**Faculty of Engineering, Applied Sciences, and Technology**

###### BCS 405 – Data Mining

###### Project

|  |  |  |
| --- | --- | --- |
| Course Code: BCS 405 |  | Course Name: Data Mining |
| Date: Nov 13, 2022 |  | Time: 9:00-12:00 |
| Location: MGT 105 |  | Instructor(s): Dr. Said Elnaffar |
| Number of Students: 27 |  | Number of Pages: 3 |

|  |  |
| --- | --- |
| **Course Learning Outcomes** | **Project** |
| **CLO1.** Describe the major issues in data mining and classification |  |
| **CLO2.** Design and implement algorithms for data preprocessing and visualization | **X** |
| **CLO3.** Apply appropriate methods on data sets, and data classification | **X** |
| **CLO4.** Explain the limits and constraints of different data classification techniques. | **X** |

**CLO Mappings**

|  |  |
| --- | --- |
| **Code & Report** | **Q&A** |
| **CLO 2, CLO 3** | **CLO4** |

**Students** (double-click the table to add ID and Names)

****

# Project Description and Requirements

One of the commonly used natural language processing techniques used to classify content is Naïve Bayes. In this project you are asked to develop a topic classifier that can distinguish whether a newspaper topic is about, for example, Sports or Technology.

Your project will consist of two modules, one for training and building the prediction model, and the other for testing the model and assessing its accuracy. Newspaper articles can be stored as text files. For example, in the training mode, your program can access a master folder called *training* which encompasses two subfolders: *sports* and *technology*. Each of the latter folders will contain enough samples of articles from each topic.

Similarly, in the testing mode, your program can access a master folder called *testing* which encompasses two subfolders: *sports* and *technology*. The accuracy of your classifier will be assessed based on the predicted label for each article under each relevant topic.

# Submission Instructions

1. Use this template. Fill out the names of students on the front page (*double-click the Excel table to edit*).
2. Attach your report to the end of this document. You should describe your design, methodology, and solution.
3. You should provide test cases and screenshots.
4. Outline the contributions of each member.
5. Attach your code (fully commented).
6. Make one submission per group.

# Rubric

|  |  |  |  |
| --- | --- | --- | --- |
| Criterion | Needs Improvement  [0-3] | Good  [4-7] | Excellent  [8-10] |
| Q&A | The student provided poor answers and insufficient evidence of contribution to the submission. | The student answered some of the questions correctly and showed some contribution to the submission. | The student answered all questions correctly and showed genuine contribution to the submission. |
| Coding & Report | The code does not work properly, many requirements are missing, and the report does not provide enough proofs (screenshots) of testing scenarios. | The code works partially and implements some of the requirements; the report shows some testing scenarios. | The code works properly and as expected per the requirements and the report shows all testing scenarios. |

*Classifier (Naïve bayes)*

*Contents: -*

1. *Design (p.3)*
2. *Methodology (p.4 – 8)*
3. *Solution (p.9 – 20)*
4. *Code (p.21 – 28)*

Design

This project was issued the task of predicting and sorting articles based on two categories: -

Technology

Sports

We set out to accomplish this task by first gathering articles from a news source of our choosing (discussed more in Methodology) and then we set out to create the algorithm to split the data into two sets: Training and testing data sets. To make the predictor more user friendly and usable by end users in the future a simple application UI was created with the ability to predict any future articles from google news in either category.

Group member tasks are as per the following: -

Luka - Developing the algorithm that determined the prediction using multinomial naïve bayes.

David - Gathering articles using a custom-made web scrapper and report writing.

Farhad - Created the pre-processing algorithm for “cleaning” the articles.

Stalin - Report editing, comment creation, and UI creator.

Software, language & article news used: -

Google news was used to gather the articles.

Visual studio code for writing and running the code.

Anaconda for getting data mining libraries for python.

Python programming language.

GitHub for code sharing, modifying.

VOIP applications for discussions & idea sharing.

Methodology

In data mining sorting is a very vital component that is needed in managing big data, therefore we wanted to create a method to both extract articles and sort them in two categories.

Web scraping

For this project the first notable task was to gather a suitable number of articles to accurately create our prediction model. The first step was to select an appropriate news site for extraction, and we choose googlenews.com as the hub for our articles. The next step was to consider how many articles we need to get accurate results and we have decided to use a total amount of 400 articles, 200 from the technology section and 200 from the sports section.

To accomplish this task, we built a custom web scrapper that sorted articles in their respective folders. The web scraper went on the google news link related to either technology or sports, read and stored the article contents on to a text document in its respected folder. The process would loop and scroll through the category to pull and store articles up to the limit of 200 each.

The code: -

import newspaper

import os

from requests\_html import HTMLSession

# ---------------------------------------sports--------------------------------------------------------

session = HTMLSession()

# URL link for the sports category

url = 'https://news.google.com/topics/CAAqJggKIiBDQkFTRWdvSUwyMHZNRFp1ZEdvU0FtVnVHZ0pWVXlnQVAB?hl=en-US&gl=US&ceid=US%3Aen'

r = session.get(url)

# this line is utilised to scroll the google news category page to gather the articles up to the specified limit.

r.html.render(sleep=1, scrolldown=6)

articles = r.html.find('article')

newsLinks = []

for i in articles:

    try:

        newsitem = i.find('a', first=True)

        rawLink = newsitem.attrs['href']

        # URL is obtained by looking at the inspect element of the web page and extracting the link that is stored in html 'a' class

        # All articles within the link contains the starting link as mentioned as a string in fullLink variable.

        fullLink = str("https://news.google.com" + rawLink[1:-1])

        newsLinks.append(fullLink)

    except:

        pass

print(len(newsLinks))

print(newsLinks[5:10])

item = 1

switch = 0

# 100 articles are stored in the directory mentioned below.

os.chdir("Python\_classifier\Sports\Training")

for url in newsLinks:

    if item <= 100 and switch == 0:

        try:

            file2write = open("TrSportsArticle" + str(item) + ".txt", 'w')

            url\_i = newspaper.Article(url=(url), language='en')

            url\_i.download()

            url\_i.parse()

            text = url\_i.text

            if(len(text) > 0):

                file2write.write(text)

                print(url\_i)

                item = item + 1

                file2write.close()

            if(item == 101):

                # storing another 100 in the testing subfolder

                os.chdir("..\Testing")

                switch = 1

        except:

            pass

    elif item > 100 and switch == 1:

        try:

            file2write = open("TeSportsArticle" + str(item) + ".txt", 'w')

            url\_i = newspaper.Article(url=(url), language='en')

            url\_i.download()

            url\_i.parse()

            text = url\_i.text

            if(len(text) > 0):

                file2write.write(text)

                print(url\_i)

                item = item + 1

                file2write.close()

        except:

            pass

    elif item >= 200:

        break

# -------------------------------------------------technology---------------------------------------------------------------------------

# same logic for technology

session = HTMLSession()

url = 'https://news.google.com/topics/CAAqJggKIiBDQkFTRWdvSUwyMHZNRGRqTVhZU0FtVnVHZ0pWVXlnQVAB?hl=en-US&gl=US&ceid=US%3Aen'

r = session.get(url)

r.html.render(sleep=1, scrolldown=6)

articles = r.html.find('article')

newsLinks = []

for i in articles:

    try:

        newsitem = i.find('a', first=True)

        rawLink = newsitem.attrs['href']

        fullLink = str("https://news.google.com" + rawLink[1:-1])

        newsLinks.append(fullLink)

    except:

        pass

print(len(newsLinks))

print(newsLinks[5:10])

item = 1

switch = 0

os.chdir("Python\_classifier\Technology\Training")

for url in newsLinks:

    if item <= 100 and switch == 0:

        try:

            file2write = open("TrTechnologyArticle" + str(item) + ".txt", 'w')

            url\_i = newspaper.Article(url=(url), language='en')

            url\_i.download()

            url\_i.parse()

            text = url\_i.text

            if(len(text) > 0):

                file2write.write(text)

                item = item + 1

                file2write.close()

            if(item == 101):

                os.chdir("..\Testing")

                switch = 1

        except:

            pass

    elif item > 100 and switch == 1:

        try:

            file2write = open("TeTechnologyArticle" + str(item) + ".txt", 'w')

            url\_i = newspaper.Article(url=(url), language='en')

            url\_i.download()

            url\_i.parse()

            text = url\_i.text

            if(len(text) > 0):

                file2write.write(text)

                item = item + 1

                file2write.close()

        except:

            pass

    elif item >= 200:

        break

Data pre-processing

Articles contain a lot of words that aren’t necessary to be known by the prediction model such as stop words, punctuation, numbers, and special characters such as “\_=+” to name a few, therefore we understood that an algorithm is needed to “clean” the articles to get efficient predictions in our model. Letters that have been capitalized have also been lowercased to make every word on the same level. Natural language processing library was used to achieve clean data. (Code in solutions)

Prediction model

The prediction model we used is called multinomial naïve bayes which is a method used for natural language processing to understand articles and it aims to calculate the probability of the label of an article. Providing the predicted label of the article. The way it predicts is first utilizing what is known as the count vector where words in an article are transformed into numbers and are then compared with the label value of either sports (1) or technology (0). This is done to see the words that commonly appear in either category to establish a prediction for the test articles and the unknown articles.

This is the formula: -

**P(A|B) = P(A)\*P(B|A)/P(B)**

**P(B) = prior prob of B**

**P(A) = prior prob of A**

**P(B|A) = occurrence of predictor B given A prob**

Dataframe and splitting

Once 200 articles were gathered for technology and sports, we placed them together in one dataframe following the dictionary format we established in the code. Since we knew beforehand the categories of the articles we assigned label 1 to sports and label 0 to technology and added the label to the dataframe. Sklearn is the library we used to split the data. Sklearn would create x, y train where x is the articles(words) and y was the label(either sports or technology). It would select 300 articles to train the naïve bayes prediction. We also had x, y test which contained 100 articles that would be utilized to see if naïve bayes was operating as expected. The article selection is based on pseudo random choice done by the algorithm. The article retrieval is done dynamically so the test and training split can be added to and improved with more articles in the future, it is not hard coded with a specific article number limitation.

Solution

Libraries needed for data, os, language & string manipulation.

import pandas as pd

import os

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import string

# nltk.download()

Preprocessing code

# Pre-processing!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

def cleanArticle(file2read):

#Declare extra words and punctuations

    stop\_words = set(stopwords.words('english') + list(string.punctuation))

#Declare the returning variable

    article = ''

#Declare extra chars of article which are not included in the previous packages

    customExtras = ["\n", "’", "“", "”", "'", "`","1","2","3","4","5","6","7","8","9","0","—",".."]

#function that converts arr of strings into one string (used later)

    def listToString(s):

        str1 = " "

        return (str1.join(s))

#Remove extra chars in article

    for i in customExtras:

        file2read = file2read.replace(i, "")

#Converts article into array of words

    word\_tokens= word\_tokenize(file2read.lower())

    # print(word\_tokens)

#Empty array of clean words

    filtered\_file2read =[]

#check if word from article is the extra,if not assign the word in the clean words array

    for w in word\_tokens:

        if w not in stop\_words:

            filtered\_file2read.append(w)

#print(filtered\_file2read)

    article = listToString(filtered\_file2read)

#print(article)

    return article

putting articles into a data frame

pulling articles from their respective directories and placing them in a data frame.

(Note - Restart kernel if rerunning code)

#Technology folder !!!!!!!!!!!!!!!!!!!!!!

#change directory to technology training

#restart kernel if rerunning code

os.chdir("Technology/Training/")

dirTecharticle = os.listdir()

#cwd = os.getcwd()

#print(cwd)

articlestr = []

genrestr = []

#creating dictionary for data frame to be used

trdata = {'article': articlestr,

        'genre': genrestr

        }

#open training technology articles

#articles will be extracted and placed in the dataframe

for file in dirTecharticle:

     file2read = open(file,'r').read()

#apply the preprocessing function

     file2read = cleanArticle(file2read)

     articlestr.append(file2read)

     genrestr.append("Technology")

datatr = pd.DataFrame(trdata)

print(datatr)

# Sports folder!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

os.chdir("../../Sports/Training/")

dirSportsarticle= os.listdir()

# cwd = os.getcwd()

# print(cwd)

#articles from sports folder

for file in dirSportsarticle:

     file2read = open(file,'r').read()

     # apply the pre-processing function

     file2read = cleanArticle(file2read)

     articlestr.append(file2read)

     genrestr.append("Sports")

datatr = pd.DataFrame(trdata)

print(datatr)



Label for the prediction model

#assign `label`` to dataframe if technology then label = 0  if not label = 1

datatr["label"] = datatr["genre"].apply(lambda x: 0 if x=="Technology" else 1)

datatr

A screenshot of a computer

Description automatically generated with medium confidence

Creating the variables for training and testing

from sklearn.model\_selection import train\_test\_split

#X - articles for testing and training y- labels for testing and training

X\_train, X\_test, y\_train, y\_test = train\_test\_split(datatr["article"], datatr["label"], random\_state=1)

# print(X\_train)

# print(y\_train)

# print(X\_test)

# print(y\_test)

Using Count Vectorizer

from sklearn.feature\_extraction.text import CountVectorizer

 #Tokenizes the documents and converts it to matrix

cv = CountVectorizer(strip\_accents="ascii")

#Learn the vocabulary dictionary and return document-term

X\_train\_cv = cv.fit\_transform(X\_train)

#Transform documents to document-term matrix.

X\_test\_cv = cv.transform(X\_test)

#print(X\_train\_cv)

Prediction model learning

from sklearn.naive\_bayes import MultinomialNB

from sklearn.svm import SVC #For confidence

conmodel = SVC(probability=True)

conmodel.fit(X\_train\_cv,y\_train)

class\_prob = conmodel.predict\_proba(X\_test\_cv)

# declare naive bayes model

naive\_bayes = MultinomialNB()

#train naive bayes model

naive\_bayes.fit(X\_train\_cv, y\_train)

#predict

predictions = naive\_bayes.predict(X\_test\_cv)

# print(predictions)

Accuracy, Precision & recall

“Technology” is positive

“Sports” is negative

Precision = True positive /True positive +False positive

Recall = True positive /True positive +False Negative

Accuracy = tp+tn/tp+fp+tn+fn

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

#display accuracy Percision and Recall

accuracy = "Accuracy score: " + str(accuracy\_score(y\_test, predictions))

precision = "Precision score: " + str(precision\_score(y\_test, predictions))

recallscore = "Recall score: " + str(recall\_score(y\_test, predictions))

print(accuracy)

print(precision)

#printing string elements from index 0 to 17

print(recallscore[0:18])

confidence\_list\_sports = []#confidence for sports

confidence\_list\_tech = []#confidence for tech

for i in range(len(class\_prob)):

    confidence\_list\_tech.append(class\_prob[i][0]\*100)

for i in range(len(class\_prob)):

    confidence\_list\_sports.append(class\_prob[i][1]\*100)

#print(class\_probabilities[0][0] \* 100) #1st [0] shows document if 2nd [0] shows confidence for sports if 2nd [1] shows confidence for tech

# print(len(confidence\_list\_tech))

Text

Description automatically generated

Confusion matrix

Useful for identifying where the error is.

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

#Storing confusion matrix in cm with the parameters of actual labels(y\_test) & predictions from naive bayes

cm = confusion\_matrix(y\_test, predictions)

#creating the graph for cm, square is the shape, annot is the number displayed in each quadrant,cmap is the colour

# & cbar is the represents the number of articles in each quadrant as colour(darker blue being the most and red the lowest)

sns.heatmap(cm, square=True, annot=True, cmap="RdBu", cbar=False,

#Quadrant labels

#positive = Technology

#negative = Sports

xticklabels=["Technology", "Sports"], yticklabels=["Technology", "Sports"])

#X & Y labels of the entire graph

plt.xlabel("true label")

plt.ylabel("predicted label")

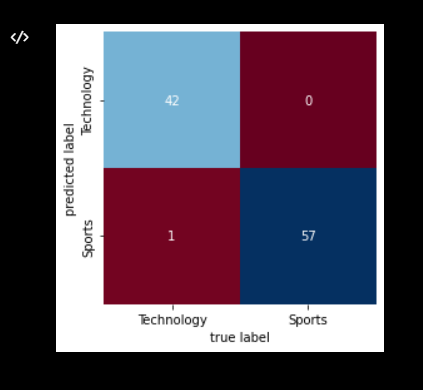
# print(cm)

# 42 articles are predicted technology and is technology.(true positive)

# 57 articles are predicted Sports and is sports. (true negative)

# 1 article was predicted Sports but was technology. (false positive)

# 0 articles were predicted as technology but was sports.(false negative)



Prediction test

With high accuracy we were confident that the model will succeed in predicting majority of the articles correctly and indeed it came out correctly.

#array which will contain predictions

tepredictions = []

#loop length = total amount of documents

#if predicion of that document is 1 which means it's technology append tepredictions with tech else append with sports

for i in range(len(X\_test)):

    if predictions[i] == 1:

        tepredictions.append("Sports")

    else:

        tepredictions.append("Technology")

#Init the dataframe for the model

model = pd.DataFrame({"actual\_label": list(y\_test), "prediction": tepredictions, "article":list(X\_test), 'confidence(tech) %':list(confidence\_list\_tech), 'confidence(sports) %': list(confidence\_list\_sports)})

#replace 0s in table with sports

model.replace(to\_replace=0, value="Technology", inplace=True)

#replace 1s in table with tech

model.replace(to\_replace=1, value="Sports", inplace=True)

model

Text

Description automatically generated

Function for custom user inputted article in .txt format

A user’s article gets preprocessed, count vectored and predicted with the naïve bayes.

#User custom article!!!!!!!!!!!!!!!!!!!!!

def predictArticle(article):

#Prediction value to be used later

    pred = []

#clean the article from extra words and chars

    clean\_article = cleanArticle(article)

#redeclare clean\_article as an iterable object (list)

    clean\_article = [clean\_article]

#convert clean\_article into an array of features

    cv\_article = cv.transform(clean\_article)

#predict the genre of the article

    temp\_nb = naive\_bayes.predict(cv\_article)

#confidence for the article

    article\_conf = conmodel.predict\_proba(cv\_article)

    conf\_tech = article\_conf[0][0]\*100

    conf\_sports = article\_conf[0][1]\*100

#set value of prediction from NB

    pred.append(temp\_nb[0])

    pred.append(conf\_tech)

    pred.append(conf\_sports)

    return pred

GUI

from pathlib import Path

import PySimpleGUI as sg

headings = [' Actual Label ', ' Prediction ', '     Article     ','confidence(sports) %','confidence(tech) %'] # headers for the table

table = model #actual table

values = table.values.tolist() # value for the table

def popup\_text(filename, text): # displays popup which contains the article the user selected

    layout = [  # declare the layout of the popup window

        [sg.Multiline(text, size=(100, 25)),], # display text on multiple lines instead of one long line, give it size of 100-width and 25-height

    ]

    # declare window with the name of the file, layout, modal = true means user wont be able to interact with any other windows

    win = sg.Window(filename, layout, modal=True)

    while True: # listening to events by using infinite loop

        event, values = win.read() # read events and their values from the window

        if event == sg.WINDOW\_CLOSED: # if event of closing the window occues break the loop

            break

    win.close() # close the window

conf\_sports = "Select a file"

conf\_tech = "select a file"

predVal = "Select a file" # predicted value (used later for displaying article's genre)

sg.theme("DarkTeal2") # declare theme frot the file

layout = [[sg.T("")],

 [sg.Text("Article Sorter")], # text in article sorter

 [sg.Text("---------------------------------------------------------------")],

 [sg.Text(accuracy)],

 [sg.Text(recallscore)],

 [sg.Text(precision)],

 # decalre tables where value  is entries converted to list from predicted Data and assign headings

 [sg.Table(values = values, headings = headings,

 auto\_size\_columns=False, # default size of columns  = false

 # column width = the size of the heading of that column + 1 for every single row, col\_widths must have a value of list

 col\_widths=list(map(lambda x:len(x)+1, headings)))],

 [sg.Text("---------------------------------------------------------------")],

 [sg.Text("Select an Article: "),

 sg.Input(key="txt"), # declare input witht the id of txt

 sg.FilesBrowse(file\_types=(('Text File',' \*.txt'),))], # browse through the files display the text files of extension .txt in file browser

 [sg.Button("Submit"),sg.Button("Exit")], # declare button for submit and exit

 [sg.Text("---------------------------------------------------------------")],

 # declare text which will display predVal(was declared before) and have id of -predVal-

 [sg.Text('Prediction: '), sg.Text(predVal, key="-predVal-")],

 [sg.Text('confidence(sports): '),sg.Text(conf\_sports,key="-conf\_sports-")],

 [sg.Text('confidence(tech): '),sg.Text(conf\_tech,key="-conf\_tech-")]]

# declare  window

window = sg.Window('Article Sorter', layout, size=(800,600))

while True: # infinite loop for listening

    event, values = window.read() # read events and values from the window

    # if event with the name of 'Exit' has occured or window has been closed break the loop and close the window

    if event == sg.WINDOW\_CLOSED or event == 'Exit':

        window.close()

        break

    elif event == 'Submit': # if event with the name of 'Submit' has occured

        print(values["txt"]) # print value of event 'txt' in the terminal (for debug purposes)

        filename = values['txt'] # assign event value which will contain the file address to variable called filename

        if Path(filename).is\_file(): # if the directory is the file and not folder or anything else

            try: # try block

                with open(filename, "rt", ) as t: # open file with "rt" which means read text

                    text = t.read() # read the text (convert to a string)

                    val\_pred = predictArticle(text) # return the prediction value and assign it to a variable

                    # print(val\_pred)

                if(val\_pred == 0): # if prediction value  is 0

                    # update the value of predVal (which is used for displaying the preidction variable on the window) to Technology

                    window['-predVal-'].update("Technology")

                else:

                    window['-predVal-'].update("Sports") # else update to Sports

                window['-conf\_sports-'].update(predictArticle(text)[2])

                window['-conf\_tech-'].update(predictArticle(text)[1])

                popup\_text(filename, text) # popup (function defined before) window with the selected text

            except Exception as e: # in case exception happens

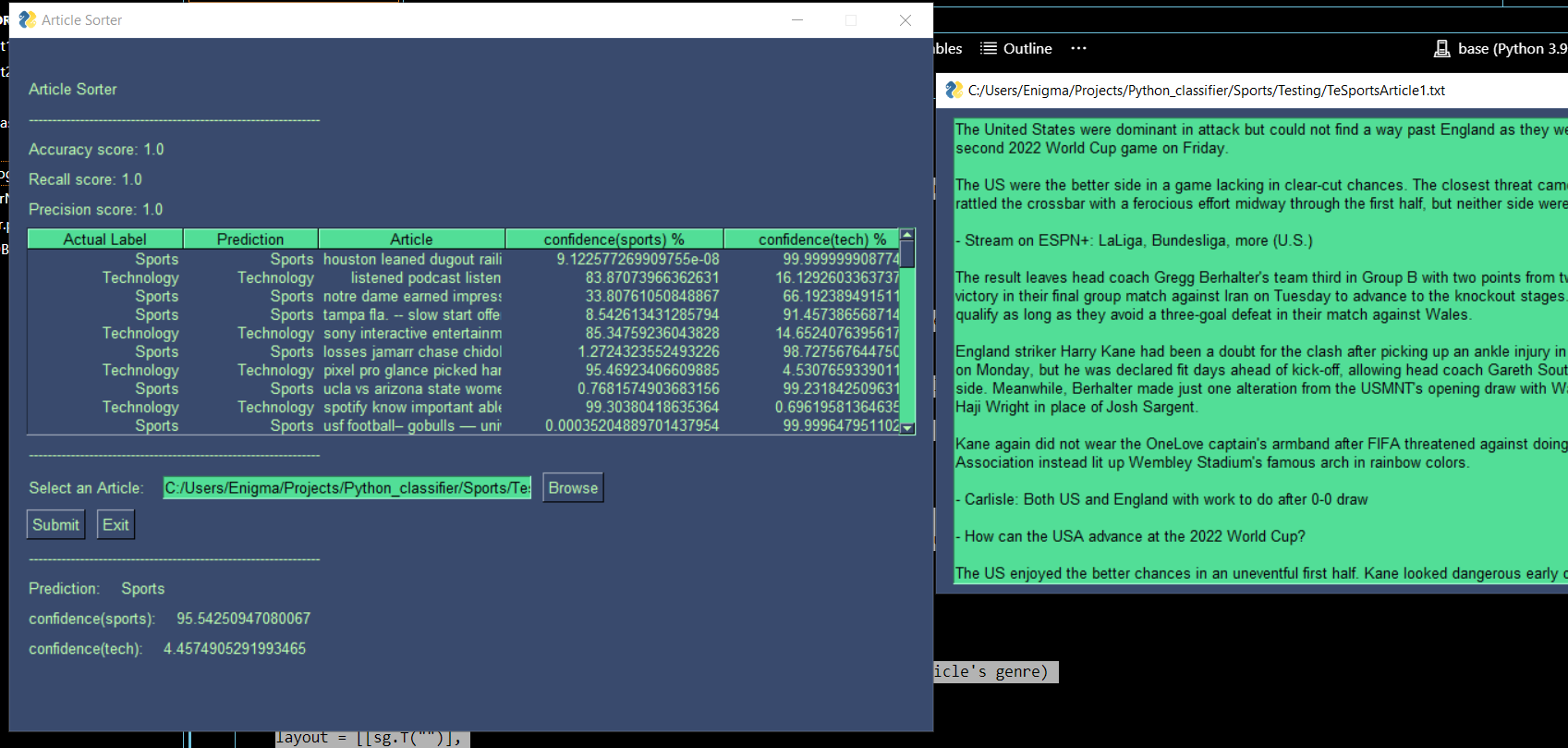
                print("Error: ", e) # console log exception

Technology test

Graphical user interface, text

Description automatically generated

Sports test



CODE

import pandas as pd

import os

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import string

# nltk.download()

# Pre-processing!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

def cleanArticle(file2read):

#declare extra words and punctuations

    stop\_words = set(stopwords.words('english') + list(string.punctuation))

#declare the returning variable

    article = ''

#declare extra chars of article which are not included in the previous packages

    customExtras = ["\n", "’", "“", "”", "'", "`","1","2","3","4","5","6","7","8","9","0","—",".."]

#function that converts arr of strings into one string (used later)

    def listToString(s):

        str1 = " "

        return (str1.join(s))

#remove extra chars in article

    for i in customExtras:

        file2read = file2read.replace(i, "")

#converts article into array of words

    word\_tokens= word\_tokenize(file2read.lower())

    # print(word\_tokens)

#empty array of clean words

    filtered\_file2read =[]

#check if word from article is the extra,if not assign the word in the clean words array

    for w in word\_tokens:

        if w not in stop\_words:

            filtered\_file2read.append(w)

#print(filtered\_file2read)

    article = listToString(filtered\_file2read)

#print(article)

    return article

#Technology folder !!!!!!!!!!!!!!!!!!!!!!

#change directory to technology training

#restart kernel if rerunning code

os.chdir("Technology/Training/")

dirTecharticle = os.listdir()

#cwd = os.getcwd()

#print(cwd)

articlestr = []

genrestr = []

#creating dictionary for data frame to be used

trdata = {'article': articlestr,

        'genre': genrestr

        }

#open training technology articles

#articles will be extracted and placed in the dataframe

for file in dirTecharticle:

     file2read = open(file,'r').read()

#apply the preprocessing function

     file2read = cleanArticle(file2read)

     articlestr.append(file2read)

     genrestr.append("Technology")

datatr = pd.DataFrame(trdata)

print(datatr)

# Sports folder!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

os.chdir("../../Sports/Training/")

dirSportsarticle= os.listdir()

# cwd = os.getcwd()

# print(cwd)

#articles from sports folder

for file in dirSportsarticle:

     file2read = open(file,'r').read()

     # apply the pre-processing function

     file2read = cleanArticle(file2read)

     articlestr.append(file2read)

     genrestr.append("Sports")

datatr = pd.DataFrame(trdata)

print(datatr)

#assign `label`` to dataframe if technology then label = 0  if not label = 1

datatr["label"] = datatr["genre"].apply(lambda x: 0 if x=="Technology" else 1)

datatr

from sklearn.model\_selection import train\_test\_split

#X - articles for testing and training y- labels for testing and training

X\_train, X\_test, y\_train, y\_test = train\_test\_split(datatr["article"], datatr["label"], random\_state=1)

# print(X\_train)

# print(y\_train)

# print(X\_test)

# print(y\_test)

from sklearn.feature\_extraction.text import CountVectorizer

 #tokenizes the documents and converts it to matrix

cv = CountVectorizer(strip\_accents="ascii")

#Learn the vocabulary dictionary and return document-term

X\_train\_cv = cv.fit\_transform(X\_train)

#Transform documents to document-term matrix.

X\_test\_cv = cv.transform(X\_test)

# print(X\_train\_cv)

from sklearn.naive\_bayes import MultinomialNB

from sklearn.svm import SVC #For confidence

conmodel = SVC(probability=True)

conmodel.fit(X\_train\_cv,y\_train)

class\_prob = conmodel.predict\_proba(X\_test\_cv)

# declare naive bayes model

naive\_bayes = MultinomialNB()

#train naive bayes model

naive\_bayes.fit(X\_train\_cv, y\_train)

#predict

predictions = naive\_bayes.predict(X\_test\_cv)

# print(predictions)

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

#display accuracy Percision and Recall

accuracy = "Accuracy score: " + str(accuracy\_score(y\_test, predictions))

precision = "Precision score: " + str(precision\_score(y\_test, predictions))

recallscore = "Recall score: " + str(recall\_score(y\_test, predictions))

print(accuracy)

print(precision)

#printing string elements from index 0 to 17

print(recallscore[0:18])

confidence\_list\_sports = []#confidence for sports

confidence\_list\_tech = []#confidence for tech

for i in range(len(class\_prob)):

    confidence\_list\_tech.append(class\_prob[i][0]\*100)

for i in range(len(class\_prob)):

    confidence\_list\_sports.append(class\_prob[i][1]\*100)

#print(class\_probabilities[0][0] \* 100) #1st [0] shows document if 2nd [0] shows confidence for sports if 2nd [1] shows confidence for tech

# print(len(confidence\_list\_tech))

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

#Storing confusion matrix in cm with the parameters of actual labels(y\_test) & predictions from naive bayes

cm = confusion\_matrix(y\_test, predictions)

#creating the graph for cm, square is the shape, annot is the number displayed in each quadrant,cmap is the colour

# & cbar is the represents the number of articles in each quadrant as colour(darker blue being the most and red the lowest)

sns.heatmap(cm, square=True, annot=True, cmap="RdBu", cbar=False,

#Quadrant labels

#positive = Technology

#negative = Sports

xticklabels=["Technology", "Sports"], yticklabels=["Technology", "Sports"])

#X & Y labels of the entire graph

plt.xlabel("true label")

plt.ylabel("predicted label")

# print(cm)

# 42 articles are predicted technology and is technology.(true positive)

# 57 articles are predicted Sports and is sports. (true negative)

# 1 article was predicted Sports but was technology. (false positive)

# 0 articles were predicted as technology but was sports.(false negative)

#array which will contain predictions

tepredictions = []

#loop length = total amount of documents

#if predicion of that document is 1 which means it's technology append tepredictions with tech else append with sports

for i in range(len(X\_test)):

    if predictions[i] == 1:

        tepredictions.append("Sports")

    else:

        tepredictions.append("Technology")

#Init the dataframe for the model

model = pd.DataFrame({"actual\_label": list(y\_test), "prediction": tepredictions, "article":list(X\_test), 'confidence(tech) %':list(confidence\_list\_tech), 'confidence(sports) %': list(confidence\_list\_sports)})

#replace 0s in table with sports

model.replace(to\_replace=0, value="Technology", inplace=True)

#replace 1s in table with tech

model.replace(to\_replace=1, value="Sports", inplace=True)

model

#User custom article!!!!!!!!!!!!!!!!!!!!!

def predictArticle(article):

#Prediction value to be used later

    pred = []

#clean the article from extra words and chars

    clean\_article = cleanArticle(article)

#redeclare clean\_article as an iterable object (list)

    clean\_article = [clean\_article]

#convert clean\_article into an array of features

    cv\_article = cv.transform(clean\_article)

#predict the genre of the article

    temp\_nb = naive\_bayes.predict(cv\_article)

#confidence for the article

    article\_conf = conmodel.predict\_proba(cv\_article)

    conf\_tech = article\_conf[0][0]\*100

    conf\_sports = article\_conf[0][1]\*100

#set value of prediction from NB

    pred.append(temp\_nb[0])

    pred.append(conf\_tech)

    pred.append(conf\_sports)

    return pred

from pathlib import Path

import PySimpleGUI as sg

headings = [' Actual Label ', ' Prediction ', '     Article     ','confidence(sports) %','confidence(tech) %'] # headers for the table

table = model #actual table

values = table.values.tolist() # value for the table

def popup\_text(filename, text): # displays popup which contains the article the user selected

    layout = [  # declare the layout of the popup window

        [sg.Multiline(text, size=(100, 25)),], # display text on multiple lines instead of one long line, give it size of 100-width and 25-height

    ]

    # declare window with the name of the file, layout, modal = true means user wont be able to interact with any other windows

    win = sg.Window(filename, layout, modal=True)

    while True: # listening to events by using infinite loop

        event, values = win.read() # read events and their values from the window

        if event == sg.WINDOW\_CLOSED: # if event of closing the window occues break the loop

            break

    win.close() # close the window

conf\_sports = "Select a file"

conf\_tech = "select a file"

predVal = "Select a file" # predicted value (used later for displaying article's genre)

sg.theme("DarkTeal2") # declare theme frot the file

layout = [[sg.T("")],

 [sg.Text("Article Sorter")], # text in article sorter

 [sg.Text("---------------------------------------------------------------")],

 [sg.Text(accuracy)],

 [sg.Text(recallscore)],

 [sg.Text(precision)],

 # decalre tables where value  is entries converted to list from predicted Data and assign headings

 [sg.Table(values = values, headings = headings,

 auto\_size\_columns=False, # default size of columns  = false

 # column width = the size of the heading of that column + 1 for every single row, col\_widths must have a value of list

 col\_widths=list(map(lambda x:len(x)+1, headings)))],

 [sg.Text("---------------------------------------------------------------")],

 [sg.Text("Select an Article: "),

 sg.Input(key="txt"), # declare input witht the id of txt

 sg.FilesBrowse(file\_types=(('Text File',' \*.txt'),))], # browse through the files display the text files of extension .txt in file browser

 [sg.Button("Submit"),sg.Button("Exit")], # declare button for submit and exit

 [sg.Text("---------------------------------------------------------------")],

 # declare text which will display predVal(was declared before) and have id of -predVal-

 [sg.Text('Prediction: '), sg.Text(predVal, key="-predVal-")],

 [sg.Text('confidence(sports): '),sg.Text(conf\_sports,key="-conf\_sports-")],

 [sg.Text('confidence(tech): '),sg.Text(conf\_tech,key="-conf\_tech-")]]

# declare  window

window = sg.Window('Article Sorter', layout, size=(800,600))

while True: # infinite loop for listening

    event, values = window.read() # read events and values from the window

    # if event with the name of 'Exit' has occured or window has been closed break the loop and close the window

    if event == sg.WINDOW\_CLOSED or event == 'Exit':

        window.close()

        break

    elif event == 'Submit': # if event with the name of 'Submit' has occured

        print(values["txt"]) # print value of event 'txt' in the terminal (for debug purposes)

        filename = values['txt'] # assign event value which will contain the file address to variable called filename

        if Path(filename).is\_file(): # if the directory is the file and not folder or anything else

            try: # try block

                with open(filename, "rt", ) as t: # open file with "rt" which means read text

                    text = t.read() # read the text (convert to a string)

                    val\_pred = predictArticle(text) # return the prediction value and assign it to a variable

                    # print(val\_pred)

                if(val\_pred == 0): # if prediction value  is 0

                    # update the value of predVal (which is used for displaying the preidction variable on the window) to Technology

                    window['-predVal-'].update("Technology")

                else:

                    window['-predVal-'].update("Sports") # else update to Sports

                window['-conf\_sports-'].update(predictArticle(text)[2])

                window['-conf\_tech-'].update(predictArticle(text)[1])

                popup\_text(filename, text) # popup (function defined before) window with the selected text

            except Exception as e: # in case exception happens

                print("Error: ", e) # console log exception