

KNN on Donors Choose Dataset

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	Description
<code>project_id</code>	A unique identifier for the proposed project. Example: p036502
<code>project_title</code>	Title of the project. Examples: <ul style="list-style-type: none"> • Art Will Make You Happy! • First Grade Fun
<code>project_grade_category</code>	Grade level of students for which the project is targeted. One of the following enumerated values: <ul style="list-style-type: none"> • Grades PreK-2 • Grades 3-5 • Grades 6-8 • Grades 9-12
<code>project_subject_categories</code>	One or more (comma-separated) subject categories for the project from the following enumerated list of values: <ul style="list-style-type: none"> • Applied Learning • Care & Hunger • Health & Sports • History & Civics • Literacy & Language • Math & Science • Music & The Arts • Special Needs • Warmth Examples: <ul style="list-style-type: none"> • Music & The Arts • Literacy & Language, Math & Science
<code>school_state</code>	State where school is located (<u>Two-letter U.S. postal code</u> (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). Example: WY
<code>project_subject_subcategories</code>	One or more (comma-separated) subject subcategories for the project. Examples: <ul style="list-style-type: none"> • Literacy • Literature & Writing, Social Sciences
<code>project_resource_summary</code>	An explanation of the resources needed for the project. Example: <ul style="list-style-type: none"> • My students need hands on literacy materials to manage sensory needs!
<code>project_essay_1</code>	First application essay*
<code>project_essay_2</code>	Second application essay*
<code>project_essay_3</code>	Third application essay*
<code>project_essay_4</code>	Fourth application essay*
<code>project_submitted_datetime</code>	Datetime when project application was submitted. Example: 2016-04-28 12:43:56.245
<code>teacher_id</code>	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4ff1c15c56
<code>teacher_prefix</code>	Teacher's title. One of the following enumerated values: <ul style="list-style-type: none"> • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
<code>teacher_number_of_previously_posted_projects</code>	Number of project applications previously submitted by the same teacher. Example: 2

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

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

Feature	Description
<code>id</code>	A <code>project_id</code> value from the <code>train.csv</code> file. Example: p036502

Feature	Description
description	Description 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]: import warnings
warnings.filterwarnings("ignore")
%matplotlib inline

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
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import cross_validation
```

1.1 Reading Data (Top 50,000 row)

```
In [3]: project_data = pd.read_csv('train_data.csv',nrows=50000)
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 (50000, 17)

The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state' 'project_submitted_datetime' 'project_grade_category' 'project_subject_categories' 'project_subject_subcategories' 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3' 'project_essay_4' 'project_resource_summary' 'teacher_number_of_previously_posted_projects' 'project_is_approved']

```
In [5]: # how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]

#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
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_grade_category	proj
31477	47750	p185738	3afe10b996b7646d8641985a4b4b570d	Mrs.	UT	2016-01-05 01:05:00	Grades PreK-2	Matr
40132	91045	p161351	40c9c33254a39827d6908ae9a6103c04	Teacher	CA	2016-01-05 01:59:00	Grades PreK-2	Spec

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

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

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

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

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

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/22898595/4084039
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]))

```

Preprocessing of teacher_prefix

```

In [9]: #“Teacher prefix” data having the dots(.) and its has been observed the some rows are empty in this feature .
#the dot(.) and empty row available in the data consider as float datatype and it does not
# accepted by the .Split() - Pandas function , so removing the same.
# cleaning has been done for the same following references are used
# 1. Removing (.) from dataframe column - used ".str.replce" funtion (padas documentation)
# 2. for empty cell in datafram column - added the "Mrs." (in train data.csv) which has me mostly occured in data set.

project_data["teacher_prefix_clean"] = project_data["teacher_prefix"].str.replace(".", "")
project_data.head(2)
print(project_data.teacher_prefix_clean.shape)

(50000,)

```

1.4 Text preprocessing

```

In [10]: # 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 [11]: project_data.head(2)

```

```

Out[11]:

```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	proj
31477	47750	p185738	3afe10b996b7646d8641985a4b4b570d	Mrs.	UT	2016-01-05 01:05:00	Grades PreK-2	Matr
40132	91045	p161351	40c9c33254a39827d6908ae9a6103c04	Teacher	CA	2016-01-05 01:59:00	Grades PreK-2	Wob Seat Wigg First Alter ...

1.4.1 Train , Cross Validation and Test Data Split

```
In [12]: #As recommended in the Lecture video, splitting the Data in Train, Test and Cross validation data set
#before applying Vectorization to avoid the data Leakage issues.
# As suggested to use stratify sampling, Referred following site for code
# https://stackoverflow.com/questions/29438265/stratified-train-test-split-in-scikit-learn

# split the data set into train and test
X_train, X_test, y_train, y_test = cross_validation.train_test_split(project_data, project_data['project_is_approved'
], test_size=0.33, stratify = project_data['project_is_approved'
])

# split the train data set into cross validation train and cross validation test
X_train, X_cv, y_train, y_cv = cross_validation.train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

```
In [13]: #Removing the class label from the data set, in our case the class label is “project is approved”
#From all Train, Test and Cross validation data set

#Train Data

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

#Test Data

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

#Cross Validation data

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

```
In [14]: # 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)
```

\n\nThe only way to learn mathematics is to do mathematics.\n\n\nPaul Halmos.\n\n\nMy students love math time and using the hands-on manipulative's to make sense of what is being taught. My students love coming to school and working as hard as they can to learn new concepts everyday. They especially love working with the math manipulative's and they are always coming up with new discoveries.\n\n\nOur school is a Title I school where over 50% of our population receives free or reduced lunch. Many of these children come from low-income families but this doesn't hinder their desire to learn. My students will be using these math manipulative's on a daily basis during our math block time. The students will each have opportunities to explore and make connections that make math more concrete and help them relate to the concepts being taught.\n\n\nThey will be learning about our base ten number system and learn about money which are both important real life skills. Learning about money is important so the students can feel independent when they are given or earn money. They will be able to have a real sense of how much they can buy with certain amounts of money. \n\n\nI'm so excited to watch them exchange pennies for other coins and learn to save their classroom money for purchases in our store.

=====
As a teacher working in a low-income/high poverty school district, my students are faced with several challenges both in and out of the classroom. Despite the many challenges they face, I am looking forward to keep things simple and provide my students with creative and meaningful learning experiences.\n\n\nLet's show all children that we support them and that through hard work and dedication all goals are possible. My students receive special education services. They are instructed in Reading, Math, Science, and Social Studies. Their disabilities range from Developmentally Delayed to Intellectual Disabilities. Their parents cannot provide the school resources they need to be successful.\n\n\nResearch has prove that having the necessary supplies available in ones classroom is a great way to reach diversity in learning styles as well as increasing students engagement. I am a firm believer that if a child knows that someone cares enough to go that extra mile for them, then they will strive to be a better student. The majority of my students struggle to keep up on a day to day basis, and I don't want the lack of supplies be another factor as to why they are behind in class. When my students come to the resource room for help, I also need certain supplies to pre-teach and reteach necessary skills to my struggling students. In my classroom, my students will have access to the necessary supplies they need to be successful student. As a special education teacher, it is my job to ensure that they have the tools necessary to be successful in the general education setting. nannan

=====
Benjamin Franklin said it best when he said, \n\nTell me and I forget. Teach me and I remember. Involve me and I learn.\n\n\nMy students understand that what they do and say matters. They know their actions can make a difference. I have a very talkative bunch of kids that are in an active, hands-on classroom. A typical day involves 5-10 eager students who are excited to learn more. Many of my students struggle with maintaining skills because they have difficulty with doing everyday tasks for different reasons. This project was inspired by my students who ask me everyday to sit or stand in different places around the classroom so they can show their full potential. I want them to be their best & have the best!\n\n\nI teach at a Title I school where English is a second language for many students.\n\n\nMy students face adversity such as coming from single-parent families and attending multiple schools during their elementary years. Over two-thirds of our students participate in the free or reduced-priced lunch program. My goal is to help my students have great hands-on experiences at school while learning the curriculum. My students constantly ask me for brain breaks throughout the day to lessen fatigue & release excess energy. They need that movement to persevere throughout the day and I've tried to incorporate more activity by allowing them the ability to find the right place to work through alternative seating. Whether it is under a stable, kneeling, standing or in wobble stools, these items make such a difference. These items allow my students the movement necessary to maximize their learning potential.\n\n\nThe requested materials will allow them the necessary movement that will boost brain activity while staying active and healthy.\n\n\nPart of my students' plan to stay active was to do yoga as part of their brain break period. My students randomly take turns leading the class in different poses or exercises. With so much curriculum to teach and only 15 minutes of recess, my students will strongly benefit from these requested items! These items are vital for student success because studies show that core exercises & balance therapy can help the students' overall well-being.\n\n\nThese donations will allow my students to become more engaged in their learning and to take ownership of their educational path (with the wiggles and all).\n\n\nMy students love to explore and learn new concepts in science. With your donation of science activities to our classroom, students will have a large selection of hands-on learning activities that reinforce the science content taught in the classroom. These activities will give students real life examples of science and help them to understand how to apply the information in real world situations.\n\n\nMy students desire to become little scientists!\n\n\nThe STEM activities and Can Do science activities will engage my students in science stations so they can apply and deepen their knowledge of the content taught in the classroom. nannan

```
In [15]: # 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
```


My classroom is full of joy and creativity! For many of the students that take my class, art is not something they have all of experience with, however once they see the results of their first project, they are hooked! Students continuously create, evaluate and reflect on their own work!\r\n\r\nMy students are from very diverse backgrounds, they range in age from 14 to 19 and classes are mixes of 9th, 10th, 11th and 12th graders.\r\nMany are from socio-economically distressed backgrounds, most are on the free-lunch program and quite a few have jobs that begin in the am before school or start right after the finish of the school day. We have set the bar high in terms of expectations of work, and by supplying the art room, this year is students will continue to produce quality artwork, which increases their confidence level and pride in their work. Even with these extra burdens that many in the surrounding areas do not face, my students are all heart, they are just great young people! It is an honor and a pleasure each day to engage and interact with such determined kids, their unique perspectives on each project inspires!\r\n\r\n\r\n\r\n\r\nFor the past four years I have awarded students who love to draw with a quality sketchbook, nothing describes how happy they are to receive this beautiful book! Many work on lined paper and have a collection of loved sketches collected in the bottom of their back pack. the sketch book allows them a special place to create and express themselves. Erasers and paper are the building blocks of an art class, we are always in need of them and having them readily at hand allows students to move through an assignment with out worry. Quality brushes make all the difference to my inspiring artists, nothing is worse than trying to paint with a thread-bear brush, these brushes will be used to investigate artistic movements, and painting techniques through out history. The canvases will allow us as a class in advanced art to create works that are worthy of display. Art is a subject that builds moral and confidence, when a child receives materials that are of value they treat their work equally!nannan

=====

My classroom is full of joy and creativity! For many of the students that take my class, art is not something they have all of experience with, however once they see the results of their first project, they are hooked! Students continuously create, evaluate and reflect on their own work! My students are from very diverse backgrounds, they range in age from 14 to 19 and classes are mixes of 9th, 10th, 11th and 12th graders. Many are from socio-economically distressed backgrounds, most are on the free-lunch program and quite a few have jobs that begin in the am before school or start right after the finish of the school day. We have set the bar high in terms of expectations of work, and by supplying the art room, this year is students will continue to produce quality artwork, which increases their confidence level and pride in their work. Even with these extra burdens that many in the surrounding areas do not face, my students are all heart, they are just great young people! It is an honor and a pleasure each day to engage and interact with such determined kids, their unique perspectives on each project inspires! For the past four years I have awarded students who love to draw with a quality sketchbook, nothing describes how happy they are to receive this beautiful book! Many work on lined paper and have a collection of loved sketches collected in the bottom of their backpack. The sketch book allows them a special place to create and express themselves. Erasers and paper are the building blocks of an art class, we are always in need of them and having them readily at hand allows students to move through an assignment without worry. Quality brushes make all the difference to my inspiring artists, nothing is worse than trying to paint with a thread-bare brush, these brushes will be used to investigate artistic movements, and painting techniques through out history. The canvases will allow us as a class in advanced art to create works that are worthy of display. Art is a subject that builds moral and confidence, when a child receives materials that are of value they treat their work equally!nannan

My classroom is full of joy and creativity For many of the students that take my class art is not something they have all of experience with however once they see the results of their first project they are hooked Students continuously create evaluate and reflect on their own work My students are from very diverse backgrounds they range in age from 14 to 19 and classes are mixes of 9th 10th 11th and 12th graders Many are from socio economically distressed backgrounds most are on the free lunch program and quite a few have jobs that begin in the am before school or start right after the finish of the school day We have set the bar high in terms of expectations of work and by supplying the art room this year is students will continue to produce quality artwork which increases their confidence level and pride in their work Even with these extra burdens that many in the surrounding areas do not face my students are all heart they are just great young people It is an honor and a pleasure each day to engage and interact with such determined kids their unique perspectives on each project inspires For the past four years I have awarded students who love to draw with a quality sketchbook nothing describes how happy they are to receive this beautiful book Many work on lined paper and have a collection of loved sketches collected in the bottom of their back pack the sketch book allows them a special place to create and express themselves Erasers and paper are the building blocks of an art class we are always in need of them and having them readily at hand allows students to move through an assignment without worry Quality brushes make all the difference to my inspiring artists nothing is worse than trying to paint with a thread bear brush these brushes will be used to investigate artistic movements and painting techniques throughout history The canvases will allow us as a class in advanced art to create works that are worthy of display Art is a subject that builds moral and confidence when a child receives materials that are of value they treat their work equally

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

1.4.2 Train - Data Processing (Essay)

```
# Combining all the above students
from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentence in tqdm(X_train['essay'].values):
    sent = decontracted(sentence)
    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_train.append(sent.lower().strip())
```

[illegible]

```
# after preprocessing
preprocessed_essays_train[1000]
```

'nelson mandela said education powerful weapon change world teach inner city school families poverty level students receive free breakfast lunch students attending school first time 4 5 years old tk kindergartners curious energetic enthusiastic learners want instill students education powerful students need technology order succeed future learn basics use technology use technology order advance academic proficiency along two working desktop computers classroom laptop requesting expose students things cannot see experience everyday students able access educational websites enhance language arts mathematical skills laptop allow students integrate technology learning across curriculum educational websites accommodate students individual needs whether enhance language development math skills decision making encourage independent learning nannan'

1.4.3 Cross Validation - Data Processing (Essay)

```
preprocessed_essays_cv = []
# tqdm is for printing the status bar
for sentence in tqdm(X_cv['essay'].values):
    sent = decontracted(sentence)
    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_cv.append(sent.lower().strip())
```

[illegible]

```
Out[23]: 'school demographic 27 african american rest made additional races not reflect racial profile county 60 free reduced
lunch less 60 proficiency grade level based standards based standardized tests however voracious readers books hands
speak rather traditional books taught english classrooms east high community one vibrant diversity rainbow cultures p
eople interspersed neighborhoods surrounding school amazing coffee tea shops international cuisine huge grocery co op
vegetable gardens course chickens students voracious readers books find relevance anchor texts date face students cha
nged years along district direction funding need connect students interests not funding buy new sets help color purpl
e long time favorite choice book students would like use anchor text plan book engage content vernacular teach write
dominant language culture analyzing language book sound interesting thought long hard engage students speak write out
side culture dominant language first full fledged attempt nannan'
```

1.4.4 Test - Data Processing (Essay)

[illegible]

```
Out[25]: '2nd grade teacher low income school recently became programming specialty school incorporated technology daily schedule love use technology education not filling pail lighting fire william butler yeatsi classroom full 25 bright eyed children come low income families little home looking give students things not see home motivated technology want give types tools keep eager learn many great apps learn grow know improve scores amount educational apps ipads endless research apps ipads students get best experience possible nothing makes heart happier see students motivated learn technology helps get motivated able math reading writing science social studies etc possibilities endless ipads not wait see smiles get use classroom motivation technology motivating kids days teach school motivation hard come excited see difference motivation students learn motivated learn scores improve project make difference academic motivation scores'
```

1.5 Preprocessing of `project title`

Wobbly Seats for Wiggly Firsties: Alternative Seating

Wobbly Seats for Wiggly Firsties: Alternative Seating

1.5.1 Train - Data Processing (Project Title)

```
100%|██████████████████████████████████████████████████████████████████████████████| 22445/22445 [00:00<00:00, 40024.23it/
s]
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11055/11055 [00:00<00:00, 40973.28it/
s]
```

```
100%|██████████████████████████████████████████████████████████████| 16500/16500 [00:00<00:00, 40701.83it/
s]
```

```
Out[32]: 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', 'teacher_prefix_clean',
               'essay'],
              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

1.6.1 Vectorizing Categorical data

- <https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/>
(<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/>)

Project_categories - Vectorization

```
In [33]: # we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)

vectorizer.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer.transform(X_train['clean_categories'].values)
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("Shape of matrix after one hot encodig ",categories_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",categories_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",categories_one_hot_test.shape)

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_S
cience', 'Literacy_Language']
Shape of matrix after one hot encodig (22445, 9)
Shape of matrix after one hot encodig (11055, 9)
Shape of matrix after one hot encodig (16500, 9)
```

Project_sub_categories - Vectorization

```
In [34]: # we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)

vectorizer.fit(X_train['clean_subcategories'].values)

sub_categories_one_hot_train = vectorizer.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)

print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",sub_categories_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_test.shape)

['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government',
'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEduc
ation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelo
pment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNee
ds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig (22445, 30)
Shape of matrix after one hot encodig (11055, 30)
Shape of matrix after one hot encodig (16500, 30)
```

School_State - Vectorization

```
In [35]: # we use count vectorizer to convert the values into one hot encoded features
from collections import Counter
my_counter_state = Counter()
for word in project_data['school_state'].values:
    my_counter_state.update(word.split())

state_dict = dict(my_counter_state)
sorted_state_dict = dict(sorted(state_dict.items(), key=lambda kv: kv[1]))

vectorizer = CountVectorizer(vocabulary=list(sorted_state_dict.keys()), lowercase=False, binary=True)

vectorizer.fit(X_train['school_state'].values)

school_state_one_hot_train = vectorizer.transform(X_train['school_state'].values)
school_state_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)
school_state_one_hot_test = vectorizer.transform(X_test['school_state'].values)

print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",school_state_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",school_state_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",school_state_one_hot_test.shape)

['VT', 'WY', 'ND', 'MT', 'RI', 'NH', 'SD', 'NE', 'AK', 'DE', 'WV', 'ME', 'NM', 'HI', 'DC', 'KS', 'ID', 'IA', 'AR', 'C
O', 'MN', 'OR', 'MS', 'KY', 'NV', 'MD', 'CT', 'TN', 'AL', 'UT', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'MA', 'LA', 'WA', 'MO',
'IN', 'OH', 'PA', 'MI', 'GA', 'SC', 'IL', 'NC', 'FL', 'TX', 'NY', 'CA']
Shape of matrix after one hot encodig (22445, 51)
Shape of matrix after one hot encodig (11055, 51)
Shape of matrix after one hot encodig (16500, 51)
```

teacher_prefix - Vectorization

```
In [36]: #“Teacher prefix” data having the dots(.) and its has been observed the some rows are empty in this feature .
#the dot(.) and empty row available in the data consider as float datatype and it does not
# accepted by the .Split() – Pandas function , so removing the same.
# cleaning has been done for the same following references are used
# 1. Removing (.) from dataframe column - used ".str.replce" funtion (padas documentation)
# 2. for empty cell in datafram column - added the "Mrs." (in train data.csv) which has me mostly occured in data
set.

project_data["teacher_prefix_clean"] = project_data["teacher_prefix"].str.replace(".", "")
project_data.head(2)
print(project_data.teacher_prefix_clean.shape)

(50000,)
```

```
In [37]: from collections import Counter
my_counter_T = Counter()
for word in project_data["teacher_prefix_clean"].values:

    my_counter_T.update(word.split())

Teacher_dict = dict(my_counter_T)
sorted_Teacher_dict = dict(sorted(Teacher_dict.items(), key=lambda kv: kv[1]))

vectorizer = CountVectorizer(vocabulary=list(Teacher_dict.keys()), lowercase=False, binary=True)
#vectorizer.fit(project_data.teacher_prefix_clean.values)

vectorizer.fit(X_train["teacher_prefix_clean"].values)
print(vectorizer.get_feature_names())

Teacher_Prefix_one_hot_train = vectorizer.transform(X_train["teacher_prefix_clean"].values)
Teacher_Prefix_one_hot_cv = vectorizer.transform(X_cv["teacher_prefix_clean"].values)
Teacher_Prefix_one_hot_test = vectorizer.transform(X_test["teacher_prefix_clean"].values)

print("Shape of matrix after one hot encodig ",Teacher_Prefix_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",Teacher_Prefix_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",Teacher_Prefix_one_hot_test.shape)

['Mrs', 'Teacher', 'Mr', 'Ms', 'Dr']
Shape of matrix after one hot encodig  (22445, 5)
Shape of matrix after one hot encodig  (11055, 5)
Shape of matrix after one hot encodig  (16500, 5)
```

project_grade_category - Vectorization

```
In [38]: # Used this as reference to avoide the space between grades and category ,
# it has split the string with comma , now getting four project grade category as required.
# https://stackoverflow.com/questions/4071396/split-by-comma-and-strip-whitespace-in-python
from collections import Counter
my_counter_project_grade_category= Counter()
for word in project_data['project_grade_category'].values:
    my_counter_project_grade_category.update(word.split(','))

project_grade_category_dict = dict(my_counter_project_grade_category)
sorted_project_grade_category_prefix_dict = dict(sorted(project_grade_category_dict.items(), key=lambda kv: kv[1]))

vectorizer = CountVectorizer(vocabulary=list(project_grade_category_dict.keys()), lowercase=False, binary=True)

vectorizer.fit(X_train["project_grade_category"].values)
print(vectorizer.get_feature_names())

project_grade_category_one_hot_train = vectorizer.transform(X_train["project_grade_category"].values)
project_grade_category_one_hot_cv = vectorizer.transform(X_cv["project_grade_category"].values)
project_grade_category_one_hot_test = vectorizer.transform(X_test["project_grade_category"].values)

print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_test.shape)

['Grades PreK-2', 'Grades 3-5', 'Grades 6-8', 'Grades 9-12']
Shape of matrix after one hot encodig  (22445, 4)
Shape of matrix after one hot encodig  (11055, 4)
Shape of matrix after one hot encodig  (16500, 4)
```

1.6.2 Vectorizing Text data

1.6.2.1 Bag of words

Train Data Vectorization - BOW (essays)

```
In [39]: # We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(preprocessed_essays_train)
bow_essays_train = vectorizer.fit_transform(preprocessed_essays_train)
print("Shape of matrix after one hot encodig ",bow_essays_train.shape)

Shape of matrix after one hot encodig  (22445, 8805)
```

CV Data Vectorization - BOW (essays)

```
In [40]: bow_essays_cv = vectorizer.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encodig ",bow_essays_cv.shape)
```

Shape of matrix after one hot encodig (11055, 8805)

Test Data Vectorization - BOW (essays)

```
In [41]: bow_essays_test = vectorizer.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",bow_essays_test.shape)
```

Shape of matrix after one hot encoding (16500, 8805)

Train Data Vectorization - BOW (Project Titles)

```
In [42]: # We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
bow_title_train = vectorizer.fit_transform(preprocessed_titles_train)
print("Shape of matrix after one hot encodig ",bow_title_train.shape)
```

Shape of matrix after one hot encodig (22445, 1228)

CV Data Vectorization - BOW (Project Titles)

```
In [43]: bow_title_cv = vectorizer.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encodig ",bow_title_cv.shape)
```

Shape of matrix after one hot encodig (11055, 1228)

Test Data Vectorization - BOW (Project Titles)

```
In [44]: bow_title_test = vectorizer.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encodig ",bow_title_test.shape)
```

Shape of matrix after one hot encodig (16500, 1228)

1.6.2.2 TFIDF vectorizer

Train Data Vectorization - TFIDF (essays)

```
In [45]: from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
tfidf_essays_train = vectorizer.fit_transform(preprocessed_essays_train)
print("Shape of matrix after one hot encodig ",tfidf_essays_train.shape)
```

Shape of matrix after one hot encodig (22445, 8805)

CV Data Vectorization - TFIDF (essays)

```
In [46]: tfidf_essays_cv = vectorizer.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encodig ",tfidf_essays_cv.shape)
```

Shape of matrix after one hot encodig (11055, 8805)

Test Data Vectorization - TFIDF (essays)

```
In [47]: tfidf_essays_test = vectorizer.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encodig ",tfidf_essays_test.shape)
```

Shape of matrix after one hot encodig (16500, 8805)

Train Data Vectorization - TFIDF (Project Titles)

```
In [48]: vectorizer = CountVectorizer(min_df=10)
tfidf_title_train = vectorizer.fit_transform(preprocessed_titles_train)
print("Shape of matrix after one hot encodig ",bow_title_train.shape)
```

Shape of matrix after one hot encodig (22445, 1228)

CV Data Vectorization - TFIDF (Project Titles)


```
In [49]: tfidf_title_cv = vectorizer.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encodig ",bow_title_cv.shape)
```

Shape of matrix after one hot encodig (11055, 1228)

Test Data Vectorization - TFIDF (Project Titles)

```
In [50]: tfidf_title_test = vectorizer.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encodig ",bow_title_test.shape)
```

Shape of matrix after one hot encodig (16500, 1228)

1.6.2.3 Using Pretrained Models: Avg W2V

```
In [51]: # 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')
```

Loading Glove Model

1917495it [04:21, 7325.72it/s]

Done. 1917495 words loaded!

```
In [52]: words = []
for i in preprocessed_essays_train:
    words.extend(i.split(' '))

for i in preprocessed_essays_train:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

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

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

all the words in the coupus 6192336

the unique words in the coupus 30410

The number of words that are present in both glove vectors and our coupus 28687 (94.334 %)

word 2 vec length 28687

```
In [54]: import pickle
         with open('glove_vectors', 'wb') as f:
             pickle.dump(words_courpus, f)
```

```
In [55]: # stringing variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variable-s-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words = set(model.keys())
```

Train Data Vectorization - AGV_W2V (essays)

```
100%|██████████████████████████████████████████████████████████████████████████████| 22445/22445 [00:05<00:00, 3756.86it/
s]

22445
300
```

CV Data Vectorization - AGV_W2V (essays)


```
In [60]: # average Word2Vec
# compute average word2vec for each review.
avg_w2v_title_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_title_cv.append(vector)

print(len(avg_w2v_title_cv))
print(len(avg_w2v_title_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 11055/11055 [00:00<00:00, 57546.92it/s]
```

```
11055
300
```

Test Data Vectorization - AGV_W2V (Project Titles)

```
In [61]: # average Word2Vec
# compute average word2vec for each review.
avg_w2v_title_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_title_test.append(vector)

print(len(avg_w2v_title_test))
print(len(avg_w2v_title_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 16500/16500 [00:00<00:00, 50201.02it/s]
```

```
16500
300
```

1.6.2.3 Using Pretrained Models: TFIDF weighted W2V

```
In [62]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

Train Data Vectorization - TFIDF_W2V (essays)

```
100%|██████████████████████████████████████████████████████████████████████████████| 22445/22445 [00:34<00:00, 659.14it/s]
22445
300
```

CV Data Vectorization - TFIDF W2V (essays)

```
100%|██████████████████████████████████████████████████████████████████████████| 11055/11055 [00:16<00:00, 654.91it/
s]

11055
300
```

Test Data Vectorization - TFIDF_W2V (essays)

```
In [66]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words_titles = set(tfidf_model.get_feature_names())
```

CV Data Vectorization - TFIDF_W2V (Project Titles)

```
In [68]: # average Word2Vec  
# compute average word2vec for each review.  
tfidf_w2v_title_cv = []; # the avg-w2v for each sentence/review is stored in this list  
for sentence in tqdm(preprocessed_titles_cv): # 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_titles ):  
            vec = model[word] # getting the vector for each word  
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.  
split()))))  
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each  
word  
            vector += (vec * tf_idf) # calculating tfidf weighted w2v  
            tf_idf_weight += tf_idf  
        if tf_idf_weight != 0:  
            vector /= tf_idf_weight  
    tfidf_w2v_title_cv.append(vector)  
  
print(len(tfidf_w2v_title_cv))  
print(len(tfidf_w2v_title_cv[0]))
```

100%|███████████████████████████████████████| 11055/11055 [00:00<00:00, 31837.21it/
s]

11055
300

Test Data Vectorization - TFIDF W2V (Project Titles)

```
In [69]: # average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_title_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_test): # 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_titles):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.
split()))))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each
word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title_test.append(vector)

print(len(tfidf_w2v_title_test))
print(len(tfidf_w2v_title_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 16500/16500 [00:00<00:00, 33995.76it/
s]

16500
300
```

1.6.3 Vectorizing Numerical features

1.6.3.1 Vectorizing Numerical features - Price

```
In [70]: price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()

# Merging the project data train , Cv , test with price from resource data
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
```

In [71]:

X_train.head(2)

Out[71]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_title
0	159832	p023402	2d6f7e333ebd4a2240b1568b2e15d4aa	Ms.	NY	2017-03-26 10:01:00	Grades PreK-2	Breakfast and Snacks For My Scholars!
1	35780	p184421	129293679500398c9997ebb800abb312	Mrs.	NC	2016-08-08 17:04:00	Grades 3-5	Research Create, a Share!

In [72]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

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

price_train = normalizer.transform(X_train['price'].values.reshape(-1,1))
price_cv = normalizer.transform(X_cv['price'].values.reshape(-1,1))
price_test = normalizer.transform(X_test['price'].values.reshape(-1,1))

print(price_train.shape)
print(price_cv.shape)
print(price_test.shape)
```

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

1.6.3.2 Vectorizing Numerical features - teacher_number_of_previously_posted_projects

In [73]:

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

prev_post_train = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_post_cv = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_post_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print(prev_post_train.shape)
print(prev_post_cv.shape)
print(prev_post_test.shape)
```

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

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 


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/>) 

### 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

### 2.4 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 [74]: # please write all the code with proper documentation, and proper titles for each subsection
go through documentations and blogs before you start coding
first figure out what to do, and then think about how to do.
reading and understanding error messages will be very much helpfull in debugging your code

when you plot any graph make sure you use
a. Title, that describes your plot, this will be very helpful to the reader
b. Legends if needed
c. X-axis Label
d. Y-axis Label
```

## 2.4.1 Applying KNN brute force on BOW, SET 1

```
In [75]: from scipy.sparse import hstack
X_train_bow = hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one_hot_train ,Teacher_Prefix_one_hot_train,project_grade_category_one_hot_train,bow_essays_train,bow_title_train,price_train,prev_post_train)).tocsr()
X_train_bow.shape
```

Out[75]: (22445, 10134)

```
In [76]: X_cv_bow = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_one_hot_cv ,Teacher_Prefix_one_hot_cv,project_grade_category_one_hot_cv,bow_essays_cv,bow_title_cv,price_cv,prev_post_cv)).tocsr()
X_cv_bow.shape
```

Out[76]: (11055, 10134)

```
In [77]: X_test_bow = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hot_test ,Teacher_Prefix_one_hot_test,project_grade_category_one_hot_test,bow_essays_test,bow_title_test,price_test,prev_post_test)).tocsr()

X_test_bow.shape
```

Out[77]: (16500, 10134)

## Finding the best hyper parameter That maximum AUC value

```
In [78]: from sklearn.metrics import roc_auc_score
def batch_predict(clf, data):
 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
 # not the predicted outputs

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

 return y_data_pred
```

```
In [80]: from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

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

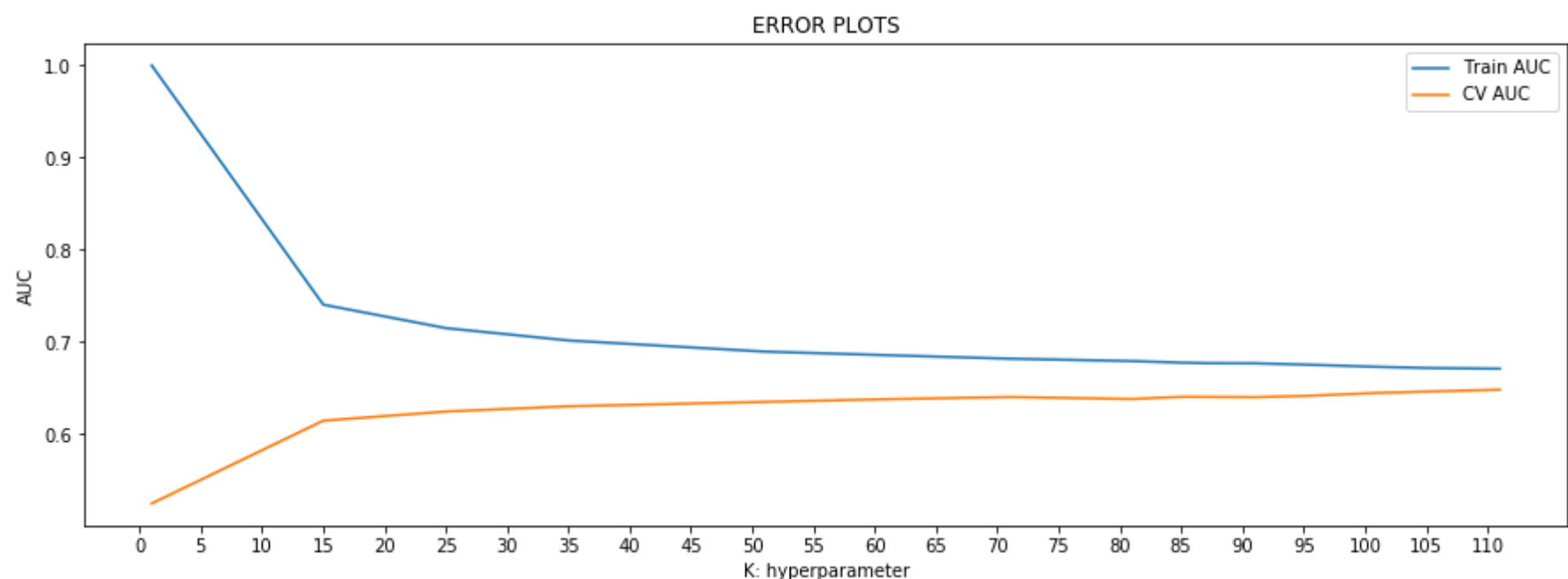
y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
K = [1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]
for i in K:
 neigh = KNeighborsClassifier(n_neighbors=i,)
 neigh.fit(X_train_bow, y_train)
 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
 # not the predicted outputs
 y_train_pred = batch_predict(neigh,X_train_bow)
 y_cv_pred = batch_predict(neigh,X_cv_bow)

 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.xticks(np.arange(0, 115, step=5))
plt.rcParams["figure.figsize"] = [16,9]
plt.show()
```



**GridSearchCV - Finding the best hyper parameter That maximum AUC value**

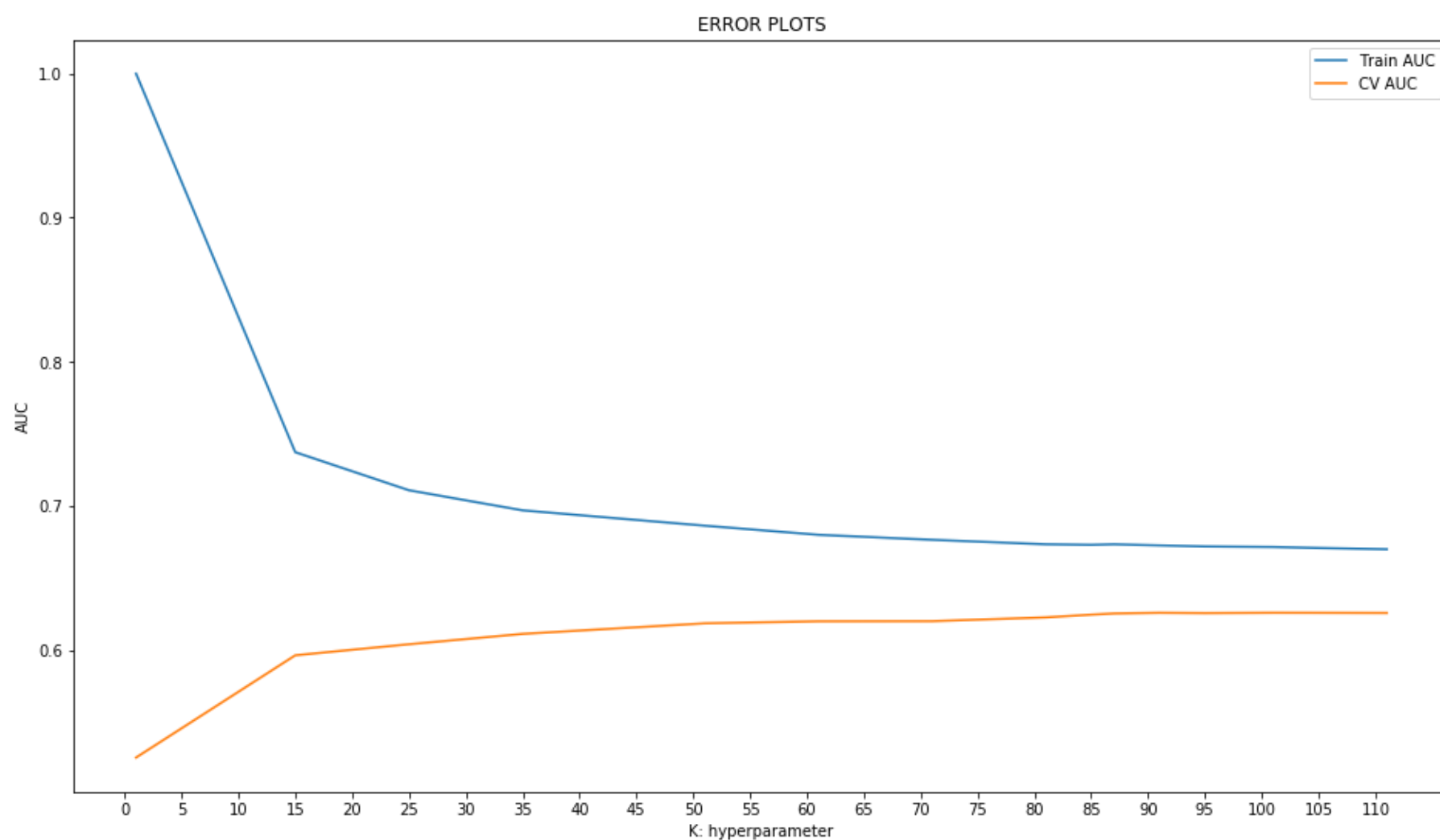
```
In [81]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()
parameters = {'n_neighbors':[1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]}
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')
plt.plot(K, cv_auc, label='CV AUC')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.xticks(np.arange(0, 115, step=5))
plt.rcParams["figure.figsize"] = [16,9]
plt.show()
```



### Using Best K Value – Training the Model

```
In [151]: best_k_bow = 95
```

```
In [152]: #https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

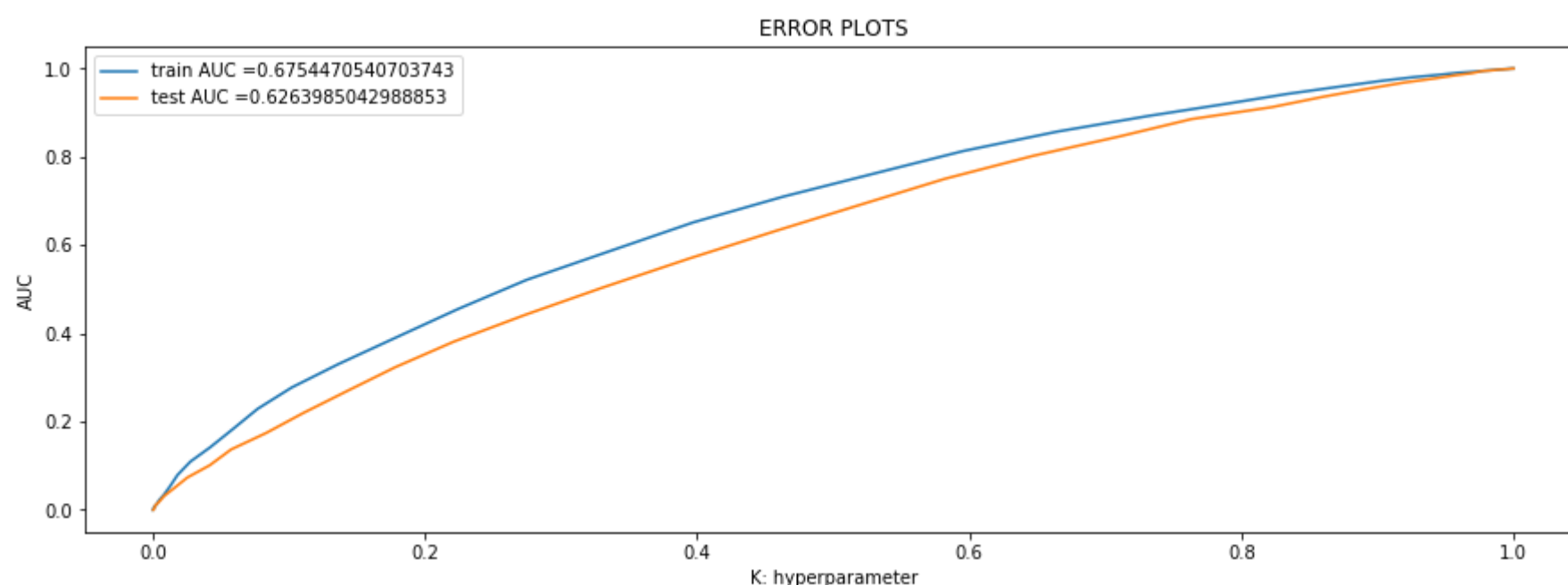
neigh = KNeighborsClassifier(n_neighbors=best_k_bow,algorithm= 'brute')
neigh.fit(X_train_bow, y_train)

roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
not the predicted outputs

y_train_pred = batch_predict(neigh, X_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="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



## Confusion Matrix

### Train confusion matrix

```
In [153]: #https://stackoverflow.com/questions/28719067/roc-curve-and-cut-off-point-python

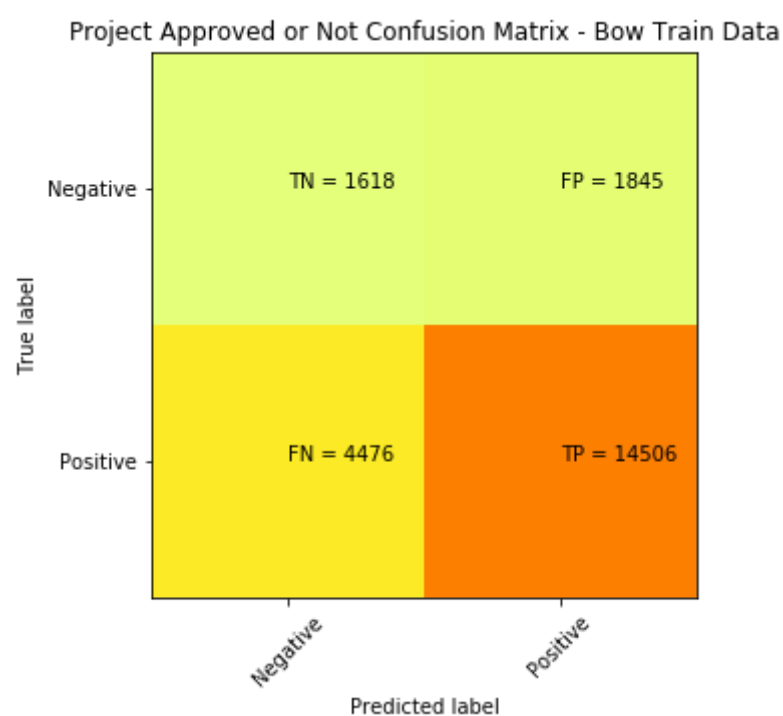
def predict(proba, threshold, fpr, tpr):
 t = threshold[np.argmax(fpr*(1-tpr))]
 print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
 predictions = []
 for i in proba:
 if i>=t:
 predictions.append(1)
 else:
 predictions.append(0)
 return predictions
```

```
In [154]: from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
bow_train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
print(bow_train_confusion_matrix)
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2489257960624794 for threshold 0.758
[[1618 1845]
 [4476 14506]]
```

In [155]: [#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/](http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/)

```
plt.clf()
plt.imshow(bow_train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - Bow Train Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(bow_train_confusion_matrix[i][j]))
plt.show()
```



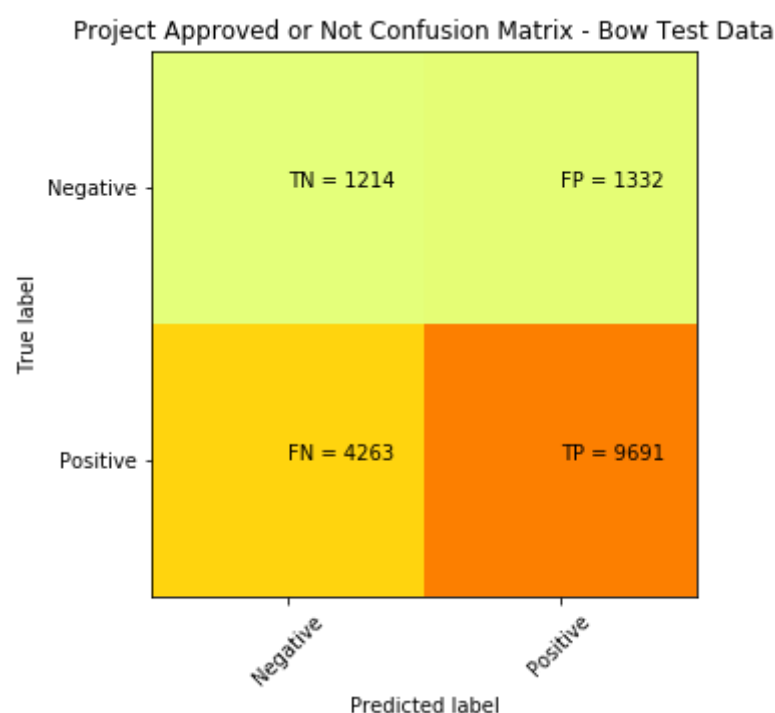
### Test confusion matrix

```
In [156]: print("Train confusion matrix")
bow_test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_fpr))
print(bow_test_confusion_matrix)
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24946298400090341 for threshold 0.768
[[1214 1332]
 [4263 9691]]
```

In [157]: <http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/>

```
plt.clf()
plt.imshow(bow_test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - Bow Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(bow_test_confusion_matrix[i][j]))
plt.show()
```



## 2.4.2 Applying KNN brute force on TFIDF, SET 2

In [100]: `X_train_tfidf = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train , Teacher_Prefix_one_hot_train, project_grade_category_one_hot_train, tfidf_essays_train, tfidf_title_train, price_train, prev_post_train)).tocsr()`  
`X_train_tfidf.shape`

Out[100]: (22445, 10134)

In [101]: `X_cv_tfidf = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_one_hot_cv , Teacher_Prefix_one_hot_cv, project_grade_category_one_hot_cv, tfidf_essays_cv, tfidf_title_cv, price_cv, prev_post_cv)).tocsr()`  
`X_cv_tfidf.shape`

Out[101]: (11055, 10134)

In [102]: `X_test_tfidf = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test , Teacher_Prefix_one_hot_test, project_grade_category_one_hot_test, tfidf_essays_test, tfidf_title_test, price_test, prev_post_test)).tocsr()`  
`X_test_tfidf.shape`

Out[102]: (16500, 10134)

## Finding the best hyper parameter That maximum AUC value

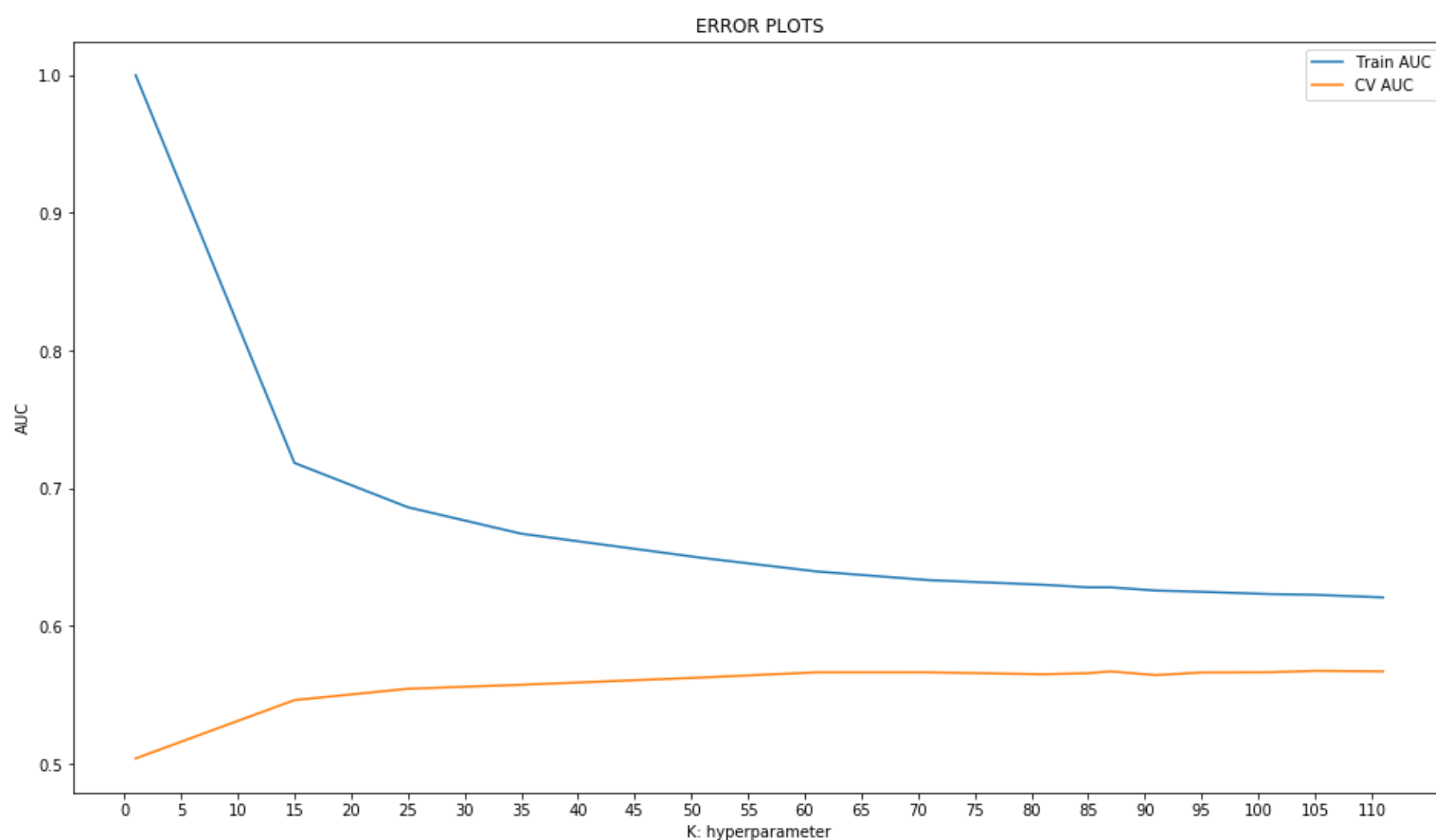
```

In [117]: train_auc = []
cv_auc = []
K = [1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]
for i in K:
 neigh = KNeighborsClassifier(n_neighbors=i,)
 neigh.fit(X_train_tfidf, y_train)
 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
 # not the predicted outputs
 y_train_pred = batch_predict(neigh,X_train_tfidf)
 y_cv_pred = batch_predict(neigh,X_cv_tfidf)

 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.xticks(np.arange(0, 115, step=5))
plt.rcParams["figure.figsize"] = [15,5]
plt.show()

```



**GridSearchCV - Finding the best hyper parameter That maximum AUC value**



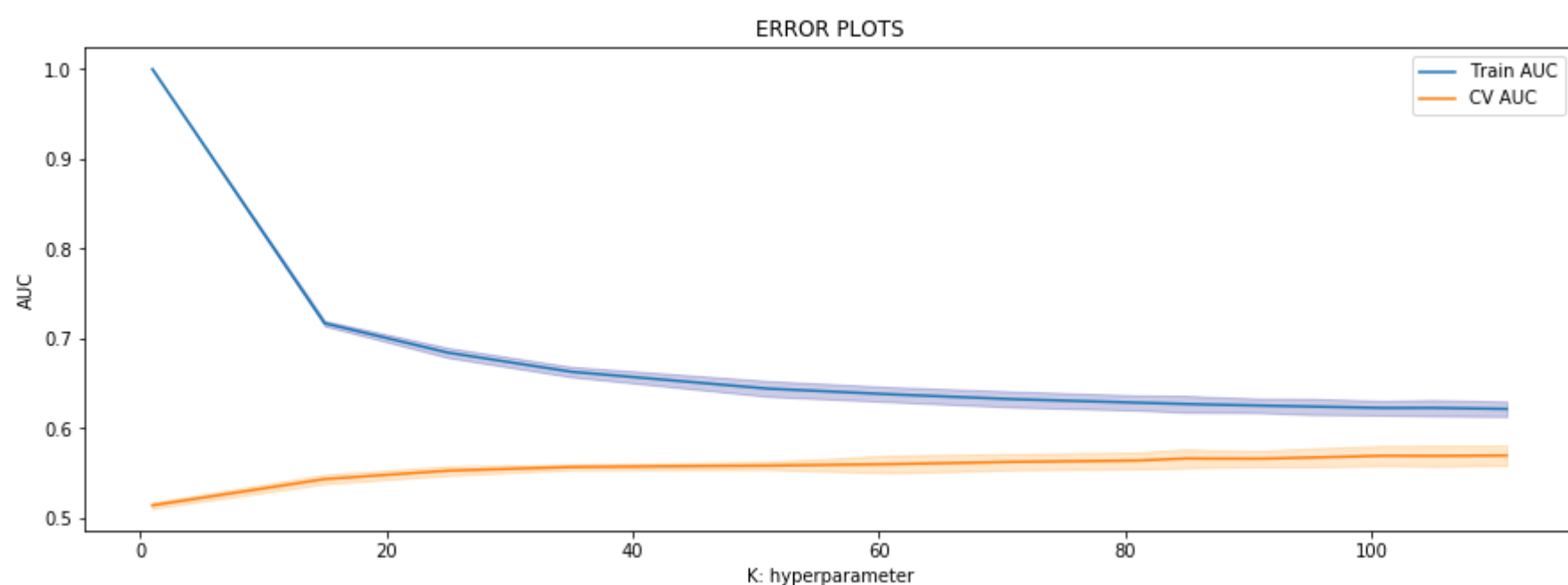
```
In [118]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()
parameters = {'n_neighbors':[1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]}
clf = GridSearchCV(neigh, parameters, cv=3, scoring='roc_auc')
clf.fit(X_train_tfidf, 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=0.2,color='darkblue')

plt.plot(K, cv_auc, label='CV AUC')
this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(K,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



### Using Best K Value – Training the Model

```
In [158]: best_k_tfidf = 91
```

```
In [159]: #https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

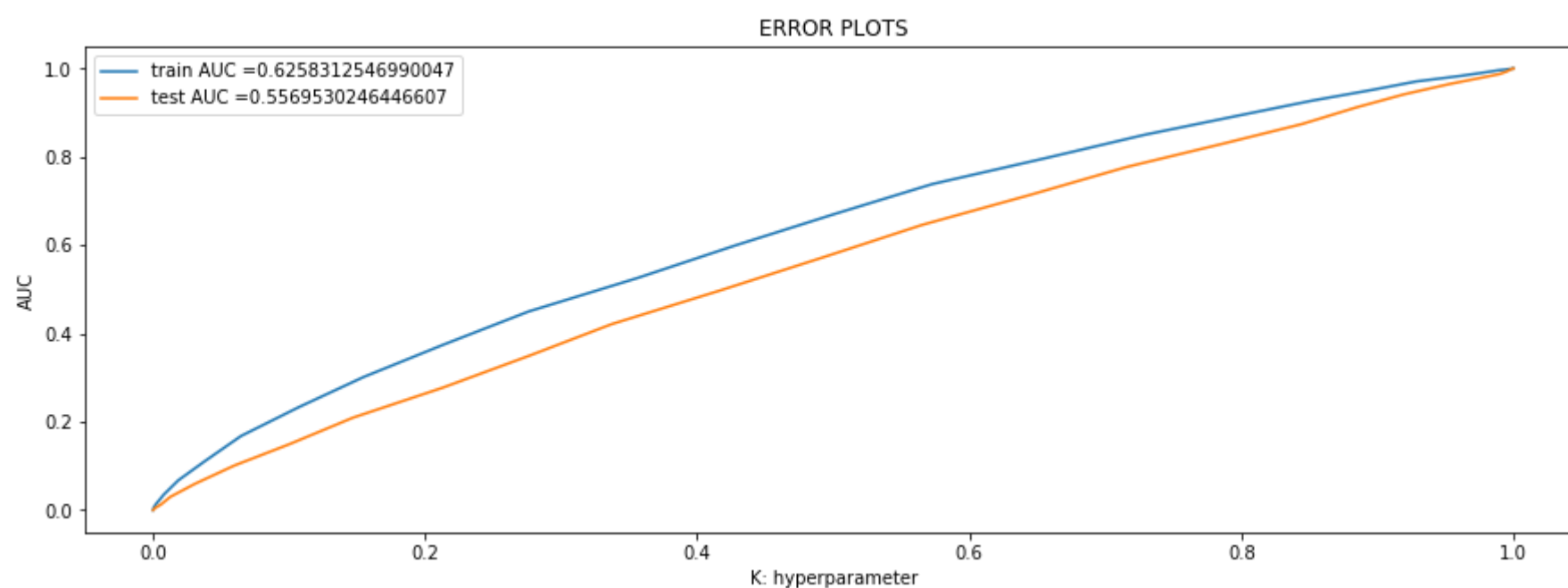
neigh = KNeighborsClassifier(n_neighbors=best_k_tfidf,algorithm= 'brute')
neigh.fit(X_train_tfidf, y_train)

roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
not the predicted outputs

y_train_pred = batch_predict(neigh, X_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="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



## Confusion Matrix

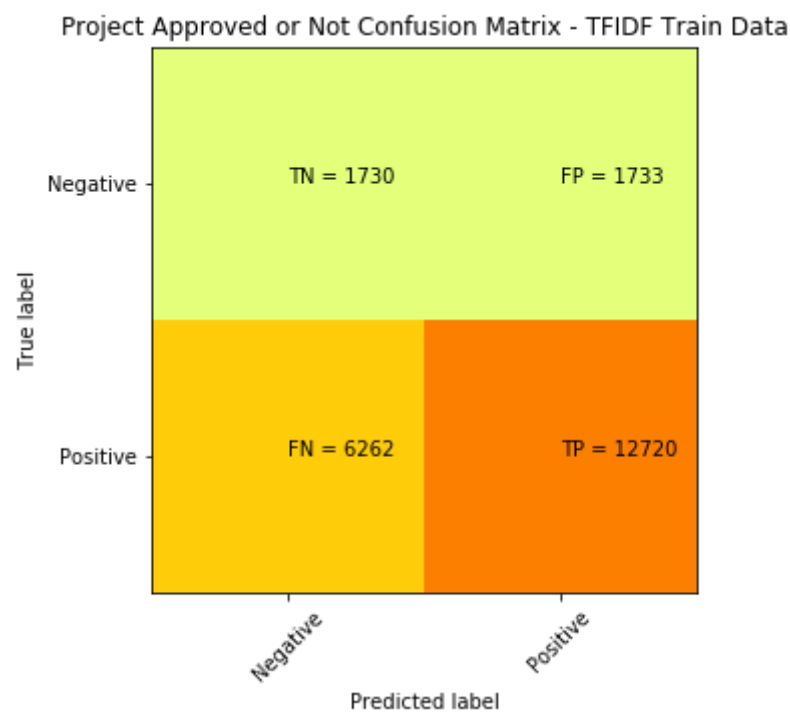
### Train confusion matrix

```
In [160]: from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
tfidf_train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
print(tfidf_train_confusion_matrix)

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999981238068975 for threshold 0.824
[[1730 1733]
 [6262 12720]]
```

In [161]: [#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/](http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/)

```
plt.clf()
plt.imshow(tfidf_train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - TFIDF Train Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(tfidf_train_confusion_matrix[i][j]))
plt.show()
```



### Test confusion matrix

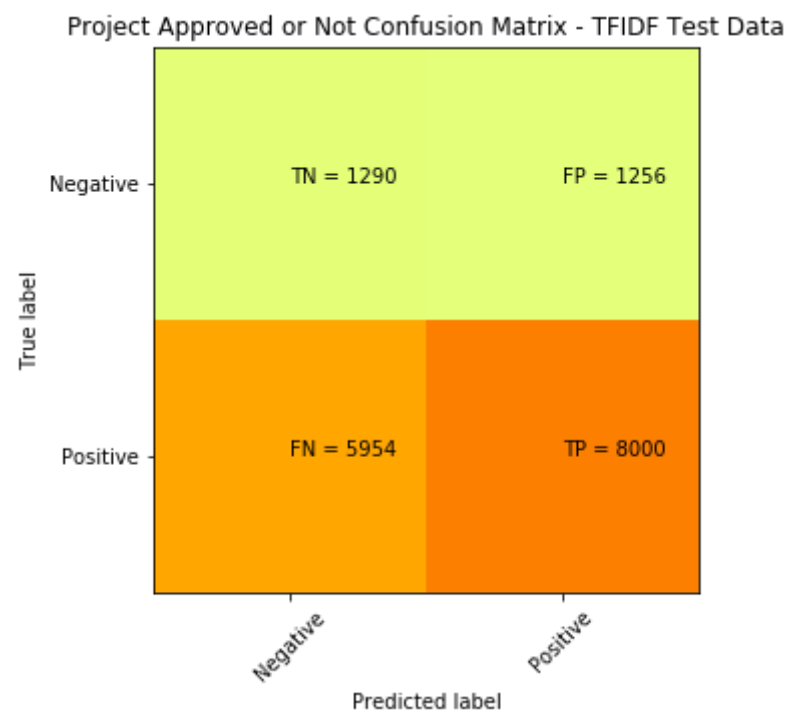
In [162]: 

```
print("Test confusion matrix")
tfidf_test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_fpr))
print(tfidf_test_confusion_matrix)
```

Test confusion matrix  
the maximum value of  $tpr \cdot (1 - fpr)$  0.24995541579323788 for threshold 0.835  
[[1290 1256]  
[5954 8000]]

In [163]: <http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/>

```
plt.clf()
plt.imshow(tfidf_test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - TFIDF Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(tfidf_test_confusion_matrix[i][j]))
plt.show()
```



### 2.4.3 Applying KNN brute force on AVG W2V, SET 3

In [119]: `X_train_avg_w2v = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train , Teacher_Prefix_one_hot_train, project_grade_category_one_hot_train, avg_w2v_essays_train, avg_w2v_title_train, price_train, prev_post_train)).tocsr()`  
`X_train_avg_w2v.shape`

Out[119]: (22445, 701)

In [120]: `X_cv_avg_w2v = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_one_hot_cv , Teacher_Prefix_one_hot_cv, project_grade_category_one_hot_cv, avg_w2v_essays_cv, avg_w2v_title_cv, price_cv, prev_post_cv)).tocsr()`  
`X_cv_avg_w2v.shape`

Out[120]: (11055, 701)

In [121]: `X_test_avg_w2v = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test , Teacher_Prefix_one_hot_test, project_grade_category_one_hot_test, avg_w2v_essays_test, avg_w2v_title_test, price_test, prev_post_test)).tocsr()`  
`X_test_avg_w2v.shape`

Out[121]: (16500, 701)

### Finding the best hyper parameter That maximum AUC value

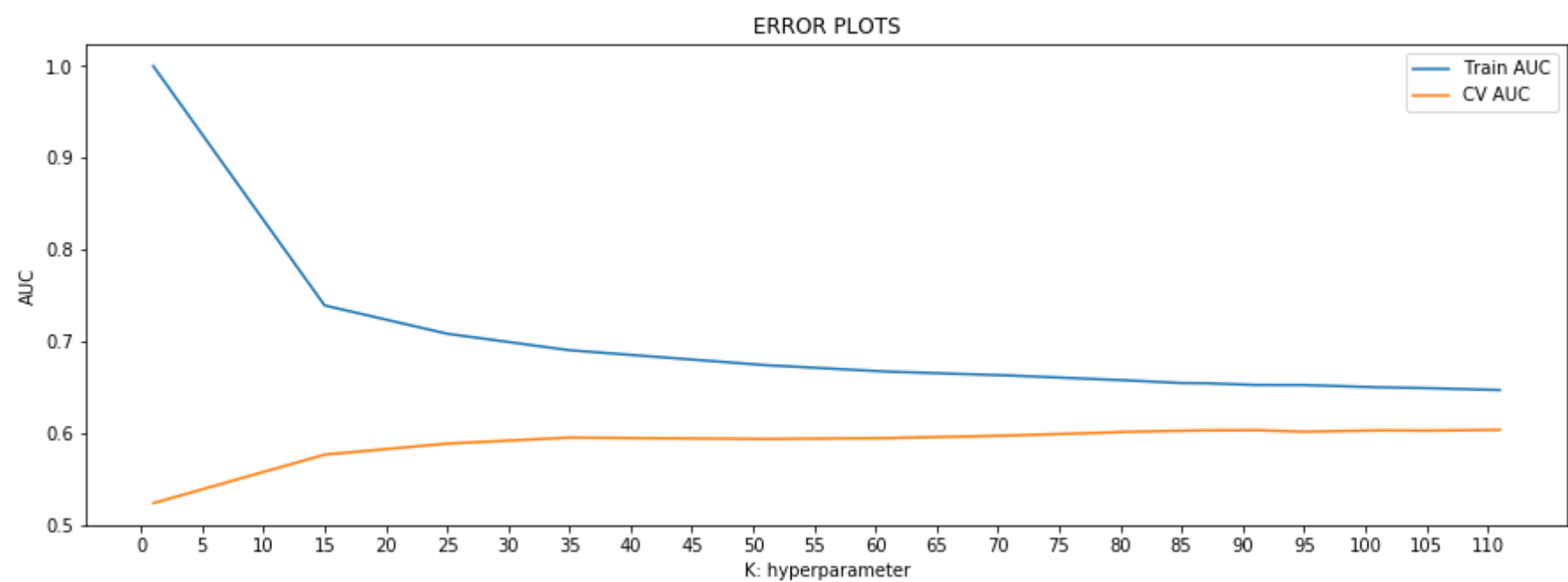
```

In [122]: train_auc = []
cv_auc = []
K = [1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]
for i in K:
 neigh = KNeighborsClassifier(n_neighbors=i,)
 neigh.fit(X_train_avg_w2v, y_train)
 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
 # not the predicted outputs
 y_train_pred = batch_predict(neigh,X_train_avg_w2v)
 y_cv_pred = batch_predict(neigh,X_cv_avg_w2v)

 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.xticks(np.arange(0, 115, step=5))
plt.rcParams["figure.figsize"] = [15,5]
plt.show()

```



**GridSearchCV - Finding the best hyper parameter That maximum AUC value**

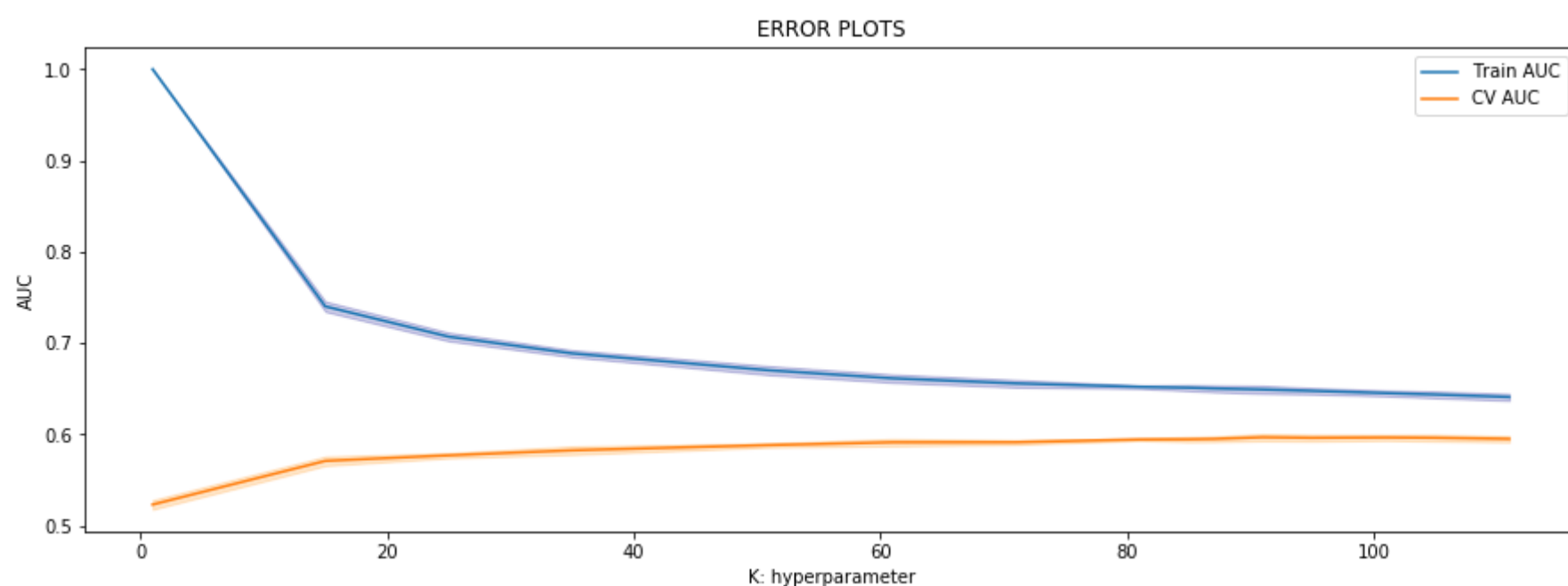
```
In [123]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()
parameters = {'n_neighbors':[1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]}
clf = GridSearchCV(neigh, parameters, cv=3, scoring='roc_auc')
clf.fit(X_train_avg_w2v, 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=0.2,color='darkblue')

plt.plot(K, cv_auc, label='CV AUC')
this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(K,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



### Using Best K Value – Training the Model

```
In [164]: best_k_avg_w2v = 91
```

```
In [165]: #https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
```

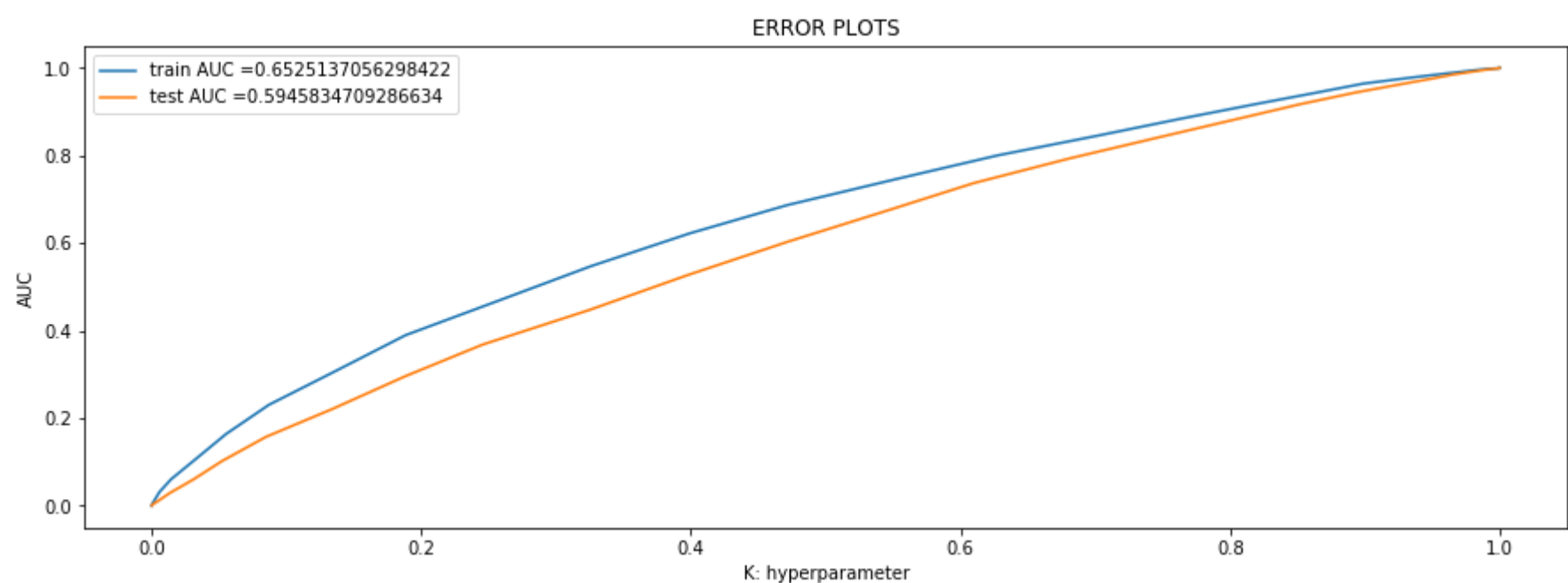
```
neigh = KNeighborsClassifier(n_neighbors=best_k_avg_w2v,algorithm= 'brute')
neigh.fit(X_train_avg_w2v, y_train)

roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_avg_w2v)
y_test_pred = batch_predict(neigh, X_test_avg_w2v)

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="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



## Confusion Matrix

### Train confusion matrix

```
In [166]: from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
avg_w2v_train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
print(avg_w2v_train_confusion_matrix)

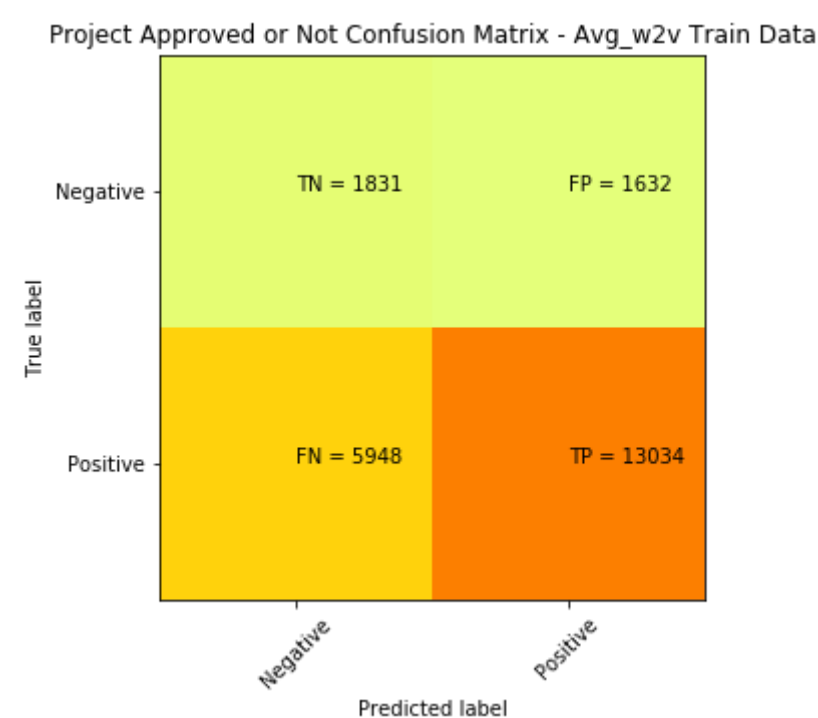
#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(avg_w2v_train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - Avg_w2v Train Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(avg_w2v_train_confusion_matrix[i][j]))
plt.show()
```

Train confusion matrix

the maximum value of  $tpr \cdot (1 - fpr)$  0.2491744541883259 for threshold 0.846

```
[[1831 1632]
 [5948 13034]]
```



**Test confusion matrix**



```

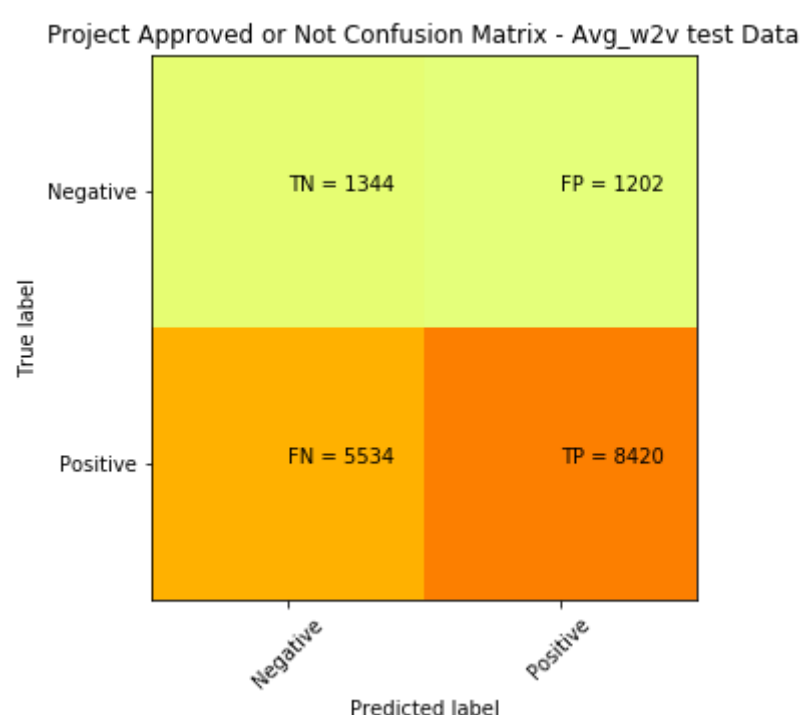
In [167]: from sklearn.metrics import confusion_matrix
print("Test confusion matrix")
avg_w2v_test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_fpr))
print(avg_w2v_test_confusion_matrix)

#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(avg_w2v_test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - Avg_w2v test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(avg_w2v_test_confusion_matrix[i][j]))
plt.show()

```

Test confusion matrix  
the maximum value of tpr\*(1-fpr) 0.24922232184675497 for threshold 0.857  
[[1344 1202]  
[5534 8420]]



#### 2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

```

In [124]: X_train_tfidf_w2v = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train , Teacher_Prefix_one_hot_train, project_grade_category_one_hot_train, tfidf_w2v_essays_train, tfidf_w2v_title_train, price_train, prev_post_train)).tocsr()
X_train_tfidf_w2v.shape

```

Out[124]: (22445, 701)

```

In [125]: X_cv_tfidf_w2v = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_one_hot_cv , Teacher_Prefix_one_hot_cv, project_grade_category_one_hot_cv, tfidf_w2v_essays_cv, tfidf_w2v_title_cv, price_cv, prev_post_cv)).tocsr()
X_cv_tfidf_w2v.shape

```

Out[125]: (11055, 701)

```

In [126]: X_test_tfidf_w2v = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test , Teacher_Prefix_one_hot_test, project_grade_category_one_hot_test, tfidf_w2v_essays_test, tfidf_w2v_title_test, price_test, prev_post_test)).tocsr()
X_test_tfidf_w2v.shape

```

Out[126]: (16500, 701)

#### Finding the best hyper parameter That maximum AUC value

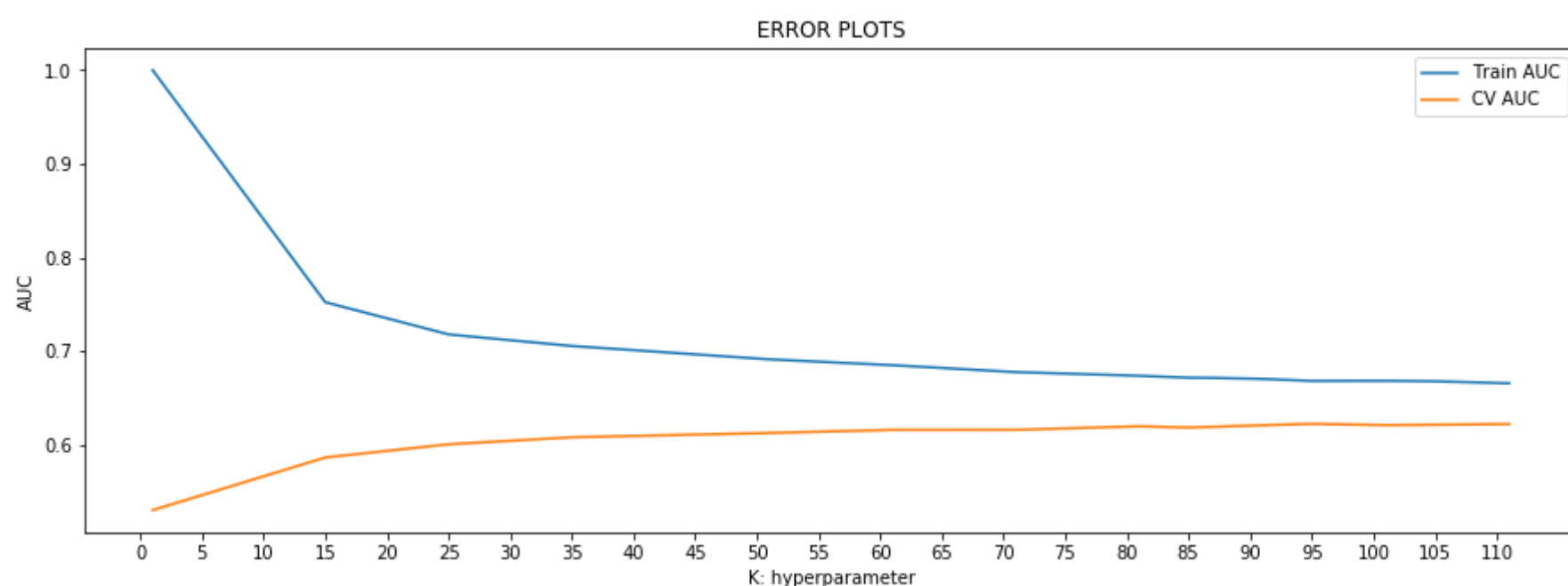
```

In [127]: train_auc = []
cv_auc = []
K = [1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]
for i in K:
 neigh = KNeighborsClassifier(n_neighbors=i,)
 neigh.fit(X_train_tfidf_w2v, y_train)
 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
 # not the predicted outputs
 y_train_pred = batch_predict(neigh,X_train_tfidf_w2v)
 y_cv_pred = batch_predict(neigh,X_cv_tfidf_w2v)

 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.xticks(np.arange(0, 115, step=5))
plt.rcParams["figure.figsize"] = [15,5]
plt.show()

```



**GridSearchCV - Finding the best hyper parameter That maximum AUC value**

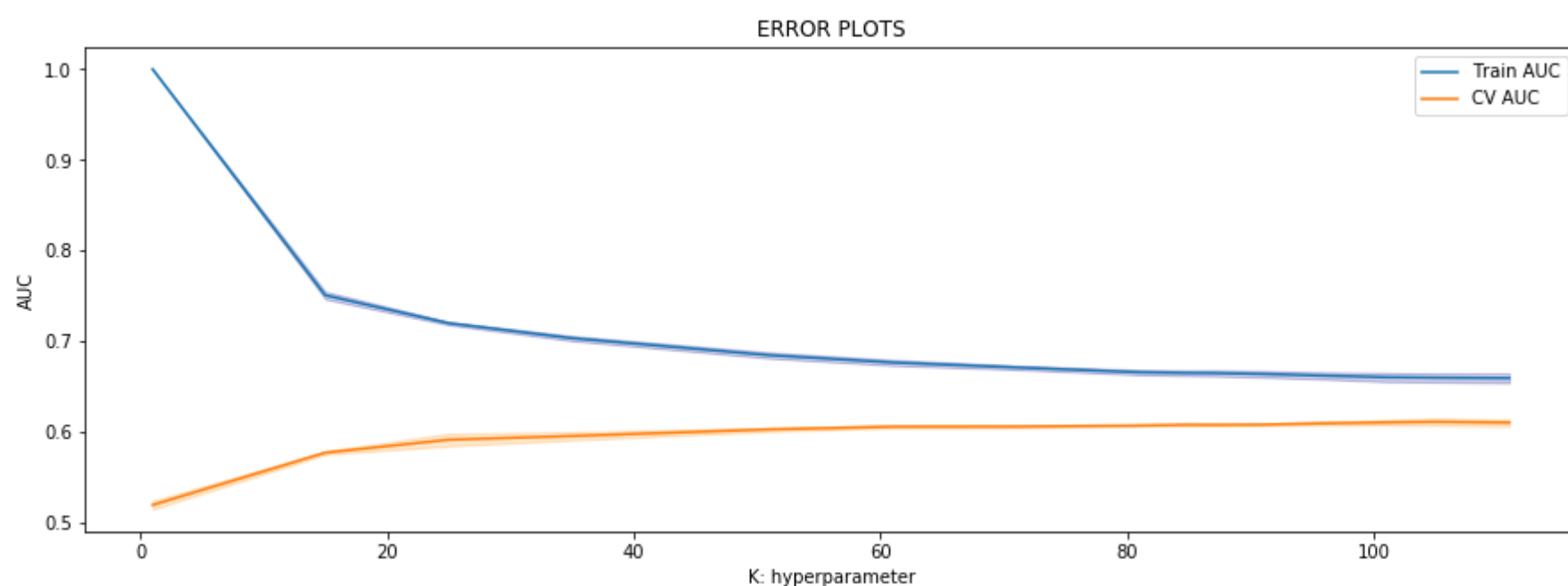
```
In [128]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()
parameters = {'n_neighbors':[1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]}
clf = GridSearchCV(neigh, parameters, cv=3, scoring='roc_auc')
clf.fit(X_train_tfidf_w2v, 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=0.2,color='darkblue')

plt.plot(K, cv_auc, label='CV AUC')
this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(K,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



### Using Best K Value – Training the Model

```
In [168]: best_k_tfidf_w2v = 95
```

```
In [169]: #https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

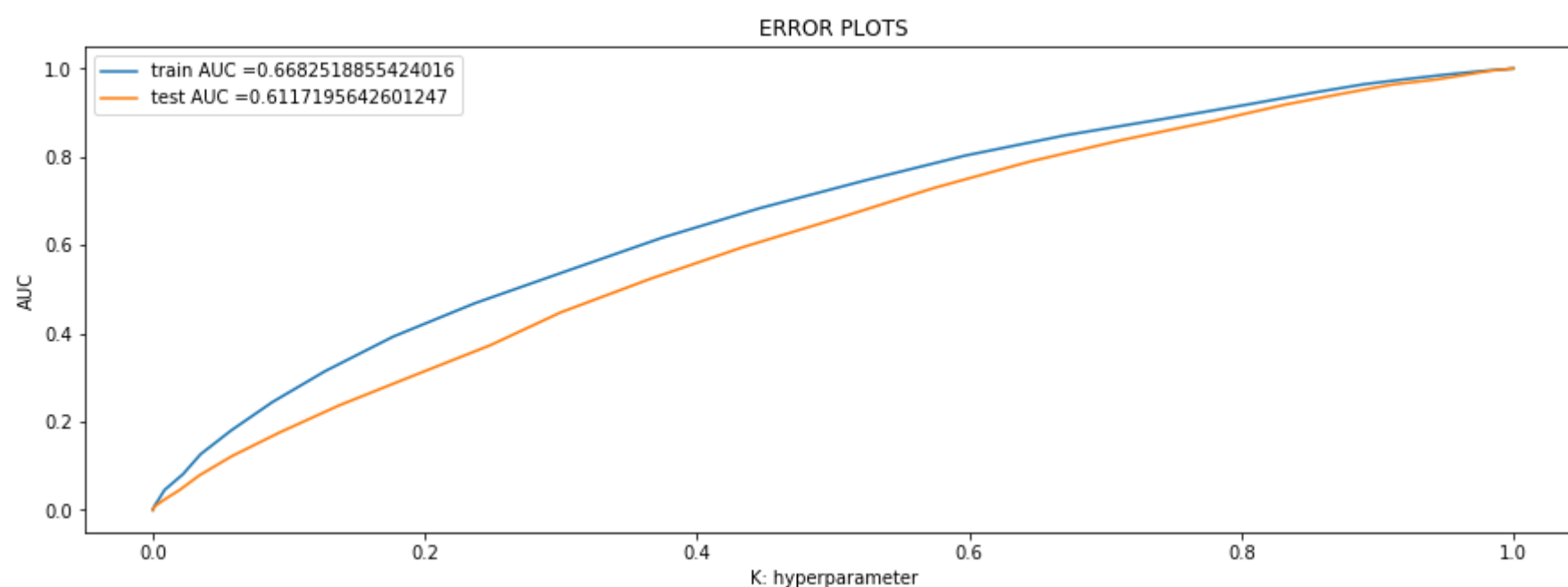
neigh = KNeighborsClassifier(n_neighbors=best_k_tfidf_w2v, algorithm= 'brute')
neigh.fit(X_train_tfidf_w2v, y_train)

roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_tfidf_w2v)
y_test_pred = batch_predict(neigh, X_test_tfidf_w2v)

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="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



## Confusion Matrix

### Train confusion matrix

```
In [170]: from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
tfidf_w2v_train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
print(tfidf_w2v_train_confusion_matrix)

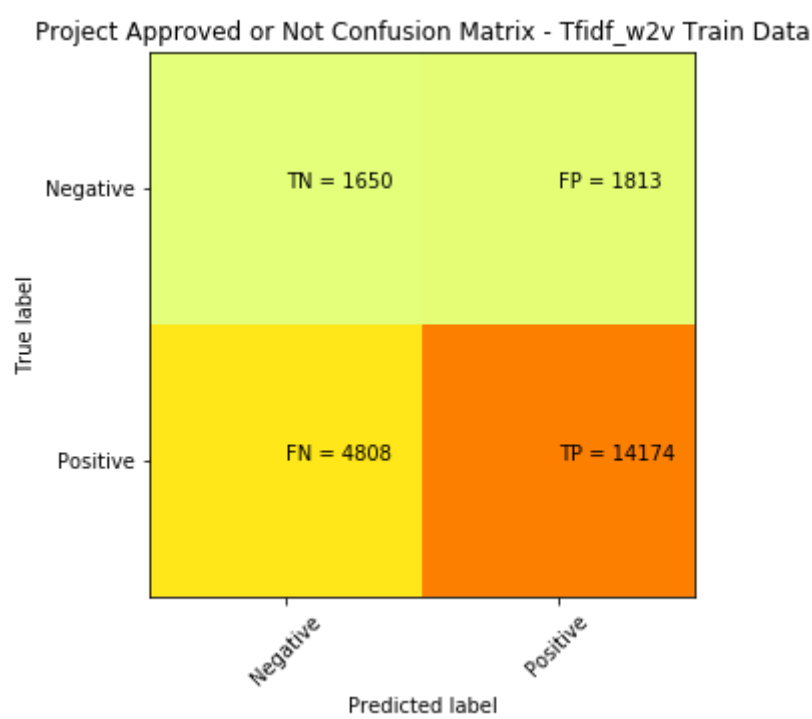
#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(tfidf_w2v_train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - Tfidf_w2v Train Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(tfidf_w2v_train_confusion_matrix[i][j]))
plt.show()
```

Train confusion matrix

the maximum value of  $tpr \cdot (1 - fpr)$  0.24944612694956267 for threshold 0.832

```
[[1650 1813]
 [4808 14174]]
```



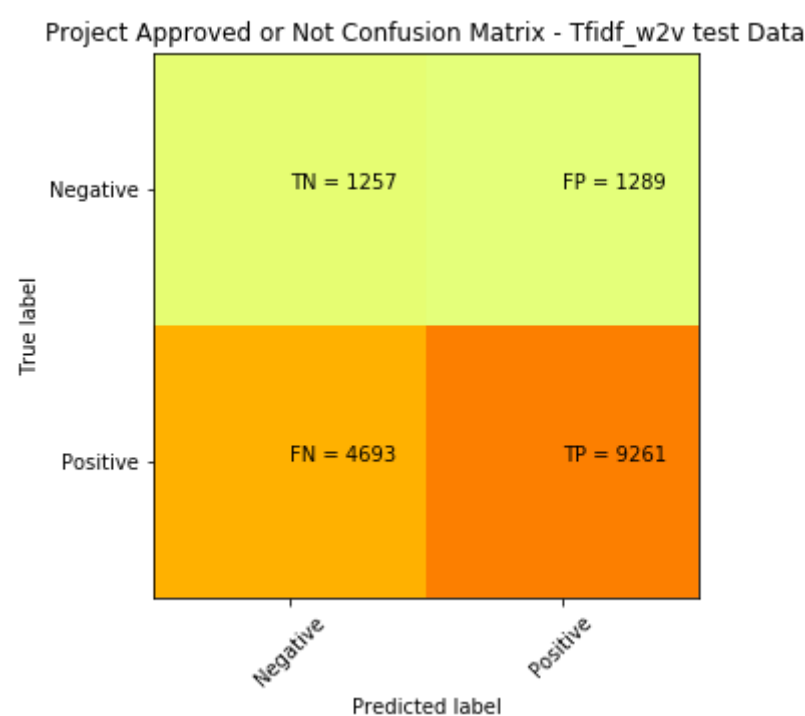
**Test confusion matrix**

```
In [171]: from sklearn.metrics import confusion_matrix
print("Test confusion matrix")
tfidf_w2v_test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_fpr))
print(tfidf_w2v_test_confusion_matrix)

#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(avg_w2v_test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - Tfidf_w2v test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(tfidf_w2v_test_confusion_matrix[i][j]))
plt.show()
```

Test confusion matrix  
the maximum value of  $tpr \cdot (1 - fpr)$  0.2499605067234218 for threshold 0.842  
[[1257 1289]  
[4693 9261]]



## 2.5 Feature selection with `SelectKBest`

```
In [183]: from sklearn.feature_selection import SelectKBest, chi2

X= SelectKBest(chi2, k=2000).fit(X_train_tfidf, y_train)
```

```
In [184]: X_train_tfidf_2000 = X.transform(X_train_tfidf)
X_cv_tfidf_2000 = X.transform(X_cv_tfidf)
X_test_tfidf_2000 = X.transform(X_test_tfidf)
```

**Finding the best hyper parameter That maximum AUC value**

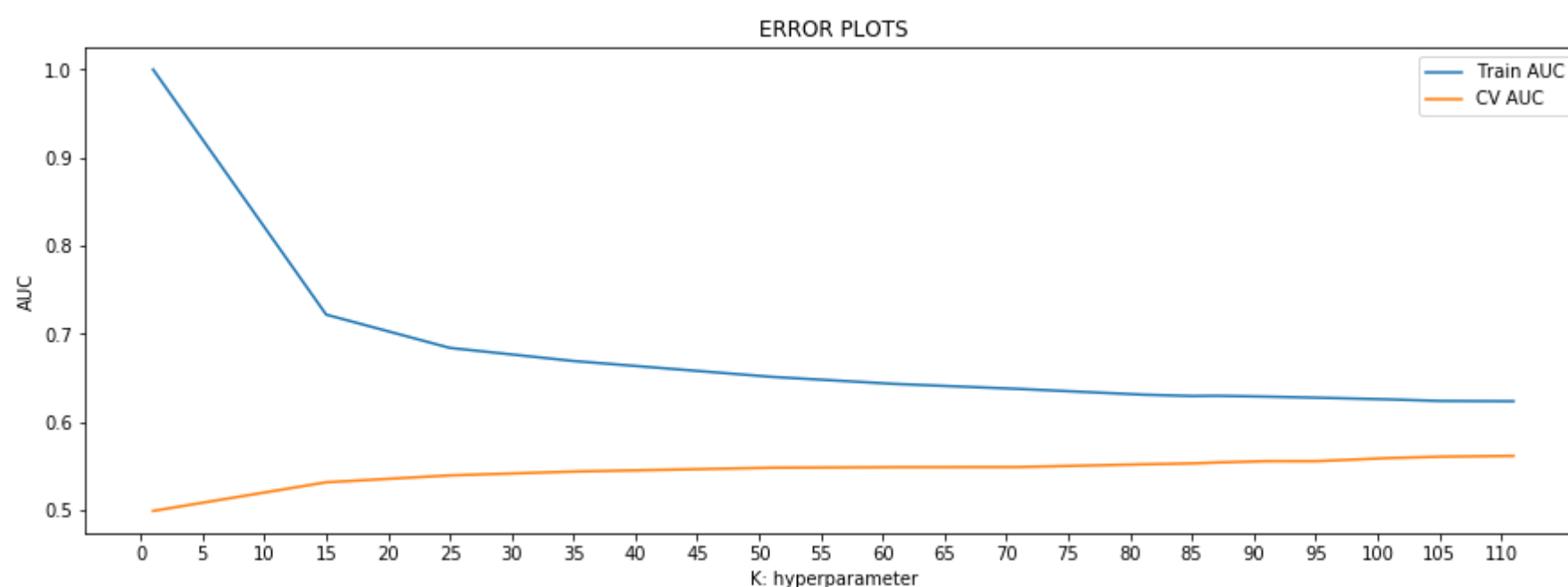
```

In [185]: train_auc = []
cv_auc = []
K = [1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]
for i in K:
 neigh = KNeighborsClassifier(n_neighbors=i,)
 neigh.fit(X_train_tfidf_2000, y_train)
 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
 # not the predicted outputs
 y_train_pred = batch_predict(neigh,X_train_tfidf_2000)
 y_cv_pred = batch_predict(neigh,X_cv_tfidf_2000)

 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.xticks(np.arange(0, 115, step=5))
plt.rcParams["figure.figsize"] = [15,5]
plt.show()

```



**GridSearchCV - Finding the best hyper parameter That maximum AUC value**

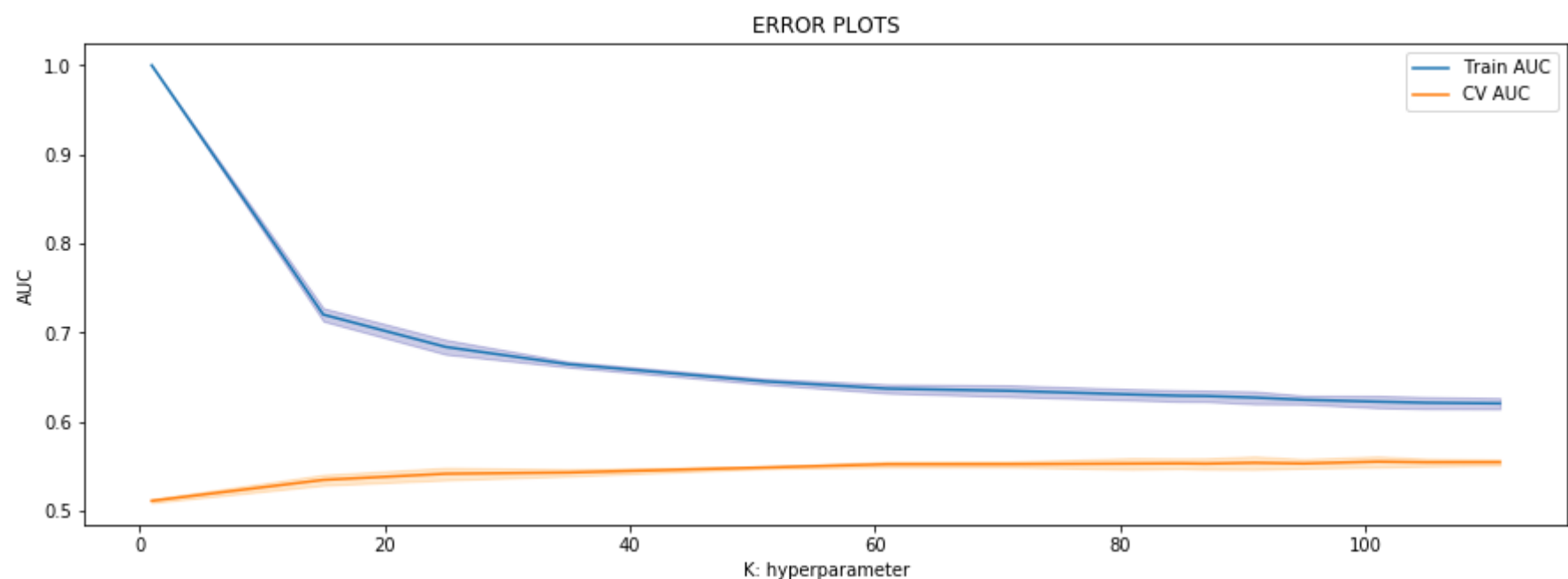
```
In [186]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()
parameters = {'n_neighbors':[1,15,25,35,51,61,71,81,85,87,91,95,101,105,111]}
clf = GridSearchCV(neigh, parameters, cv=3, scoring='roc_auc')
clf.fit(X_train_tfidf_2000, 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=0.2,color='darkblue')

plt.plot(K, cv_auc, label='CV AUC')
this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(K,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



### Using Best K Value – Training the Model

```
In [187]: best_k_tfidf_2000 = 91
```



```
In [188]: #https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

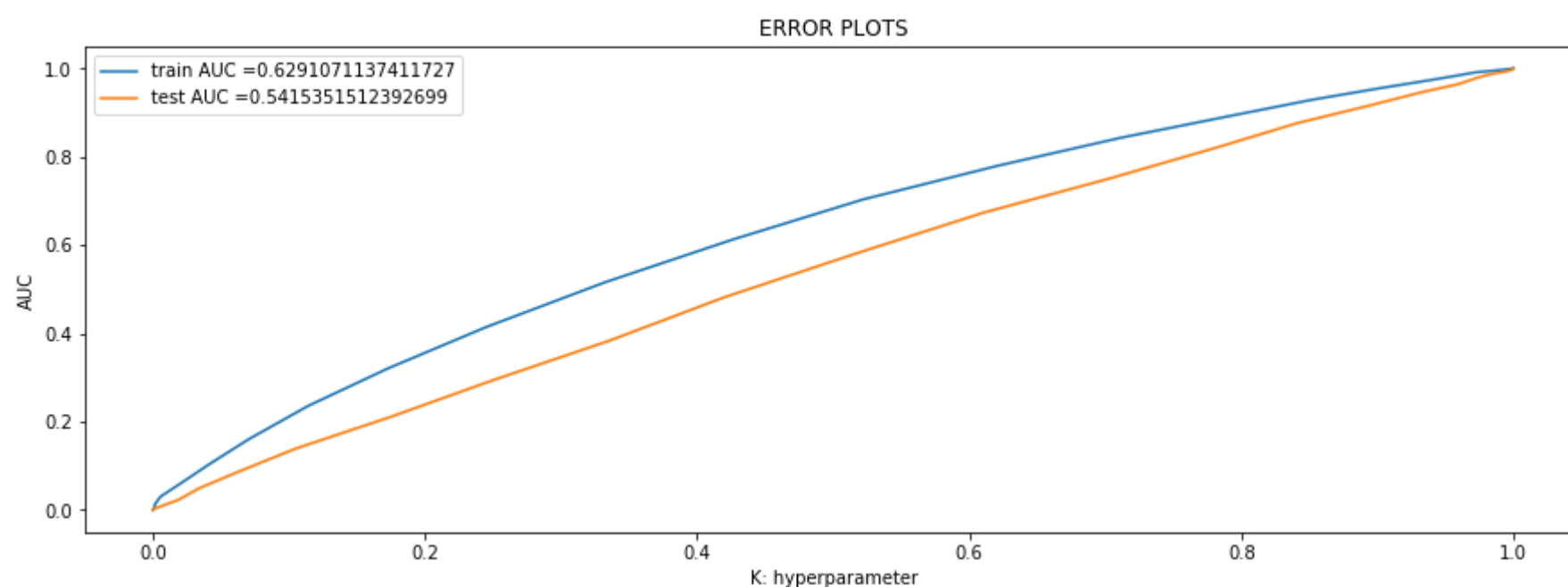
neigh = KNeighborsClassifier(n_neighbors=best_k_tfidf_2000,algorithm= 'brute')
neigh.fit(X_train_tfidf_2000, y_train)

roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
not the predicted outputs

y_train_pred = batch_predict(neigh, X_train_tfidf_2000)
y_test_pred = batch_predict(neigh, X_test_tfidf_2000)

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="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



## Confusion Matrix

### Train confusion matrix

```
In [189]: from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
tfidf_2000_train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
print(tfidf_2000_train_confusion_matrix)

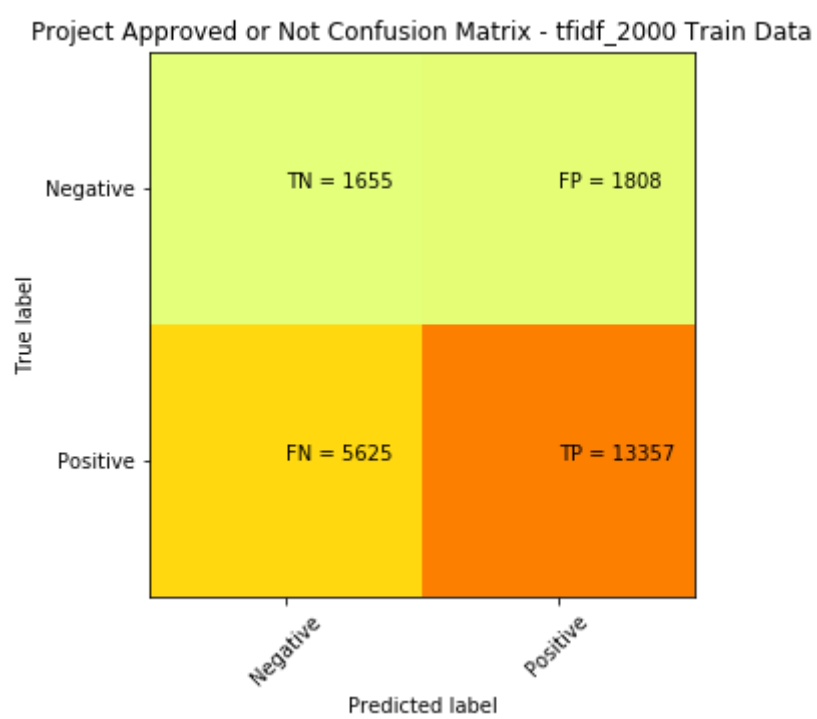
#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(tfidf_2000_train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - tfidf_2000 Train Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j, i, str(s[i][j])+" = "+str(tfidf_2000_train_confusion_matrix[i][j]))
plt.show()
```

Train confusion matrix

the maximum value of  $tpr \cdot (1 - fpr)$  0.24951200217404917 for threshold 0.824

```
[[1655 1808]
 [5625 13357]]
```



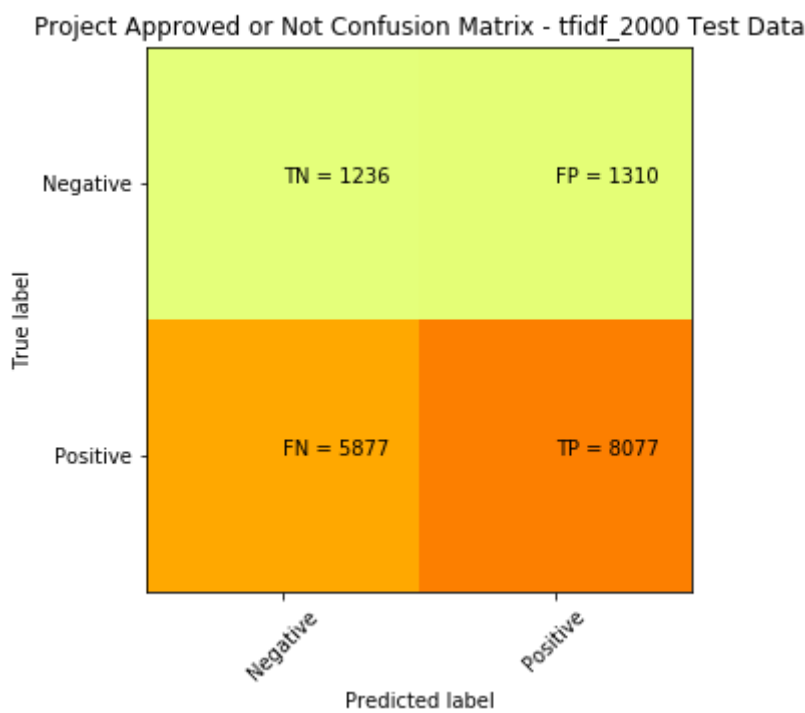
**Test confusion matrix**

```
In [190]: print("Train confusion matrix")
tfidf_2000_test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_fpr))
print(tfidf_2000_test_confusion_matrix)

##http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(tfidf_2000_test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative', 'Positive']
plt.title('Project Approved or Not Confusion Matrix - tfidf_2000 Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN', 'FP'], ['FN', 'TP']]
for i in range(2):
 for j in range(2):
 plt.text(j,i, str(s[i][j])+ " = "+str(tfidf_2000_test_confusion_matrix[i][j]))
plt.show()
```

Train confusion matrix  
the maximum value of tpr\*(1-fpr) 0.24978880353267358 for threshold 0.835  
[[1236 1310]  
[5877 8077]]



3. Conclusions

```
In [192]: from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]
x.add_row(["BOW", "Brute", 95, 0.63])
x.add_row(["TFIDF", "Brute", 91, 0.56])
x.add_row(["AVG W2V", "Brute", 91, 0.59])
x.add_row(["TFIDF W2V", "Brute", 95, 0.61])
x.add_row(["TFIDF", "Top 2000 with Brute ", 91, 0.54])
print(x)
```

| Vectorizer | Model               | Hyper Parameter | AUC  |
|------------|---------------------|-----------------|------|
| BOW        | Brute               | 95              | 0.63 |
| TFIDF      | Brute               | 91              | 0.56 |
| AVG W2V    | Brute               | 91              | 0.59 |
| TFIDF W2V  | Brute               | 95              | 0.61 |
| TFIDF      | Top 2000 with Brute | 91              | 0.54 |