**Приложение А**

**from** tkinter.filedialog **import** \*  
**from** sklearn.svm **import** SVC  
**from** pylab **import** \*  
**from** tkinter **import** \*  
**from** PIL **import** Image, ImageTk  
**from** tkinter **import** messagebox  
**import** pickle  
**import** ast  
**from** sklearn.feature\_extraction.text **import** CountVectorizer  
**from** sklearn.svm.libsvm **import** decision\_function  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.svm **import** SVC  
**import** json  
cv = CountVectorizer(max\_features = 700)  
name=[]  
coefX=[]  
clas=[]  
X=[]  
name.clear()  
F0 = np.array([[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]])  
F1 = np.array([[-1, -1, 1, 1], [-1, -1, 1, 1], [-1, -1, 1, 1], [-1, -1, 1, 1]])  
F2 = np.array([[1, 1, 1, 1], [1, 1, 1, 1], [-1, -1, -1, -1], [-1, -1, -1, -1], ])  
F3 = np.array([[-1, -1, 1, 1], [-1, -1, 1, 1], [1, 1, -1, -1], [1, 1, -1, -1]])  
F4 = np.array([[-1, 1, 1, -1], [-1, 1, 1, -1], [-1, 1, 1, -1], [-1, 1, 1, -1]])  
F5 = np.array([[-1, -1, -1, -1], [1, 1, 1, 1], [1, 1, 1, 1], [-1, -1, -1, -1]])  
F6 = np.array([[-1, 1, 1, -1], [-1, 1, 1, -1], [1, -1, -1, 1], [1, -1, -1, 1]])  
F7 = np.array([[1, 1, -1, -1], [-1, -1, 1, 1], [-1, -1, 1, 1], [1, 1, -1, -1]])  
F8 = np.array([[1, -1, -1, 1], [-1, 1, 1, -1], [-1, 1, 1, -1], [1, -1, -1, 1]])  
F9 = np.array([[1, -1, 1, -1], [1, -1, 1, -1], [1, -1, 1, -1], [1, -1, 1, -1]])  
F10 = np.array([[-1, -1, -1, -1], [1, 1, 1, 1], [-1, -1, -1, -1], [1, 1, 1, 1]])  
F11 = np.array([[1, -1, 1, -1], [1, -1, 1, -1], [-1, 1, -1, 1], [-1, 1, -1, 1]])  
F12 = np.array([[1, 1, -1, -1], [-1, -1, 1, 1], [1, 1, -1, -1], [-1, -1, 1, 1]])  
F13 = np.array([[-1, 1, -1, 1], [1, -1, 1, -1], [1, -1, 1, -1], [-1, 1, -1, 1]])  
F14 = np.array([[1, -1, -1, 1], [-1, 1, 1, -1], [-1, 1, 1, -1], [-1, -1, 1, 1]])  
F15 = np.array([[-1, 1, -1, 1], [1, -1, 1, -1], [-1, 1, -1, 1], [1, -1, 1, -1]])  
  
**def** gray\_scale(source\_name):  
 source = Image.open(source\_name)  
 result = Image.new(**'L'**, source.size)  
 **for** x **in** range(source.size[0]):  
 **for** y **in** range(source.size[1]):  
 r, g, b = source.getpixel((x, y))  
 gray = round((r + g + b) / 3)  
 result.putpixel((x, y), (gray))  
 **return** result  
  
  
**def** image\_to\_massive(source):  
 width = source.size[0] *# Определяем ширину.* height = source.size[1] *# Определяем высоту.* massive = np.zeros((height, width))  
 **for** x **in** range(width):  
 **for** y **in** range(height):  
 gray = source.getpixel((x, y))  
 massive[y][x] = gray  
 **return** massive  
  
**def** minimum(source):  
 minimum = 255  
 **for** x **in** range(source.shape[0]):  
 **for** y **in** range(source.shape[1]):  
 **if** (source[x][y] < minimum):  
 minimum = source[x][y]  
 **for** x **in** range(source.shape[0]):  
 **for** y **in** range(source.shape[1]):  
 source[x][y] = source[x][y] - minimum  
 **if** (source[x][y] < 0):  
 source[x][y] = 0  
 **return** source  
  
**def** maximum(source):  
 maximum = 0  
 **for** x **in** range(source.shape[0]):  
 **for** y **in** range(source.shape[1]):  
 **if** (source[x][y] > maximum):  
 maximum = source[x][y]  
 **for** x **in** range(source.shape[0]):  
 **for** y **in** range(source.shape[1]):  
 source[x][y] = round(source[x][y] / maximum, 1)  
 **return** source  
  
**def** norma(source):  
 massive = image\_to\_massive(source)  
 massive = minimum(massive)  
 massive = maximum(massive)  
 **return** massive  
  
**def** q\_preobr(source):  
 q = 4  
 N = int(source.shape[1] / 4) *# stolb* M = int(source.shape[0] / 4) *# stroka* result = np.zeros((q, q))  
 **for** i **in** range(q): *# stolb* **for** j **in** range(q): *# stroka* sum = 0  
 **for** y **in** range(i \* M, (i + 1) \* M):  
 **for** x **in** range(j \* N, (j + 1) \* N):  
 sum = sum + source[y][x]  
 result[i][j] = sum  
 **return** result  
  
**def** m\_create(source, f):  
 res = 0  
 **for** x **in** range(source.shape[0]):  
 **for** y **in** range(source.shape[1]):  
 res = res + source[x][y] \* f[x][y]  
 **return** round(res, 1)  
  
**def** m\_mass\_create(source):  
 mass = []  
 mass.append(m\_create(source, F0))  
 mass.append(m\_create(source, F1))  
 mass.append(m\_create(source, F2))  
 mass.append(m\_create(source, F3))  
 mass.append(m\_create(source, F4))  
 mass.append(m\_create(source, F5))  
 mass.append(m\_create(source, F6))  
 mass.append(m\_create(source, F7))  
 mass.append(m\_create(source, F8))  
 mass.append(m\_create(source, F9))  
 mass.append(m\_create(source, F10))  
 mass.append(m\_create(source, F11))  
 mass.append(m\_create(source, F12))  
 mass.append(m\_create(source, F13))  
 mass.append(m\_create(source, F14))  
 mass.append(m\_create(source, F15))  
 **return** mass  
  
**def** classification(ImageSourse):  
 x,y = (getclassific()[:])  
 clf =SVC()  
 clf.fit(x, y)  
 SVC(C=1.0, cache\_size=200, class\_weight=**None**, coef0=0.0,  
 decision\_function\_shape=**'ovr'**, degree=3, gamma=**'auto'**, kernel=**'rbf'**,  
 max\_iter=-1, probability=**False**, random\_state=**None**, shrinking=**True**,  
 tol=0.001, verbose=**False**)  
 norm\_face(ImageSourse, **"test"**)  
 gray1 = gray\_scale(ImageSourse)  
 norm\_mass = norma(gray1)  
 q4 = m\_mass\_create(q\_preobr(norm\_mass))  
 **if** (clf.decision\_function([q4])<1):  
 flag=0  
 **else**:  
 flag=1  
 pred = clf.predict([q4])  
 y.clear()  
 **return** pred,flag  
  
**def** MassAppend(namefile, ImageSourse): *# заполняем одну из строк в массиве* norm\_face( ImageSourse, namefile)  
 gray1 = gray\_scale(ImageSourse)  
 norm\_mass = norma(gray1)  
 q1 = m\_mass\_create(q\_preobr(norm\_mass))  
 appendXCoefFile(namefile, q1)  
  
**def** norm\_face(SourseImage, namefile):  
 im = Image.open(SourseImage)  
 imshow(im)  
 print(**"Please click 2 points"**)  
 X = ginput(2)  
 print(**"you clicked:"**, X)  
 show()  
 x1 = int(X[0][0])  
 y1 = int(X[0][1])  
 x2 = int(X[1][0])  
 y2 = int(X[1][1])  
 box = (x1, y1, x2, y2)  
 region = im.crop(box)  
 X.clear()  
 print(namefile)  
 region.save(str(namefile) + **'.jpg'**)  
  
**def** openfile():  
 op = askopenfilename() *# окно выбора файла* **return** op  
  
**def** openNameFile():*# Y* f = open(**'name.txt'**, **'a'**)  
 f.close()  
 f = open(**'name.txt'**, **'r'**)  
 name = [line.strip() **for** line **in** f]  
 f.close()  
 **return** name  
  
**def** appendNameFile(mystr): *#input new name* f = open(**'name.txt'**, **'a'**)  
 f.write(mystr + **'\n'**)  
 f.close()  
  
**def** openXCoefFile(namefile):  
 f = open(str(namefile)+**'.txt'**, **'r'**)  
 npCoef=[line.strip() **for** line **in** f]  
 **return** npCoef  
  
**def** appendXCoefFile(namefile, coef):*# photo one human* f = open(str(namefile) + **'.txt'**, **'a'**)  
 f.write(**"%s\n"** % coef)  
 f.close()  
  
**def** getclassific():  
 name.clear()  
 clas.clear()  
 **for** Names **in** openNameFile():  
 **for** i **in** openXCoefFile(Names):  
 print(i)  
 clas.append(json.loads(i))  
 name.append(Names)  
 ret=clas[:]  
 res=name[:]  
 **return** ret,res  
  
  
**def** button\_1():  
 result,fl=classification(openfile())  
 print ()  
 messagebox.showinfo(**"Информация"**, **"Формирование признака завершено"**)  
 **if** (fl==0):  
 top1 = Toplevel()  
 lable4 = Label(top1,  
 text=**"Результат идентификации: "** + str(result[0]),  
 font=**' arial 15 '** , fg=**'green'**)  
 lable4.grid(row=1, column=0,columnspan = 2 )  
 button9 = Button(top1, text=**'Закрыть'**,  
 font=**'arial 14'**,fg= **"red"**, command=top1.destroy)  
 button9.grid(row=3, column=1)  
 image1 = Image.open(**"test.jpg"**)  
 photo1 = ImageTk.PhotoImage(image1)  
 image2 = Image.open(str(result[0]) + **".jpg"**)  
 photo2 = ImageTk.PhotoImage(image2)  
 label5 = Label(top1,image=photo1)  
 label5.image = photo1  
 label5.grid(row=2, column=0)  
 label6 = Label(top1,image=photo2)  
 label6.image = photo2  
 label6.grid(row=2, column=1)  
 **else**:  
 top1 = Toplevel()  
 lable4 = Label(top1,  
 text=**"Результат идентификации: Человек не идентифицирован"**,  
 font=**' arial 15 '** , fg=**'green'**)  
 lable4.grid(row=1, column=0, columnspan=2)  
 button9 = Button(top1, text=**'Закрыть'**,  
 font=**'arial 14'**, fg=**"red"**, command=top1.destroy)  
 button9.grid(row=3, column=1)  
 image1 = Image.open(**"test.jpg"**)  
 photo1 = ImageTk.PhotoImage(image1)  
 label5 = Label(top1, image=photo1)  
 label5.image = photo1  
 top1.mainloop()  
  
**def** button\_2():  
 top = Toplevel()  
 mas=[]  
 mystring = StringVar()  
 lable2 = Label(top,  
 text=**'Введите имя человека,\n чье фото желаете добавить'**,  
 font=**' arial 11 '**)  
 lable2.grid(row=1, column=0,sticky=W)  
 text1 = Entry(top,textvariable = mystring)  
 text1.grid(row=2, column=0)  
  
 **def** cliced2():  
 **if** openNameFile().count(mystring.get())==0:  
 messagebox.showerror(**"Ошибка"**, **"Такого человека нет в базе!"**)  
 top.destroy  
 **else**:  
 MassAppend(mystring.get(),openfile())  
 messagebox.showinfo(**"Информация"**, **"Формирование признака завершено"**)  
 mas.clear()  
 button6 = Button(top, text=**'ok'**, font=**'arial 14'**, command=cliced2)  
 button6.grid(row=3, column=0)  
 button4 = Button(top, text=**'Закрыть'**, font=**'arial 14'**,fg= **"red"**,command=top.destroy)  
 button4.grid(row=3, column=1)  
 top.mainloop()  
  
**def** button\_3():  
 top3 = Toplevel()  
 mystring3 = StringVar()  
 lable3 = Label(top3,  
 text=**'Введите имя человека,\n которого желаете добавить'**,  
 font=**' arial 11 '**)  
 lable3.grid(row=1, column=0,sticky=W)  
 text2 = Entry(top3,textvariable = mystring3)  
 text2.grid(row=2, column=0)  
  
 **def** cliced3():  
 **if** openNameFile().count(mystring3.get())!=0:  
 messagebox.showerror(**"Ошибка"**, **"Такой человек уже есть в базе!"**)  
 top3.destroy  
 **else**:  
 appendNameFile(mystring3.get())  
 MassAppend(mystring3.get(),openfile())  
 messagebox.showinfo(**"Информация"**, **"Формирование признака завершено"**)  
 print(openNameFile())  
 button7 = Button(top3, text=**'ok'**, font=**'arial 14'**, command=cliced3)  
 button7.grid(row=3, column=0)  
 button8 = Button(top3, text=**'Закрыть'**, font=**'arial 14'**,fg= **"red"**, command=top3.destroy)  
 button8.grid(row=3, column=1)  
  
**def** main():  
 root = Tk()  
 root.title( **' Идентификация '** )  
 root.configure(background=**"light gray"**)  
 lable1 = Label( root ,  
 text = **' Эта программа прeдназначена для проведения идентификации по фото. \n '   
 ' Пожалуйста выберите один из 3х вариантов '** ,  
 font = **' arial 11 '** )  
 lable1.grid( row = 1 , column = 0 , columnspan = 3 )  
 button1 = Button(root,text=**'Сравнить'**,background=**"light gray"**, font=**'arial 14'**,command=button\_1)  
 button1.grid(row=3,column=0)  
 button2=Button(root,text=**'Добавить фото'**,background=**"light gray"**,font=**'arial 14'**,command=button\_2)  
 button2.grid(row=3,column=1)  
 button3=Button(root,text=**'Добавить человека'**,background=**"light gray"**,font=**'arial 14'**,command=button\_3)  
 button3.grid(row=3,column=2)  
 button5 = Button(root, text=**'Выход'**, background=**"light gray"**, fg= **"red"**, font=**'arial 14'**,command=root.quit)  
 button5.grid(row=4, column = 1)  
 root.mainloop()  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()  
 name.clear()  
 coefX.clear()  
 clas.clear()