DonorsChoose

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 competition 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.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	D
`project_id`	A unique identifier for the proposed project. **Example:**
`project_title`	Title of the project. • `Art Will Make Yo` • `First G`
`project_grade_category`	Grade level of students for which the project is targeted. One of th enumera • `Grade` • `Gr` • `Gr` • `Gra`

Feature	D
project_subject_categories	One or more (comma-separated) subject categories for the proje following enumerated lis • Applied • Care • Health • History • Literacy & • Math & • Music & • Speci • • • Music & • Literacy & Language, Math &
school_state	State where school is located (Two-letter U.S. (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos **Exa**
project_subject_subcategories	One or more (comma-separated) subject subcategories for • • Literature & Writing, Social
project_resource_summary	An explanation of the resources needed for the project. • My students need hands on literacy materials t sensory need
project_essay_1	First applica
project_essay_2	Second applica
project_essay_3	Third applica
project_essay_4	Fourth applica
project_submitted_datetime	Datetime when project application was submitted. **Example:** 201 12:4
teacher_id	A unique identifier for the teacher of the proposed project bdf8baa8fedef6bfeec7ae4f
teacher_prefix	Teacher's title. One of the following enumera • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the sa **E**

* 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

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_3:__ "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.

In [2]:
```
%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
```

1.1 Reading Data

In [3]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [4]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_
state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [5]:
```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/40840
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[5]:

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

In [6]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[6]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [7]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a

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

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

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of `project_subject_subcategories`

In [8]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a

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

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

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 Text preprocessing

In [9]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [10]:
```
project_data.head(2)
```

Out[10]:

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

In [11]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [12]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom as we
ll as the STEM journals, which my students really enjoyed.  I would love to implem
ent more of the Lakeshore STEM kits in my classroom for the next school year as th
ey provide excellent and engaging STEM lessons.My students come from a variety of
backgrounds, including language and socioeconomic status.  Many of them don't have
a lot of experience in science and engineering and these kits give me the material
s to provide these exciting opportunities for my students.Each month I try to do s
everal science or STEM/STEAM projects.  I would use the kits and robot to help gui
de my science instruction in engaging and meaningful ways.  I can adapt the kits t
o my current language arts pacing guide where we already teach some of the materia
l in the kits like tall tales (Paul Bunyan) or Johnny Appleseed.  The following un
its will be taught in the next school year where I will implement these kits: magn
ets, motion, sink vs. float, robots.  I often get to these units and don't know If
I am teaching the right way or using the right materials.    The kits will give me
additional ideas, strategies, and lessons to prepare my students in science.It is
challenging to develop high quality science activities.  These kits give me the ma
terials I need to provide my students with science activities that will go along w
ith the curriculum in my classroom.  Although I have some things (like magnets) in
my classroom, I don't know how to use them effectively.  The kits will provide me
with the right amount of materials and show me how to use them in an appropriate w
ay.
==================================================
I teach high school English to students with learning and behavioral disabilities.
My students all vary in their ability level. However, the ultimate goal is to incr
ease all students literacy levels. This includes their reading, writing, and commu
nication levels.I teach a really dynamic group of students. However, my students f
ace a lot of challenges. My students all live in poverty and in a dangerous neighb
orhood. Despite these challenges, I have students who have the the desire to defea
t these challenges. My students all have learning disabilities and currently all a
re performing below grade level. My students are visual learners and will benefit
from a classroom that fulfills their preferred learning style.The materials I am r
equesting will allow my students to be prepared for the classroom with the necessa
ry supplies.  Too often I am challenged with students who come to school unprepare
d for class due to economic challenges.  I want my students to be able to focus on
learning and not how they will be able to get school supplies.  The supplies will
last all year.  Students will be able to complete written assignments and maintain
a classroom journal.  The chart paper will be used to make learning more visual in
class and to create posters to aid students in their learning.  The students have
access to a classroom printer.  The toner will be used to print student work that
is completed on the classroom Chromebooks.I want to try and remove all barriers fo
r the students learning and create opportunities for learning. One of the biggest
barriers is the students not having the resources to get pens, paper, and folders.
My students will be able to increase their literacy skills because of this projec
t.
==================================================
\"Life moves pretty fast. If you don't stop and look around once in awhile, you co
uld miss it.\"  from the movie, Ferris Bueller's Day Off.  Think back...what do yo
```

u remember about your grandparents? How amazing would it be to be able to flip th
rough a book to see a day in their lives?My second graders are voracious readers!
They love to read both fiction and nonfiction books. Their favorite characters in
clude Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson. They also love
to read about insects, space and plants. My students are hungry bookworms! My stud
ents are eager to learn and read about the world around them. My kids love to be a
t school and are like little sponges absorbing everything around them. Their paren
ts work long hours and usually do not see their children. My students are usually
cared for by their grandparents or a family friend. Most of my students do not hav
e someone who speaks English at home. Thus it is difficult for my students to acqu
ire language.Now think forward... wouldn't it mean a lot to your kids, nieces or n
ephews or grandchildren, to be able to see a day in your life today 30 years from
now? Memories are so precious to us and being able to share these memories with fu
ture generations will be a rewarding experience. As part of our social studies cu
rriculum, students will be learning about changes over time. Students will be stu
dying photos to learn about how their community has changed over time. In particu
lar, we will look at photos to study how the land, buildings, clothing, and school
s have changed over time. As a culminating activity, my students will capture a s
lice of their history and preserve it through scrap booking. Key important events
in their young lives will be documented with the date, location, and names. Stud
ents will be using photos from home and from school to create their second grade m
emories. Their scrap books will preserve their unique stories for future generat
ions to enjoy.Your donation to this project will provide my second graders with an
opportunity to learn about social studies in a fun and creative manner. Through t
heir scrapbooks, children will share their story with others and have a historical
document for the rest of their lives.
==========
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest stud
ents with the biggest enthusiasm for learning. My students learn in many different
ways using all of our senses and multiple intelligences. I use a wide range of tec
hniques to help all my students succeed. \r\nStudents in my class come from a vari
ety of different backgrounds which makes for wonderful sharing of experiences and
cultures, including Native Americans.\r\nOur school is a caring community of succe
ssful learners which can be seen through collaborative student project based learn
ing in and out of the classroom. Kindergarteners in my class love to work with han
ds-on materials and have many different opportunities to practice a skill before i
t is mastered. Having the social skills to work cooperatively with friends is a cr
ucial 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 kitc
hen 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 delicio
us healthy food for snack time. My students will have a grounded appreciation for
the work that went into making the food and knowledge of where the ingredients cam
e from as well as how it's healthy for their bodies. This project would 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 f
rom our classroom garden in the spring. We will also create our own cookbooks to b
e printed and shared with families. \r\nStudents will gain math and literature ski
lls as well as a life long enjoyment for healthy cooking.nannan
==========
My classroom consists of twenty-two amazing sixth graders from different cultures
and backgrounds. They are a social bunch who enjoy working in partners and working
with groups. They are hard-working and eager to head to middle school next year. M
y job is to get them ready to make this transition and make it as smooth as possib
le. In order to do this, my students need to come to school every day and feel saf
e and ready to learn. Because they are getting ready to head to middle school, I g
ive them lots of choice- choice on where to sit and work, the order to complete as
signments, choice of projects, etc. Part of the students feeling safe is the abili

```
ty for them to come into a welcoming, encouraging environment. My room is colorful
and the atmosphere is casual. I want them to take ownership of the classroom becau
se we ALL share it together. Because my time with them is limited, I want to ensur
e they get the most of this time and enjoy it to the best of their abilities.Curre
ntly, we have twenty-two desks of differing sizes, yet the desks are similar to th
e ones the students will use in middle school. We also have a kidney table with cr
ates for seating. I allow my students to choose their own spots while they are wor
king independently or in groups. More often than not, most of them move out of the
ir desks and onto the crates. Believe it or not, this has proven to be more succes
sful than making them stay at their desks! It is because of this that I am looking
toward the “Flexible Seating” option for my classroom.\r\n The students look forwa
rd to their work time so they can move around the room. I would like to get rid of
the constricting desks and move toward more “fun” seating options. I am requesting
various seating so my students have more options to sit. Currently, I have a stool
and a papasan chair I inherited from the previous sixth-grade teacher as well as f
ive milk crate seats I made, but I would like to give them more options and reduce
the competition for the “good seats”. I am also requesting two rugs as not only mo
re seating options but to make the classroom more welcoming and appealing. In orde
r for my students to be able to write and complete work without desks, I am reques
ting a class set of clipboards. Finally, due to curriculum that requires groups to
work together, I am requesting tables that we can fold up when we are not using th
em to leave more room for our flexible seating options.\r\nI know that with more s
eating options, they will be that much more excited about coming to school! Thank
you for your support in making my classroom one students will remember forever!nan
nan
==================================================
```

In [13]:

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

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [14]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smallest stu
dents with the biggest enthusiasm for learning. My students learn in many differen
t ways using all of our senses and multiple intelligences. I use a wide range of t
echniques to help all my students succeed. \r\nStudents in my class come from a va
riety of different backgrounds which makes for wonderful sharing of experiences an
d cultures, including Native Americans.\r\nOur school is a caring community of suc
cessful learners which can be seen through collaborative student project based lea
rning in and out of the classroom. Kindergarteners in my class love to work with h
ands-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 c
rucial 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 kitc
hen 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 delicio
us healthy food for snack time. My students will have a grounded appreciation for
the work that went into making the food and knowledge of where the ingredients cam
e from as well as how it is healthy for their bodies. This project would expand ou
r 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 classroom garden in the spring. We will also create our own cookbooks to
be printed and shared with families. \r\nStudents will gain math and literature sk
ills as well as a life long enjoyment for healthy cooking.nannan
==================================================
```

In [15]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smallest stude
nts with the biggest enthusiasm for learning. My students learn in many different
ways using all of our senses and multiple intelligences. I use a wide range of tec
hniques to help all my students succeed.  Students in my class come from a variet
y of different backgrounds which makes for wonderful sharing of experiences and cu
ltures, including Native Americans.  Our school is a caring community of successfu
l 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-o
n 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  w
here 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 th
at went into making the food and knowledge of where the ingredients came from as w
ell 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 c
lassroom 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 [16]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
 A person is a person no matter how small Dr Seuss I teach the smallest students w
ith the biggest enthusiasm for learning My students learn in many different ways u
sing 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 differ
ent backgrounds which makes for wonderful sharing of experiences and cultures incl
uding Native Americans Our school is a caring community of successful learners whi
ch 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 ki
ndergarten 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 childh
ood classroom I have had several kids ask me Can we try cooking with REAL food I w
ill take their idea and create Common Core Cooking Lessons where we learn importan
t 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 knowledge 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 agricultu
ral cooking recipes by having us peel our own apples to make homemade applesauce m
ake our own bread and mix up healthy plants from our classroom garden in the sprin
g We will also create our own cookbooks to be printed and shared with families Stu
dents will gain math and literature skills as well as a life long enjoyment for he
althy cooking nannan
```

In [17]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'throu
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 't
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've",
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn'
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'm
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
            'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 10
9248/109248 [02:25<00:00, 751.24it/s]
```

In [19]:
```
# after preprocesing

preprocessed_essays[20000]
preprocessed_essays = pd.DataFrame({'preprocessed_essays': preprocessed_essays})
```

In [20]:
```
#project_data.drop(['preprocessed_essays'], axis=1, inplace=True)

project_data = pd.concat([project_data, preprocessed_essays], axis=1)
project_data.head(1)
```

Out[20]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grade

1.4 Preprocessing of `project_title`

In [21]:
```
# similarly you can preprocess the titles also


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

```
100%|██████████████████████████████████████████████████████████████████████████████| 1092
48/109248 [00:05<00:00, 20652.14it/s]
```

In [22]:
```
preprocessed_titles = pd.DataFrame({'preprocessed_titles': preprocessed_titles})


project_data = pd.concat([project_data, preprocessed_titles], axis=1)
project_data.head(1)
```

Out[22]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grade

1.5 Preparing data for models

In [23]:
```
project_data.columns
```

Out[23]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', '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',
       'clean_categories', 'clean_subcategories', 'essay',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

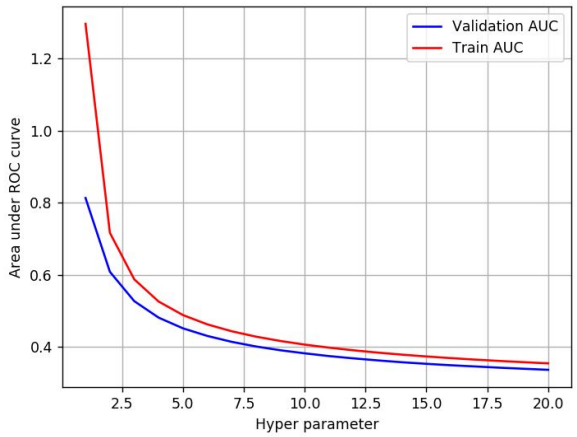
1.5.1 Merge with resource data

In [24]:

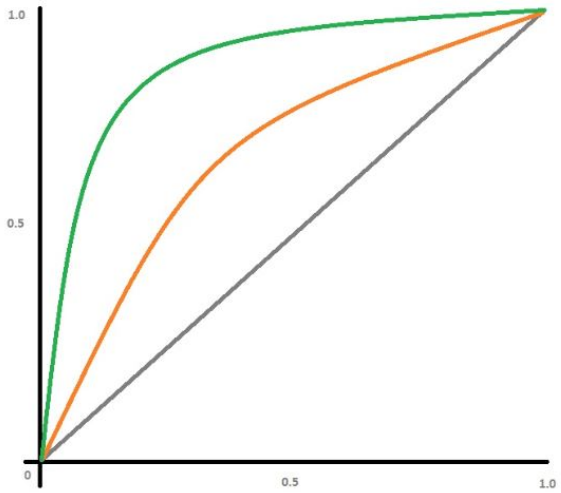
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).rese
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

Assignment 3: Apply KNN

1. **[Task-1] Apply KNN(brute force version) on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)
   - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)
   - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)
2. **Hyper paramter tuning to find best K**
   - Find the best hyper parameter which results in the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
   - Find the best hyper paramter using k-fold cross validation (or) simple cross validation data
   - Use gridsearch-cv or randomsearch-cv or write your own for loops to do this task
3. **Representation of results**
   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure



- Once you find the best hyper parameter, you need to train your model-M using the best hyper-param. Now, find the AUC on test data and plot the ROC curve on both train and test using model-M.



- Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

4. **[Task-2]**

- Select top 2000 features from feature Set 2 using `SelectKBest` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html) and then apply KNN on top of these features

-
```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2

X, y = load_digits(return_X_y=True)
X.shape
X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
X_new.shape
========
output:
(1797, 64)
(1797, 20)
```

- Repeat the steps 2 and 3 on the data matrix after feature selection

5. **Conclusion**

- You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link (http://zetcode.com/python/prettytable/)

Vectorizer	Model	Hyper parameter	AUC
BOW	Brute	7	0.78
TFIDF	Brute	12	0.79
W2V	Brute	10	0.78
TFIDFW2V	Brute	6	0.78

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. K Nearest Neighbor

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

In [25]:
```
#Stratify vs ramdom sampling. oversampling for imbalanced data
#https://stats.stackexchange.com/questions/250273/benefits-of-stratified-vs-random-s




from sklearn.model_selection import train_test_split

# train = project_data.drop(['project_is_approved'], axis=1, inplace=True)  # this w

X_train, X_test, y_train, y_test = train_test_split(project_data, project_data['proj
                                                    test_size=0.3, stratify = projec

X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.3, str


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

In [26]:
```
print(X_test.shape)
print(y_test.shape)
print(X_cv.shape)
print(y_cv.shape)
```

```
(32775, 21)
(32775,)
(22942, 21)
(22942,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [27]:

```
# Encoding of Categorical Features:

# Category:
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=Fals
categories_one_hot_train = vectorizer.fit_transform(X_train['clean_categories'].valu
categories_one_hot_cv = vectorizer.transform(X_cv['clean_categories'].values)
categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].values)

print(vectorizer.get_feature_names())
print("category Shape of matrix after one hot encodig ",categories_one_hot_train.sha



# Subcategory
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=
sub_categories_one_hot_train = vectorizer.fit_transform(X_train['clean_subcategories
sub_categories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].val


print(vectorizer.get_feature_names())
print("subctg Shape of matrix after one hot encodig ",sub_categories_one_hot_train.s


#you can do the similar thing with state, teacher_prefix and project_grade_category

vectorizer = CountVectorizer(lowercase=False, binary=True)
state_one_hot_train = vectorizer.fit_transform(X_train['school_state'].values)
state_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)
state_one_hot_test = vectorizer.transform(X_test['school_state'].values)


print("state Shape of matrix after one hot encodig ",state_one_hot_train.shape)



vectorizer = CountVectorizer(lowercase=False, binary=True)


tp_one_hot_train = vectorizer.fit_transform(X_train['teacher_prefix'].apply(lambda x
tp_one_hot_cv = vectorizer.transform(X_cv['teacher_prefix'].apply(lambda x: np.str_(
tp_one_hot_test = vectorizer.transform(X_test['teacher_prefix'].apply(lambda x: np.s


print("tp Shape of matrix after one hot encodig ",tp_one_hot_train.shape)




# Project Grade List
from collections import Counter
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.splitlines())



grade_list = dict(my_counter)
```

```
print(grade_list)


# If not generating the above list and put into vocabulary, the vector will some mes
# This is because of space and new lines. Otherwise no need for vocabulary

vectorizer = CountVectorizer(vocabulary=list(grade_list.keys()),lowercase=False, bir



pg_one_hot_train = vectorizer.fit_transform(X_train['project_grade_category'].values
pg_one_hot_cv = vectorizer.transform(X_cv['project_grade_category'].values)
pg_one_hot_test = vectorizer.transform(X_test['project_grade_category'].values)


print("pg Shape of matrix after one hot encodig ",pg_one_hot_train.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'Spec
ialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
category Shape of matrix after one hot encodig  (53531, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extra
curricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmt
h', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'Team
Sports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_Life
Science', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'Visua
lArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writin
g', 'Mathematics', 'Literacy']
subctg Shape of matrix after one hot encodig  (53531, 30)
state Shape of matrix after one hot encodig  (53531, 51)
tp Shape of matrix after one hot encodig  (53531, 6)
{'Grades PreK-2': 44225, 'Grades 6-8': 16923, 'Grades 3-5': 37137, 'Grades 9-12':
10963}
pg Shape of matrix after one hot encodig  (53531, 4)
```

In [28]:
```
print(pg_one_hot_test.shape)
print(pg_one_hot_train.shape)

print(state_one_hot_test.shape)
print(state_one_hot_train.shape)


print(tp_one_hot_test.shape)
print(tp_one_hot_train.shape)
```

```
(32775, 4)
(53531, 4)
(32775, 51)
(53531, 51)
(32775, 6)
(53531, 6)
```

In [29]:
```
# Numerical Data


from sklearn.preprocessing import Normalizer

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.    .
# Reshape your data either using array.reshape(-1, 1)


#instead of standardize, try normalization since chi2 requires non-negative


price_scalar = Normalizer()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and stand
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.

#Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(X_train['price'].values.reshape(-1

price_standardized_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_standardized_test = price_scalar.transform(X_test['price'].values.reshape(-1,


print(price_standardized_train.mean())
print(price_standardized_train.std())

print(price_standardized_train[100])
```

```
1.0
0.0
[1.]
```

In [30]:

```
previous_scalar = Normalizer()
previous_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.r

# finding the mean and standard deviation of this data
#print(f"Mean : {previous_scalar.mean_[0]}, Standard deviation : {np.sqrt(previous_s

# Now standardize the data with above maen and variance.
previous_standardized_train = previous_scalar.transform(X_train['teacher_number_of_p



previous_standardized_cv = previous_scalar.transform(X_cv['teacher_number_of_previou


previous_standardized_test = previous_scalar.transform(X_test['teacher_number_of_pre


print(previous_standardized_train.mean())
print(previous_standardized_train.std())

print(previous_standardized_train[100])
```

```
0.7257850591246193
0.44611781748333273
[1.]
```

In [31]:

```
print(price_standardized_test.shape)
print(price_standardized_train.shape)


print(previous_standardized_test.shape)
print(previous_standardized_train.shape)
```

```
(32775, 1)
(53531, 1)
(32775, 1)
(53531, 1)
```

2.3 Make Data Model Ready: encoding essay, and project_title

1.5.2.1 Bag of words

In [53]:
```
# We are considering only the words which appeared in at least 10 documents(rows or
vectorizer = CountVectorizer(min_df=10,max_features = 5000)
text_train_bow = vectorizer.fit_transform(X_train['preprocessed_essays'].values)


# should fit_transferm only on train data . Transform on test data
text_cv_bow = vectorizer.transform(X_cv['preprocessed_essays'].values)
text_test_bow = vectorizer.transform(X_test['preprocessed_essays'].values)

print("Shape of matrix after one hot encodig ",text_train_bow.shape)
print("Shape of matrix after one hot encodig ",text_test_bow.shape)
print("Shape of matrix after one hot encodig ",text_cv_bow.shape)
print(text_train_bow[1])
```

```
Shape of matrix after one hot encodig  (53531, 5000)
Shape of matrix after one hot encodig  (32775, 5000)
Shape of matrix after one hot encodig  (22942, 5000)
  (0, 4128)     1
  (0, 2563)     1
  (0, 3747)     1
  (0, 4088)     1
  (0, 4954)     1
  (0, 3666)     1
  (0, 308)      1
  (0, 4545)     1
  (0, 2410)     1
  (0, 253)      1
  (0, 651)      1
  (0, 2570)     1
  (0, 4205)     1
  (0, 1638)     1
  (0, 3656)     1
  (0, 751)      1
  (0, 1097)     1
  (0, 1705)     1
  (0, 2731)     1
  (0, 2021)     1
  (0, 232)      1
  (0, 3792)     1
  (0, 2817)     1
  (0, 307)      1
  (0, 4298)     5
  :     :
  (0, 2738)     2
  (0, 2398)     1
  (0, 2438)     1
  (0, 2154)     1
  (0, 4218)     1
  (0, 3413)     1
  (0, 1343)     1
  (0, 4589)     1
  (0, 330)      1
  (0, 2711)     1
  (0, 1551)     1
  (0, 585)      1
  (0, 93)       1
  (0, 2895)     1
  (0, 3004)     1
  (0, 4757)     1
```

```
  (0, 4120)     1
  (0, 2628)     4
  (0, 2771)     2
  (0, 4938)     1
  (0, 2397)     1
  (0, 2624)     2
  (0, 3954)     3
  (0, 249)      1
  (0, 4367)     6
```

In [54]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it


vectorizer = CountVectorizer(min_df=10,max_features = 5000)
title_train_bow = vectorizer.fit_transform(X_train['preprocessed_titles'].values)
title_cv_bow = vectorizer.transform(X_cv['preprocessed_titles'].values)
title_test_bow = vectorizer.transform(X_test['preprocessed_titles'].values)

print("Shape of matrix after one hot encodig ",title_train_bow.shape)
print("Shape of matrix after one hot encodig ",title_cv_bow.shape)
print("Shape of matrix after one hot encodig ",title_test_bow.shape)
```

```
Shape of matrix after one hot encodig  (53531, 2227)
Shape of matrix after one hot encodig  (22942, 2227)
Shape of matrix after one hot encodig  (32775, 2227)
```

1.5.2.2 TFIDF vectorizer

In [55]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10, max_features = 5000)
text_train_tfidf = vectorizer.fit_transform(X_train['preprocessed_essays'].values)
text_cv_tfidf = vectorizer.transform(X_cv['preprocessed_essays'].values)
text_test_tfidf = vectorizer.transform(X_test['preprocessed_essays'].values)

print("Shape of matrix after one hot encodig ",text_train_tfidf.shape)
print("Shape of matrix after one hot encodig ",text_cv_tfidf.shape)
print("Shape of matrix after one hot encodig ",text_test_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (53531, 5000)
Shape of matrix after one hot encodig  (22942, 5000)
Shape of matrix after one hot encodig  (32775, 5000)
```

In [56]:
```
# Similarly you can vectorize for title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
title_train_tfidf = vectorizer.fit_transform(X_train['preprocessed_titles'].values)
title_cv_tfidf = vectorizer.transform(X_cv['preprocessed_titles'].values)
title_test_tfidf = vectorizer.transform(X_test['preprocessed_titles'].values)

print("Shape of matrix after one hot encodig ",title_train_tfidf.shape)
print("Shape of matrix after one hot encodig ",title_cv_tfidf.shape)
print("Shape of matrix after one hot encodig ",title_test_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (53531, 2227)
Shape of matrix after one hot encodig  (22942, 2227)
Shape of matrix after one hot encodig  (32775, 2227)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [123]:
```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors

# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
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.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')



# Word2Vec does not provide good result since only vectorize by letter, not words
'''

#this line of code trains your w2v model on the give list of sentances
w2v_model=Word2Vec(X_train['preprocessed_essays'].values, min_count=5,size=50, worke


glove_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(glove_words))
print("sample words ", glove_words)

'''
```

```
Loading Glove Model

1917494it [12:06, 2639.45it/s]

Done. 1917494  words loaded!
```

Out[123]:
```
'\n\n#this line of code trains your w2v model on the give list of sentances\nw2v_m
odel=Word2Vec(X_train[\'preprocessed_essays\'].values, min_count=5,size=50, worker
s=4)\n\n\nglove_words = list(w2v_model.wv.vocab)\nprint("number of words that occu
red minimum 5 times ",len(glove_words))\nprint("sample words ", glove_words)\n\n'
```

In [153]:
```
with open('glove_vectors', 'rb') as f:
    w2v_model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [154]:
```
print(X_train['preprocessed_titles'].values[1])
```

```
group work made easy
```

In [157]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(X_train['preprocessed_essays'].values): # for each review/sente
    vector = np.zeros(300) # as word vectors are of zero length, if word2vec then us
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
              #vector += w2v_model.wv[word] # this is for w2v_model from Word2Vec fu
            vector += w2v_model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 5
3531/53531 [00:38<00:00, 1381.77it/s]

53531
```

In [162]:
```
#this line of code trains your w2v model on the give list of sentances
'''

w2v_model=Word2Vec(X_cv['preprocessed_essays'].values,min_count=5,size=50, workers=4


glove_words = list(w2v_model.wv.vocab)


print("number of words that occured minimum 5 times ",len(glove_words))
print("sample words ", glove_vector[0:50])
'''
```

Out[162]:
```
'\n\nw2v_model=Word2Vec(X_cv[\'preprocessed_essays\'].values,min_count=5,size=50,
workers=4)\n\n\nglove_words = list(w2v_model.wv.vocab)\n\n\nprint("number of words
that occured minimum 5 times ",len(glove_words))\nprint("sample words ", glove_vec
tor[0:50])\n'
```

In [164]:
```
avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this li
for sentence in tqdm(X_cv['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            #vector += w2v_model.wv[word] # this is for w2v_model from Word2Vec func
            vector += w2v_model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 2
2942/22942 [00:14<00:00, 1593.62it/s]

22942
```

In [165]:
```
#w2v_model=Word2Vec(X_test['preprocessed_essays'].values,min_count=5,size=50, worker


#glove_words = list(w2v_model.wv.vocab)
#print("number of words that occured minimum 5 times ",len(glove_words))
#print("sample words ", glove_words[0:50])
```

In [166]:
```
avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(X_test['preprocessed_essays'].values): # for each review/senter
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            #vector += w2v_model.wv[word] # this is for w2v_model from Word2Vec fur
            vector += w2v_model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 3
2775/32775 [00:19<00:00, 1704.76it/s]

32775
```

In [167]:
```
# Similarly you can vectorize for title also

#w2v_model=Word2Vec(X_train['preprocessed_titles'].values,min_count=5,size=50, worke


#glove_words = list(w2v_model.wv.vocab)
#print("number of words that occured minimum 5 times ",len(glove_words))
#print("sample words ", glove_words[0:50])

avg_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_train['preprocessed_titles'].values): # for each review/sente
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
             #vector += w2v_model.wv[word] # this is for w2v_model from Word2Vec fur
            vector += w2v_model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_train.append(vector)

print(len(avg_w2v_vectors_titles_train))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 53
531/53531 [00:02<00:00, 22827.47it/s]

53531
```

In [168]:
```
# Similarly you can vectorize for title also

#w2v_model=Word2Vec(X_cv['preprocessed_titles'],min_count=5,size=50, workers=4)


#glove_words = list(w2v_model.wv.vocab)
#print("number of words that occured minimum 5 times ",len(glove_words))
#print("sample words ", glove_words[0:50])

avg_w2v_vectors_titles_cv = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(X_cv['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
             #vector += w2v_model.wv[word] # this is for w2v_model from Word2Vec fur
            vector += w2v_model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_cv.append(vector)

print(len(avg_w2v_vectors_titles_cv))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 22
942/22942 [00:00<00:00, 23934.43it/s]

22942
```

In [169]:
```
# Similarly you can vectorize for title also

#w2v_model=Word2Vec(X_test['preprocessed_titles'].values,min_count=5,size=50, worker


#glove_words = list(w2v_model.wv.vocab)
#print("number of words that occured minimum 5 times ",len(glove_words))
#print("sample words ", glove_words[0:50])

avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored i
for sentence in tqdm(X_test['preprocessed_titles'].values): # for each review/senter
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
             #vector += w2v_model.wv[word] # this is for w2v_model from Word2Vec fur
            vector += w2v_model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_test.append(vector)

print(len(avg_w2v_vectors_titles_test))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 32
775/32775 [00:01<00:00, 22302.43it/s]

32775
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [170]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
#w2v_model=Word2Vec(X_train['preprocessed_essays'].values,min_count=5,size=50, worke
#glove_words = list(w2v_model.wv.vocab)


tfidf_model_train= TfidfVectorizer()
tfidf_model_train.fit(X_train['preprocessed_essays'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_train.get_feature_names(), list(tfidf_model_train.
tfidf_words_train = set(tfidf_model_train.get_feature_names())
```

In [171]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in th
for sentence in tqdm(X_train['preprocessed_essays'].values): # for each review/sente
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words_train):
            vec = w2v_model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            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))
```

```
100%|██████████████████████████████████████████████████████████████████████████████|
53531/53531 [04:07<00:00, 216.03it/s]

53531
```

In [67]:
```
#w2v_model=Word2Vec(X_cv['preprocessed_essays'].values,min_count=5,size=50, workers=
#glove_words = list(w2v_model.wv.vocab)


tfidf_model_cv= TfidfVectorizer()
tfidf_model_cv.fit(X_cv['preprocessed_essays'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_cv.get_feature_names(), list(tfidf_model_cv.idf_)))
tfidf_words_cv = set(tfidf_model_cv.get_feature_names())



tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this

for sentence in tqdm(X_cv['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words_cv):
            vec = w2v_model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            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))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 2
2942/22942 [00:04<00:00, 4992.21it/s]

22942
```

In [172]:
```
#w2v_model=Word2Vec(X_test['preprocessed_essays'].values,min_count=5,size=50, worker
#glove_words = list(w2v_model.wv.vocab)


tfidf_model_test= TfidfVectorizer()
tfidf_model_test.fit(X_test['preprocessed_essays'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_test.get_feature_names(), list(tfidf_model_test.id
tfidf_words_test = set(tfidf_model_test.get_feature_names())


tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in thi


for sentence in tqdm(X_test['preprocessed_essays'].values): # for each review/senter
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words_test):
            vec = w2v_model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            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))
```

```
100%|██████████████████████████████████████████████████████████████████████████████████████████████████|
32775/32775 [02:44<00:00, 199.33it/s]

32775
```

In [77]:
```
# Similarly you can vectorize for title also
```

In [173]:

```
    # average Word2Vec
# compute average word2vec for each review.

#w2v_model=Word2Vec(X_train['preprocessed_titles'].values,min_count=5,size=50, worke
#glove_words = list(w2v_model.wv.vocab)



tfidf_model_title_train= TfidfVectorizer()
tfidf_model_title_train.fit(X_train['preprocessed_titles'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_title_train.get_feature_names(), list(tfidf_model_
tfidf_words_title_train = set(tfidf_model_title_train.get_feature_names())



tfidf_w2v_vectors_title_train = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_train['preprocessed_titles'].values): # for each review/sente
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words_title_train):
            vec = w2v_model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            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]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 53
531/53531 [00:04<00:00, 10948.73it/s]

53531
300
```

In [174]:

```
    # average Word2Vec
# compute average word2vec for each review.

#w2v_model=Word2Vec(X_cv['preprocessed_titles'],min_count=5,size=50, workers=4)
#glove_words = list(w2v_model.wv.vocab)



tfidf_model_title_cv= TfidfVectorizer()
tfidf_model_title_cv.fit(X_cv['preprocessed_titles'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_title_cv.get_feature_names(), list(tfidf_model_tit
tfidf_words_title_cv = set(tfidf_model_title_cv.get_feature_names())



tfidf_w2v_vectors_title_cv = []; # the avg-w2v for each sentence/review is stored ir
for sentence in tqdm(X_cv['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words_title_cv):
            vec = w2v_model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            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]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 22
942/22942 [00:02<00:00, 11069.03it/s]

22942
300
```

In [175]:

```
    # average Word2Vec
# compute average word2vec for each review.

#w2v_model=Word2Vec(X_test['preprocessed_titles'].values,min_count=5,size=50, worker
#glove_words = list(w2v_model.wv.vocab)



tfidf_model_title_test= TfidfVectorizer()
tfidf_model_title_test.fit(X_test['preprocessed_titles'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_title_test.get_feature_names(), list(tfidf_model_t
tfidf_words_title_test = set(tfidf_model_title_test.get_feature_names())



tfidf_w2v_vectors_title_test = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_test['preprocessed_titles'].values): # for each review/senter
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words_title_test):
            vec = w2v_model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            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_test.append(vector)

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

```
100%|██████████████████████████████████████████████████████████████████████████| 32
775/32775 [00:02<00:00, 14678.92it/s]

32775
300
```

2.4 Appling KNN on different kind of featurization as mentioned in the instructions

Apply KNN on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instructions

In [176]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%100(
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

2.4.1 Applying KNN brute force on BOW, SET 1

In [72]:
```
# Please write all the code with proper documentation


from scipy.sparse import hstack

X_train_bow = hstack((categories_one_hot_train, sub_categories_one_hot_train, state_
X_test_bow= hstack((categories_one_hot_test, sub_categories_one_hot_test, state_one_
X_cv_bow = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, state_one_hot_c
```

In [73]:
```
print(X_test_bow.shape)
print(y_test.shape)

print(X_train_bow.shape)
print(y_train.shape)

print(X_cv_bow.shape)
print(y_cv.shape)
```

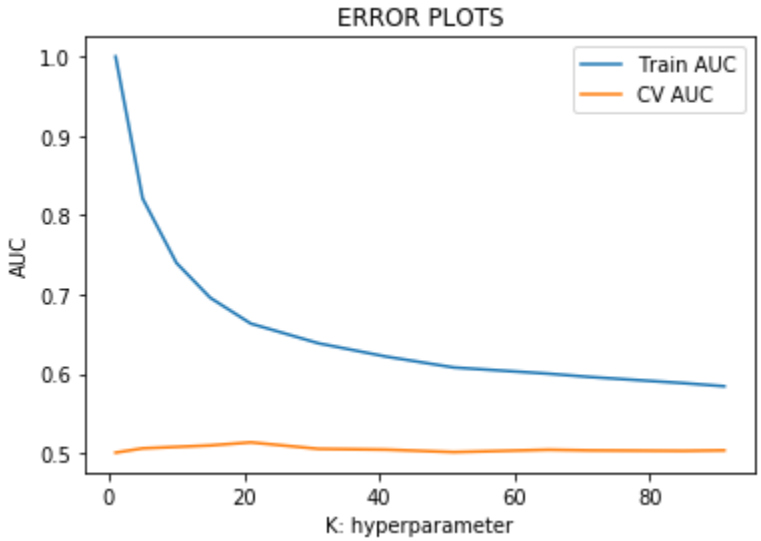
```
(32775, 7329)
(32775,)
(53531, 7329)
(53531,)
(22942, 7329)
(22942,)
```

In [74]:

```
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_bow, y_train)

    y_train_pred = batch_predict(neigh, X_train_bow)
    y_cv_pred = batch_predict(neigh, X_cv_bow)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))


plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()



#Below method will kill the kernel overnight
'''
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSear
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

neigh = KNeighborsClassifier()
parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51]}
clf = GridSearchCV(neigh, parameters, cv=3, scoring='roc_auc')
clf.fit(X_train_bow, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(K, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(K,train_auc - train_auc_std,train_auc + train_auc_std,alpha=0

plt.plot(K, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
```

```
plt.gca().fill_between(K,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='da
plt.legend()
plt.xlabel("TPR")
plt.ylabel("FPR")
plt.title("AUC-CURVE")
plt.show()
'''
```

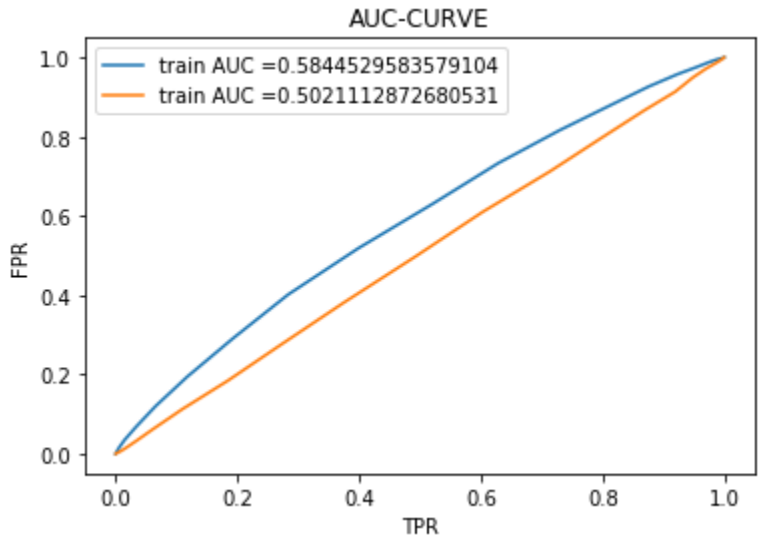
```
100%|██████████████████████████████████████████████████████████████████████████████
███| 12/12 [2:00:31<00:00, 588.04s/it]
```



Out[74]:
```
'\'\n# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.G
ridSearchCV.html\nfrom (https://scikit-learn.org/stable/modules/generated/sklearn.
model_selection.GridSearchCV.html\nfrom) sklearn.model_selection import GridSearch
CV\nfrom sklearn.neighbors import KNeighborsClassifier\nfrom sklearn.metrics impor
t roc_auc_score\nimport matplotlib.pyplot as plt\n\nneigh = KNeighborsClassifier()
\nparameters = {\'n_neighbors\':[1, 5, 10, 15, 21, 31, 41, 51]}\nclf = GridSearchC
V(neigh, parameters, cv=3, scoring=\'roc_auc\')\nclf.fit(X_train_bow, y_train)\n\n
train_auc= clf.cv_results_[\'mean_train_score\']\ntrain_auc_std= clf.cv_results_
[\'std_train_score\']\ncv_auc = clf.cv_results_[\'mean_test_score\'] \ncv_auc_std=
clf.cv_results_[\'std_test_score\']\n\nplt.plot(K, train_auc, label=\'Train AUC\')
\n# this code is copied from here: https://stackoverflow.com/a/48803361/4084039\np
lt.gca().fill_between(K,train_auc (https://stackoverflow.com/a/48803361/4084039\np
lt.gca().fill_between(K,train_auc) - train_auc_std,train_auc + train_auc_std,alpha
=0.2,color=\'darkblue\')\n\nplt.plot(K, cv_auc, label=\'CV AUC\')\n# this code is
 copied from here: https://stackoverflow.com/a/48803361/4084039\nplt.gca().fill_be
tween(K,cv_auc (https://stackoverflow.com/a/48803361/4084039\nplt.gca().fill_betwe
en(K,cv_auc) - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color=\'darkorange\')\npl
t.legend()\nplt.xlabel("TPR")\nplt.ylabel("FPR")\nplt.title("AUC-CURVE")\nplt.show
()\n'
```

In [75]:
```
best_k = 91
```

In [76]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#s
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_train_bow, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates o
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_bow)
y_test_pred = batch_predict(neigh, X_test_bow)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("TPR")
plt.ylabel("FPR")
plt.title("AUC-CURVE")
plt.show()


print("="*100)
```




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

In [143]:

```
# how to choose threshold for Confusion matrix
#https://stackoverflow.com/questions/32627926/scikit-changing-the-threshold-to-creat


from sklearn.metrics import confusion_matrix


def predict(proba, threshold, fpr, tpr):
    t = threshold[np.argmax(fpr*(1-tpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions

# plot confusion matrix
# https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matri

#https://stackoverflow.com/questions/19984957/scikit-predict-default-threshold
# defaul threshold is 0.5
#print(confusion_matrix(y_train, neigh.predict(y_train_pred, tr_thresholds, train_fp
```

In [145]:

```
print("Train confusion matrix")

print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, trai

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tp
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.33629462596773574 for threshold 0.846
[[ 4223  3882]
 [16661 28765]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2832985377569942 for threshold 0.857
[[ 2616  2347]
 [12864 14948]]
```

2.4.2 Applying KNN brute force on TFIDF, SET 2

In [78]:

```
from scipy.sparse import hstack

X_train_tfidf = hstack((categories_one_hot_train, sub_categories_one_hot_train, stat
X_test_tfidf = hstack((categories_one_hot_test, sub_categories_one_hot_test, state_c
X_cv_tfidf = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, state_one_hot
```

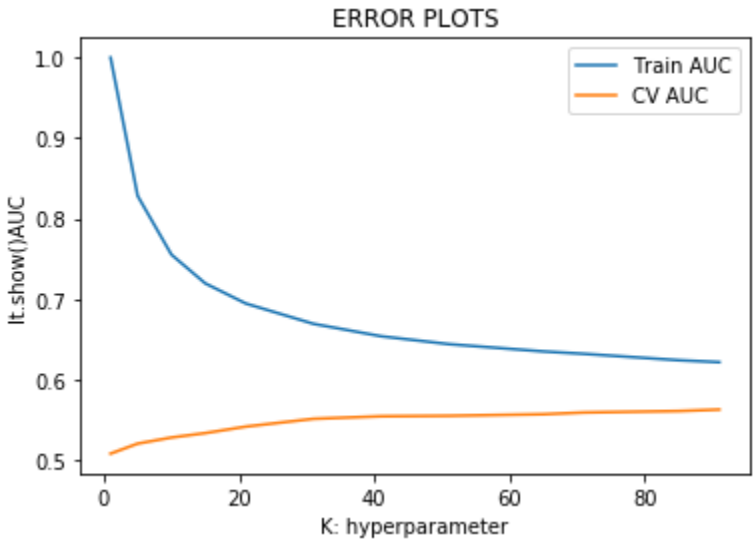
In [79]:
```
print(X_test_tfidf.shape)
print(y_test.shape)

print(X_train_tfidf.shape)
print(y_train.shape)

print(X_cv_tfidf.shape)
print(y_cv.shape)
```

```
(32775, 7329)
(32775,)
(53531, 7329)
(53531,)
(22942, 7329)
(22942,)
```

In [80]:

```
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_tfidf, y_train)

    y_train_pred = batch_predict(neigh, X_train_tfidf)
    y_cv_pred = batch_predict(neigh, X_cv_tfidf)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))



plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("lt.show()AUC")
plt.title("ERROR PLOTS")
plt.show()

print("="*100)
```

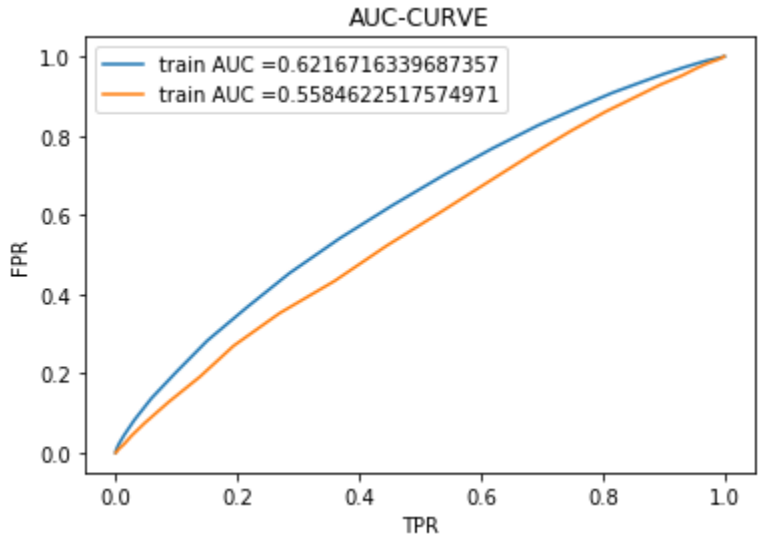
```
100%|██████████████████████████████████████████████████████████████████████████████
██| 12/12 [1:57:04<00:00, 556.35s/it]
```




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

In [81]:
```
best_k = 91
```

In [82]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#s
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_train_tfidf, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates o
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_tfidf)
y_test_pred = batch_predict(neigh, X_test_tfidf)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()

plt.xlabel("TPR")
plt.ylabel("FPR")
plt.title("AUC-CURVE")
plt.show()

print("="*100)
```




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

In [146]:
```
print("Train confusion matrix")

print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, trai

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tp
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.33629462596773574 for threshold 0.846
[[ 4223  3882]
 [16661 28765]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2832985377569942 for threshold 0.857
[[ 2616  2347]
 [12864 14948]]
```

2.4.3 Applying KNN brute force on AVG W2V, SET 3

In [89]:
```
# This is taking too long to run KNN, but preprocessing has been done properly



X_train_avg = hstack((categories_one_hot_train, sub_categories_one_hot_train, state_
X_test_avg = hstack((categories_one_hot_test, sub_categories_one_hot_test, state_one
X_cv_avg = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, state_one_hot_c
```

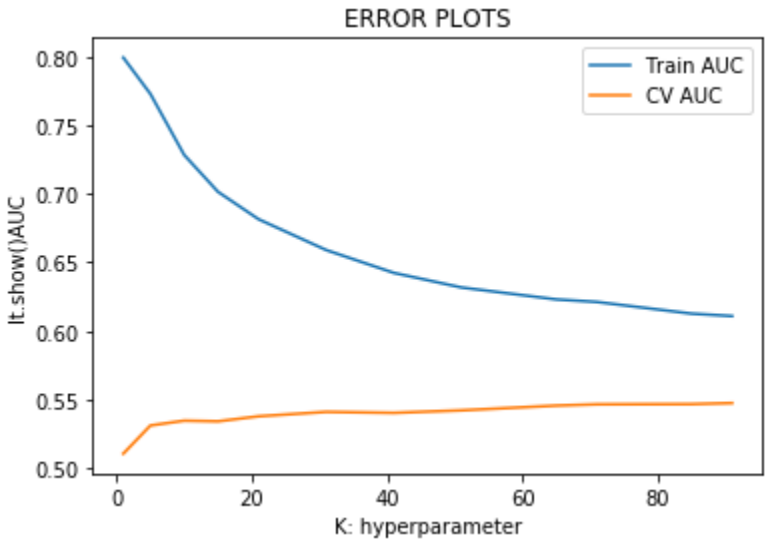
In [91]:
```
print(X_test_avg.shape)
print(y_test.shape)

print(X_train_avg.shape)
print(y_train.shape)

print(X_cv_avg.shape)
print(y_cv.shape)
```

```
(32775, 202)
(32775,)
(53531, 202)
(53531,)
(22942, 202)
(22942,)
```

In [92]:

```
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_avg, y_train)

    y_train_pred = batch_predict(neigh, X_train_avg)
    y_cv_pred = batch_predict(neigh, X_cv_avg)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))



plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()

print("="*100)
```

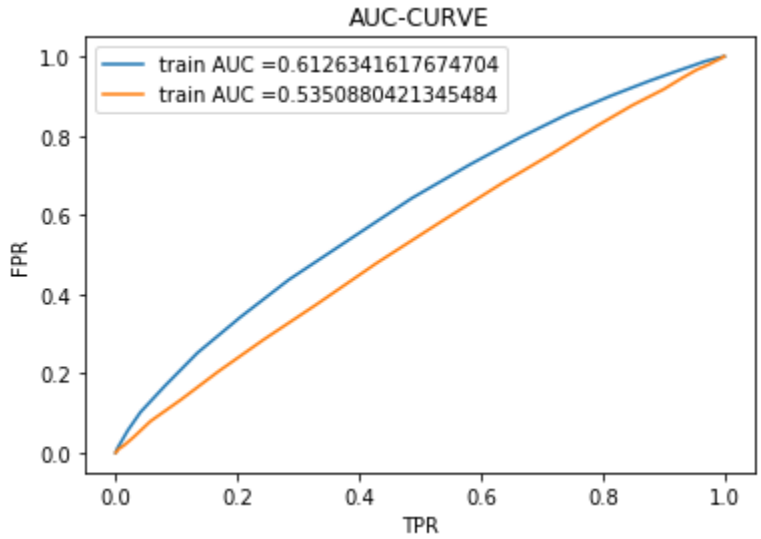
```
100%|██████████████████████████████████████████████████████████████████████████████
██| 12/12 [1:23:06<00:00, 354.83s/it]
```




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

In [93]:
```
best_k = 85
```

In [94]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#s
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_train_avg, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates o
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_avg)
y_test_pred = batch_predict(neigh, X_test_avg)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()


plt.xlabel("TPR")
plt.ylabel("FPR")
plt.title("AUC-CURVE")
plt.show()

print("="*100)
```




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

In [148]:
```
print("Train confusion matrix")

print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, trai

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tp
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.33629462596773574 for threshold 0.846
[[ 4223  3882]
 [16661 28765]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2832985377569942 for threshold 0.857
[[ 2616  2347]
 [12864 14948]]
```

2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [100]:
```
# This is taking too long to run KNN, but preprocessing has been done properly



X_train_tfidfw2v = hstack((categories_one_hot_train, sub_categories_one_hot_train, s
X_test_tfidfw2v = hstack((categories_one_hot_test, sub_categories_one_hot_test, stat
X_cv_tfidfw2v = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, state_one_
```

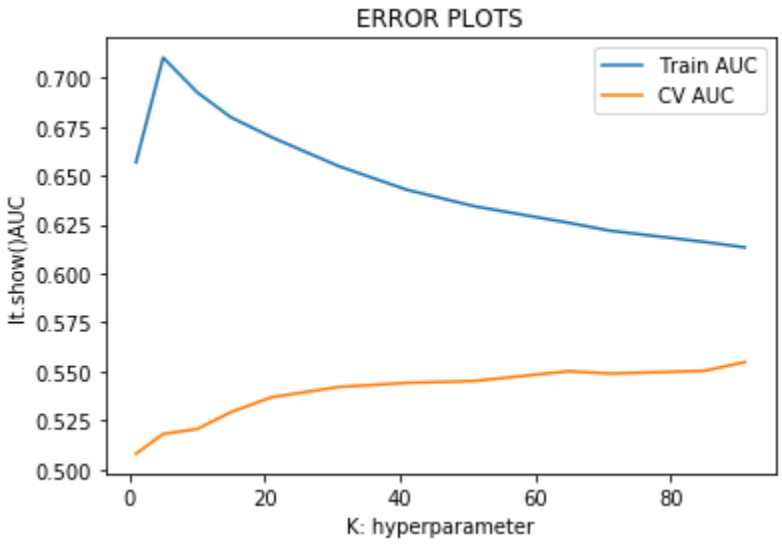
In [102]:
```
print(X_test_tfidfw2v.shape)


print(X_train_tfidfw2v.shape)


print(X_cv_tfidfw2v.shape)
```

```
(32775, 702)
(53531, 702)
(22942, 702)
```

In [103]:

```
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_tfidfw2v, y_train)

    y_train_pred = batch_predict(neigh, X_train_tfidfw2v)
    y_cv_pred = batch_predict(neigh, X_cv_tfidfw2v,)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))



plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("lt.show()AUC")
plt.title("ERROR PLOTS")
plt.show()

print("="*100)
```

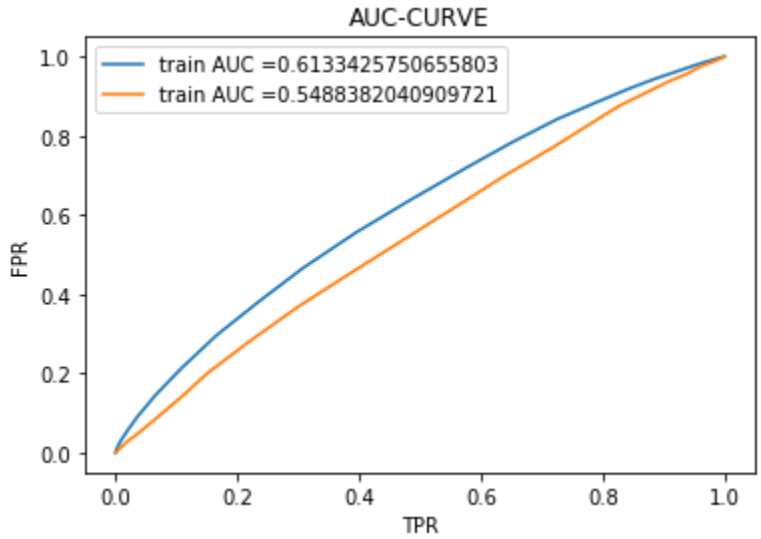
```
100%|██████████████████████████████████████████████████████████████████████████████
████| 12/12 [52:33<00:00, 272.58s/it]
```




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

In [104]:
```
best_k = 91
```

In [105]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#s
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_train_tfidfw2v, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates o
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_tfidfw2v)
y_test_pred = batch_predict(neigh, X_test_tfidfw2v)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()


plt.xlabel("TPR")
plt.ylabel("FPR")
plt.title("AUC-CURVE")
plt.show()

print("="*100)
```




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

In [149]:
```
print("Train confusion matrix")

print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, trai

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tp
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.33629462596773574 for threshold 0.846
[[ 4223  3882]
 [16661 28765]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2832985377569942 for threshold 0.857
[[ 2616  2347]
 [12864 14948]]
```

2.5 Feature selection with `SelectKBest`

In [115]:
```
from scipy.sparse import hstack

X_train_select = hstack((categories_one_hot_train, sub_categories_one_hot_train, sta
X_test_select = hstack((categories_one_hot_test, sub_categories_one_hot_test, state_
X_cv_select = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, state_one_ho
```

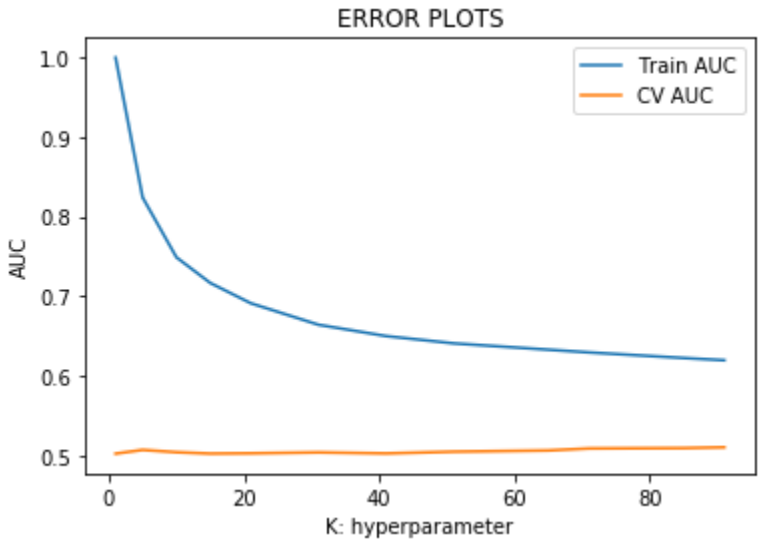
In [130]:
```
from sklearn.feature_selection import SelectKBest, chi2


# Be careful how to do proper fit_transform to cv and test data
selectk = SelectKBest(chi2, k=2000)

X_train_new = selectk.fit_transform(X_train_select, y_train)
X_test_new = selectk.transform(X_test_select)
X_cv_new = selectk.transform(X_cv_select)
```

In [131]:
```


print(X_train_select.shape, y_train.shape)
print(X_test_select.shape, y_cv.shape)
print(X_cv_select.shape, y_test.shape)

print(X_train_new.shape, y_train.shape)
print(X_test_new.shape, y_cv.shape)
print(X_cv_new.shape, y_test.shape)
```

```
(53531, 7329) (53531,)
(32775, 7329) (22942,)
(22942, 7329) (32775,)
(53531, 2000) (53531,)
(32775, 2000) (22942,)
(22942, 2000) (32775,)
```

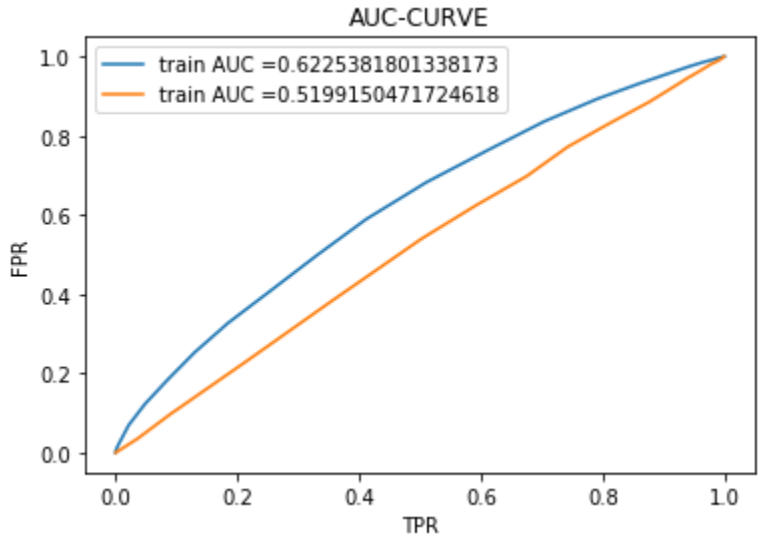
In [88]:
```
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train_new, y_train)

    y_train_pred = batch_predict(neigh, X_train_new)
    y_cv_pred = batch_predict(neigh, X_cv_new)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))


plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████| 12/12 [40:19<00:00, 199.99s/it]
```



In [89]:
```
best_k = 85
```

In [92]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#s
from sklearn.metrics import roc_curve, auc

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt



neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_train_new, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates o
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_new)
y_test_pred = batch_predict(neigh, X_test_new)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("TPR")
plt.ylabel("FPR")
plt.title("AUC-CURVE")
plt.show()

print("="*100)
```




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

In [151]:
```
print("Train confusion matrix")

print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, trai

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tp
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.33629462596773574 for threshold 0.846
[[ 4223  3882]
 [16661 28765]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2832985377569942 for threshold 0.857
[[ 2616  2347]
 [12864 14948]]
```

3. Conclusions

In [1]:
```
# Please compare all your models using Prettytable library

from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]

x.add_row(["BOW", "Brute", 91, 0.50])
x.add_row(["TFIDF", "Brute", 91, 0.56])
x.add_row(["AVG W2V", "Brute", 85, 0.54])
x.add_row(["TFIDF W2V", "Brute", 91, 0.55])
x.add_row(["TFIDF", "Top 200", 85, 0.52])

print(x)
```

```
+------------+---------+-----------------+------+
| Vectorizer |  Model  | Hyper Parameter | AUC  |
+------------+---------+-----------------+------+
|    BOW     |  Brute  |        91       | 0.5  |
|   TFIDF    |  Brute  |        91       | 0.56 |
|  AVG W2V   |  Brute  |        85       | 0.54 |
| TFIDF W2V  |  Brute  |        91       | 0.55 |
|   TFIDF    | Top 200 |        85       | 0.52 |
+------------+---------+-----------------+------+
```

In [ ]: