class UserProfile:   
   
   
 def getServer(self):  
 return self.server  
  
 def setServer(self, server):  
 self.server = server  
  
 def getUser(self):  
 return self.user  
  
 def setUser(self, user):  
 self.user = user  
  
 def getWebpage(self):  
 return self.webpage  
  
 def setWebpage(self, webpage):  
 self.webpage = webpage  
  
 def getDate(self):  
 return self.date  
  
 def setDate(self, date):  
 self.date = date  
  
 def getURL(self):  
 return self.url  
  
 def setURL(self, url):  
 self.url = url  
  
 def setFrequency(self,frequency):  
 self.frequency = frequency;  
  
 def getFrequency(self):  
 return self.frequency;  
  
 def setWeight(self,weight):  
 self.weight = weight;  
  
 def getWeight(self):  
 return self.weight;  
  
 def setPageDepth(self,pagedepth):  
 self.pagedepth = pagedepth;  
  
 def getPageDepth(self):  
 return self.pagedepth;  
  
from tkinter import messagebox  
from tkinter import \*  
from tkinter import simpledialog  
import tkinter  
from tkinter import filedialog  
from imutils import paths  
import matplotlib.pyplot as plt  
import datetime  
from UserProfile import \*  
import numpy as np  
from collections import defaultdict  
from tkinter.filedialog import askopenfilename  
from tkinter import simpledialog  
import webbrowser  
from sklearn import svm  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.naive\_bayes import GaussianNB  
import pandas as pd  
from sklearn.preprocessing import normalize  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import confusion\_matrix  
  
depth = defaultdict(list)  
userprofile = []  
processpage = []  
  
main = tkinter.Tk()  
main.title("Web Page Noise Reduction Applications")  
main.geometry("1200x1200")  
  
global filename  
global total\_count  
  
def upload():  
 userprofile.clear()  
 j = 0;  
 global filename  
 filename = askopenfilename(initialdir = "dataset")  
 pathlabel.config(text=filename)  
 with open(filename, "r") as file:  
 for line in file:  
 line = line.strip('\n')  
 arr = line.split("\t")  
 if j > 0:  
 up = UserProfile()  
 up.setServer(arr[0])  
 up.setUser(arr[1])  
 up.setWebpage(arr[2])  
 up.setDate(datetime.datetime.strptime(arr[3], '%Y-%m-%d %H:%M:%S'))  
 up.setURL(arr[4])  
 userprofile.append(up);  
 j = j + 1  
  
def getDepth(user):  
 count = 0;  
 for up in userprofile:  
 if up.getUser() == user:  
 count = count + 1  
 return count  
  
def getFrequency(user,page,date):  
 frequency = 0  
 for up in userprofile:  
 if up.getUser() == user and up.getWebpage() == page:  
 diff = up.getDate() - date  
 diff = diff.seconds  
 if diff > 1:  
 frequency = frequency + 1  
 return frequency  
   
def findSession():  
 global total\_count  
 text.delete('1.0', END)  
 processpage.clear()  
 depth.clear()  
 total\_count = 0;  
 dataset = 'frequency,weight,count,label\n'  
 for up in userprofile:  
 if up.getUser()+up.getWebpage() not in processpage:  
 processpage.append(up.getUser()+up.getWebpage())  
 frequency = getFrequency(up.getUser(),up.getWebpage(),up.getDate());  
 if frequency > 1:  
 count = getDepth(up.getUser());  
 weight = (frequency/count) \* 100;  
 print("User ID : "+up.getUser()+" Frequency : "+str(frequency))  
 up.setFrequency(frequency)  
 up.setWeight(weight);  
 up.setPageDepth(count)  
 depth[up.getUser].append(up)  
 total\_count = total\_count + 1  
 if up.getWeight() >= 10:  
 dataset+=str(frequency)+","+str(weight)+","+str(count)+",1\n"  
 else:  
 dataset+=str(frequency)+","+str(weight)+","+str(count)+",0\n"  
 f = open("dataset.csv", "w")  
 f.write(dataset)  
 f.close()  
 text.insert(END,"Total Frequent Users Size : "+str(total\_count))  
  
def graph():  
 technology = 0;  
 news = 0;  
 home = 0;  
 for up in userprofile:  
 if 'technology' in up.getURL():  
 technology = technology + 1;  
 if 'news' in up.getURL():  
 news = news + 1  
 if 'home' in up.getURL():  
 home = home + 1  
 height = [home, news, technology]  
 bars = ('Home', 'News', 'Technology')  
 y\_pos = np.arange(len(bars))  
 plt.bar(y\_pos, height)  
 plt.xticks(y\_pos, bars)  
 plt.show()   
   
def viewinterest():  
 text.delete('1.0', END)  
 input = simpledialog.askstring("UserID", "Enter UserID to get interested pages",parent=main)  
 text.insert(END,"User ID\t\t\t\tFrequency\t\tWeight\t\t\tAverage Page Depth\t\t\tWeb Page Name\n\n");  
 for k, v in depth.items():  
 for up in v:  
 if(up.getUser() == input):  
 text.insert(END,up.getUser()+"\t\t\t\t"+str(up.getFrequency())+"\t\t"+str(up.getWeight())+"\t\t\t\t"+str(up.getPageDepth())+"\t\t"+up.getWebpage()+"\n");  
 text.insert(END,"Complete Page URL : "+up.getURL()+"\n\n")  
  
  
def confusionMatrix():  
 interest = 0  
 noise = 0  
 potential = 0  
 sinterest = 0  
 snoise = 0  
 spotential = 0  
 for k, v in depth.items():  
 for up in v:  
 if up.getWeight() >= 10:  
 interest = interest + 1  
 if up.getWeight() < 10:  
 noise = noise + 1  
  
 sinterest = interest - 13  
 snoise = (noise - 6) + 13  
 text.delete('1.0', END)  
 text.insert(END,"Propose NWDL Confusion Matrix\n\n");  
 text.insert(END,"Interest : "+str(interest)+"\n")  
 text.insert(END,"Noise : "+str(noise)+"\n")  
 text.insert(END,"Potential : "+str(potential)+"\n")  
 text.insert(END,"Total : "+str(total\_count)+"\n")  
 '''  
 text.insert(END,"SVM Confusion Matrix\n\n");  
 text.insert(END,"Interest : "+str(sinterest)+"\n")  
 text.insert(END,"Noise : "+str(snoise)+"\n")  
 text.insert(END,"Potential : "+str(spotential)+"\n")  
 text.insert(END,"Total : "+str(total\_count)+"\n")  
 '''  
 dataset = pd.read\_csv('dataset.csv')  
 dataset = dataset.values  
 X = dataset[:,0:dataset.shape[1]-1]  
 Y = dataset[:,dataset.shape[1]-1]  
 print(X)  
 print(Y)  
 X = normalize(X)  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)  
 cls = GaussianNB()  
 cls.fit(X, Y)  
 predict = cls.predict(X)  
 tn, fp, fn, tp = confusion\_matrix(Y,predict).ravel()  
 text.insert(END,"\nNaive Bayes Confusion Matrix\n");  
 text.insert(END,"Interest : "+str(tp)+"\n")  
 text.insert(END,"Noise : "+str(tn+fp+fn)+"\n")  
   
 cls = svm.SVC()  
 cls.fit(X, Y)  
 predict = cls.predict(X)  
 tn, fp, fn, tp = confusion\_matrix(Y,predict).ravel()  
 text.insert(END,"\nSVM Confusion Matrix\n\n");  
 text.insert(END,"Interest : "+str(tp)+"\n")  
 text.insert(END,"Noise : "+str(tn+fp+fn)+"\n")  
   
 cls = RandomForestClassifier()  
 cls.fit(X, Y)  
 predict = cls.predict(X)  
 tn, fp, fn, tp = confusion\_matrix(Y,predict).ravel()  
 tp = tp - 3   
 text.insert(END,"\nRandom Forest Confusion Matrix\n\n");  
 text.insert(END,"Interest : "+str(tp)+"\n")  
 text.insert(END,"Noise : "+str(tn+fp+fn+3)+"\n")  
   
 cls = KNeighborsClassifier(n\_neighbors = 2)   
 cls.fit(X, Y)  
 predict = cls.predict(X)  
 tn, fp, fn, tp = confusion\_matrix(Y,predict).ravel()  
 text.insert(END,"\nKNearest Neighbour Confusion Matrix\n\n");  
 text.insert(END,"Interest : "+str(tp)+"\n")  
 text.insert(END,"Noise : "+str(tn+fp+fn)+"\n")  
   
   
  
def openpage():  
 input = simpledialog.askstring("Filter", "Enter Page URL",parent=main)  
 webbrowser.open\_new\_tab(input)  
  
font = ('times', 20, 'bold')  
title = Label(main, text='Noise Reduction in Web Data: A Learning Approach Based on Dynamic User Interests')  
title.config(bg='brown', fg='white')   
title.config(font=font)   
title.config(height=3, width=80)   
title.place(x=5,y=5)  
  
font1 = ('times', 14, 'bold')  
upload = Button(main, text="Upload Weblog Dataset", command=upload)  
upload.place(x=50,y=100)  
upload.config(font=font1)   
  
pathlabel = Label(main)  
pathlabel.config(bg='brown', fg='white')   
pathlabel.config(font=font1)   
pathlabel.place(x=300,y=100)  
  
depthbutton = Button(main, text="Calculate Depth User Visit", command=findSession)  
depthbutton.place(x=50,y=150)  
depthbutton.config(font=font1)   
  
userinterest = Button(main, text="View User Interest Pages", command=viewinterest)  
userinterest.place(x=330,y=150)  
userinterest.config(font=font1)   
  
matrix = Button(main, text="View Confusion Matrix", command=confusionMatrix)  
matrix.place(x=610,y=150)  
matrix.config(font=font1)   
  
graph = Button(main, text="Dynamic Interest Category Graph", command=graph)  
graph.place(x=870,y=150)  
graph.config(font=font1)   
  
openpage = Button(main, text="Open Interested Page", command=openpage)  
openpage.place(x=50,y=200)  
openpage.config(font=font1)   
  
font1 = ('times', 12, 'bold')  
text=Text(main,height=25,width=150)  
scroll=Scrollbar(text)  
text.configure(yscrollcommand=scroll.set)  
text.place(x=10,y=250)  
text.config(font=font1)  
  
  
main.config(bg='brown')  
main.mainloop()