ohe_train=vectorizer.transform(X_train['clean_subcategories'].values)#applying learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",subcategories_ohe_train.shape)
print("train data after one hot encoding",subcategories_ohe_train[0:5,:])
subcategories_ohe_cv=vectorizer.transform(X_cv['clean_subcategories'].values)
print('='*50)
print("Shape of CV data after one hot encoding",subcategories_ohe_cv.shape)
print("CV data after one hot encoding",subcategories_ohe_cv[0:5,:])
subcategories_ohe_test=vectorizer.transform(X_test['clean_subcategories'].values)
print('='*50)
print("Shape of test data after one hot encoding",subcategories_ohe_test.shape)
print("test data after one hot encoding",subcategories_ohe_test[0:5,:])
```

```
['economics', 'communityservice', 'financialliteracy', 'parentinvolvement', 'extracurricular',
'civicsgovernment', 'foreignlanguages', 'nutritioneducation', 'carehunger', 'warmth',
'socialsciences', 'performingarts', 'charactereducation', 'teamsports', 'other',
'collegecareerprep', 'music', 'historygeography', 'healthlifescience', 'earlydevelopment', 'esl',
'gymfitness', 'environmentalscience', 'visualarts', 'healthwellness', 'appliedsciences',
'specialneeds', 'literaturewriting', 'mathematics', 'literacy']
==================================================
Shape of train data after one hot encoding (35395, 30)
train data after one hot encoding   (0, 22)	1
  (0, 23)	1
  (1, 26)	1
  (2, 28)	1
  (3, 26)	1
  (4, 22)	1
==================================================
Shape of CV data after one hot encoding (23598, 30)
CV data after one hot encoding   (0, 28)	1
  (1, 26)	1
  (1, 28)	1
  (2, 20)	1
  (2, 29)	1
  (3, 18)	1
  (4, 26)	1
==================================================
Shape of test data after one hot encoding (39330, 30)
test data after one hot encoding   (0, 25)	1
  (1, 28)	1
  (2, 21)	1
  (2, 24) 1
  (3, 20) 1
  (3, 28) 1
  (4, 18) 1
  (4, 22) 1
```

School-State feature

In [67]:

```
#counting number of words in the project grade category and then coverting into dictionary
from collections import Counter
my_counter=Counter()
for state in train_data['school_state'].values:
    my_counter.update(state.split())

#Converting to dictionary
school_state_dict=dict(my_counter)
#sorting
sorted_school_state_dict=dict(sorted(school_state_dict.items(),key=lambda kv:(kv[1],kv[0])))
```

In [68]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lowercase=False, bi
nary=True)
vectorizer.fit(X_train['school_state'].values)
print(vectorizer.get_feature_names())
print('='*50)
state_ohe_train=vectorizer.transform(X_train['school_state'].values)#applying learned parameters t
o train,test and cv values
print("Shape of train data after one hot encoding",state_ohe_train.shape)
print("train data after one hot encoding",state_ohe_train[0:5,:])
state_ohe_cv=vectorizer.transform(X_cv['school_state'].values)
print('='*50)
print("Shape of CV data after one hot encoding",state_ohe_cv.shape)
print("CV data after one hot encoding",state_ohe_cv[0:5,:])
state_ohe_test=vectorizer.transform(X_test['school_state'].values)
print('='*50)
print("Shape of test data after one hot encoding",state_ohe_test.shape)
print("test data after one hot encoding",state_ohe_test[0:5,:])
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'I
A', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ',
'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX
', 'CA']
==================================================
Shape of train data after one hot encoding (35395, 51)
train data after one hot encoding   (0, 47) 1
  (1, 45) 1
  (2, 37) 1
  (3, 49) 1
  (4, 29) 1
==================================================
Shape of CV data after one hot encoding (23598, 51)
CV data after one hot encoding   (0, 3) 1
  (1, 38) 1
  (2, 5) 1
  (3, 50) 1
  (4, 41) 1
==================================================
Shape of test data after one hot encoding (39330, 51)
test data after one hot encoding   (0, 50) 1
  (1, 39) 1
  (2, 43) 1
  (3, 47) 1
  (4, 43) 1
```

Project_Grade feature

In [69]:

```
from collections import Counter
my_counter1 = Counter()
for word in train_data['preprocessed_grade'].values:
    my_counter1.update(word.split())

#converting to dictionary
project_grade_dict=dict(my_counter1)
#Now sorting the dictionary
sorted_project_grade_dict = dict(sorted(project_grade_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
print(sorted_project_grade_dict)
```

```
{'Grades_9_12': 10963, 'Grades_6_8': 16923, 'Grades_3_5': 37137, 'Grades_PreK_2': 44225}
```

In [70]:

```
#How to remove values from a dictionary in python-> https://thispointer.com/different-ways-to-remo
ve-a-key-from-dictionary-in-python/
if 'Grades' in sorted_project_grade_dict:
    del sorted_project_grade_dict['Grades']

print("Updated Dictionary :" , sorted_project_grade_dict)
```

```
Updated Dictionary : {'Grades_9_12': 10963, 'Grades_6_8': 16923, 'Grades_3_5': 37137,
'Grades_PreK_2': 44225}
```

In [71]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_dict.keys()), lowercase=False, b
inary=True)
vectorizer.fit(X_train['preprocessed_grade'].values)
print(vectorizer.get_feature_names())
print('='*50)
grade_ohe_train=vectorizer.transform(X_train['preprocessed_grade'].values)#applying learned
parameters to train,test and cv values
print("Shape of train data after one hot encoding",grade_ohe_train.shape)
print("train data after one hot encoding",grade_ohe_train[0:5,:])
grade_ohe_cv=vectorizer.transform(X_cv['preprocessed_grade'].values)
print('='*50)
print("Shape of CV data after one hot encoding",grade_ohe_cv.shape)
print("cv data after one hot encoding",grade_ohe_cv[0:5,:])
grade_ohe_test=vectorizer.transform(X_test['preprocessed_grade'].values)
print('='*50)
print("Shape of test data after one hot encoding",grade_ohe_test.shape)
print("test data after one hot encoding",grade_ohe_test[0:5,:])
```

```
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'Grades_PreK_2']
==================================================
Shape of train data after one hot encoding (35395, 4)
train data after one hot encoding   (0, 1)	1
  (1, 2)	1
  (2, 1)	1
  (3, 3)	1
  (4, 2)	1
==================================================
Shape of CV data after one hot encoding (23598, 4)
cv data after one hot encoding   (0, 3)	1
  (1, 1)	1
  (2, 3)	1
  (3, 0)	1
  (4, 2)	1
==================================================
Shape of test data after one hot encoding (39330, 4)
test data after one hot encoding   (0, 0)	1
  (1, 2)	1
  (2, 2)	1
  (3, 3)	1
  (4, 3)	1
```

Teacher-Prefix feature

In [72]:

In [72]:

```
train_data['preprocessed_prefix']= train_data['preprocessed_prefix'].fillna('missing')
print("="*50)
print(train_data['preprocessed_prefix'].value_counts())
```

```
==================================================
mrs        57269
ms         38955
mr         10648
teacher     2360
dr            13
nan            3
Name: preprocessed_prefix, dtype: int64
```

In [73]:

```
from collections import Counter
my_counter1 = Counter()
for word in train_data['preprocessed_prefix'].values:
    my_counter1.update(word.split())

#converting to dictionary
teacher_prefix_dict=dict(my_counter1)
#Now sorting the dictionary
sorted_teacher_prefix_grade_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
print(sorted_teacher_prefix_grade_dict)
```

```
{'nan': 3, 'dr': 13, 'teacher': 2360, 'mr': 10648, 'ms': 38955, 'mrs': 57269}
```

In [74]:

```
#to counter error: np.nan is an invalid document, expected byte or unicode string.
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is-an-invalid-document

vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_grade_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['preprocessed_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())
print('='*50)
prefix_ohe_train=vectorizer.transform(X_train['preprocessed_prefix'].values.astype('U'))#applying learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",prefix_ohe_train.shape)
print("train data after one hot encoding",prefix_ohe_train[0:5,:])
prefix_ohe_cv=vectorizer.transform(X_cv['preprocessed_prefix'].values.astype('U'))
print('='*50)
print("Shape of CV data after one hot encoding",prefix_ohe_cv.shape)
print("cv data after one hot encoding",prefix_ohe_cv[0:5,:])
prefix_ohe_test=vectorizer.transform(X_test['preprocessed_prefix'].values.astype('U'))
print('='*50)
print("Shape of test data after one hot encoding",prefix_ohe_test.shape)
print("test data after one hot encoding",prefix_ohe_test[0:5,:])
```

```
['nan', 'dr', 'teacher', 'mr', 'ms', 'mrs']
==================================================
Shape of train data after one hot encoding (35395, 6)
train data after one hot encoding   (0, 5)	1
  (1, 5)	1
  (2, 4)	1
  (3, 5)	1
  (4, 5)	1
==================================================
Shape of CV data after one hot encoding (23598, 6)
cv data after one hot encoding   (0, 5)	1
  (1, 4)	1
  (2, 5)	1
  (3, 2)	1
  (4, 4)	1
==================================================
Shape of test data after one hot encoding (39330, 6)
test data after one hot encoding   (0, 4)	1
  (1, 5)	1
  (2, 3) 1
  (3, 5) 1
  (4, 5) 1
```

Numerical Features

Price feature

In [75]:

```
from sklearn.preprocessing import Normalizer
price_scalar = Normalizer()
price_scalar.fit(X_train['price'].values.reshape(-1,1))


price_train=price_scalar.transform(X_train['price'].values.reshape(-1, 1))
print("Shape of price train data after normalization",price_train.shape)
price_cv=price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
print("Shape of price CV data after normalization",price_cv.shape)
price_test=price_scalar.transform(X_test['price'].values.reshape(-1, 1))
print("Shape of price test data after normalization",price_test.shape)
```

```
Shape of price train data after normalization (35395, 1)
Shape of price CV data after normalization (23598, 1)
Shape of price test data after normalization (39330, 1)
```

Quantity Feature

In [76]:

```
quantity_scalar = Normalizer()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data


quantity_train=quantity_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
print("Shape of quantity train data after normalization",quantity_train.shape)
quantity_cv=quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
print("Shape of quantity CV data after normalization",quantity_cv.shape)
quantity_test=quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
print("Shape of quantity test data after normalization",quantity_test.shape)
```

```
Shape of quantity train data after normalization (35395, 1)
Shape of quantity CV data after normalization (23598, 1)
Shape of quantity test data after normalization (39330, 1)
```

Teacher number of previously posted projects feature

In [77]:

```
tnp_scalar = Normalizer()
tnp_scalar.fit(X_train["teacher_number_of_previously_posted_projects"].values.reshape(-1,1)) # find
ing the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
tnp_train = tnp_scalar.transform(X_train["teacher_number_of_previously_posted_projects"].values.re
shape(-1, 1))
print('='*50)
print(tnp_train.shape)
tnp_cv = tnp_scalar.transform(X_cv["teacher_number_of_previously_posted_projects"].values.reshape(
-1, 1))
print(tnp_cv.shape)
tnp_test =
tnp_scalar.transform(X_test["teacher_number_of_previously_posted_projects"].values.reshape(-1, 1))
print(tnp_test.shape)
```

```
==================================================
(35395, 1)
(23598, 1)
(39330, 1)
```

Bag of Words

Preprocessed Essay

In [82]:

```
model =  CountVectorizer()
model.fit(X_train["preprocessed_essays"])
train_bow_essay = model.transform(X_train["preprocessed_essays"])
print("Shape of matrix ",train_bow_essay.shape)
print("="*50)
cv_bow_essay=model.transform(X_cv["preprocessed_essays"]) #BoW of CV
print("Shape of matrix ",cv_bow_essay.shape)
print("="*50)
test_bow_essay = model.transform(X_test["preprocessed_essays"]) #BoW of Test
print("Shape of matrix ",test_bow_essay.shape)
```

```
Shape of matrix  (35395, 36613)
==================================================
Shape of matrix  (23598, 36613)
==================================================
Shape of matrix  (39330, 36613)
```

Preprocessed Title

In [78]:

```
model =  CountVectorizer()
model.fit(X_train["preprocessed_title"])
train_bow_title = model.transform(X_train["preprocessed_title"])
print("Shape of matrix ",train_bow_title.shape)
cv_bow_title=model.transform(X_cv["preprocessed_title"]) #BoW of test
print("Shape of matrix ",cv_bow_title.shape)
test_bow_title = model.transform(X_test["preprocessed_title"]) #BoW of Cross Validation
print("Shape of matrix ",test_bow_title.shape)
```

```
Shape of matrix  (35395, 10084)
Shape of matrix  (23598, 10084)
Shape of matrix  (39330, 10084)
```

Tf-idf

Preprocessed Essay

In [79]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model = TfidfVectorizer(min_df=10) #df tells us that we will only consider those words which is
present atleast in 10 documents
model.fit(X_train["preprocessed_essays"])
train_tfidf_essay = model.transform(X_train["preprocessed_essays"])
print("Shape of matrix ",train_tfidf_essay.shape)
cv_tfidf_essay=model.transform(X_cv["preprocessed_essays"]) #BoW of test
print("Shape of matrix ",cv_tfidf_essay.shape)
test_tfidf_essay= model.transform(X_test["preprocessed_essays"]) #BoW of Cross Validation
print("Shape of matrix ",test_tfidf_essay.shape)
```

```
Shape of matrix  (35395, 10703)
Shape of matrix  (23598, 10703)
Shape of matrix  (39330, 10703)
```

Preprocessed Title

In [80]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model = TfidfVectorizer(min_df=10) #df tells us that we will only consider those words which is
present atleast in 10 documents
model.fit(X_train["preprocessed_title"])
train_tfidf_title = model.transform(X_train["preprocessed_title"])
print("Shape of matrix ",train_tfidf_title.shape)
cv_tfidf_title=model.transform(X_cv["preprocessed_title"]) #BoW of cv
print("Shape of matrix ",cv_tfidf_title.shape)
test_tfidf_title= model.transform(X_test["preprocessed_title"]) #BoW of test
print("Shape of matrix ",test_tfidf_title.shape)
```

```
Shape of matrix  (35395, 1715)
Shape of matrix  (23598, 1715)
Shape of matrix  (39330, 1715)
```

Applying KNN

Merging all the features

Set 1: Categorical Features,Numerical Features+Preprocessed Essay(BOW)+Preprocessed Title(BOW)

In [83]:

```
from scipy.sparse import hstack
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train,train_bow_essay,train_bow_title)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv,cv_bow_essay,cv_bow_title)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test,test_bow_essay,test_bow_title)).tocsr()
```

In [84]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

```
(35395, 46800) (35395,)
==================================================
(23598, 46800) (23598,)
==================================================
(39330, 46800) (39330,)
```

Simple Brute Force

Finding Hyper parameter using AUC value

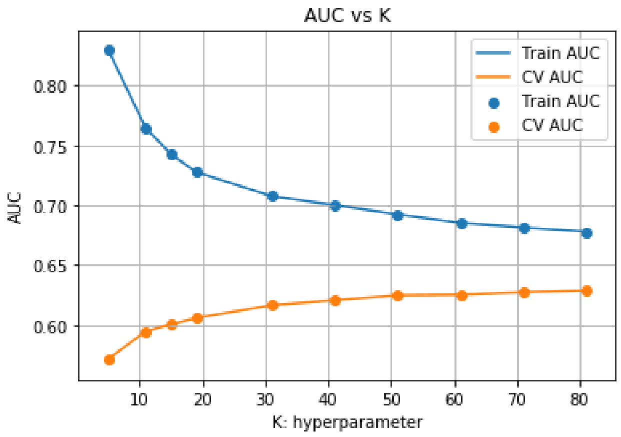
In [85]:

```
#writting function for using batch-wise prediction
def batch_predict(neigh,data):
    '''Batch-Wise prediction is used to predict the class label in batches to fast process the knn
```

```
algorithm'''
    y_train_pred = []
    loop_value=data.shape[0]-data.shape[0]%1000
    for i in range(0, loop_value, 1000): #range will be from 0 to 49041 with step of 1000pts each t
ime so are values will be between 0-4000
        y_train_pred.extend(neigh.predict_proba(data[i:i+1000])[:,1])
    y_train_pred.extend(neigh.predict_proba(data[loop_value:])[:,1])

    return y_train_pred
```

In [86]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
train_auc = []
cv_auc = []
K = [5, 11, 15, 19, 31, 41, 51,61,71,81]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  batch_predict(neigh,X_tr)
    y_cv_pred =  batch_predict(neigh,X_cv)
    #print(y_train.shape)
    #print(len(y_train_pred))
     #roc_auc_score->Compute(ROC AUC) from prediction scores.
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))


plt.plot(K, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(K, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(K, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(K, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("K: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs K") #adding title of the plot
plt.grid()
plt.show()
```

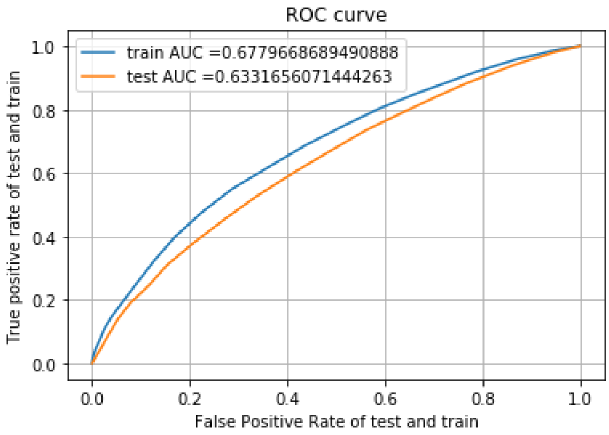


Looking at the plot our best hyperparameter is 81

Testing on Test Data(using our best hyper parameter=81)

In [87]:

```
from sklearn.metrics import roc_curve,auc

neigh=KNeighborsClassifier(n_neighbors= 81)
neigh.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=batch_predict(neigh,X_tr)
y_test_predict=batch_predict(neigh,X_te)
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation
of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation -
>https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("ROC curve")
plt.grid()
plt.show()
```



Confusion Matrix

In [88]:

```
def pred(proba,thresh, fpr ,tpr):
    """This function calculations and return the prediction with highest tpr and lowest tpr"""
    t=thresh[np.argmax(tpr*(1-fpr))] #t creates a numpy array with the max fpr and lowest tpr
    print("the maximum tpr*(1-fpr)",max(tpr*(1-fpr)),"for threshold",np.round(t,3))
    predictions=[]
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Train Data

In [89]:

```
from sklearn.metrics import confusion_matrix

print("Train confusion matrix")
cm=confusion_matrix(y_train, pred(y_train_predict, train_thresholds,train_fpr,train_tpr))
print(cm)
```

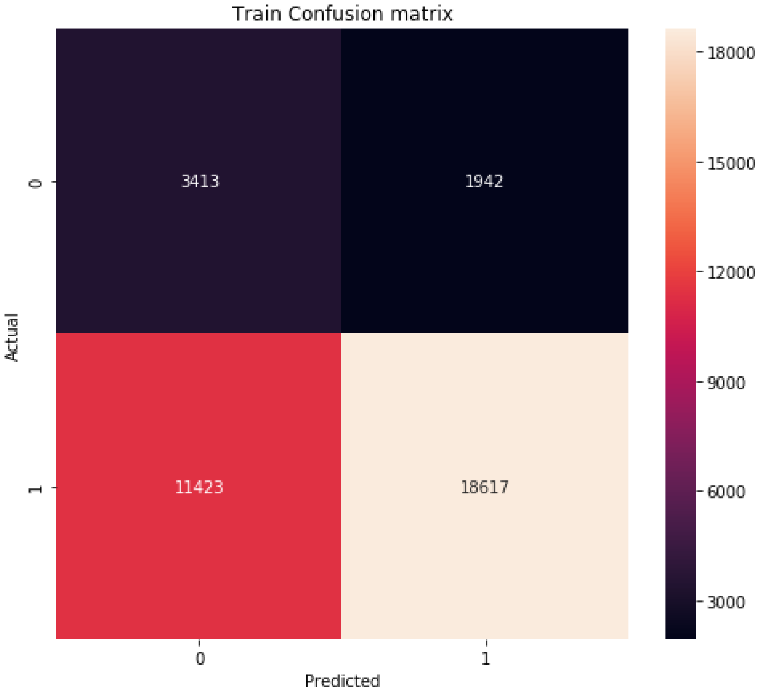
```
Train confusion matrix
the maximum tpr*(1-fpr) 0.39499043914059184 for threshold 0.79
the maximum tpr*(1-fpr) 0.39499043914059184 for threshold 0.79
[[ 3413  1942]
 [11423 18617]]
```

In [90]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[90]:

```
Text(0.5, 42.0, 'Predicted')
```



For the training data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=62%
2. Misclassification Rate= 1-Accuracy = 38% i.e. our model made 38% predictions wrong.
3. Sensitivity or Recall= 62% percentage of total number of positive correct predictions
4. Specificity= 63% percentage of total number of correct negative predictions.
5. Precision= 90% percentage of time when we predicted yes we are correct.

Test Data

In [91]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
cm1=confusion_matrix(y_test, pred(y_test_predict, test_thresholds,test_fpr,test_tpr))
print(cm1)
```

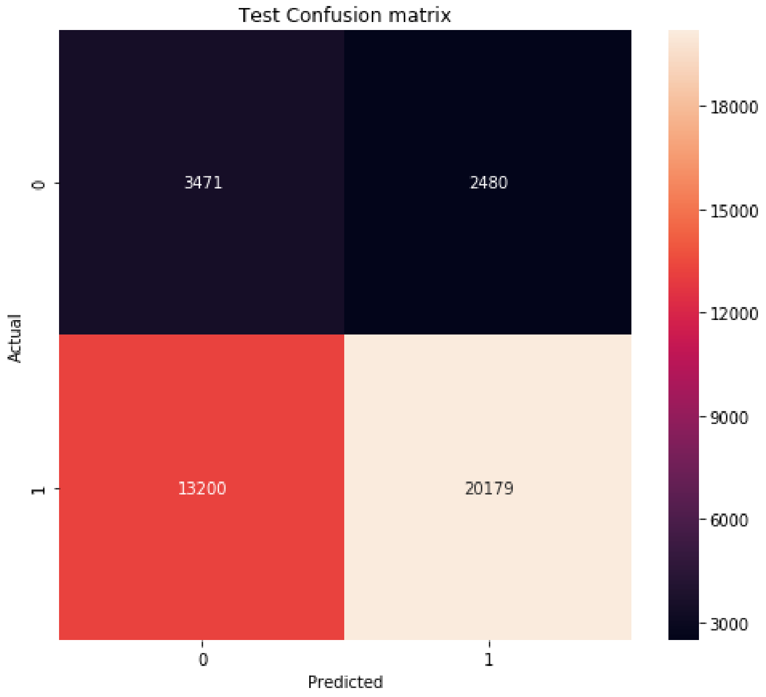
```
Test confusion matrix
the maximum tpr*(1-fpr) 0.35260704261812303 for threshold 0.79
[[ 3471  2480]
 [13200 20179]]
```

In [92]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[92]:

```
Text(0.5, 42.0, 'Predicted')
```



For the testing data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=60%
2. Misclassification Rate= 1-Accuracy = 40% i.e. our model made 38% predictions wrong.
3. Sensitivity or Recall= 60% percentage of total number of positive correct predictions
4. Specificity= 58% percentage of total number of correct negative predictions.
5. Precision= 89% percentage of time when we predicted yes we are correct.

Set 2: Categorical Features,Numerical Features+Preprocessed Essay(Tfidf)+Preprocessed Title(Tfidf)

In [93]:

```
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train,train_tfidf_title,train_tfidf_essay)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv,cv_tfidf_essay,cv_tfidf_title)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test,test_tfidf_essay,test_tfidf_title)).tocsr()
```

In [94]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

```
(35395, 12521) (35395,)
==================================================
(23598, 12521) (23598,)
==================================================
(39330, 12521) (39330,)
```

In [95]:

```
X_tr1= X_tr
y_train1= y_train
X_cv1= X_cv
y_cv1= y_cv
X_te1= X_te
y_test1= y_test
```

In [96]:

```
#checking the final matrix are of same dimension or not
print(X_tr1.shape,y_train1.shape)
print("="*50)
print(X_cv1.shape,y_cv1.shape)
print("="*50)
print(X_te1.shape,y_test1.shape)
```

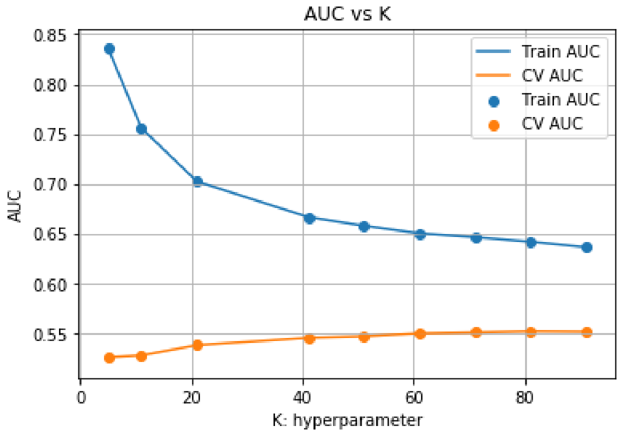
```
(35395, 12521) (35395,)
==================================================
(23598, 12521) (23598,)
==================================================
(39330, 12521) (39330,)
```

In [97]:

```
#writting function for using batch-wise prediction
def batch_predict(neigh,data):
    '''Batch-Wise prediction is used to predict the class label in batches to fast process the knn
algorithm'''
    y_train_pred = []
    loop_value=data.shape[0]-data.shape[0]%1000
    for i in range(0, loop_value, 1000): #range will be from 0 to 49041 with step of 1000pts each t
ime so are values will be between 0-4000
        y_train_pred.extend(neigh.predict_proba(data[i:i+1000])[:,1])
    y_train_pred.extend(neigh.predict_proba(data[loop_value:])[:,1])

    return y_train_pred
```

In [98]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
train_auc = []
cv_auc = []
K = [5, 11, 21, 41, 51,61,71,81,91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(X_tr, y_train)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  batch_predict(neigh,X_tr)
    y_cv_pred =  batch_predict(neigh,X_cv)

    #print(y_train.shape)
    #print(len(y_train_pred))
     #roc_auc_score->Compute(ROC AUC) from prediction scores.
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->computes(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))


plt.plot(K, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(K, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(K, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(K, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("K: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs K") #adding title of the plot
plt.grid()
plt.show()
```

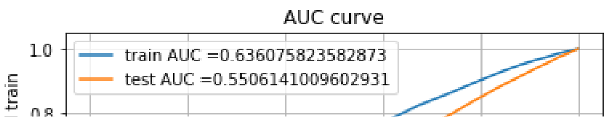


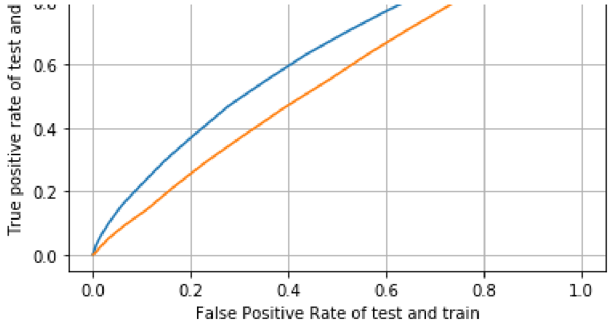
Looking at the plot our best hyperparameter is 91

Testing on Test Data(using our best hyper parameter=91)

In [99]:

```
from sklearn.metrics import roc_curve,auc

neigh=KNeighborsClassifier(n_neighbors= 91)
neigh.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=batch_predict(neigh,X_tr)
y_test_predict=batch_predict(neigh,X_te)
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation
of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation -
>https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("AUC curve")
plt.grid()
plt.show()
```





Confusion Matrix

In [100]:

```
def pred(proba,thresh, fpr ,tpr):
    """This function calculations and return the prediction with highest tpr and lowest tpr"""
    t=thresh[np.argmax(tpr*(1-fpr))] #t creates a numpy array with the max fpr and lowest tpr
    print("the maximum tpr*(1-fpr)",max(tpr*(1-fpr)),"for threshold",np.round(t,3))
    predictions=[]
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Train Data

In [101]:

```
from sklearn.metrics import confusion_matrix

print("Train confusion matrix")
cm=confusion_matrix(y_train, pred(y_train_predict, train_thresholds,train_fpr,train_tpr))
print(cm)
```

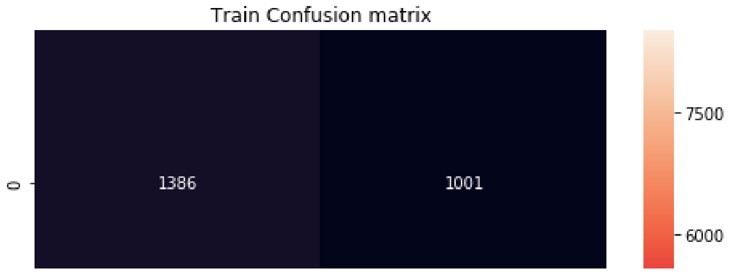
```
Train confusion matrix
the maximum tpr*(1-fpr) 0.355611553098825 for threshold 0.846
[[ 3007  2348]
 [11016 19024]]
```

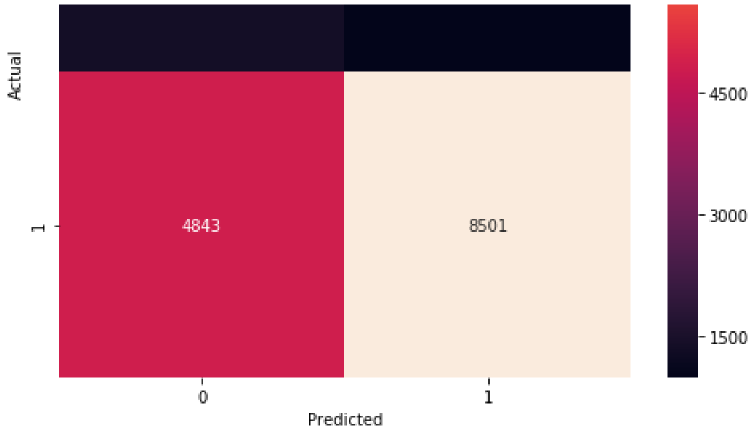
In [161]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[161]:

```
Text(0.5, 42.0, 'Predicted')
```





For the training data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=62%
2. Misclassification Rate= 1-Accuracy = 38% i.e. our model made 38% predictions wrong.
3. Sensitivity or Recall= 63% percentage of total number of positive correct predictions
4. Specificity= 58% percentage of total number of correct negative predictions.
5. Precision= 89% percentage of time when we predicted yes we are correct.

Test Data

In [103]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
cm1=confusion_matrix(y_test, pred(y_test_predict, test_thresholds,test_fpr,test_tpr))
print(cm1)
```

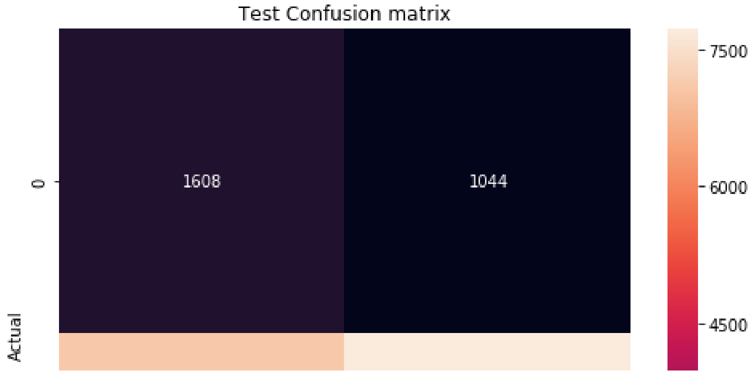
```
Test confusion matrix
the maximum tpr*(1-fpr) 0.2847772220349165 for threshold 0.857
[[ 3066  2885]
 [14929 18450]]
```

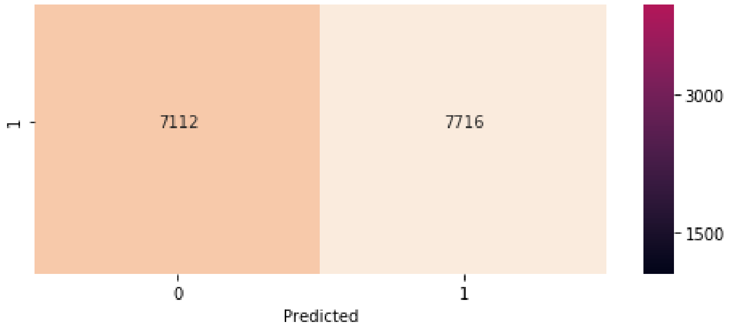
In [162]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[162]:

```
Text(0.5, 42.0, 'Predicted')
```





For the test data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=53%
2. Misclassification Rate= 1-Accuracy = 47% i.e. our model made 47% predictions wrong.
3. Sensitivity or Recall= 52% percentage of total number of positive correct predictions
4. Specificity= 60% percentage of total number of correct negative predictions.
5. Precision= 88% percentage of time when we predicted yes we are correct.

Preparing Data For Splitting

In [105]:

```
print(sample_data_2.shape)
print(y2.shape)
```

```
(43699, 13)
(43699,)
```

In [106]:

```
sample_data_2.head(3)
```

Out[106]:

	Unnamed: 0	teacher_prefix	school_state	teacher_number_of_previously_posted_projects	price	quantity	clean_categories	clea
90859	94690	Mrs.	TX	5	14.85	96	literacylanguage	
103455	139845	Ms.	CA	51	133.49	5	historycivics specialneeds	
83219	103768	Mrs.	KY	70	153.87	26	appliedlearning	

In [107]:

```
# split the data set into train and test
#how to stratify using knn->https://stackoverflow.com/questions/34842405/parameter-stratify-from-m
ethod-train-test-split-scikit-learn
X_1, X_test, y_1, y_test =model_selection.train_test_split(sample_data_2,y2, test_size=0.40, random
_state=5,stratify= y2)#random spliiting of data into test and train
```

In [108]:

```
X_train, X_cv, y_train, y_cv = train_test_split(X_1, y_1, test_size=0.40,random_state=5,stratify= y
_1) # this is random splitting of train data into train anc cross-validation
```

In [109]:

```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(15731, 13) (15731,)
(10488, 13) (10488,)
(17480, 13) (17480,)
====================================================================================================
```

Vectorization

One Hot Encoding of Categorical Data

Category feature

In [110]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True) #creating vocabulary
vectorizer.fit(X_train['clean_categories'].values) #learning from the train data
print(vectorizer.get_feature_names())
print('='*50)
categories_ohe_train=vectorizer.transform(X_train['clean_categories'].values)#applying learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",categories_ohe_train.shape)
print("train data after one hot encoding",categories_ohe_train[0:5, :])
categories_ohe_cv=vectorizer.transform(X_cv['clean_categories'].values)
print('='*50)
print("Shape of CV data after one hot encoding",categories_ohe_cv.shape)
print("CV data after one hot encoding",categories_ohe_cv[0:5, :])
categories_ohe_test=vectorizer.transform(X_test['clean_categories'].values)
print('='*50)
print("Shape of test data after one hot encoding",categories_ohe_test.shape)
print("test data after one hot encoding",categories_ohe_test[0:5, :])
```

```
['carehunger', 'warmth', 'historycivics', 'musicarts', 'appliedlearning', 'specialneeds',
'healthsports', 'mathscience', 'literacylanguage']
==================================================
Shape of train data after one hot encoding (15731, 9)
train data after one hot encoding   (0, 2)	1
  (0, 8)	1
  (1, 7)	1
  (2, 7)	1
  (3, 7)	1
  (4, 5)	1
==================================================
Shape of CV data after one hot encoding (10488, 9)
CV data after one hot encoding   (0, 6)	1
  (1, 4)	1
  (1, 7)	1
  (2, 7)	1
  (2, 8)	1
  (3, 8)	1
  (4, 5)	1
  (4, 7)	1
==================================================
Shape of test data after one hot encoding (17480, 9)
test data after one hot encoding   (0, 8)	1
  (1, 8)	1
  (2, 3)	1
  (2, 7)	1
  (3, 2)	1
  (3, 7)	1
  (4, 5)	1
```

Sub-Category feature

In [111]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_subcat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print('='*50)
subcategories_ohe_train=vectorizer.transform(X_train['clean_subcategories'].values)#applying learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",subcategories_ohe_train.shape)
print("train data after one hot encoding",subcategories_ohe_train[0:5,:])
subcategories_ohe_cv=vectorizer.transform(X_cv['clean_subcategories'].values)
print('='*50)
print("Shape of CV data after one hot encoding",subcategories_ohe_cv.shape)
print("CV data after one hot encoding",subcategories_ohe_cv[0:5,:])
subcategories_ohe_test=vectorizer.transform(X_test['clean_subcategories'].values)
print('='*50)
print("Shape of test data after one hot encoding",subcategories_ohe_test.shape)
print("test data after one hot encoding",subcategories_ohe_test[0:5,:])
```

```
['economics', 'communityservice', 'financialliteracy', 'parentinvolvement', 'extracurricular',
'civicsgovernment', 'foreignlanguages', 'nutritioneducation', 'carehunger', 'warmth',
'socialsciences', 'performingarts', 'charactereducation', 'teamsports', 'other',
'collegecareerprep', 'music', 'historygeography', 'healthlifescience', 'earlydevelopment', 'esl',
'gymfitness', 'environmentalscience', 'visualarts', 'healthwellness', 'appliedsciences',
'specialneeds', 'literaturewriting', 'mathematics', 'literacy']
==================================================
Shape of train data after one hot encoding (15731, 30)
train data after one hot encoding   (0, 17) 1
  (0, 29) 1
  (1, 28) 1
  (2, 22) 1
  (2, 25) 1
  (3, 25) 1
  (4, 26) 1
==================================================
Shape of CV data after one hot encoding (10488, 30)
CV data after one hot encoding   (0, 21) 1
  (0, 24) 1
  (1, 15) 1
  (1, 25) 1
  (2, 28) 1
  (2, 29) 1
  (3, 20) 1
  (3, 29) 1
  (4, 26) 1
  (4, 28) 1
==================================================
Shape of test data after one hot encoding (17480, 30)
test data after one hot encoding   (0, 29) 1
  (1, 27) 1
  (1, 29) 1
  (2, 23) 1
  (2, 25) 1
  (3, 10) 1
  (3, 22) 1
  (4, 26) 1
```

School-State feature

In [112]:

```
#counting number of words in the project grade category and then coverting into dictionary
from collections import Counter
my_counter=Counter()
for state in train_data['school_state'].values:
    my_counter.update(state.split())

#Converting to dictionary
school_state_dict=dict(my_counter)
#sorting
sorted_school_state_dict=dict(sorted(school_state_dict.items(),key=lambda kv:(kv[1],kv[0])))
```

In [113]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lowercase=False, bi
nary=True)
vectorizer.fit(X_train['school_state'].values)
print(vectorizer.get_feature_names())
print('='*50)
state_ohe_train=vectorizer.transform(X_train['school_state'].values)#applying learned parameters t
o train,test and cv values
print("Shape of train data after one hot encoding",state_ohe_train.shape)
print("train data after one hot encoding",state_ohe_train[0:5,:])
state_ohe_cv=vectorizer.transform(X_cv['school_state'].values)
print('='*50)
print("Shape of CV data after one hot encoding",state_ohe_cv.shape)
print("CV data after one hot encoding",state_ohe_cv[0:5,:])
state_ohe_test=vectorizer.transform(X_test['school_state'].values)
print('='*50)
print("Shape of test data after one hot encoding",state_ohe_test.shape)
print("test data after one hot encoding",state_ohe_test[0:5,:])
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'I
A', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ',
'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX
', 'CA']
==================================================
Shape of train data after one hot encoding (15731, 51)
train data after one hot encoding   (0, 50)	1
  (1, 32)	1
  (2, 46)	1
  (3, 31)	1
  (4, 45)	1
==================================================
Shape of CV data after one hot encoding (10488, 51)
CV data after one hot encoding   (0, 42)	1
  (1, 43)	1
  (2, 39)	1
  (3, 50)	1
  (4, 30)	1
==================================================
Shape of test data after one hot encoding (17480, 51)
test data after one hot encoding   (0, 25)	1
  (1, 44)	1
  (2, 43)	1
  (3, 44)	1
  (4, 47)	1
```

Project_Grade feature

In [114]:

```
from collections import Counter
my_counter1 = Counter()
for word in train_data['preprocessed_grade'].values:
    my_counter1.update(word.split())

#converting to dictionary
project_grade_dict=dict(my_counter1)
#Now sorting the dictionary
sorted_project_grade_dict = dict(sorted(project_grade_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
print(sorted_project_grade_dict)
```

```
{'Grades_9_12': 10963, 'Grades_6_8': 16923, 'Grades_3_5': 37137, 'Grades_PreK_2': 44225}
```

In [115]:

```
#How to remove values from a dictionary in python-> https://thispointer.com/different-ways-to-remo
ve-a-key-from-dictionary-in-python/
if 'Grades' in sorted_project_grade_dict:
    del sorted_project_grade_dict['Grades']

print("Updated Dictionary :" , sorted_project_grade_dict)
```

```
Updated Dictionary : {'Grades_9_12': 10963, 'Grades_6_8': 16923, 'Grades_3_5': 37137,
'Grades_PreK_2': 44225}
```

In [116]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_dict.keys()), lowercase=False, b
inary=True)
vectorizer.fit(X_train['preprocessed_grade'].values)
print(vectorizer.get_feature_names())
print('='*50)
grade_ohe_train=vectorizer.transform(X_train['preprocessed_grade'].values)#applying learned
parameters to train,test and cv values
print("Shape of train data after one hot encoding",grade_ohe_train.shape)
print("train data after one hot encoding",grade_ohe_train[0:5,:])
grade_ohe_cv=vectorizer.transform(X_cv['preprocessed_grade'].values)
print('='*50)
print("Shape of CV data after one hot encoding",grade_ohe_cv.shape)
print("cv data after one hot encoding",grade_ohe_cv[0:5,:])
grade_ohe_test=vectorizer.transform(X_test['preprocessed_grade'].values)
print('='*50)
print("Shape of test data after one hot encoding",grade_ohe_test.shape)
print("test data after one hot encoding",grade_ohe_test[0:5,:])
```

```
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'Grades_PreK_2']
==================================================
Shape of train data after one hot encoding (15731, 4)
train data after one hot encoding   (0, 3) 1
  (1, 3) 1
  (2, 1) 1
  (3, 2) 1
  (4, 2) 1
==================================================
Shape of CV data after one hot encoding (10488, 4)
cv data after one hot encoding   (0, 3) 1
  (1, 1) 1
  (2, 2) 1
  (3, 2) 1
  (4, 1) 1
==================================================
Shape of test data after one hot encoding (17480, 4)
test data after one hot encoding   (0, 3) 1
  (1, 2) 1
  (2, 0) 1
  (3, 2) 1
  (4, 3) 1
```

Teacher-Prefix feature

In [117]:

```
train_data['preprocessed_prefix']= train_data['preprocessed_prefix'].fillna('missing')
print("="*50)
print(train_data['preprocessed_prefix'].value_counts())
```

```
==================================================
mrs        57269
ms         38955
mr         10648
teacher     2360
dr            13
nan            3
Name: preprocessed_prefix, dtype: int64
```

In [118]:

```
from collections import Counter
my_counter1 = Counter()
for word in train_data['preprocessed_prefix'].values:
    my_counter1.update(word.split())

#converting to dictionary
teacher_prefix_dict = dict(my_counter1)
```

```
teacher_prefix_dict=dict(my_counter1)
#Now sorting the dictionary
sorted_teacher_prefix_grade_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv:(kv[1] ,k
v[0])))
print(sorted_teacher_prefix_grade_dict)
```

```
{'nan': 3, 'dr': 13, 'teacher': 2360, 'mr': 10648, 'ms': 38955, 'mrs': 57269}
```

In [119]:

```
#to counter error: np.nan is an invalid document, expected byte or unicode string.
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is
-an-invalid-document

vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_grade_dict.keys()), lowercase=F
alse, binary=True)
vectorizer.fit(X_train['preprocessed_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())
print('='*50)
prefix_ohe_train=vectorizer.transform(X_train['preprocessed_prefix'].values.astype('U'))#applying
learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",prefix_ohe_train.shape)
print("train data after one hot encoding",prefix_ohe_train[0:5,:])
prefix_ohe_cv=vectorizer.transform(X_cv['preprocessed_prefix'].values.astype('U'))
print('='*50)
print("Shape of CV data after one hot encoding",prefix_ohe_cv.shape)
print("cv data after one hot encoding",prefix_ohe_cv[0:5,:])
prefix_ohe_test=vectorizer.transform(X_test['preprocessed_prefix'].values.astype('U'))
print('='*50)
print("Shape of test data after one hot encoding",prefix_ohe_test.shape)
print("test data after one hot encoding",prefix_ohe_test[0:5,:])
```

```
['nan', 'dr', 'teacher', 'mr', 'ms', 'mrs']
==================================================
Shape of train data after one hot encoding (15731, 6)
train data after one hot encoding   (0, 5) 1
  (1, 4) 1
  (2, 5) 1
  (3, 5) 1
  (4, 4) 1
==================================================
Shape of CV data after one hot encoding (10488, 6)
cv data after one hot encoding   (0, 5) 1
  (1, 5) 1
  (2, 4) 1
  (3, 4) 1
  (4, 4) 1
==================================================
Shape of test data after one hot encoding (17480, 6)
test data after one hot encoding   (0, 5) 1
  (1, 4) 1
  (2, 5) 1
  (3, 4) 1
  (4, 4) 1
```

Numerical Features

Price feature

In [120]:

```
from sklearn.preprocessing import Normalizer
price_scalar = Normalizer()
price_scalar.fit(X_train['price'].values.reshape(-1,1))
```

Out[120]:

```
Normalizer(copy=True, norm='l2')
```

In [121]:

```
price_train=price_scalar.transform(X_train['price'].values.reshape(-1, 1))
print("Shape of price train data after normalization",price_train.shape)
price_cv=price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
print("Shape of price CV data after normalization",price_cv.shape)
price_test=price_scalar.transform(X_test['price'].values.reshape(-1, 1))
print("Shape of price test data after normalization",price_test.shape)
```

```
Shape of price train data after normalization (15731, 1)
Shape of price CV data after normalization (10488, 1)
Shape of price test data after normalization (17480, 1)
```

Quantity Feature

In [122]:

```
quantity_scalar = Normalizer()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
```

Out[122]:

```
Normalizer(copy=True, norm='l2')
```

In [123]:

```
quantity_train=quantity_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
print("Shape of quantity train data after normalization",quantity_train.shape)
quantity_cv=quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
print("Shape of quantity CV data after normalization",quantity_cv.shape)
quantity_test=quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
print("Shape of quantity test data after normalization",quantity_test.shape)
```

```
Shape of quantity train data after normalization (15731, 1)
Shape of quantity CV data after normalization (10488, 1)
Shape of quantity test data after normalization (17480, 1)
```

Teacher number of previously posted projects feature

In [124]:

```
tnp_scalar = Normalizer()
tnp_scalar.fit(X_train["teacher_number_of_previously_posted_projects"].values.reshape(-1,1)) # find
ing the mean and standard deviation of this data
```

Out[124]:

```
Normalizer(copy=True, norm='l2')
```

In [125]:

```
# Now standardize the data with above maen and variance.
tnp_train = tnp_scalar.transform(X_train["teacher_number_of_previously_posted_projects"].values.re
shape(-1, 1))
print('='*50)
print(tnp_train.shape)
tnp_cv = tnp_scalar.transform(X_cv["teacher_number_of_previously_posted_projects"].values.reshape(
-1, 1))
print(tnp_cv.shape)
tnp_test =
tnp_scalar.transform(X_test["teacher_number_of_previously_posted_projects"].values.reshape(-1, 1))
print(tnp_test.shape)
```

```
==================================================
(15731, 1)
(10488, 1)
(17480, 1)
```

Average word2vector(avg w2v)

In [126]:

```
#https://stackoverflow.com/questions/49083826/get-trouble-to-load-glove-840b-300d-vector
import numpy as np
from tqdm import tqdm
from tqdm import tqdm_notebook as tqdm
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding='utf8')
    model = {}
    for line in tqdm(f):
        splitLine = line.split(' ')
        word = splitLine[0]
        embedding = np.asarray(splitLine[1:], dtype='float32')
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
```

In [127]:

```
model = loadGloveModel('glove.840B.300d.txt')
```

```
Loading Glove Model


Done. 2196016  words loaded!
```

In [128]:

```
words = []
for i in X_train["preprocessed_essays"]:
    words.extend(i.split(' '))
```

In [129]:

```
print("all the words in the corpus", len(words))
words = set(words)
print("the unique words in the corpus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our corpus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

train_words_corpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        train_words_corpus[i] = model[i]
print("word 2 vec length", len(train_words_corpus))
```

```
all the words in the corpus 2362490
the unique words in the corpus 26889
The number of words that are present in both glove vectors and our corpus 24922 ( 92.685 %)
word 2 vec length 24922
```

In [130]:

```
import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(train_words_corpus, f) # save training datasets into a pickle file for machine
learning
```

In [131]:

```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
    glove_words = set(model.keys())
```

Train Essays

In [132]:

```
# average Word2Vec
# compute average word2vec for each training data

from tqdm import tqdm_notebook as tqdm
avg_w2v_vectors_train = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_train["preprocessed_essays"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

```
15731
300
```

Test Essays

In [133]:

```
# average Word2Vec
# compute average word2vec for each test data

from tqdm import tqdm_notebook as tqdm
avg_w2v_vectors_test = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_test["preprocessed_essays"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
17480
300
```

Cross-Validation Essays

In [134]:

```
# average Word2Vec
# compute average word2vec for each CV data

from tqdm import tqdm_notebook as tqdm
avg_w2v_vectors_cv = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_cv["preprocessed_essays"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
```

```
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))
```

```
10488
300
```

Train Titles

In [135]:

```
# average Word2Vec
# compute average word2vec for each training data

from tqdm import tqdm_notebook as tqdm
avg_w2v_vectors_title_train = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_train["preprocessed_title"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_train.append(vector)

print(len(avg_w2v_vectors_title_train))
print(len(avg_w2v_vectors_title_train[0]))
```

```
15731
300
```

Test Titles

In [136]:

```
# average Word2Vec
# compute average word2vec for each test data

from tqdm import tqdm_notebook as tqdm
avg_w2v_vectors_title_test = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_test["preprocessed_title"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_test.append(vector)

print(len(avg_w2v_vectors_title_test))
print(len(avg_w2v_vectors_title_test[0]))
```

```
17480
300
```

Cross-Validation Ttiles

In [137]:

```
# average Word2Vec
# compute average word2vec for each CV data

from tqdm import tqdm_notebook as tqdm
avg_w2v_vectors_title_cv = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_cv["preprocessed_title"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_cv.append(vector)

print(len(avg_w2v_vectors_title_cv))
print(len(avg_w2v_vectors_title_cv[0]))
```

```
10488
300
```

Tf-idf weighted W2V

Using Pretrained Model for finding the tf-idf weighted word2vec

Train Essays

In [138]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train["preprocessed_essays"])
# 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 [139]:

```
# compute average word2vec for Training Data
from tqdm import tqdm_notebook as tqdm
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence
for sentence in tqdm(X_train["preprocessed_essays"]): # for each 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
    for word in sentence.split(): # for each word in a 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_w2v_vectors_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))
```

```
15731
300
```

Cross-Validation Essays

In [140]:

```
# compute average word2vec for Cross Validation data
from tqdm import tqdm_notebook as tqdm
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence
for sentence in tqdm(X_cv["preprocessed_essays"]): # for each 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
    for word in sentence.split(): # for each word in a 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_w2v_vectors_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

```
10488
300
```

Test Essays

In [141]:

```
# compute average word2vec for test data
from tqdm import tqdm_notebook as tqdm
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence
for sentence in tqdm(X_test["preprocessed_essays"]): # for each 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
    for word in sentence.split(): # for each word in a 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_w2v_vectors_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
17480
300
```

Train Titles

In [142]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train["preprocessed_title"])
# 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 [143]:

```
# compute average word2vec for Training Data
from tqdm import tqdm_notebook as tqdm
```

```
tfidf_w2v_vectors_title_train = []; # the avg-w2v for each sentence
for sentence in tqdm(X_train["preprocessed_title"]): # for each 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
    for word in sentence.split(): # for each word in a 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_w2v_vectors_title_train.append(vector)

print(len( tfidf_w2v_vectors_title_train))
print(len( tfidf_w2v_vectors_title_train[0]))
```

```
15731
300
```

Cross-Validation Titles

In [144]:

```
# compute average word2vec for Cross-Validation Data
from tqdm import tqdm_notebook as tqdm
tfidf_w2v_vectors_title_cv = []; # the avg-w2v for each sentence
for sentence in tqdm(X_cv["preprocessed_title"]): # for each 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
    for word in sentence.split(): # for each word in a 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_w2v_vectors_title_cv.append(vector)

print(len( tfidf_w2v_vectors_title_cv))
print(len( tfidf_w2v_vectors_title_cv[0]))
```

```
10488
300
```

Test titles

In [145]:

```
# compute average word2vec for Test Data
from tqdm import tqdm_notebook as tqdm
tfidf_w2v_vectors_title_test = []; # the avg-w2v for each sentence
for sentence in tqdm(X_test["preprocessed_title"]): # for each 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
    for word in sentence.split(): # for each word in a 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
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_test.append(vector)

print(len( tfidf_w2v_vectors_title_test))
print(len( tfidf_w2v_vectors_title_test[0]))
```

```
17480
300
```

Set 3: Categorical Features,Numerical Features+Preprocessed Essay(Avg W2V)+Preprocessed Title(Avg W2V)

In [146]:

```
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train,avg_w2v_vectors_train,avg_w2v_vectors_title_train)).t
ocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv,avg_w2v_vectors_cv,avg_w2v_vectors_title_cv)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test,avg_w2v_vectors_test,avg_w2v_vectors_title_test)).tocsr()
```

In [147]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

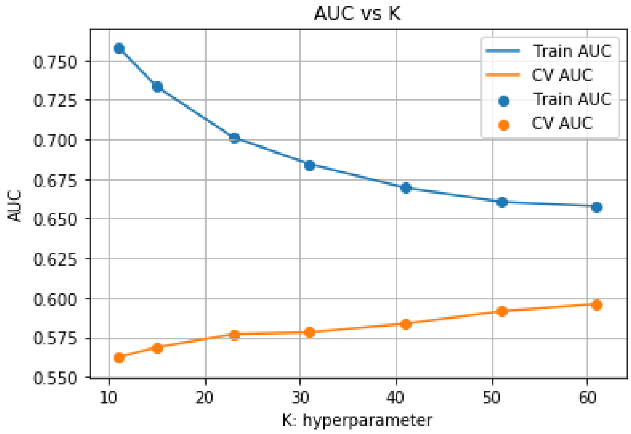
```
(15731, 703) (15731,)
==================================================
(10488, 703) (10488,)
==================================================
(17480, 703) (17480,)
```

In [148]:

```
#writting function for using batch-wise prediction
def batch_predict(neigh,data):
    '''Batch-Wise prediction is used to predict the class label in batches to fast process the knn
algorithm'''
    y_train_pred = []
    loop_value=data.shape[0]-data.shape[0]%1000
    for i in range(0, loop_value, 1000): #range will be from 0 to 49041 with step of 1000pts each t
ime so are values will be between 0-4000
        y_train_pred.extend(neigh.predict_proba(data[i:i+1000])[:,1])
    y_train_pred.extend(neigh.predict_proba(data[loop_value:])[:,1])

    return y_train_pred
```

In [149]:

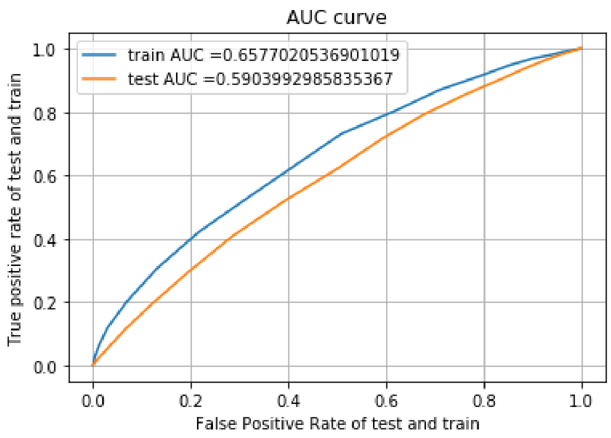
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
train_auc = []
cv_auc = []
K = [11, 15, 23,31, 41, 51,61]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(X_tr, y_train)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
tive class
    # not the predicted outputs
    y_train_pred =  batch_predict(neigh,X_tr)
    y_cv_pred =  batch_predict(neigh,X_cv)

    #print(y_train.shape)
    #print(len(y_train_pred))
     #roc_auc_score->Compute(ROC AUC) from prediction scores.
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))


plt.plot(K, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(K, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(K, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(K, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("K: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs K") #adding title of the plot
plt.grid()
plt.show()
```



Testing on Test Data(using our best hyper parameter=61)

In [150]:

```
from sklearn.metrics import roc_curve,auc

neigh=KNeighborsClassifier(n_neighbors= 61)
neigh.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=batch_predict(neigh,X_tr)
y_test_predict=batch_predict(neigh,X_te)
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation
of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation -
>https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("AUC curve")
plt.grid()
plt.show()
```



Confusion Matrix

In [151]:

```
def pred(proba,thresh, fpr ,tpr):
    """This function calculations and return the prediction with highest tpr and lowest tpr"""
    t=thresh[np.argmax(tpr*(1-fpr))] #t creates a numpy array with the max fpr and lowest tpr
    print("the maximum tpr*(1-fpr)",max(tpr*(1-fpr)),"for threshold",np.round(t,3))
    predictions=[]
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Train Data

In [152]:

```
from sklearn.metrics import confusion_matrix

print("Train confusion matrix")
cm=confusion_matrix(y_train, pred(y_train_predict, train_thresholds,train_fpr,train_tpr))
print(cm)
```

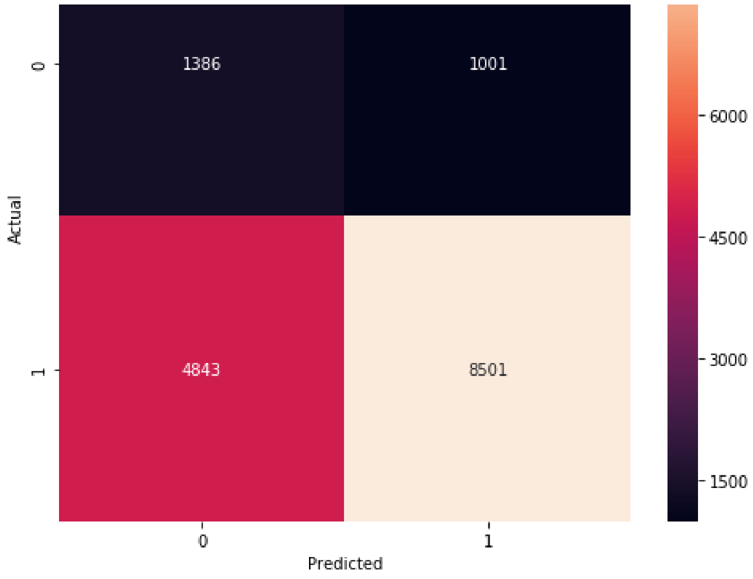
```
Train confusion matrix
the maximum tpr*(1-fpr) 0.36990891158041306 for threshold 0.852
[[1386 1001]
 [4843 8501]]
```

In [163]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[163]:

```
Text(0.5, 42.0, 'Predicted')
```



For the training data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=63%
2. Misclassification Rate= 1-Accuracy = 37% i.e. our model made 38% predictions wrong.
3. Sensitivity or Recall= 63% percentage of total number of positive correct predictions
4. Specificity= 58% percentage of total number of correct negative predictions.
5. Precision= 89% percentage of time when we predicted yes we are correct.

Test Data

In [154]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
cm1=confusion_matrix(y_test, pred(y_test_predict, test_thresholds,test_fpr,test_tpr))
print(cm1)
```

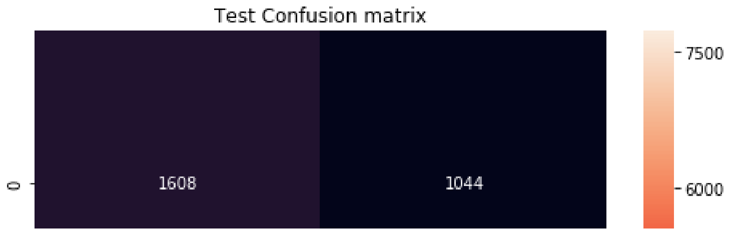
```
Test confusion matrix
the maximum tpr*(1-fpr) 0.3155165658220293 for threshold 0.869
[[1608 1044]
 [7112 7716]]
```

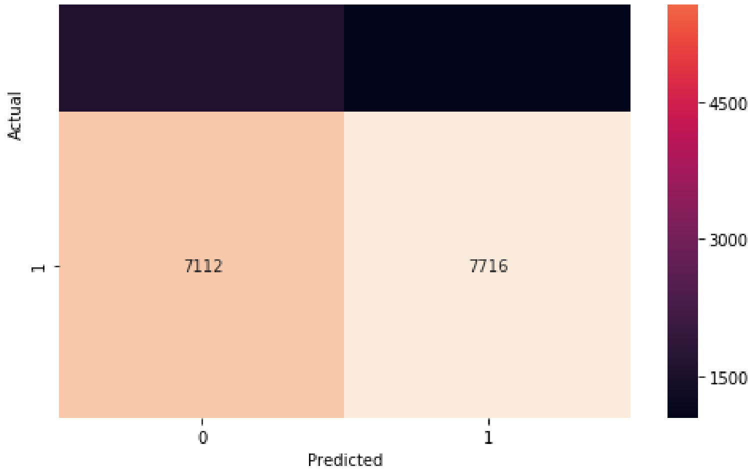
In [164]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[164]:

```
Text(0.5, 42.0, 'Predicted')
```





For the test data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=53%
2. Misclassification Rate= 1-Accuracy = 47% i.e. our model made 38% predictions wrong.
3. Sensitivity or Recall= 52% percentage of total number of positive correct predictions
4. Specificity= 60% percentage of total number of correct negative predictions.
5. Precision= 88% percentage of time when we predicted yes we are correct.

Set 4: Categorical Features,Numerical Features+Preprocessed Essay(TFIDF-W2V)+Preprocessed Title(TFIDF- W2V)

In [156]:

```
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train,tfidf_w2v_vectors_train,tfidf_w2v_vectors_title_train
)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv,tfidf_w2v_vectors_cv,tfidf_w2v_vectors_title_cv)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test,tfidf_w2v_vectors_test,tfidf_w2v_vectors_title_test)).tocsr()
```

In [157]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

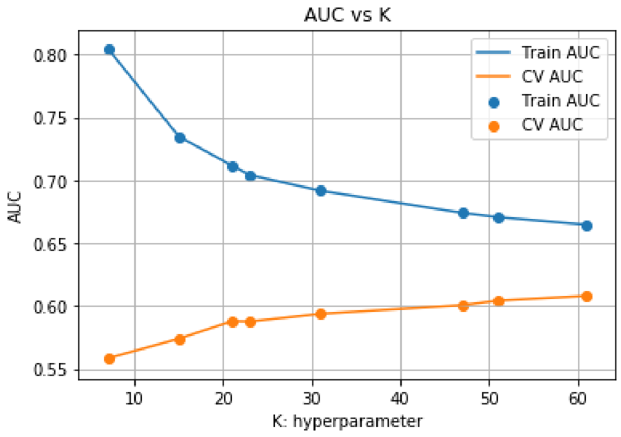
```
(15731, 703) (15731,)
==================================================
(10488, 703) (10488,)
==================================================
(17480, 703) (17480,)
```

In [158]:

```
#writting function for using batch-wise prediction
def batch_predict(neigh,data):
    '''Batch-Wise prediction is used to predict the class label in batches to fast process the knn
algorithm'''
    y_train_pred = []
    loop_value=data.shape[0]-data.shape[0]%1000
    for i in range(0, loop_value, 1000): #range will be from 0 to 49041 with step of 1000pts each t
ime so are values will be between 0-4000
        y_train_pred.extend(neigh.predict_proba(data[i:i+1000])[:,1])
    y_train_pred.extend(neigh.predict_proba(data[loop_value:])[:,1])
    return y_train_pred
```

In [159]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
train_auc = []
cv_auc = []
K = [7,15, 21,23,31, 47,51,61]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(X_tr, 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 =  batch_predict(neigh,X_tr)
    y_cv_pred =  batch_predict(neigh,X_cv)

    #print(y_train.shape)
    #print(len(y_train_pred))
     #roc_auc_score->Compute(ROC AUC) from prediction scores.
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from prediction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))


plt.plot(K, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(K, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(K, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(K, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("K: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs K") #adding title of the plot
plt.grid()
plt.show()
```

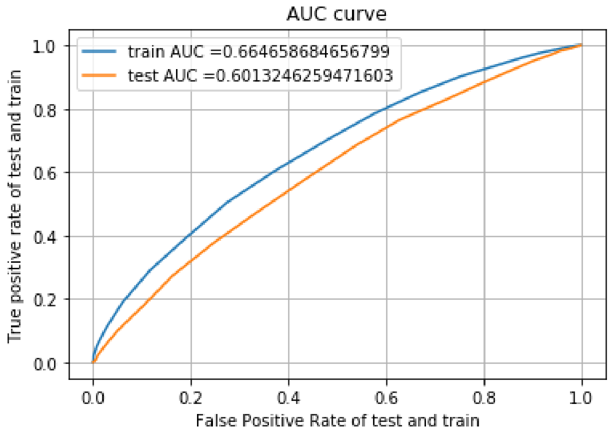


Testing

In [160]:

```
from sklearn.metrics import roc_curve,auc

neigh=KNeighborsClassifier(n_neighbors= 61)
neigh.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc curve returns three values fpr,tpr and thresholds
```

```
y_train_predict=batch_predict(neigh,X_tr)
y_test_predict=batch_predict(neigh,X_te)
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation
of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation -
>https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("AUC curve")
plt.grid()
plt.show()
```



Confusion Matrix

In [232]:

```
def pred(proba,thresh, fpr ,tpr):
    """This function calculations and return the prediction with highest tpr and lowest tpr"""
    t=thresh[np.argmax(tpr*(1-fpr))] #t creates a numpy array with the max fpr and lowest tpr
    print("the maximum tpr*(1-fpr)",max(tpr*(1-fpr)),"for threshold",np.round(t,3))
    predictions=[]
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Train Data

In [233]:

```
from sklearn.metrics import confusion_matrix

print("Train confusion matrix")
cm=confusion_matrix(y_train,pred(y_train_predict, train_thresholds,train_fpr,train_tpr))
print(cm)
```

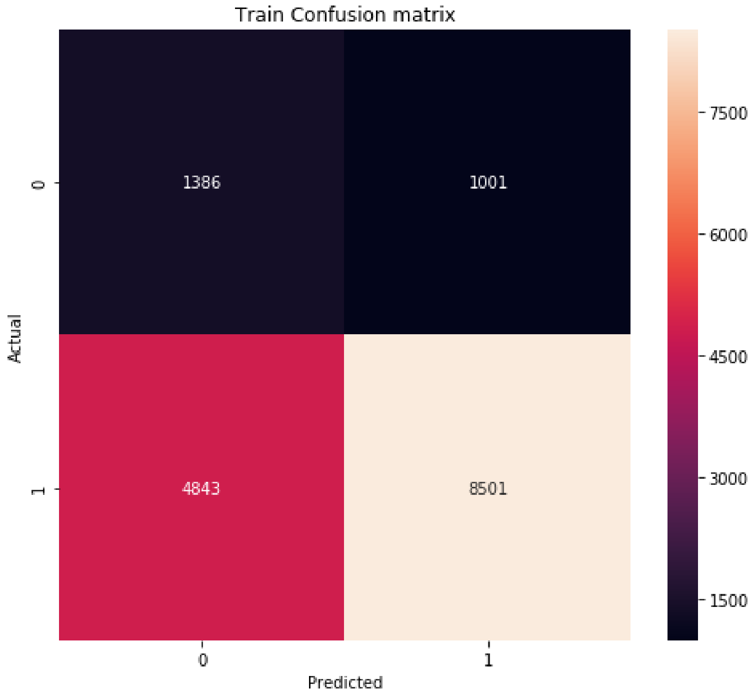
```
Train confusion matrix
the maximum tpr*(1-fpr) 0.3777064468141621 for threshold 0.852
[[1406  976]
 [4807 8542]]
```

In [165]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[165]:

```
Text(0.5, 42.0, 'Predicted')
```



For the training data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=63%
2. Misclassification Rate= 1-Accuracy = 38% i.e. our model made 38% predictions wrong.
3. Sensitivity or Recall= 63% percentage of total number of positive correct predictions
4. Specificity= 58% percentage of total number of correct negative predictions.
5. Precision= 89% percentage of time when we predicted yes we are correct.

Test Data

In [235]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
cm1=confusion_matrix(y_test, pred(y_test_predict, test_thresholds,test_fpr,test_tpr))
print(cm1)
```

```
Test confusion matrix
the maximum tpr*(1-fpr) 0.319727177794552 for threshold 0.852
[[1357 1289]
 [5586 9248]]
```

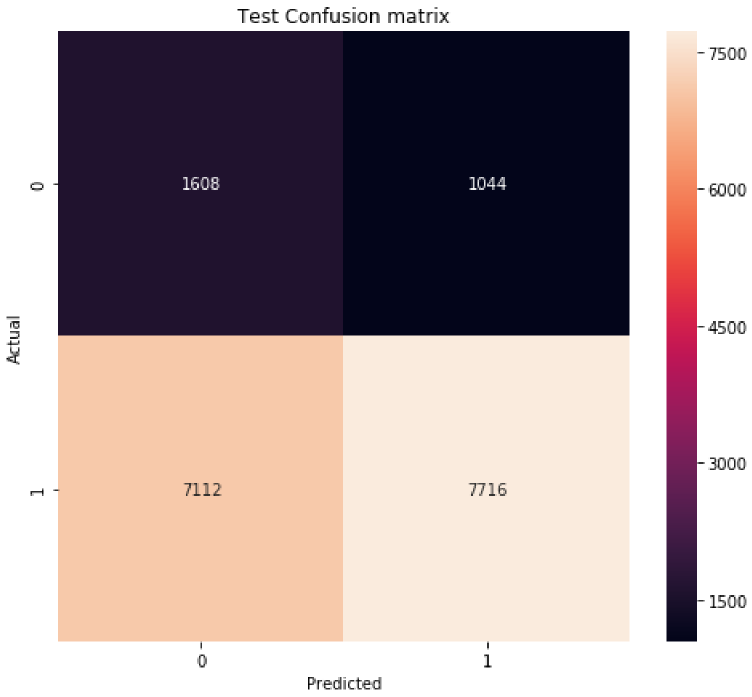
In [166]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
```

```
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[166]:

```
Text(0.5, 42.0, 'Predicted')
```



For the test data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=53%
2. Misclassification Rate= 1-Accuracy = 47% i.e. our model made 47% predictions wrong.
3. Sensitivity or Recall= 52% percentage of total number of positive correct predictions
4. Specificity= 60% percentage of total number of correct negative predictions.
5. Precision= 88% percentage of time when we predicted yes we are correct.

Feature Selection using 'SelectKBest'

For Set2

In [167]:

```
print(X_tr1.shape,y_train1.shape)
print("="*50)
print(X_cv1.shape,y_cv1.shape)
print("="*50)
print(X_te1.shape,y_test1.shape)
```

```
(35395, 12521) (35395,)
==================================================
(23598, 12521) (23598,)
==================================================
(39330, 12521) (39330,)
```

In [168]:

```
#SelectKbest documentation ->https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
#chi2 documentation -> https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html
#How to handle negative values SelectKBest->https://stackoverflow.com/questions/25792012/feature-selection-using-scikit-learn
```

```
election-using-scikit-learn

from sklearn.feature_selection import SelectKBest, chi2

bestk=SelectKBest(chi2, k=2000).fit(X_tr1,y_train1)

X_tr_best=bestk.transform(X_tr1)
X_te_best=bestk.transform(X_te1)
X_cv_best=bestk.transform(X_cv1)
```

In [169]:

```
#datamatrix with new features

print(X_tr_best.shape,y_train1.shape)
print("="*50)
print(X_cv_best.shape,y_cv1.shape)
print("="*50)
print(X_te_best.shape,y_test1.shape)
```

```
(35395, 2000) (35395,)
==================================================
(23598, 2000) (23598,)
==================================================
(39330, 2000) (39330,)
```

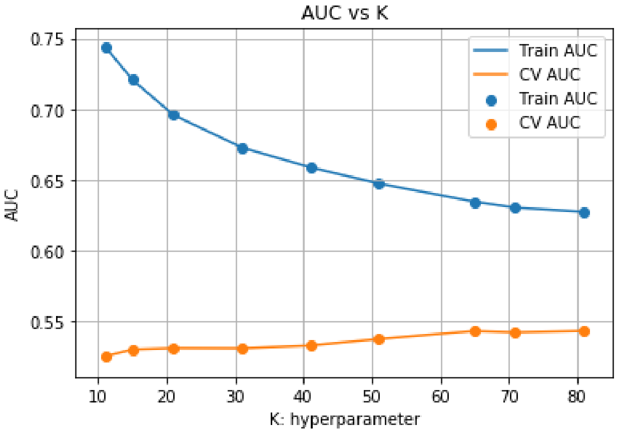
Finding the best hyperparameter

In [170]:

```
#writting function for using batch-wise prediction
def batch_predict(neigh,data):
    '''Batch-Wise prediction is used to predict the class label in batches to fast process the knn
algorithm'''
    y_train_pred = []
    loop_value=data.shape[0]-data.shape[0]%1000
    for i in range(0, loop_value, 1000): #range will be from 0 to 49041 with step of 1000pts each t
ime so are values will be between 0-4000
        y_train_pred.extend(neigh.predict_proba(data[i:i+1000])[:,1])
    y_train_pred.extend(neigh.predict_proba(data[loop_value:])[:,1])

    return y_train_pred
```

In [171]:

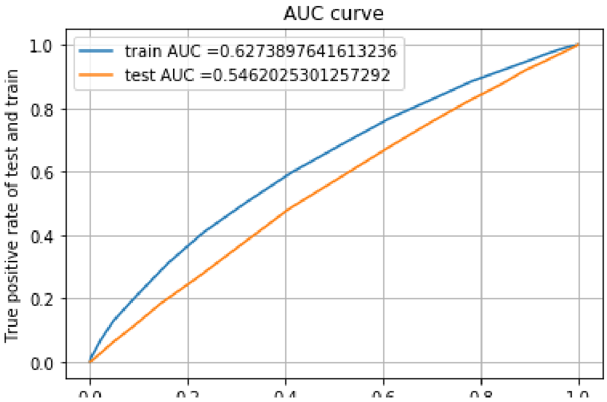
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
train_auc = []
cv_auc = []
K = [11,15, 21,31,41,51,65,71,81]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tr_best, y_train1)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  batch_predict(neigh,X_tr_best)
    y_cv_pred =  batch_predict(neigh,X_cv_best)

    #print(y_train.shape)
    #print(len(y_train_pred))
     #roc_auc_score->Compute(ROC AUC) from prediction scores.
    train_auc.append(roc_auc_score(y_train1,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pr
ediction scores.
    cv_auc.append(roc_auc_score(y_cv1, y_cv_pred))


plt.plot(K, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(K, train_auc, label='Train AUC') #Scatter plot of K vs auc train
plt.plot(K, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(K, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("K: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs K") #adding title of the plot
plt.grid()
plt.show()
```



Testing using best hyperparameter

In [172]:

```
from sklearn.metrics import roc_curve,auc

neigh=KNeighborsClassifier(n_neighbors= 81)
neigh.fit(X_tr_best,y_train1)

#documentation of roc_curve ->https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=batch_predict(neigh,X_tr_best)
y_test_predict=batch_predict(neigh,X_te_best)
train_fpr,train_tpr,train_thresholds= roc_curve(y_train1,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test1,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation
of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation -
>https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("AUC curve")
plt.grid()
plt.show()
```



Confusion matrix

In [173]:

```
def pred(proba,thresh, fpr ,tpr):
    """This function calculations and return the prediction with highest tpr and lowest tpr"""
    t=thresh[np.argmax(tpr*(1-fpr))] #t creates a numpy array with the max fpr and lowest tpr
    print("the maximum tpr*(1-fpr)",max(tpr*(1-fpr)),"for threshold",np.round(t,3))
    predictions=[]
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Train Data

In [174]:

```
from sklearn.metrics import confusion_matrix

print("Train confusion matrix")
cm=confusion_matrix(y_train1, pred(y_train_predict, train_thresholds,train_fpr,train_tpr))
print(cm)
```

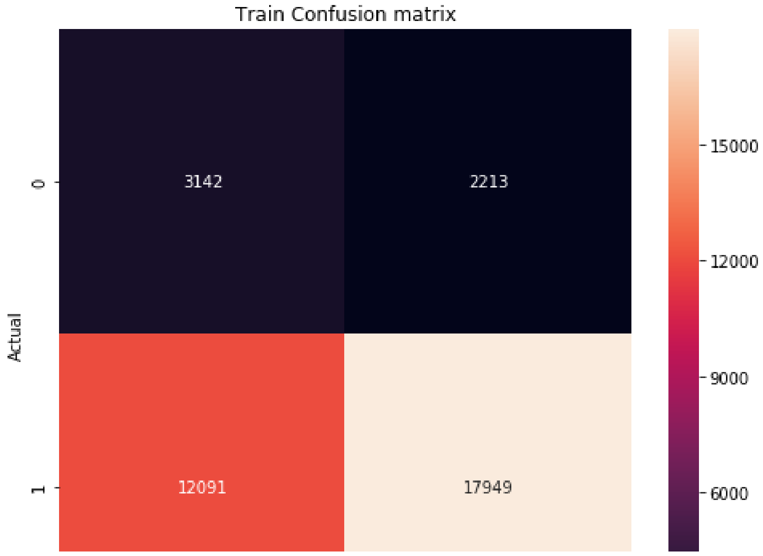
```
Train confusion matrix
the maximum tpr*(1-fpr) 0.35057991771941804 for threshold 0.852
[[ 3142  2213]
 [12091 17949]]
```

In [175]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[175]:

```
Text(0.5, 42.0, 'Predicted')
```



For the training data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=59%
2. Misclassification Rate= 1-Accuracy = 41% i.e. our model made 41% predictions wrong.
3. Sensitivity or Recall= 59.7% percentage of total number of positive correct predictions
4. Specificity= 58% percentage of total number of correct negative predictions.
5. Precision= 89% percentage of time when we predicted yes we are correct.

Test Data

In [176]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
cm1=confusion_matrix(y_test1, pred(y_test_predict, test_thresholds,test_fpr,test_tpr))
print(cm)
```

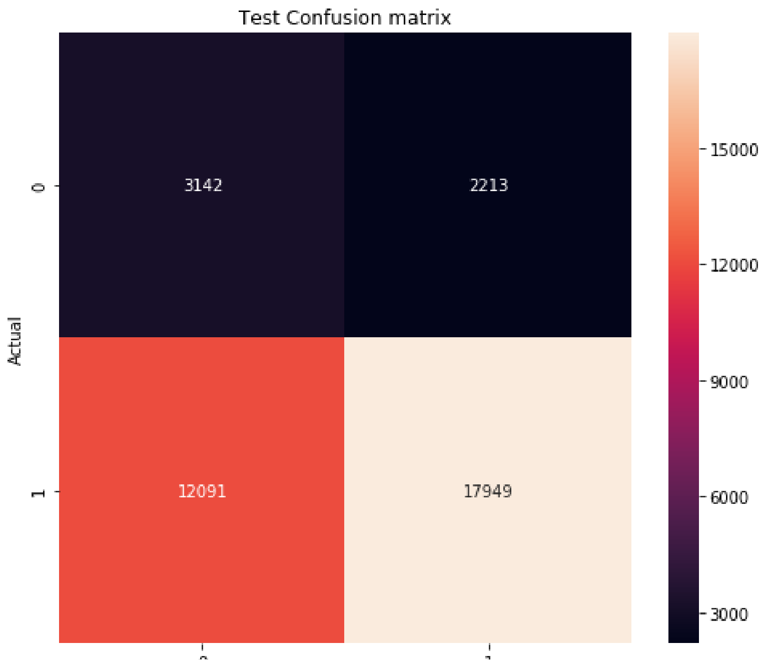
```
Test confusion matrix
the maximum tpr*(1-fpr) 0.2855377999390037 for threshold 0.864
[[ 3142  2213]
 [12091 17949]]
```

In [177]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm1=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[177]:

```
Text(0.5, 42.0, 'Predicted')
```



For the test data we calculate some metrics with the help of confusion matrix

1. Accuracy=Total number of correct predictions divided by total=59%
2. Misclassification Rate= 1-Accuracy = 41% i.e. our model made 41% predictions wrong.
3. Sensitivity or Recall= 59% percentage of total number of positive correct predictions
4. Specificity= 58% percentage of total number of correct negative predictions.
5. Precision= 89% percentage of time when we predicted yes we are correct.

Conclusion

In [178]:

```
#Refer->http://zetcode.com/python/prettytable/
#Refer->https://het.as.utexas.edu/HET/Software/Numpy/reference/generated/numpy.percentile.html
#Refer->https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.round_.html
from prettytable import PrettyTable
x=PrettyTable()

x.field_names=["Vectorizer", "Model", "Best Hyperparameter","Test AUC"] #column headers

x.add_row(["BOW", "Brute", 81 , 0.633])
x.add_row(["TFIDF", "Brute", 91, 0.550])
x.add_row(["AVG W2V", "Brute" ,61, 0.590])
x.add_row(["TFIDF W2V", "Brute", 61, 0.601])
x.add_row(["TFIDF", "TOP 2000", 81, 0.546])

print(x)
```

```
+------------+----------+---------------------+----------+
| Vectorizer |  Model   | Best Hyperparameter | Test AUC |
+------------+----------+---------------------+----------+
|    BOW     |  Brute   |          81         |  0.633   |
|   TFIDF    |  Brute   |          91         |   0.55   |
|  AVG W2V   |  Brute   |          61         |   0.59   |
| TFIDF W2V  |  Brute   |          61         |  0.601   |
|   TFIDF    | TOP 2000 |          81         |  0.546   |
+------------+----------+---------------------+----------+
```

References

https://github.com/harrismohammed/DonorsChoose.org-knn for some ideas

www.google.com for quick documentation

In [ ]: