full_knn_analysis

July 14, 2019

1 DonorsChoose

[ ]:
```

```

<p> 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. </p> <p> 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: <ul> <li> 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</li> <li>How to increase the consistency of project vetting across different volunteers to improve the experience for teachers</li> <li>How to focus volunteer time on the applications that need the most assistance</li> </ul> </p> <p> 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. </p>

1.1 About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`

`project_title` | Title of the project. **Examples:**
Art Will Make You Happy!
First Grade Fun
`project_grade_category` | Grade level of students for which the project is targeted. One of the following enumerated values:
Grades PreK-2
Grades 3-5
Grades 6-8
Grades 9-12
`project_subject_categories` | One or more (comma-separated) subject categories for the project from the following enumerated list of values:

Applied Learning
Care & Hunger
Health & Sports
History & Civics
Literacy & Language
Math & Science
Music & The Arts
Special Needs
Warmth
Examples:
Music & The Arts
Literacy & Language, Math & Science
`school_state` | State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories` | One or more (comma-separated) subject subcategories for the project. **Examples:**
Literacy
Literature & Writing, Social Sciences
`project_resource_summary` | An explanation of the resources needed for the project. **Example:**
My students need hands on literacy materials to manage sensory needs!</code
`project_essay_1` | First application essay
`project_essay_2` | Second application essay `project_essay_3` | Third application essay *`project_essay_4` | Fourth application essay* `project_submitted_datetime` | Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id` | A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix` | Teacher's title. One of the following enumerated values:
nan
Dr.
Mr.
Mrs.
Ms.
Teacher.
`teacher_number_of_previously_posted_projects` | Number of project applications previously submitted by the same teacher. **Example:** `2`
* See the section Notes on the Essay Data for more details about these features.
Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

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

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

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

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

1.1.1 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.

1.2 Step 1: Import the necessary Libraries

we will need to import libraries that allow for data analysis and data visualization to get acclimated to the dataset. We will be using pandas, numpy, matplotlib and seaborn to conduct this. Data Exploration libraries

[1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
warnings.filterwarnings(action='ignore', category=UserWarning, module='gensim')
warnings.filterwarnings("ignore",'detected Windows; aliasing chunkize to␣
 ↪chunkize_serial')
warnings.filterwarnings("ignore", message="numpy.dtype size changed")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.3 Step 2: Read in the dataset.

We will use the pandas .read_csv() method to read in the dataset. Then we will use the. head() method to observe the first few rows of the data, to understand the information better. In our case, the feature(column) headers tell us pretty little. This is fine because we are merely trying to gain insight via classifying new data points by referencing it's neighboring elements.

1.4 1.1 Reading Data

[2]:
```
project_data = pd.read_csv("C:\\VipinML\\Assignment␣
 ↪2\\Assignments_DonorsChoose_2018\\train_data.csv")
resource_data = pd.read_csv("C:\\VipinML\Assignment␣
 ↪2\\Assignments_DonorsChoose_2018\\resources.csv")
#Limit the data for testing purpose since processing takes few hours for full␣
 ↪set..

project_data = project_data.head(40000)
```

```
resource_data = resource_data.head(40000)
```

[3]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (40000, 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']
```

[4]:
```
# 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(1)
```

[4]:
```
     Unnamed: 0       id                        teacher_id teacher_prefix  \
473      100660  p234804  cbc0e38f522143b86d372f8b43d4cff3           Mrs.

    school_state                Date project_grade_category  \
473           GA 2016-04-27 00:53:00          Grades PreK-2

    project_subject_categories project_subject_subcategories  \
473           Applied Learning             Early Development

                                 project_title  \
473  Flexible Seating for Flexible Learning

                                       project_essay_1  \
473  I recently read an article about giving studen...

                                       project_essay_2  \
473  I teach at a low-income (Title 1) school. Ever...

                                        project_essay_3  \
473  We need a classroom rug that we can use as a c...

                                        project_essay_4  \
473  Benjamin Franklin once said, \"Tell me and I f...

                              project_resource_summary  \
473  My students need flexible seating in the class...

     teacher_number_of_previously_posted_projects  project_is_approved
473                                             2                    1
```

1.5 1.2 preprocessing of `project_subject_categories`

[5]:
```python
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.
 ↪com/a/47301924/4084039

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

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]))
```

1.6 1.3 preprocessing of `project_subject_subcategories`

[6]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.
 ↪com/a/47301924/4084039

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science",␣
 ↪"Warmth", "Care & 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 placeing all the ' '(space) with␣
 ↪''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the␣
 ↪trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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]))
```

1.7 1.3 Text preprocessing

[7]:
```
# 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)
```

[8]:
```
project_data.head(1)
```

[8]:
```
     Unnamed: 0       id                        teacher_id teacher_prefix  \
473      100660  p234804  cbc0e38f522143b86d372f8b43d4cff3           Mrs.

    school_state                Date project_grade_category  \
473           GA 2016-04-27 00:53:00          Grades PreK-2

                              project_title  \
473  Flexible Seating for Flexible Learning

                                        project_essay_1  \
473  I recently read an article about giving studen...

                                        project_essay_2  \
473  I teach at a low-income (Title 1) school. Ever...

                                        project_essay_3  \
473  We need a classroom rug that we can use as a c...

                                        project_essay_4  \
473  Benjamin Franklin once said, \"Tell me and I f...

                               project_resource_summary  \
473  My students need flexible seating in the class...

     teacher_number_of_previously_posted_projects  project_is_approved  \
473                                             2                    1

    clean_categories clean_subcategories  \
473  AppliedLearning    EarlyDevelopment

                                                  essay
473  I recently read an article about giving studen...
```

[9]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

[10]:
```
# 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
```

[11]:
```
sent = decontracted(project_data['essay'].values[500])
print(sent[1:200])
print("="*100)
```

```
tudents in my math classes work collaboratively to problem solve. They struggle
together and succeed together as they work as a team, applying math and science
concepts they have learned over the yea
================================================================================
====================
```

[12]:
```
# \r \n \t remove from string python: http://texthandler.com/info/
  ↪remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent[1:200])
print(sent[1:200])
```

```
tudents in my math classes work collaboratively to problem solve. They struggle
together and succeed together as they work as a team, applying math and science
concepts they have learned over the yea
tudents in my math classes work collaboratively to problem solve. They struggle
together and succeed together as they work as a team, applying math and science
concepts they have learned over the yea
```

[13]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent[1:200])
```

```
tudents in my math classes work collaboratively to problem solve They struggle
together and succeed together as they work as a team applying math and science
concepts they have learned over the years
```

[14]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',␣
 ↪"you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',␣
 ↪'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',␣
 ↪'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',␣
 ↪'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',␣
 ↪'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',␣
 ↪'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',␣
 ↪'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',␣
 ↪'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',␣
 ↪'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so',␣
 ↪'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should',␣
 ↪"should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',␣
 ↪"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",␣
 ↪'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',␣
 ↪"shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

1.7.1 1.4.3 Merging price with project_data

[15]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).
 ↪reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
print (price_data[1:3])
project_data.head(1)
```

```
        id   price  quantity
1  p000147   13.13        25
```

```
   2  p000157  3508.32         9
```

[15]:
```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0      100660  p234804  cbc0e38f522143b86d372f8b43d4cff3           Mrs.

  school_state                Date project_grade_category  \
0           GA 2016-04-27 00:53:00          Grades PreK-2

                         project_title  \
0  Flexible Seating for Flexible Learning

                                     project_essay_1  \
0  I recently read an article about giving studen...

                                     project_essay_2  \
0  I teach at a low-income (Title 1) school. Ever...

                                     project_essay_3  \
0  We need a classroom rug that we can use as a c...

                                     project_essay_4  \
0  Benjamin Franklin once said, \"Tell me and I f...

                            project_resource_summary  \
0  My students need flexible seating in the class...

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             2                    1

  clean_categories clean_subcategories  \
0  AppliedLearning    EarlyDevelopment

                                               essay   price  quantity
0  I recently read an article about giving studen...  481.04       9.0
```

[16]:
```
#Convert NaN value to mean of the column
project_data.fillna(project_data.mean(), inplace=True)
project_data.head(1)
```

[16]:
```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0      100660  p234804  cbc0e38f522143b86d372f8b43d4cff3           Mrs.

  school_state                Date project_grade_category  \
0           GA 2016-04-27 00:53:00          Grades PreK-2

                         project_title  \
0  Flexible Seating for Flexible Learning
```

```
                                     project_essay_1  \
0  I recently read an article about giving studen...

                                     project_essay_2  \
0  I teach at a low-income (Title 1) school. Ever...

                                     project_essay_3  \
0  We need a classroom rug that we can use as a c...

                                     project_essay_4  \
0  Benjamin Franklin once said, \"Tell me and I f...

                            project_resource_summary  \
0  My students need flexible seating in the class...

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             2                    1

  clean_categories clean_subcategories  \
0  AppliedLearning    EarlyDevelopment

                                               essay   price  quantity
0  I recently read an article about giving studen...  481.04       9.0
```

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

[17]:
```python
y = project_data['project_is_approved'].values
X = project_data.drop(['project_is_approved'], axis=1)

# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
 ↪stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.
 ↪33, stratify=y_train)
```

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

```
100%|| 17956/17956 [00:08<00:00, 2043.49it/s]
```

[19]:
```
# Combining all the above stundents
from tqdm import tqdm
X_test_preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    X_test_preprocessed_essays.append(sent.lower().strip())
```

```
100%|| 13200/13200 [00:06<00:00, 1975.09it/s]
```

[20]:
```
# Combining all the above stundents
from tqdm import tqdm
X_cv_preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    X_cv_preprocessed_essays.append(sent.lower().strip())
```

```
100%|| 8844/8844 [00:04<00:00, 2040.35it/s]
```

1.9 Step 3: Standardize (normalize) the data scale to prep for KNN algorithm.

Because the distance between pairs of points plays a critical part on the classification, it is necessary to normalize the data This will generate an array of values. Again, KNN depends on the distance between each feature. Please see Section 1 for all normalization.

1.9.1 1.4.1 Vectorizing Categorical data

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

1.9.2 Vectorization of clean_categories for X_train,X_test, X_cv

[21]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=100,␣
 ↪vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
X_train_categories_one_hot = vectorizer.transform(X_train['clean_categories'].
 ↪values)
X_test_categories_one_hot = vectorizer.transform(X_test['clean_categories'].
 ↪values)
X_cv_categories_one_hot = vectorizer.transform(X_cv['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix X_train_categories_one_hot  after one hot encodig␣
 ↪",X_train_categories_one_hot.shape)
print("Shape of matrix X_test_categories_one_hot after one hot encodig␣
 ↪",X_test_categories_one_hot.shape)
print("Shape of matrix X_cv_categories_one_hot after one hot encodig␣
 ↪",X_cv_categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix X_train_categories_one_hot  after one hot encodig  (17956, 9)
Shape of matrix X_test_categories_one_hot after one hot encodig  (13200, 9)
Shape of matrix X_cv_categories_one_hot after one hot encodig  (8844, 9)
```

1.9.3 Vectorization of clean_subcategories for X_train,X_test, X_cv

[22]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4),␣
 ↪max_features=100,vocabulary=list(sorted_sub_cat_dict.keys()),␣
 ↪lowercase=False, binary=True)
X_train_sub_categories_one_hot = vectorizer.
 ↪transform(X_train['clean_subcategories'].values)
X_test_sub_categories_one_hot = vectorizer.
 ↪transform(X_test['clean_subcategories'].values)
X_cv_sub_categories_one_hot = vectorizer.transform(X_cv['clean_subcategories'].
 ↪values)
print(vectorizer.get_feature_names())
print("Shape of matrix X_train_sub_categories_one_hot  after one hot encodig␣
 ↪",X_train_sub_categories_one_hot.shape)
```

```
print("Shape of matrix X_test_sub_categories_one_hot after␣
 ↪oneX_test_sub_categories_one_hot  hot encodig␣
 ↪",X_test_sub_categories_one_hot.shape)
print("Shape of matrixX_cv_sub_categories_one_hot after one hot encodig␣
 ↪",X_cv_sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'ForeignLanguages', 'Civics_Government',
'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences',
'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience',
'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts',
'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing',
'Mathematics', 'Literacy']
Shape of matrix X_train_sub_categories_one_hot  after one hot encodig  (17956,
30)
Shape of matrix X_test_sub_categories_one_hot after
oneX_test_sub_categories_one_hot  hot encodig  (13200, 30)
Shape of matrixX_cv_sub_categories_one_hot after one hot encodig  (8844, 30)
```

[23]:
```
# you can do the similar thing with state, teacher_prefix and␣
 ↪project_grade_category also
```

1.9.4 1.4.2 Vectorizing Text data

[24]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/
 ↪how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('C:\\VipinML\\InputData\\glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

1.9.5 Vectorization of preprocessed_essays for X_train,X_test, X_cv

[25]:
```
# average Word2Vec
# compute average word2vec for each review.
X_train_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored␣
 ↪in this list
for sentence in tqdm(X_train_preprocessed_essays): # 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
    X_train_avg_w2v_vectors.append(vector)

print(len(X_train_avg_w2v_vectors))
print(len(X_train_avg_w2v_vectors[0]))
```

```
100%|| 17956/17956 [00:04<00:00, 3693.72it/s]

17956
300
```

[26]:
```
# average Word2Vec
# compute average word2vec for each review.
X_test_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored␣
 ↪in this list
for sentence in tqdm(X_test_preprocessed_essays): # 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
    X_test_avg_w2v_vectors.append(vector)

print(len(X_test_avg_w2v_vectors))
print(len(X_test_avg_w2v_vectors[0]))
```

```
100%|| 13200/13200 [00:03<00:00, 3563.13it/s]

13200
300
```

[27]:
```
# average Word2Vec
# compute average word2vec for each review.
X_cv_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in␣
 ↪this list
for sentence in tqdm(X_cv_preprocessed_essays): # 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
    X_cv_avg_w2v_vectors.append(vector)

print(len(X_cv_avg_w2v_vectors))
print(len(X_cv_avg_w2v_vectors[0]))
```

```
100%|| 8844/8844 [00:02<00:00, 3738.85it/s]

8844
300
```

1.9.6 Vectorization of teacher_prefix for X_train,X_test, X_cv

[28]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4),␣
 ↪max_features=100,vocabulary=list(sorted_cat_dict.keys()), lowercase=False,␣
 ↪binary=True)
X_train_teacher_prefix_data = X_train['teacher_prefix']
X_train_teacher_prefix_data.fillna("Mrs.", inplace = True)
teacher_prefix_notnull = X_train_teacher_prefix_data[pd.
 ↪notnull(X_train_teacher_prefix_data)]
vectorizer.fit(teacher_prefix_notnull.values)
print(vectorizer.get_feature_names())
X_train_teacher_prefix_one_hot = vectorizer.transform(teacher_prefix_notnull.
 ↪values)
print("Shape of matrix after one hot encodig ",X_train_teacher_prefix_one_hot.
 ↪shape)
print (X_train_teacher_prefix_one_hot)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (17956, 9)
```

[29]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4),␣
 ↪max_features=100,vocabulary=list(sorted_cat_dict.keys()), lowercase=False,␣
 ↪binary=True)
X_test_teacher_prefix_data = X_test['teacher_prefix']
X_test_teacher_prefix_data.fillna("Mrs.", inplace = True)
teacher_prefix_notnull = X_test_teacher_prefix_data[pd.
 ↪notnull(X_test_teacher_prefix_data)]
vectorizer.fit(teacher_prefix_notnull.values)
print(vectorizer.get_feature_names())
```

```
X_test_teacher_prefix_one_hot = vectorizer.transform(teacher_prefix_notnull.
 ↪values)
print("Shape of matrix after one hot encodig ",X_test_teacher_prefix_one_hot.
 ↪shape)
print (X_test_teacher_prefix_one_hot)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (13200, 9)
```

[30]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4),␣
 ↪max_features=100,vocabulary=list(sorted_cat_dict.keys()), lowercase=False,␣
 ↪binary=True)
X_cv_teacher_prefix_data = X_cv['teacher_prefix']
X_cv_teacher_prefix_data.fillna("Mrs.", inplace = True)
teacher_prefix_notnull = X_cv_teacher_prefix_data[pd.
 ↪notnull(X_cv_teacher_prefix_data)]
vectorizer.fit(teacher_prefix_notnull.values)
print(vectorizer.get_feature_names())
X_cv_teacher_prefix_one_hot = vectorizer.transform(teacher_prefix_notnull.
 ↪values)
print("Shape of matrix after one hot encodig ",X_cv_teacher_prefix_one_hot.
 ↪shape)
print (X_cv_teacher_prefix_one_hot)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (8844, 9)
```

1.9.7 Vectorization of price for X_train,X_test, X_cv

[31]:
```
X_train.head(1)
X_test.head(1)
X_cv.head(1)
```

[31]:
```
       Unnamed: 0       id                        teacher_id teacher_prefix  \
24777       24805  p242967  4ded3e9cc69f0d4559c8f8dcb3d1599a           Mrs.

      school_state                 Date project_grade_category  \
24777           WY 2016-11-16 10:54:07            Grades 9-12

                          project_title  \
24777  Perceiving Patterns in Painting
```

```
                                        project_essay_1  \
24777  My students have a strong pride in their schoo...

                                        project_essay_2 project_essay_3  \
24777  In the words of Isaiah Berlin, \"To understand...             NaN

      project_essay_4                         project_resource_summary  \
24777             NaN  My students need a Cricut cutting tool to crea...

       teacher_number_of_previously_posted_projects clean_categories  \
24777                                             3       Music_Arts

      clean_subcategories                                             essay  \
24777          VisualArts  My students have a strong pride in their schoo...

            price   quantity
24777  282.226627  17.967552
```

[32]:
```python
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

#normalizer.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))
X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print("="*100)
```

```
After vectorizations
(17956, 1) (17956,)
(13200, 1) (13200,)
(8844, 1) (8844,)
================================================================================
====================
```

1.10 Bag of words of preprocessed_essays for X_train,X_test, X_cv

[33]:
```
# We are considering only the words which appeared in at least 10␣
↪documents(rows or projects).
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4))
X_train_text_bow = vectorizer.fit_transform(X_train_preprocessed_essays)
X_test_text_bow = vectorizer.transform(X_test_preprocessed_essays)
X_cv_text_bow = vectorizer.transform(X_cv_preprocessed_essays)
print("Shape of matrix X_train_text_bow after one hot encodig␣
↪",X_train_text_bow.shape)
print("Shape of matrix X_test_text_bow after one hot encodig ",X_test_text_bow.
↪shape)
print("Shape of matrix X_cv_text_bow after one hot encodig ",X_cv_text_bow.
↪shape)
```

```
Shape of matrix X_train_text_bow after one hot encodig  (17956, 56706)
Shape of matrix X_test_text_bow after one hot encodig  (13200, 56706)
Shape of matrix X_cv_text_bow after one hot encodig  (8844, 56706)
```

1.11 Bag of words of project_title for X_train,X_test, X_cv

[34]:
```
# PROJECT_TITLE BOW
# We are considering only the words which appeared in at least 10␣
↪documents(rows or projects).
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4))
X_train_project_title_bow = vectorizer.fit_transform(X_train['project_title'])
X_test_project_title_bow = vectorizer.transform(X_test['project_title'])
X_cv_project_title_bow = vectorizer.transform(X_cv['project_title'])
print("Shape of matrix X_train_project_title_bow after one hot encodig␣
↪",X_train_project_title_bow .shape)
print("Shape of matrix X_test_project_title_bow after one hot encodig␣
↪",X_test_project_title_bow .shape)
print("Shape of matrix X_cv_project_title_bow after one hot encodig␣
↪",X_cv_project_title_bow .shape)
```

```
Shape of matrix X_train_project_title_bow after one hot encodig  (17956, 2135)
Shape of matrix X_test_project_title_bow after one hot encodig  (13200, 2135)
Shape of matrix X_cv_project_title_bow after one hot encodig  (8844, 2135)
```

1.12 TFIDF of preprocessed_essays for X_train,X_test, X_cv

[35]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train_preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
X_train_tfidf_words = set(tfidf_model.get_feature_names())
```

```
[36]: tfidf_model = TfidfVectorizer()
      tfidf_model.fit(X_test_preprocessed_essays)
      # we are converting a dictionary with word as a key, and the idf as a value
      dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
      X_test_tfidf_words = set(tfidf_model.get_feature_names())
```

```
[37]: tfidf_model = TfidfVectorizer()
      tfidf_model.fit(X_cv_preprocessed_essays)
      # we are converting a dictionary with word as a key, and the idf as a value
      dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
      X_cv_tfidf_words = set(tfidf_model.get_feature_names())
```

1.13 TFIDF of preprocessed_essays for X_train,X_test, X_cv

```
[38]: from sklearn.feature_extraction.text import TfidfVectorizer
      vectorizer = TfidfVectorizer()
      X_train_text_tfidf = vectorizer.fit_transform(X_train_preprocessed_essays)
      X_test_text_tfidf = vectorizer.transform(X_test_preprocessed_essays)
      X_cv_text_tfidf = vectorizer.transform(X_cv_preprocessed_essays)
      print("Shape of matrix X_train_text_tfidf after one hot encodig␣
       ↪",X_train_text_tfidf.shape)
      print("Shape of matrix X_test_text_tfidf after one hot encodig␣
       ↪",X_test_text_tfidf.shape)
      print("Shape of matrix X_cv_text_tfidf after one hot encodig ",X_cv_text_tfidf.
       ↪shape)
```

```
Shape of matrix X_train_text_tfidf after one hot encodig  (17956, 28182)
Shape of matrix X_test_text_tfidf after one hot encodig  (13200, 28182)
Shape of matrix X_cv_text_tfidf after one hot encodig  (8844, 28182)
```

1.14 TFIDF of Project Title for X_train,X_test, X_cv

```
[39]: from sklearn.feature_extraction.text import TfidfVectorizer
      vectorizer = TfidfVectorizer(min_df=10)
      X_train_project_title_tfidf = vectorizer.
       ↪fit_transform((X_train['project_title']))
      X_test_project_title_tfidf = vectorizer.transform((X_test['project_title']))
      X_cv_project_title_tfidf = vectorizer.transform((X_cv['project_title']))
      print("Shape of matrix  X_train_project_title_tfidf after one hot encodig␣
       ↪",X_train_project_title_tfidf.shape)
      print("Shape of matrix  X_test_project_title_tfidf after one hot encodig␣
       ↪",X_test_project_title_tfidf.shape)
      print("Shape of matrix  X_cv_project_title_tfidf after one hot encodig␣
       ↪",X_cv_project_title_tfidf.shape)
```

```
Shape of matrix  X_train_project_title_tfidf after one hot encodig  (17956,
1045)
```

```
Shape of matrix  X_test_project_title_tfidf after one hot encodig  (13200, 1045)
Shape of matrix  X_cv_project_title_tfidf after one hot encodig  (8844, 1045)
```

1.14.1 TFIDF AVG W2V for Project Title for X_train,X_test, X_cv

[40]:
```
# average Word2Vec
# compute average word2vec for each review.
X_train_project_title_avg_w2v_vectors = []; # the avg-w2v for each sentence/
 ↪review is stored in this list
for sentence in tqdm(X_train['project_title']): # 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
    X_train_project_title_avg_w2v_vectors.append(vector)

print(len(X_train_project_title_avg_w2v_vectors))
print(len(X_train_project_title_avg_w2v_vectors[0]))
```

```
100%|| 17956/17956 [00:00<00:00, 122475.17it/s]

17956
300
```

[41]:
```
# average Word2Vec
# compute average word2vec for each review.
X_test_project_title_avg_w2v_vectors = []; # the avg-w2v for each sentence/
 ↪review is stored in this list
for sentence in tqdm(X_test['project_title']): # 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
    X_test_project_title_avg_w2v_vectors.append(vector)

print(len(X_test_project_title_avg_w2v_vectors))
print(len(X_test_project_title_avg_w2v_vectors[0]))
```

```
100%|| 13200/13200 [00:00<00:00, 121409.54it/s]
```

```
13200
300
```

[42]:
```
# average Word2Vec
# compute average word2vec for each review.
X_cv_project_title_avg_w2v_vectors = []; # the avg-w2v for each sentence/review␣
 ↪is stored in this list
for sentence in tqdm(X_cv['project_title']): # 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
    X_cv_project_title_avg_w2v_vectors.append(vector)

print(len(X_cv_project_title_avg_w2v_vectors))
print(len(X_cv_project_title_avg_w2v_vectors[0]))
```

```
100%|| 8844/8844 [00:00<00:00, 112250.20it/s]

8844
300
```

[43]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack


X_tr =␣
 ↪hstack((X_train_price_norm,X_train_sub_categories_one_hot,X_train_teacher_prefix_one_hot)).
 ↪tocsr()
X_cr =␣
 ↪hstack((X_cv_price_norm,X_cv_sub_categories_one_hot,X_cv_teacher_prefix_one_hot)).
 ↪tocsr()
X_te =␣
 ↪hstack((X_test_price_norm,X_test_sub_categories_one_hot,X_test_teacher_prefix_one_hot)).
 ↪tocsr()

#print (X_train_price_norm)
X_tr_bow =␣
 ↪hstack((X_train_price_norm,X_train_sub_categories_one_hot,X_train_teacher_prefix_one_hot,X_
 ↪tocsr()
X_cr_bow =␣
 ↪hstack((X_cv_price_norm,X_cv_sub_categories_one_hot,X_cv_teacher_prefix_one_hot,X_cv_text_b
 ↪tocsr()
```

```
X_te_bow =␣
 ↪hstack((X_test_price_norm,X_test_sub_categories_one_hot,X_test_teacher_prefix_one_hot,X_tes
 ↪tocsr()

X_tr_tfidf =␣
 ↪hstack((X_train_sub_categories_one_hot,X_train_teacher_prefix_one_hot,X_train_price_norm,X_
 ↪tocsr()
X_cr_tfidf =␣
 ↪hstack((X_cv_sub_categories_one_hot,X_cv_teacher_prefix_one_hot,X_cv_price_norm,X_cv_projec
 ↪tocsr()
X_te_tfidf =␣
 ↪hstack((X_test_sub_categories_one_hot,X_test_teacher_prefix_one_hot,X_test_price_norm,X_tes
 ↪tocsr()

X_tr_tfidf_w2v =␣
 ↪hstack((X_train_sub_categories_one_hot,X_train_teacher_prefix_one_hot,X_train_price_norm,X_
 ↪tocsr()
X_cr_tfidf_w2v =␣
 ↪hstack((X_cv_sub_categories_one_hot,X_cv_teacher_prefix_one_hot,X_cv_price_norm,X_cv_projec
 ↪tocsr()
X_te_tfidf_w2v =␣
 ↪hstack((X_test_sub_categories_one_hot,X_test_teacher_prefix_one_hot,X_test_price_norm,X_tes
 ↪tocsr()



X_tr_avg_w2v =␣
 ↪hstack((X_train_sub_categories_one_hot,X_train_teacher_prefix_one_hot,X_train_price_norm,X_
 ↪tocsr()
X_cr_avg_w2v =␣
 ↪hstack((X_cv_sub_categories_one_hot,X_cv_teacher_prefix_one_hot,X_cv_price_norm,X_cv_projec
 ↪tocsr()
X_te_avg_w2v =␣
 ↪hstack((X_test_sub_categories_one_hot,X_test_teacher_prefix_one_hot,X_test_price_norm,X_tes
 ↪tocsr()

#print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)

print("="*100)

print(X_tr_tfidf.shape, y_train.shape)
print(X_cr_tfidf.shape, y_cv.shape)
print(X_te_tfidf.shape, y_test.shape)
print("="*100)
```

```
#print(X_tr_tfidf_w2v.shape, y_train.shape)
#print(X_cr_tfidf_w2v.shape, y_cv.shape)
#print(X_te_tfidf_w2v.shape, y_test.shape)
#print("="*100)
```

```
(17956, 40) (17956,)
(8844, 40) (8844,)
(13200, 40) (13200,)
================================================================================
====================
(17956, 29267) (17956,)
(8844, 29267) (8844,)
(13200, 29267) (13200,)
================================================================================
====================
```

[ ]:

```
```

2 Assignment 3: Apply KNN

[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)
   Hyper paramter tuning to find best K
   Find the best hyper parameter which results in the maximum AUC 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

```
</ul>
</li>
<br>
<li>
<strong>Representation of results</strong>
    <ul>
<li>You need to plot the performance of model both on train data and cross validation data for
<img src='train_cv_auc.JPG' width=300px></li>
<li>Once you find the best hyper parameter, you need to train your model-M using the best hype
<img src='train_test_auc.JPG' width=300px></li>
<li>Along with plotting ROC curve, you need to print the <a href='https://www.appliedaicourse.
<img src='confusion_matrix.png' width=300px></li>
```

```
    </ul>
</li>
<li><strong> [Task-2] </strong>
    <ul>
        <li>Select top 2000 features from feature <font color='red'>Set 2</font> using <a href=
```

and then apply KNN on top of these features

```
    <li>
            <pre>
            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)
            </pre>
        </li>
        <li>Repeat the steps 2 and 3 on the data matrix after feature selection</li>
    </ul>
</li>
<br>
<li><strong>Conclusion</strong>
    <ul>
<li>You need to summarize the results at the end of the notebook, summarize it in the table fo
    <img src='summary.JPG' width=400px>
</li>
    </ul>
```

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.

2. K Nearest Neighbor

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

[44]:
```python
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 tr_loop will be 49041 -␣
↪49041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

[45]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very␣
↪high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for␣
↪threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

[46]:
```
def knn_cross_validation(X_tr,y_train,X_cr,y_cv):
    import matplotlib.pyplot as plt
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import roc_auc_score
    """
    y_true : array, shape = [n_samples] or [n_samples, n_classes]
    True binary labels or binary label indicators.

    y_score : array, shape = [n_samples] or [n_samples, n_classes]
    Target scores, can either be probability estimates of the positive class,␣
↪confidence values, or 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 = [3, 15, 25, 51, 101]
    for i in tqdm(K):
        neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
        neigh.fit(X_tr, y_train)

        y_train_pred = batch_predict(neigh,X_tr)
        y_cv_pred = batch_predict(neigh,X_cr)

        # roc_auc_score(y_true, y_score) the 2nd parameter should be␣
↪probability estimates of the positive class
        # not the predicted outputs
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

[47]:
```
def knn_cross_validation_OLD(X_tr,y_train,X_cv,y_cv,X_test,y_test):
    from sklearn import model_selection
    from mlxtend.plotting import plot_decision_regions
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score
    # Import classification report and confusion matrix to evaluate predictions
    from sklearn.metrics import classification_report, confusion_matrix

    # split the data set into train and test
    # Break up the data as you see test data to 30%, train data is 70%. so we␣
↪break data into two matrix.
    # x1 and y1 would be 70% as train data and x-test and y_test will be test␣
↪data as 30%.
    #X_1, X_test, y_1, y_test = model_selection.train_test_split(X, y,␣
↪test_size=0.3, random_state=0)
```

```
    # split the train data set into cross validation train and cross validation␣
↪test
    #X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1,␣
↪test_size=0.3)

    neighbors = np.arange(1,50,5)
    train_accuracy =np.empty(len(neighbors))
    test_accuracy = np.empty(len(neighbors))

    for i,k in tqdm(enumerate(neighbors)):
        knn = KNeighborsClassifier(n_neighbors=k) # Number of nearest neighbor␣
↪is i.

        # fitting the model on crossvalidation train
        knn.fit(X_tr, y_train)
        # predict the response on the crossvalidation train
       # pred = knn.predict(X_cv)  # predicting the value using cross␣
↪validation data.
```

2.0.1 Split the normalized data into training and test sets

Logic below is simialr as covred in kanalysis_cross_validation(X,y), but here logic is for to calculate confusion matrix, acuarcy ration for best K as we already foound best K after trying best accuracy for multiple K values. We can apply K -fold CV to either the hyperparameter tuning, performance reporting, or both. The advantage of this approach is that the performance is less sensitive to unfortunate splits of data. In addition, it utilize data better since each example can be used for both training and validation/testing.

Let's use K -Fold CV to select the hyperparamter n_neighbors of the KNeighborsClassifier:

[48]:
```
def knn_best_nfold_validation_OLD(X_tr,y_tr,X_cv,y_cv,X_test, y_test,k):
    from sklearn import model_selection
    from mlxtend.plotting import plot_decision_regions
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score
    # Import classification report and confusion matrix to evaluate predictions
    from sklearn.metrics import classification_report, confusion_matrix

    # split the data set into train and test
    # Break up the data as you see test data to 30%, train data is 70%. so we␣
↪break data into two matrix.
    # x1 and y1 would be 70% as train data and x-test and y_test will be test␣
↪data as 30%.

    #X_1, X_test, y_1, y_test = model_selection.train_test_split(X, y,␣
↪test_size=0.3, random_state=0)
```

```
    # split the train data set into cross validation train and cross validation␣
↪test
    #X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1,␣
↪test_size=0.3)

    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors=k) # Number of nearest neighbor is k.


    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred = knn.predict(X_cv)  # predicting the value using cross validation␣
↪data.

    # evaluate CV accuracy
    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)  # I get the␣
↪accuracy score.
    print (k)
    print('\nCV accuracy for k = %d is %d%%' % (k, acc))

    # We are doing the same thing on test data what we did  above to find␣
↪accuracy.
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_tr,y_tr)
    pred = knn.predict(X_test)
    acc = accuracy_score(y_test, pred, normalize=True) * float(100)
    print('\n****Test accuracy for k = %d is %d%%' % (k, acc))

    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,␣
↪pred)
    roc_auc = auc(false_positive_rate, true_positive_rate)
    plt.title('Receiver Operating Characteristic')
    plt.plot(false_positive_rate, true_positive_rate, 'b',
    label='AUC = %0.2f'% roc_auc)
    plt.legend(loc='lower right')
    plt.plot([0,1],[0,1],'r--')
    plt.xlim([-0.1,1.2])
    plt.ylim([-0.1,1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()

    # Print out classification report and confusion matrix
    print(classification_report(y_test, pred))
    # Print out confusion matrix
    cmat = confusion_matrix(y_test, pred)
    #print(cmat)
    print('TP - True Negative {}'.format(cmat[0,0]))
    print('FP - False Positive {}'.format(cmat[0,1]))
    print('FN - False Negative {}'.format(cmat[1,0]))
    print('TP - True Positive {}'.format(cmat[1,1]))
    print('Accuracy Rate: {}'.format(np.divide(np.sum([cmat[0,0],cmat[1,1]]),np.
↪sum(cmat))))
    print('Misclassification Rate: {}'.format(np.divide(np.
↪sum([cmat[0,1],cmat[1,0]]),np.sum(cmat))))
```

[49]:
```
def knn_best_nfold_validation(X_tr,y_tr,X_cv,y_cv,X_te, y_test,best_k):
    # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
↪roc_curve.html#sklearn.metrics.roc_curve
    from sklearn.metrics import roc_curve, auc
    from sklearn.neighbors import KNeighborsClassifier

    neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
    neigh.fit(X_tr, y_train)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability␣
↪estimates of the positive class
    # not the predicted outputs

    y_train_pred = batch_predict(neigh, X_tr)
    y_test_pred = batch_predict(neigh, X_te)

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

    plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr,␣
↪train_tpr)))
    plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr,␣
↪test_tpr)))
    plt.legend()
    plt.xlabel("K: hyperparameter")
    plt.ylabel("AUC")
    plt.title("ERROR PLOTS")
    plt.grid()
    plt.show()

    print("="*100)
    from sklearn.metrics import confusion_matrix
    best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
    print("Train confusion matrix")
    print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
    print("Test confusion matrix")
    print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

2.0.2 Split the normalized data into training and test sets

This step is required to prepare us for the fitting (i.e. training) the #model later. The "X" variable is a collection of all the features. The "y" variable is the target label which specifies the #classification of 1 or 0 based. Our goal will be to identify which category the new data point should fall into. Evaluate the predictions. Evaluate the Model by reviewing the classification report or confusion matrix. By reviewing these tables, we are able to evaluate how accurate our model is with new values. Pleazse refer this website also. very well covered , how to draw confusion matrix or calculate accuracy ration. https://medium.com/@kbrook10/day-11-machine-learning-using-knn-k-nearest-neighbors-with-scikit-learn-350c3a1402e6

[50]:
```
def kanalysis_cross_validation(X_tr,y_tr,X_cr,y_cv):
    from sklearn import model_selection
    from mlxtend.plotting import plot_decision_regions
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score
    # Import classification report and confusion matrix to evaluate predictions
    from sklearn.metrics import classification_report, confusion_matrix

    # split the data set into train and test
    # Break up the data as you see test data to 30%, train data is 70%. so we␣
↪break data into two matrix.
    # x1 and y1 would be 70% as train data and x-test and y_test will be test␣
↪data as 30%.

    error_rate = []

   # X_1, X_test, y_1, y_test = model_selection.train_test_split(X, y,␣
↪test_size=0.3, random_state=0)
    # split the train data set into cross validation train and cross validation␣
↪test
    #X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1,␣
↪test_size=0.3)

    #Evaluate alternative K-values for better predictionsTo simplify the␣
↪process of evaluating multiple cases of k-values,
    #we create a function to derive the error using the average where our␣
↪predictions were not equal to the test values.
    for i in range(1,50,5):  # taking only odd numbers.  1,3,5...
        # instantiate learning model (k = 30)
        knn = KNeighborsClassifier(n_neighbors=i) # Number of nearest neighbor␣
↪is i.
```

```
        # fitting the model on crossvalidation train
        knn.fit(X_tr, y_tr)

        # predict the response on the crossvalidation train
        pred = knn.predict(X_cr)  # predicting the value using cross validation␣
 ↪data.

        # evaluate CV accuracy
        acc = accuracy_score(y_cv, pred, normalize=True) * float(100)  # I get␣
 ↪the accuracy score.
        print('\nCV accuracy for k = %d is %d%%' % (i, acc))


        # evaluate CV accuracy
        error_rate.append(np.mean(pred != y_cv))
        print (error_rate)

    # Configure and plot error rate over k values
    plt.figure(figsize=(10,4))
    plt.plot(range(1,50,5), error_rate, color='blue', linestyle='dashed',␣
 ↪marker='o', markerfacecolor='red', markersize=10)
    plt.title('Error Rate vs. K-Values')
    plt.xlabel('K-Values')
    plt.ylabel('Error Rate')
```

2.2 Make Data Model Ready: encoding numerical, categorical features

[51]:
```
# 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
# make sure you featurize train and test data separatly

# 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
```

[52]:
```
def knn(X,y, k):
    from mlxtend.plotting import plot_decision_regions
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split
    # X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=101)
     knn = KNeighborsClassifier(n_neighbors=k)
     #model.fit(X, y)
    # Fit (i.e. traing) the model
    # knn.fit(X_train, y_train)

     knn.fit(X,y)
     pred = knn.predict(X)
     print (pred)

     value = 1.5
     width = 0.75
```

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

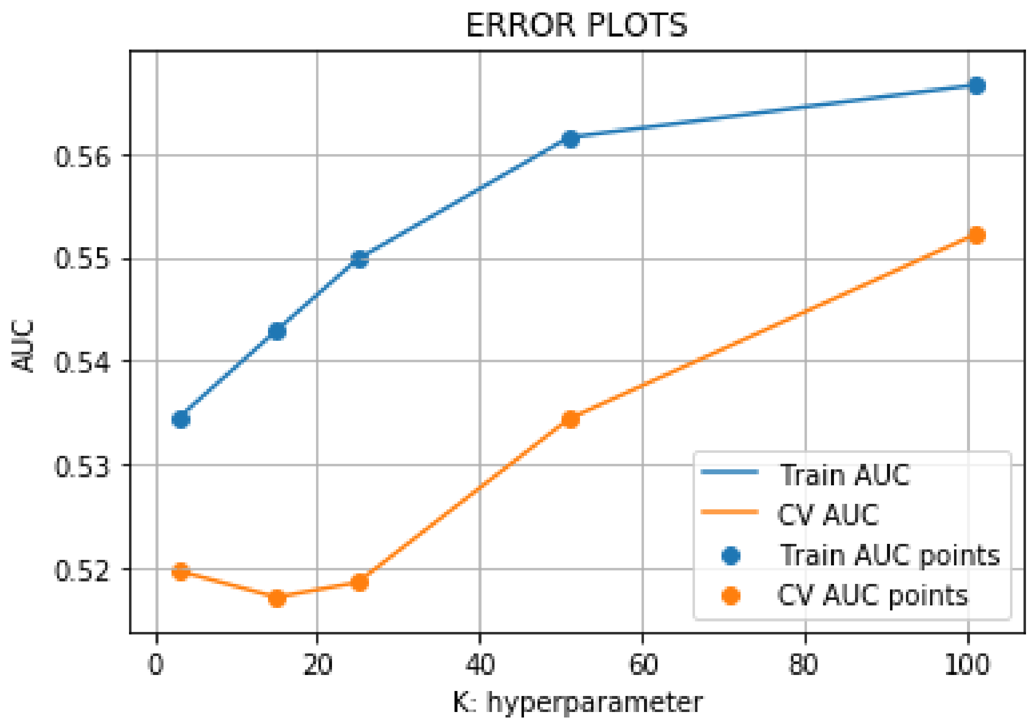
Apply KNN on different kind of featurization as mentioned in the instructions For Every model that you work on make sure you do the step 2 and step 3 of instructions

2.3 Make Data Model Ready: encoding eassay, and project_title

2.0.3 2.4.0 Applying KNN brute force Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW), SET 1

[53]:
```
knn_cross_validation(X_tr,y_train,X_cr,y_cv)
```

```
100%|| 5/5 [01:14<00:00, 14.93s/it]
```

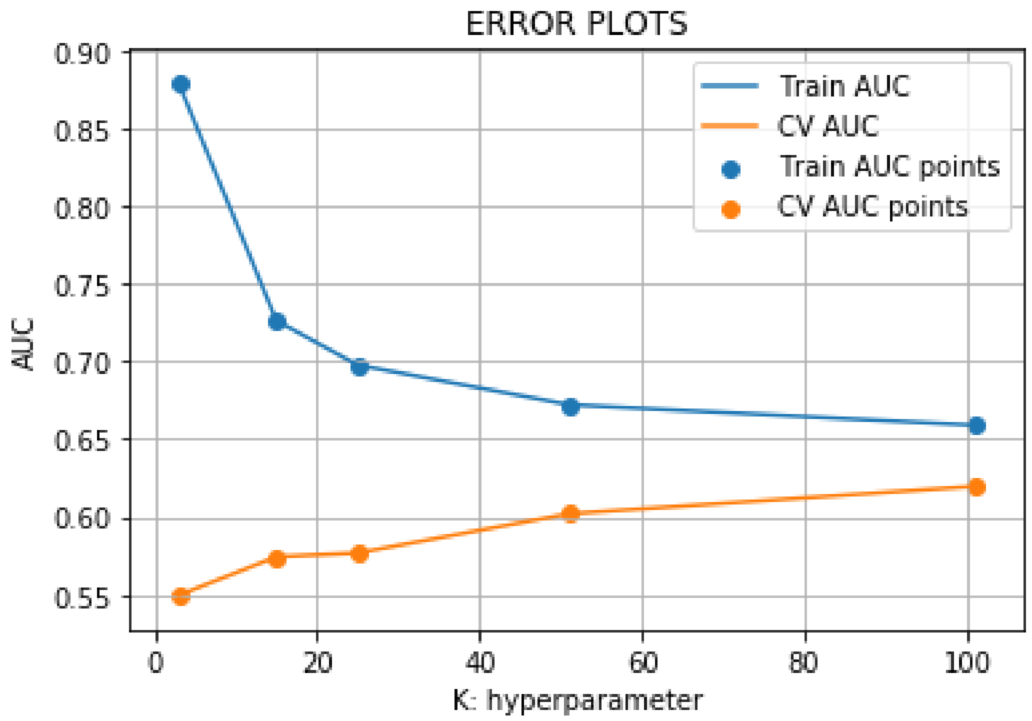


2.0.4 2.4.1 Applying KNN brute force on BOW - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW), SET 1

[54]:
```
# Please write all the code with proper documentation
```

[55]:
```
knn_cross_validation(X_tr_bow,y_train,X_cr_bow,y_cv)
```

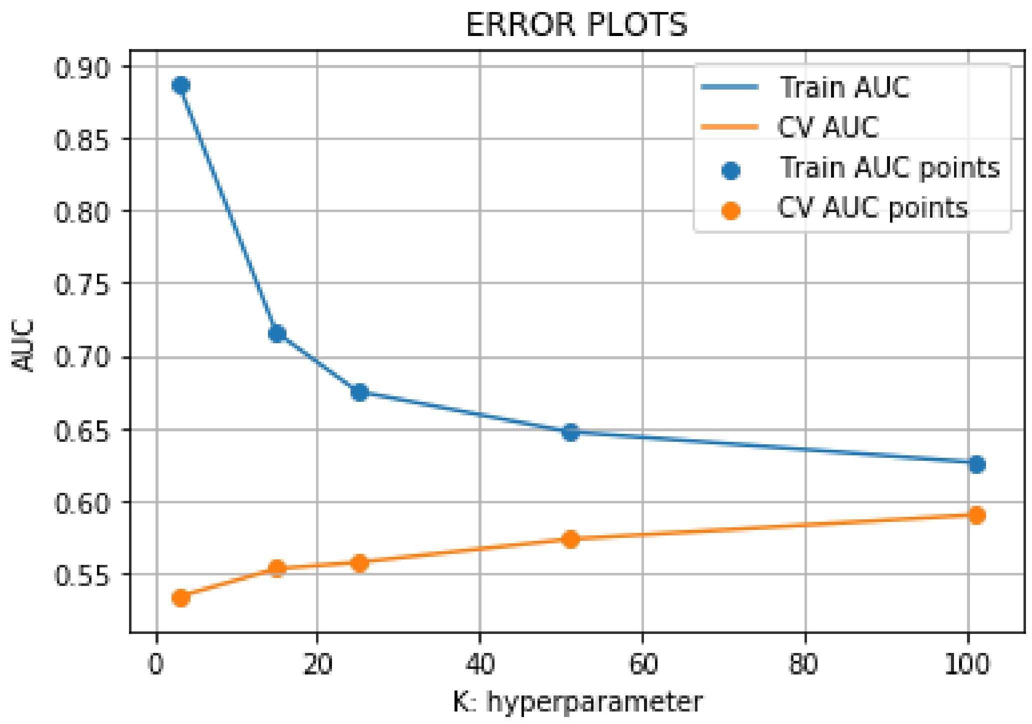
```
100%|| 5/5 [01:49<00:00, 21.91s/it]
```



2.0.5 2.4.2 Applying KNN brute force on TFIDF Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF), SET 2

[56]:
```
knn_cross_validation(X_tr_tfidf,y_train,X_cr_tfidf,y_cv)
```

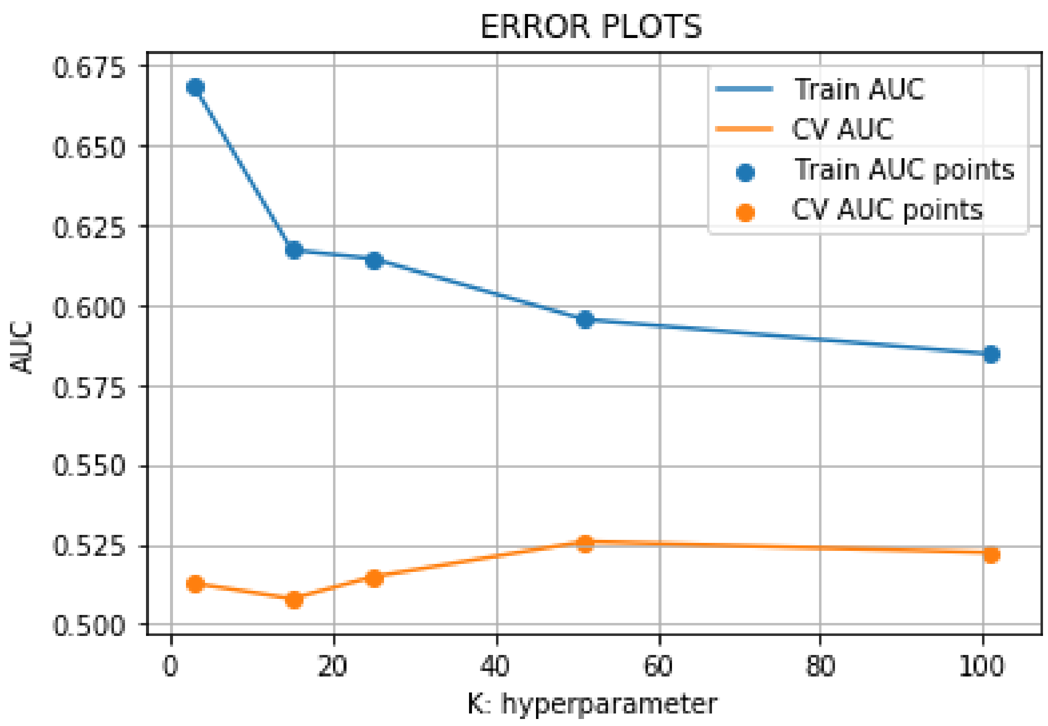
```
100%|| 5/5 [01:40<00:00, 20.11s/it]
```



2.0.6 2.4.3 Applying KNN brute force on AVG W2V, SET 3

[57]:
```
knn_cross_validation(X_tr_avg_w2v,y_train,X_cr_avg_w2v,y_cv)
```

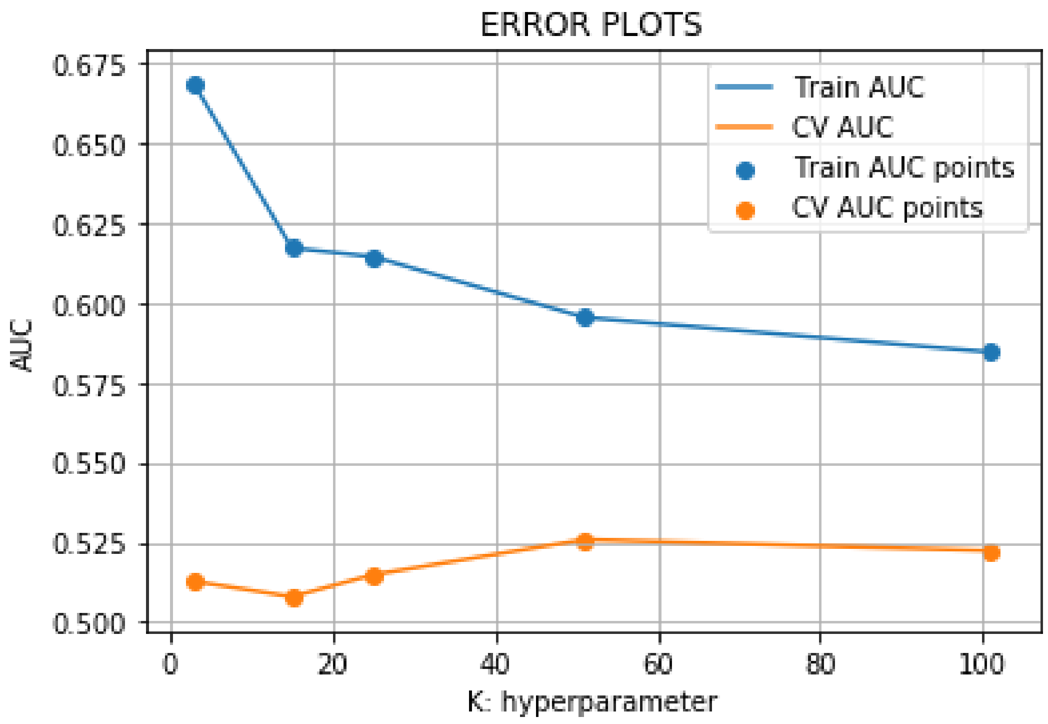
```
100%|| 5/5 [03:33<00:00, 42.90s/it]
```



2.0.7 2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

[58]:
```
knn_cross_validation(X_tr_tfidf_w2v,y_train,X_cr_tfidf_w2v,y_cv)
```

```
100%|| 5/5 [03:32<00:00, 42.26s/it]
```



2.5 Feature selection with `SelectKBest`

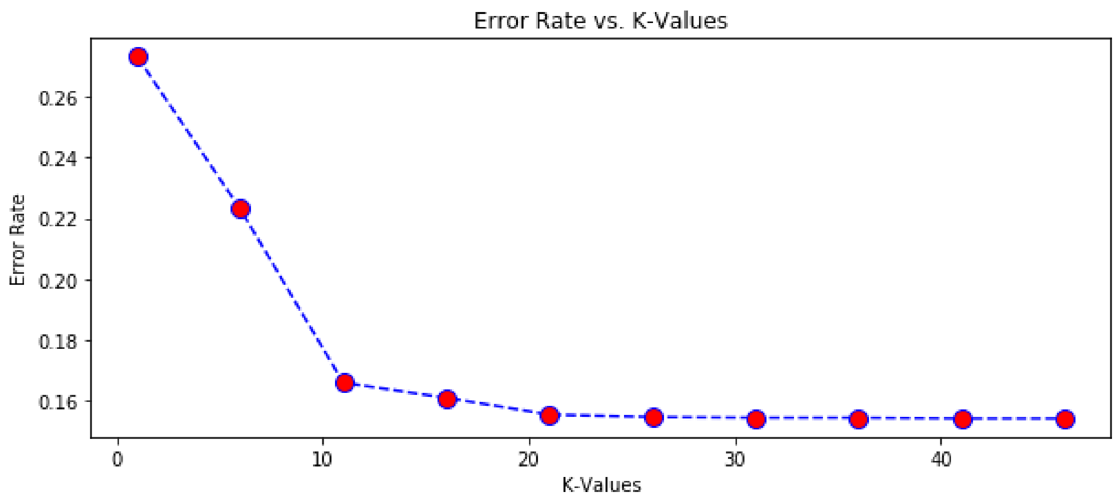
[59]:
```
# please write all the code with proper documentation, and proper titles for␣
↪each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in␣
↪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
```

2.1 2.5.1 K-Analysis categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW), SET 1

[60]:
```
kanalysis_cross_validation(X_tr_bow,y_train,X_cr_bow,y_cv)
```

```
CV accuracy for k = 1 is 72%
[0.2730664857530529]
```

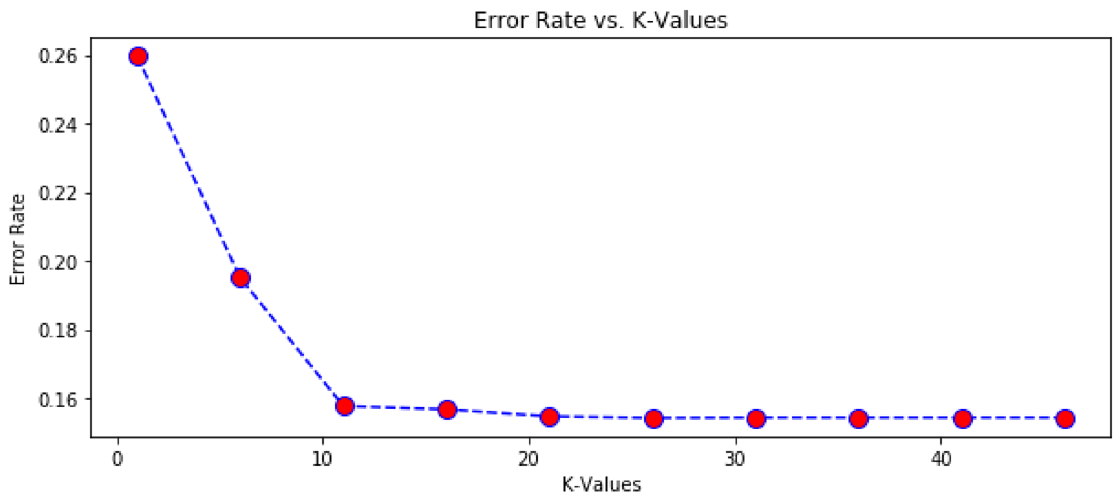
```
CV accuracy for k = 6 is 77%
[0.2730664857530529, 0.2230890999547716]

CV accuracy for k = 11 is 83%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968]

CV accuracy for k = 16 is 83%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968,
0.16112618724559022]

CV accuracy for k = 21 is 84%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968,
0.16112618724559022, 0.1555857078245138]

CV accuracy for k = 26 is 84%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968,
0.16112618724559022, 0.1555857078245138, 0.15479421076436]

CV accuracy for k = 31 is 84%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968,
0.16112618724559022, 0.1555857078245138, 0.15479421076436, 0.15456806874717322]

CV accuracy for k = 36 is 84%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968,
0.16112618724559022, 0.1555857078245138, 0.15479421076436, 0.15456806874717322,
0.15456806874717322]

CV accuracy for k = 41 is 84%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968,
0.16112618724559022, 0.1555857078245138, 0.15479421076436, 0.15456806874717322,
0.15456806874717322, 0.15434192672998642]

CV accuracy for k = 46 is 84%
[0.2730664857530529, 0.2230890999547716, 0.16610131162369968,
0.16112618724559022, 0.1555857078245138, 0.15479421076436, 0.15456806874717322,
0.15456806874717322, 0.15434192672998642, 0.15434192672998642]
```



2.1.1 2.5.2 K-Analysis on TFIDF Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF), SET 2

[61]:
```
kanalysis_cross_validation(X_tr_tfidf,y_train,X_cr_tfidf,y_cv)
```

```
CV accuracy for k = 1 is 74%
[0.2596110357304387]

CV accuracy for k = 6 is 80%
[0.2596110357304387, 0.19504748982360923]

CV accuracy for k = 11 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833]

CV accuracy for k = 16 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833,
0.15671641791044777]

CV accuracy for k = 21 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833,
0.15671641791044777, 0.1546811397557666]

CV accuracy for k = 26 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833,
0.15671641791044777, 0.1546811397557666, 0.15422885572139303]

CV accuracy for k = 31 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833,
0.15671641791044777, 0.1546811397557666, 0.15422885572139303,
0.15434192672998642]

CV accuracy for k = 36 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833,
0.15671641791044777, 0.1546811397557666, 0.15422885572139303,
0.15434192672998642, 0.15434192672998642]

CV accuracy for k = 41 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833,
0.15671641791044777, 0.1546811397557666, 0.15422885572139303,
0.15434192672998642, 0.15434192672998642, 0.15434192672998642]

CV accuracy for k = 46 is 84%
[0.2596110357304387, 0.19504748982360923, 0.15773405698778833,
0.15671641791044777, 0.1546811397557666, 0.15422885572139303,
0.15434192672998642, 0.15434192672998642, 0.15434192672998642,
0.15434192672998642]
```

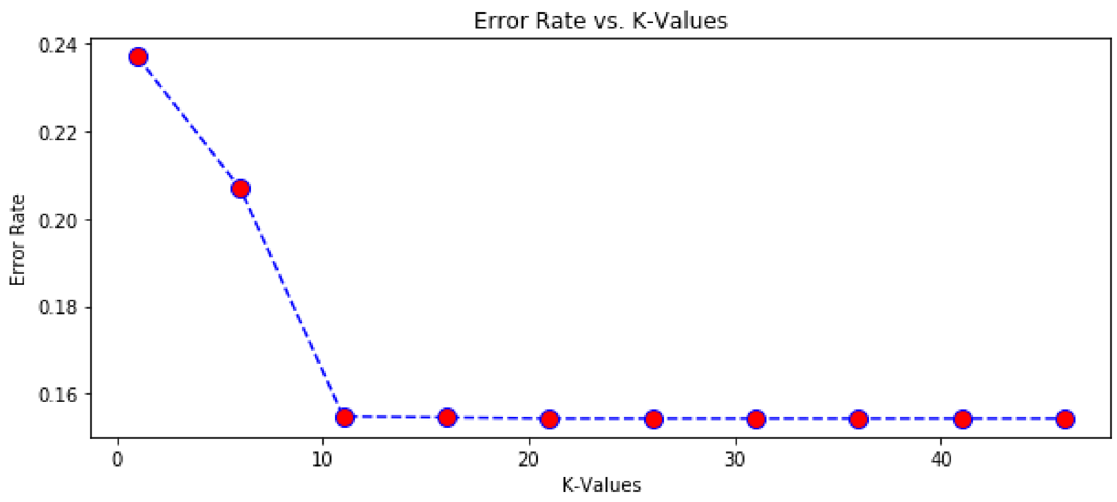


2.1.2 2.5.3 K-Analysis on AVG W2V - categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V), SET 3

[62]:
```
kanalysis_cross_validation(X_tr_avg_w2v,y_train,X_cr_avg_w2v,y_cv)
```

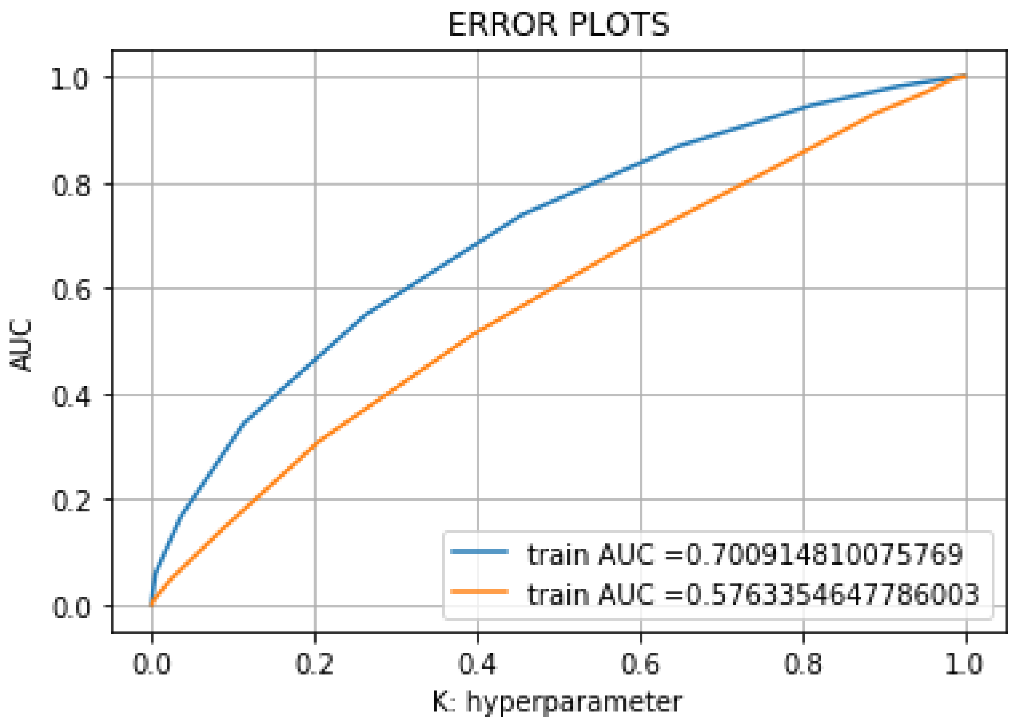
```
CV accuracy for k = 1 is 76%
[0.2369968340117594]

CV accuracy for k = 6 is 79%
```

```
[0.2369968340117594, 0.20691994572591588]

CV accuracy for k = 11 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436]

CV accuracy for k = 16 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436, 0.15456806874717322]

CV accuracy for k = 21 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436, 0.15456806874717322,
0.15434192672998642]

CV accuracy for k = 26 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436, 0.15456806874717322,
0.15434192672998642, 0.15434192672998642]

CV accuracy for k = 31 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436, 0.15456806874717322,
0.15434192672998642, 0.15434192672998642, 0.15434192672998642]

CV accuracy for k = 36 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436, 0.15456806874717322,
0.15434192672998642, 0.15434192672998642, 0.15434192672998642,
0.15434192672998642]

CV accuracy for k = 41 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436, 0.15456806874717322,
0.15434192672998642, 0.15434192672998642, 0.15434192672998642,
0.15434192672998642, 0.15434192672998642]

CV accuracy for k = 46 is 84%
[0.2369968340117594, 0.20691994572591588, 0.15479421076436, 0.15456806874717322,
0.15434192672998642, 0.15434192672998642, 0.15434192672998642,
0.15434192672998642, 0.15434192672998642, 0.15434192672998642]
```



2.1.3 3.1 K-BEST categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW), SET 1

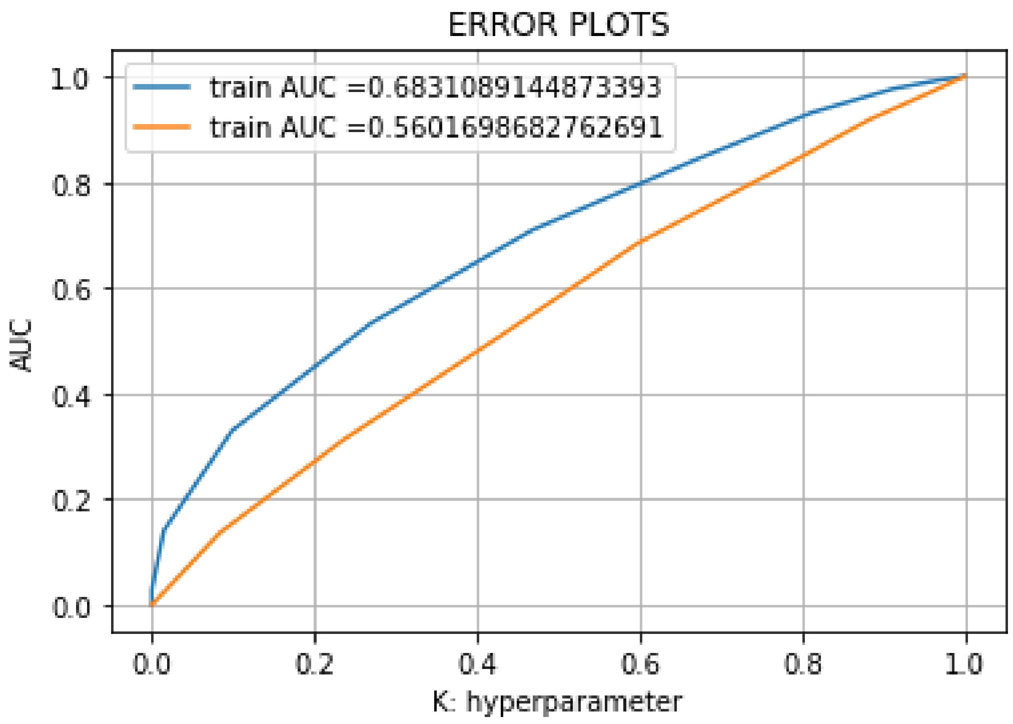
[63]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

# with the same hstack function we are concatinating a sparse matrix and a␣
↪dense matirx :)
#X = hstack((categories_one_hot, sub_categories_one_hot, price_standardized,␣
↪project_title_bow,text_bow))

Best_k=22
knn_best_nfold_validation(X_tr_bow,y_train,X_cr_bow,y_cv,X_te_bow,␣
↪y_test,Best_k)
```



```
================================================================================
====================
the maximum value of tpr*(1-fpr) 0.4044698167129674 for threshold 0.818
Train confusion matrix
[[2042  730]
 [6847 8337]]
Test confusion matrix
[[1233  804]
 [5461 5702]]
```

2.1.4 3.2 K-Best on TFIDF Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF), SET 2

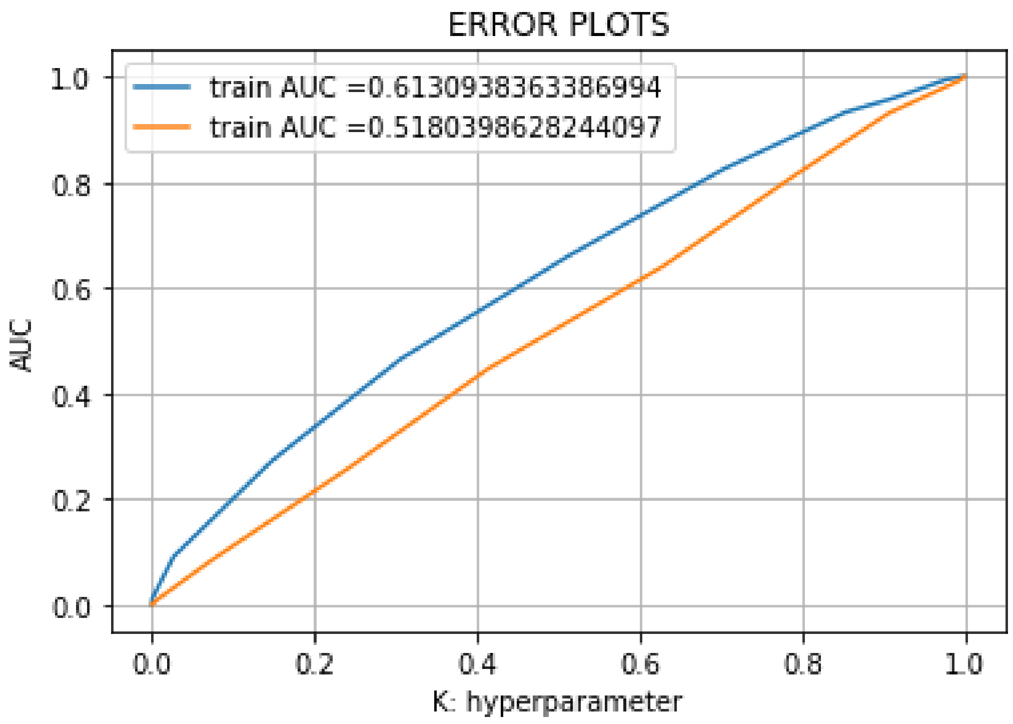
[64]:
```
# Please write all the code with proper documentation

Best_k=22
knn_best_nfold_validation(X_tr_tfidf,y_train,X_cr_tfidf,y_cv,X_te_tfidf,␣
 ↪y_test,Best_k)
```



```
================================================================================
====================
the maximum value of tpr*(1-fpr) 0.3893153079796915 for threshold 0.864
Train confusion matrix
[[2023  749]
 [7084 8100]]
Test confusion matrix
[[1165  872]
 [5494 5669]]
```

2.1.5 3.3 K-Best on AVG W2V - categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V), SET 3

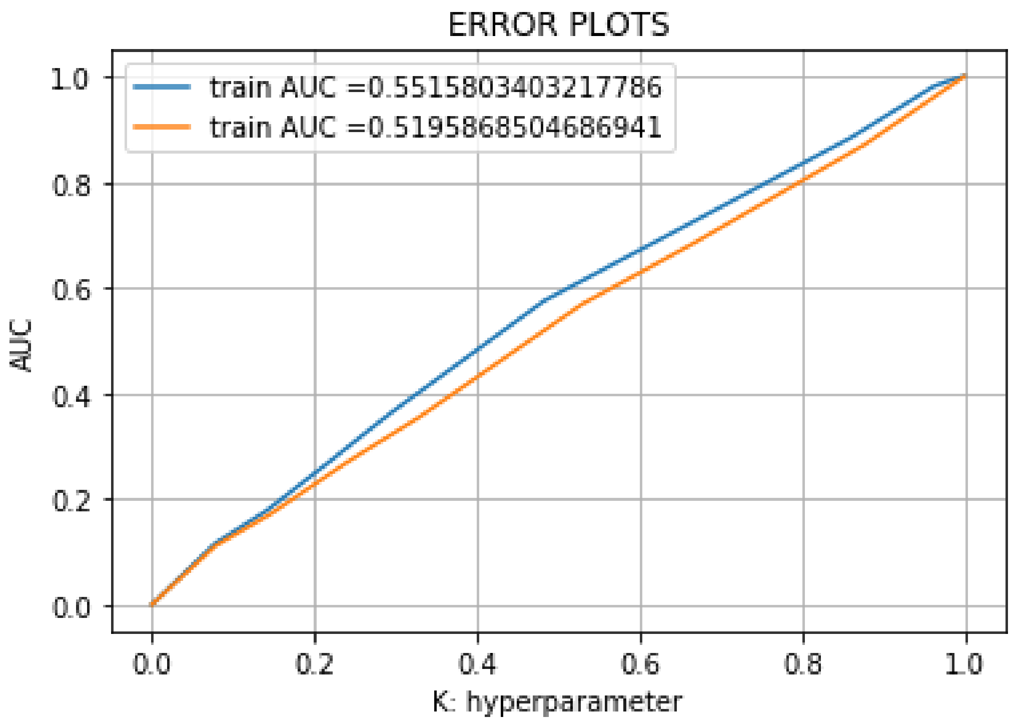
[65]:
```
Best_k= 26
knn_best_nfold_validation(X_tr_avg_w2v,y_train,X_cr_avg_w2v,y_cv,X_te_avg_w2v,␣
 ↪y_test,Best_k)
```



```
================================================================================
====================
the maximum value of tpr*(1-fpr) 0.32269264696490724 for threshold 0.885
Train confusion matrix
[[1923  849]
 [8121 7063]]
Test confusion matrix
[[1194  843]
 [6177 4986]]
```

2.1.6 3.4 K-Best on on TFIDF W2V - categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V), SET 4

[66]:
```
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
#from scipy.sparse import hstack

Best_k= 26
knn_best_nfold_validation(X_tr,y_train,X_cr,y_cv,X_te, y_test,Best_k)
```



```
================================================================================
====================
the maximum value of tpr*(1-fpr) 0.29756250218578983 for threshold 0.846
Train confusion matrix
[[1433 1339]
 [6444 8740]]
Test confusion matrix
[[ 958 1079]
 [4809 6354]]
```

3. Conclusions

2.2 Summary of above program as below:

2.2.1 Step 1: Import the necessary Libraries

we will need to import libraries that allow for data analysis and data visualization to get acclimated to the dataset. We will be using pandas, numpy, matplotlib and seaborn to conduct this. Data Exploration libraries

2.2.2 Step 2: Read in the dataset.

We will use the pandas .read_csv() method to read in the dataset. Then we will use the. head() method to observe the first few rows of the data, to understand the information better. In our case, the feature(column) headers tell us pretty little. This is fine because we are merely trying to gain insight via classifying new data points by referencing it's neighboring elements.

2.2.3 Step 3: Standardize (normalize) the data scale to prep for KNN algorithm.

Because the distance between pairs of points plays a critical part on the classification, it is necessary to normalize the data This will generate an array of values. Again, KNN depends on the distance between each feature. Please see Section 1 for all normalization.

2.2.4 Step 4: Split the normalized data into training and test sets.

This step is required to prepare us for the fitting (i.e. training) the model later. The "X" variable is a collection of all the features. The "y" variable is the target label which specifies the classification of 1 or 0 based. Our goal will be to identify which category the new data point should fall into.

Please see functions as covered below, used in above program: def kanalysis_cross_validation(X,y): def knn_best_nfold_validation(X,y, k): def knn(X,y, k):

2.2.5 Step 5: Create and Train the Model.

Here we create a KNN Object and use the .fit() method to train the model. Upon completion of the model we should receive confirmation that the training has been complete

Please see functions as covered below, used in above program: def kanalysis_cross_validation(X,y): def knn_best_nfold_validation(X,y, k): def knn(X,y, k):

2.2.6 Step 6: Make Predictions.

Here we review where our model was accurate and where it misclassified elements.

Please see functions as covered below, used in above program: def kanalysis_cross_validation(X,y): def knn_best_nfold_validation(X,y, k):

2.2.7 Step 7: Evaluate the predictions.

Evaluate the Model by reviewing the classification report or confusion matrix. By reviewing these tables, we are able to evaluate how accurate our model is with new values.

def kanalysis_cross_validation(X,y): def knn_best_nfold_validation(X,y, k):

2.2.8 Setp 8:Classification Report :

This tells us our model was around 84% accurate… Print out classification report and confusion matrix

print(classification_report(y_test, pred))

I have covered various set to show confusion matrix.

Please see section 2. covered various data sets and created confusion matrix.

2.2.9 Step 9: Evaluate alternative K-values for better predictions.

To simplify the process of evaluating multiple cases of k-values, we create a function to derive the error using the average where our predictions were not equal to the test values.

Please see section 2. covered various data sets and created error accuracy reports.

2.2.10 Step 10: Plot Error Rate

Here we see that the error rate continues to decrease as we increase the k-value. A picture tells a thousand words. Or at least here, we are able to understand what value of k leads to an optimal model. The k-value of 17 seems to give a decent error rate without too much noise, as we see with k-values of 28 and larger.

2.2.11 Step 11: Adjust K value per error rate evaluations

This is just fine tuning our model to increase accuracy. We will need to retrain our model with the new k-value. Please see section 3 in above program. we have created confusion matrix for optimal KNN value for various data sets. As we can see for optimal KNN, Accuracy is much higher - so prediction is much better.

[ ]:

```

```