**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ(МИИТ)**)

**Институт управления и цифровых технологий**

Кафедра «Вычислительные системы, сети и информационная безопасность»

**ОТЧЕТ ПО лабораторной работЕ №8**

**«РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ»**

**ПО дисциплине** **«Нейроинформатика»**

*Направление:* 10.03.01*Информационная безопасность*

*Профиль:**Безопасность компьютерных систем*

Выполнил:   
студент группы УИБ-311

Москаленко В.А.

Проверил:

Доцент Малинский С.В.

(должность, ФИО)

**МОСКВА 2024**

**СОДЕРЖАНИЕ**

[1 ЗАДАНИЕ 3](#_Toc184175522)

[2 ТЕОРЕТИЧЕСКАЯ ИНФОРМАЦИЯ 4](#_Toc184175523)

[2.1 Метод градиентного спуска 4](#_Toc184175524)

[2.2 Нормализация данных 4](#_Toc184175525)

[3 РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ 6](#_Toc184175526)

[РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 11](#_Toc184175527)

[ЗАКЛЮЧЕНИЕ 12](#_Toc184175528)

[ПРИЛОЖЕНИЕ А 13](#_Toc184175529)

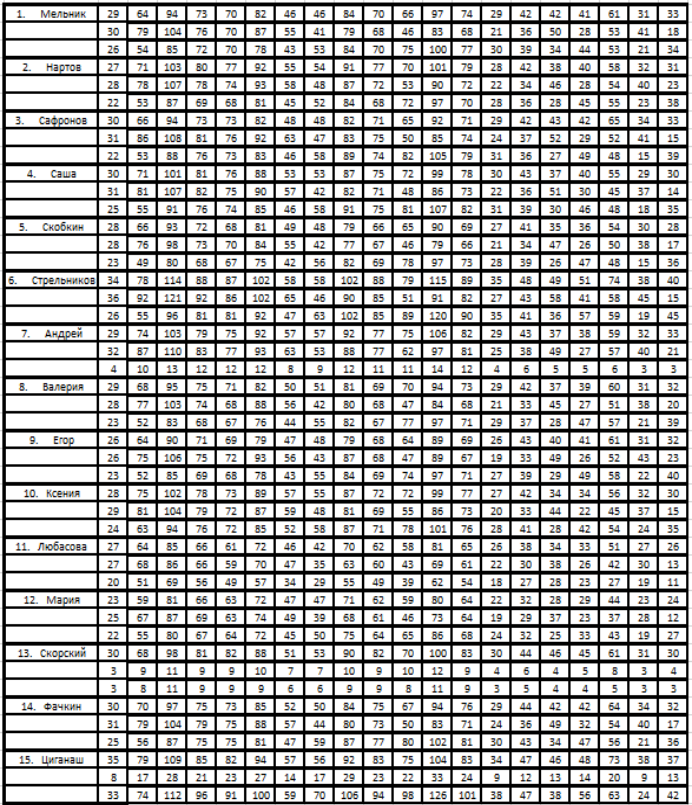
[ПРИЛОЖЕНИЕ Б 17](#_Toc184175530)

**1 ЗАДАНИЕ**

В файле собраны метрики для 15 человек. Для каждого представлено три строки, каждая из которых содержит набор из 20 признаков. Требуется разработать систему распознавания лиц, применяющую алгоритм градиентного спуска для классификации с использованием персептронов.

Вариант 3.1

Таблица 1 – Обучающая выборка для задачи распознавания лиц



**2 ТЕОРЕТИЧЕСКАЯ ИНФОРМАЦИЯ**

## **2.1 Метод градиентного спуска**

Градиентный спуск – это фундаментальный метод оптимизации, который применяется для нахождения минимального значения функции. В контексте машинного обучения эта функция представляет собой функциюпотерь, которая оценивает, насколько хорошо модель предсказывает результаты. Задача состоит в минимизации этой функции, чтобы добиться наиболее точных предсказаний.

Метод градиентного спуска работает итеративно, обновляя параметры модели по следующей формуле (1).

(1)

Где – скорость обучения,

– разница ожидаемой и предсказанной суммами,

– вектор признаков объектов.

Скорость обучения) – ключевой параметр градиентного спуска. Правильный выбор значения скорости обучения может существенно повлиять на сходимость алгоритма.

## **2.2 Нормализация данных**

Нормализация данных является важным этапом подготовки данных перед обучением моделей машинного обучения. Цель нормализации – привести все признаки к единому масштабу, что позволяет избежать доминирования признаков с большими значениями над признаками с меньшими.

В данной работе использовался метод минимально-максимальной нормализации (Min-Max Scaling). Он приводит данные к диапазону от 0 до 1. Это позволяет алгоритму быстрее сходиться и лучше учитывать все признаки. Этот метод можно описать следующей формулой (3).

(3)

Где – множество объектов.

Без нормализации градиентный спуск может работать нестабильно или медленно сходиться, так как значения признаков разного масштаба создают проблемы для корректного обновления весов.

# **3 РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ**

В процессе реализации данной задачи на каждом этапе были проведены работы по разделению лиц на две категории: однозначно различимые и сложные для распознавания лица.

Такой подход позволяет решать задачу поэтапно, постепенно снижая общую ошибку модели. Для каждой итерации алгоритма была сформулирована стратегия обучения, которая заключается в следующем:

* На шаге однослойный персептрон настраивается для эффективного распознавания лиц, принадлежащих к однозначно различимым классам. Эти лица являются "своими" для данного шага.
* Также персептрон обучается распознавать ещё одну дополнительную категорию – лица, которые ранее были классифицированы плохо или с низкой точностью.

Таким образом, итоговая система построена как последовательность шагов. На каждом из них для были найдены оптимальные веса для персептронов. Каждый шаг состоит из обучения персептрона на определенном наборе классов, выбранных заранее. При невозможности классификации класса происходит переход к следующему шагу – к обучению персептрона на остальных классах.

График точности персептрона на первом шаге работы программы представлен на рисунке 1.

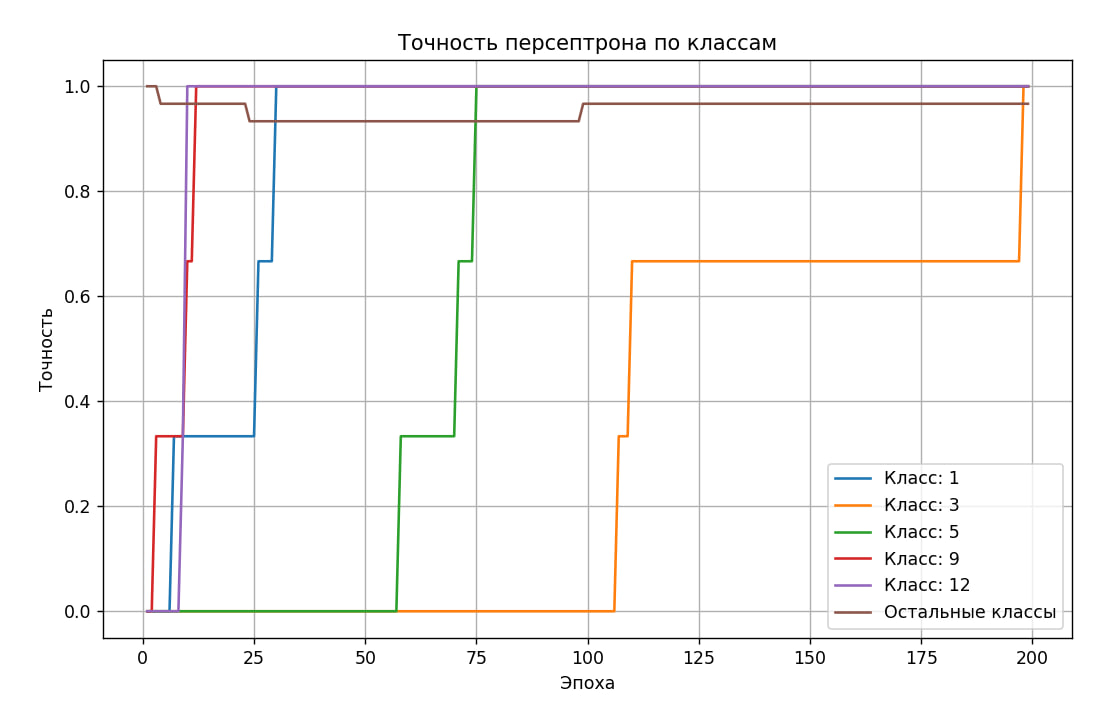


Рисунок 1 – График обучения первого персептрона

Класс: 1 – Мельник, Класс: 3 – Сафронов, Класс: 5 – Скобкин, Класс: 9 – Егор, Класс: 12 – Мария. График точности персептрона на втором шаге работы программы представлен на рисунке 2.

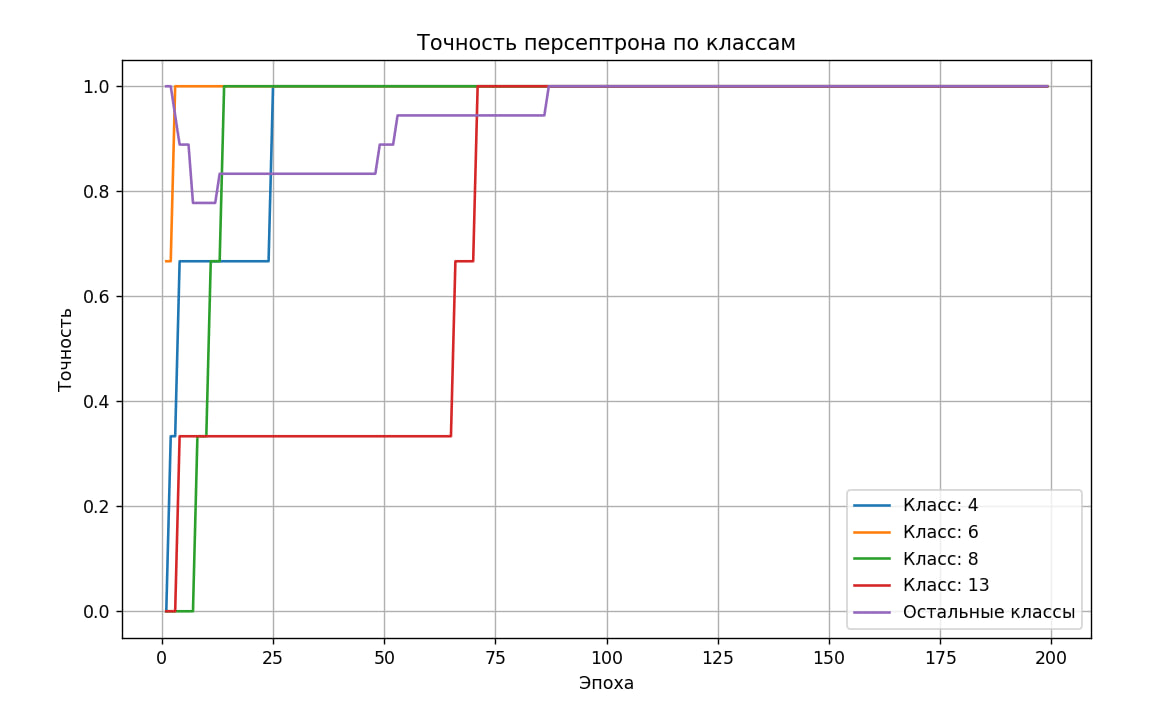


Рисунок 2 – График обучения второго персептрона

Класс: 4 – Саша, Класс: 6 – Стрельников, Класс: 8 – Валерия, Класс: 13 – Скорский. График точности персептрона на третьем шаге работы программы представлен на рисунке 3.

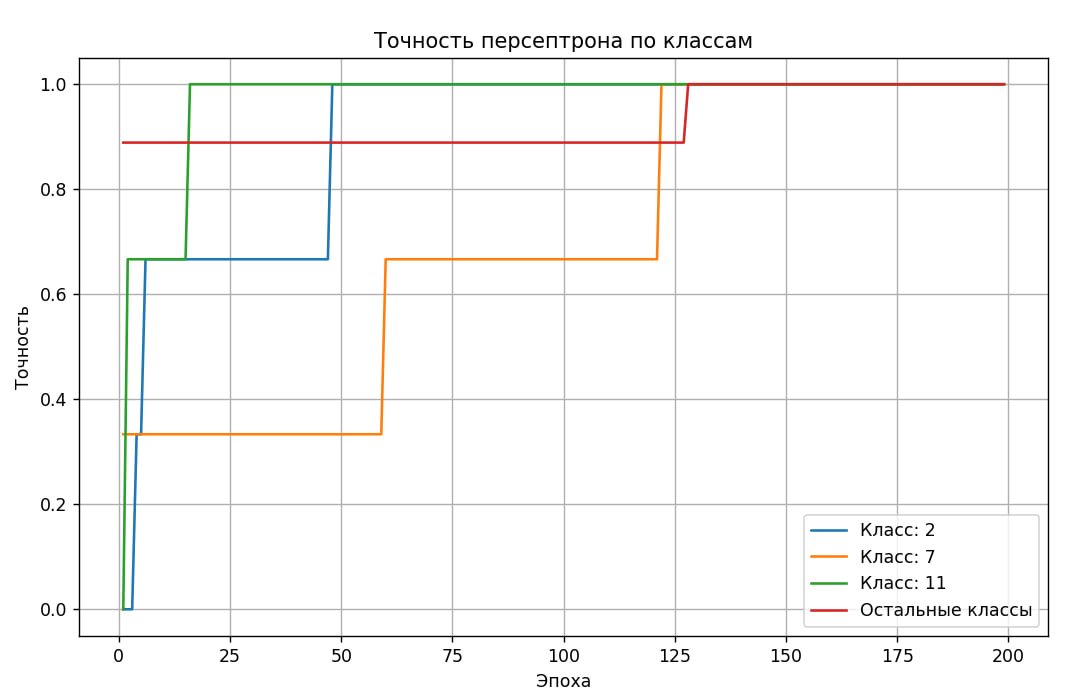


Рисунок 3 – График обучения третьего персептрона

Класс: 2 – Нартов, Класс: 7 – Андрей, Класс: 11 – Любасова. График точности персептрона на четвертом шаге работы программы представлен на рисунке 3.

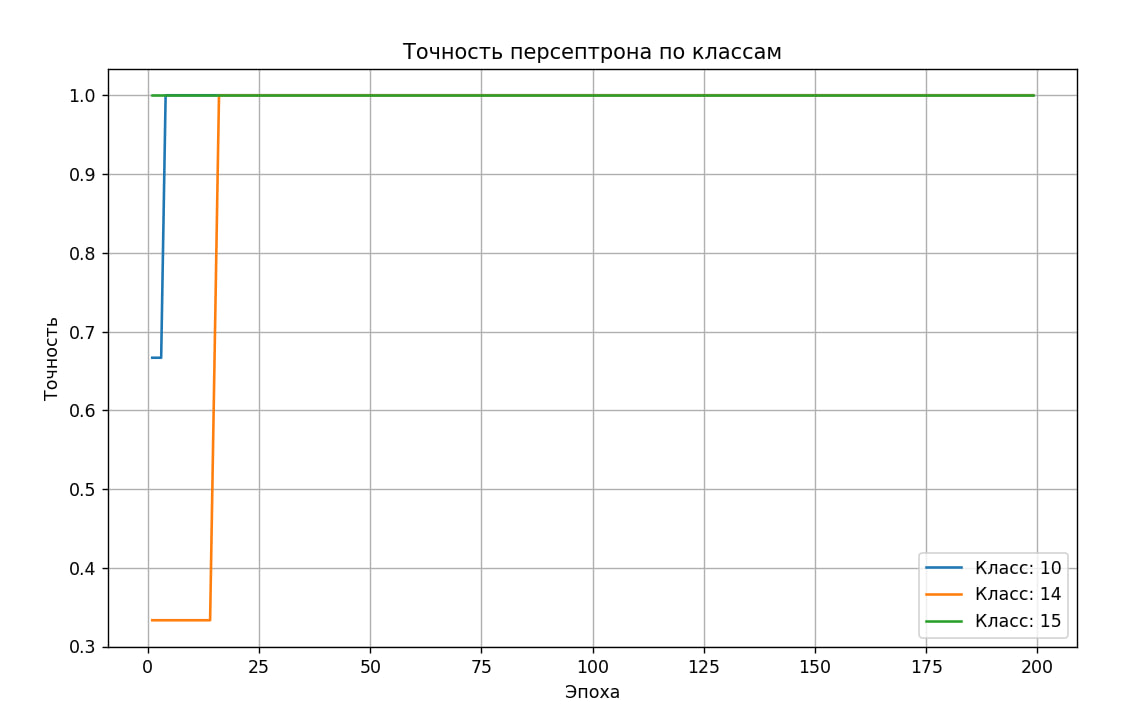


Рисунок 4 – График обучения четвертого персептрона

Класс: 10 – Ксения, Класс: 14 – Фачкин, Класс: 15 – Цигинаш. В таблице 2 можно увидеть 4 шага работы программы, какие классы распознает определенный персептрон.

Таблица 2 – Персептроны и распознаваемые ими классы

|  |  |
| --- | --- |
| Персептрон № | Классы |
| 1 | Мельник, Сафронов, Скобкин, Егор, Мария |
| 2 | Саша, Стрельников, Валерия, Скорский |
| 3 | Нартов, Андрей, Любасова |
| 4 | Ксения, Фачкин, Цигинаш |

# **РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

В результате была разработана программа для распознавания лиц, применяющая алгоритм градиентного спуска для классификации с использованием персептронов. Каждый персептрон использует в качестве входных данных вектор из 20 значений, который описывает лицо человека. Пользователь вводит набор из 20 признаков в программу, и далее происходит распознавание с применением 4 персептронов для каждого шага работы программы. Также пользователю предоставляется возможность вывода на экран графиков точности персептронов для классов и их весов на каждом шаге. Примеры работы программы представлены на рисунках 5 – 6.

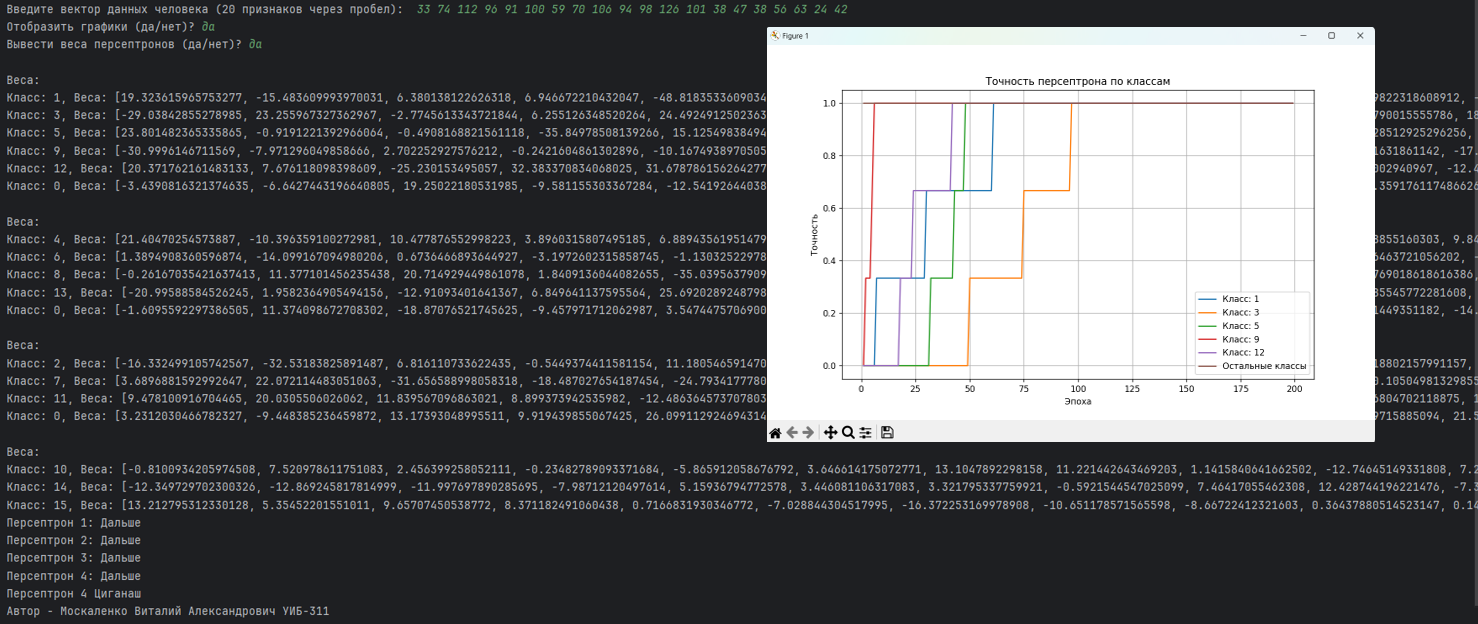


Рисунок 5 – Результат работы программы с выводом весов персептронов и графиком точности на экран

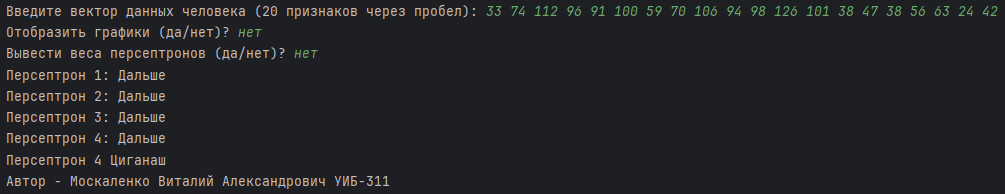


Рисунок 6 – Результат работы программы с без вывода весов персептронов и графиком точности на экран

# **ЗАКЛЮЧЕНИЕ**

В результате данной работы была создана система распознавания лиц, основанная на использовании алгоритма градиентного спуска для выполнения задачи классификации. Основной целью разработки стало построение модели, способной эффективно разделять определять данные, представляющие метрики 15 различных людей.

Для достижения этой цели была проведено распознавание признаков каждого лица. На входе система принимала набор данных, где для каждого человека были представлены числовые характеристики – вектор из 20 признаков.

Разработанная модель использует однослойные персептроны, веса которых обновлялись с помощью алгоритма градиентного спуска. Этот процесс позволил минимизировать функцию потерь, что способствовало точной настройке параметров модели и улучшению её способности различать данные.

# **ПРИЛОЖЕНИЕ А**

#Lab8AltF.py – основная программа

from InfoFileLab8 import vectoryLic, klassovyeGruppy, soedinenieKlassov  
import numpy as np  
import copy  
import random  
import matplotlib.pyplot as plt  
  
def Obuchenie(obuchDannie, vybrannyeKlassy=[], pokazGrafiki=False, pokazVesa=False, mestoDlyaCiklaSShagom=""):  
 dannieDlyaObucheniya = copy.deepcopy(obuchDannie)  
 skorostObucheniya = 0.2  
 klassy = list(range(0, len(vybrannyeKlassy)))  
  
 for i, (klass, atributy) in enumerate(dannieDlyaObucheniya):  
 dannieDlyaObucheniya[i] = (klass, np.array(atributy) / 50.0 - 1)  
 dannye = []  
 for klass, atributy in dannieDlyaObucheniya:  
 if klass in vybrannyeKlassy:  
 dannye.append([klassy[vybrannyeKlassy.index(klass)], atributy])  
 else:  
 dannye.append([klassy[vybrannyeKlassy.index(-1)], atributy])  
 for i in range(6):  
 dannye += dannye  
  
 random.shuffle(dannye)  
 vesa = [np.random.uniform(-0.1, 0.1, size=len(dannye[0][1])) for \_ in klassy]  
 poteriPoKlassam = {klass: [] for klass in klassy}  
 tochnostPoKlassam = {klass: [] for klass in klassy}  
 znacheniyaEpokh = []  
 luchshieVesa = None  
  
 for epokha in range(1, 200):  
 obshayaPoteria = 0  
 summarnyePoteriPoKlassam = {klass: 0 for klass in klassy}  
 pravilnoOpredelennyeKlass = {klass: 0 for klass in klassy}  
 vsegoPoKlassam = {klass: 0 for klass in klassy}  
  
 for klass, atributy in dannye:  
 summyVesov = [np.dot(ves, atributy) for ves in vesa]  
 maxSumma = np.max(summyVesov)  
 eksponenty = np.exp(summyVesov - maxSumma)  
 softmax = eksponenty / np.sum(eksponenty)  
 oneHot = np.zeros(len(vesa))  
 oneHot[klass] = 1  
 poteria = -np.sum(oneHot \* np.log(softmax))  
 obshayaPoteria += poteria  
 summarnyePoteriPoKlassam[klass] += poteria  
 vsegoPoKlassam[klass] += 1  
 oshibka = softmax - oneHot  
 grad = np.outer(oshibka, atributy)  
 vesa -= skorostObucheniya \* grad  
  
 matricaSputannosti = [[0 for klass in klassy] for klass in klassy]  
  
 for testKlass, testVector in dannye:  
 summResultat = raspoznaniePerc(testVector, vesa)  
 predskazannyiKlass = max(summResultat, key=summResultat.get)  
 matricaSputannosti[predskazannyiKlass][testKlass] += 1  
  
 if predskazannyiKlass == testKlass:  
 pravilnoOpredelennyeKlass[testKlass] += 1  
  
 for klass in klassy:  
 ideal = True  
 if vsegoPoKlassam[klass] > 0:  
 tochnost = pravilnoOpredelennyeKlass[klass] / vsegoPoKlassam[klass]  
 if tochnost != 1:  
 ideal = False  
 sredPoteria = summarnyePoteriPoKlassam[klass] / vsegoPoKlassam[klass]  
 tochnostPoKlassam[klass].append(tochnost)  
 poteriPoKlassam[klass].append(sredPoteria)  
 else:  
 tochnostPoKlassam[klass].append(0)  
 poteriPoKlassam[klass].append(0)  
  
 znacheniyaEpokh.append(epokha)  
  
 if ideal and luchshieVesa is None:  
 luchshieVesa = vesa  
  
 if luchshieVesa is None:  
 luchshieVesa = vesa  
  
 if pokazGrafiki:  
 plt.figure(figsize=(10, 6))  
 for klass in klassy:  
 plt.plot(znacheniyaEpokh, tochnostPoKlassam[klass], label=f'{"Класс: " + str(vybrannyeKlassy[klass] + 1) if vybrannyeKlassy[klass] != -1 else "Остальные классы"}')  
 plt.xlabel("Эпоха")  
 plt.ylabel("Точность")  
 plt.title("Точность персептрона по классам")  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
 if pokazVesa:  
 print("\nВеса:")  
 for klass, ves in enumerate(luchshieVesa):  
 print(f"Класс: {vybrannyeKlassy[klass] + 1}, Веса: {ves.tolist()}")  
  
 return luchshieVesa  
  
def raspoznaniePerc(vhodnoiVektor, vesa):  
 vzveshennyeSummy = [np.dot(ves, vhodnoiVektor) for ves in vesa]  
 softmax = np.exp(vzveshennyeSummy) / np.sum(np.exp(vzveshennyeSummy))  
 return {i: softmax[i] for i in range(len(vesa))}  
  
def normirovkaPriznakov(priznaki):  
 normirovannyePriznaki = (priznaki - np.mean(priznaki)) / np.std(priznaki)  
 return normirovannyePriznaki  
  
def kaskadnayaKlassifikaciya(priznaki, spisokVesov, indeksSoedineniyaKlassov):  
 tekushiePriznaki = normirovkaPriznakov(np.array(priznaki))  
 rezultaty = []  
 for i, ves in enumerate(spisokVesov):  
 if ves is None:  
 raise ValueError(f"Персептрон {i + 1}: Оптимальные веса отсутствуют!")  
 vyrabotki = np.dot(ves, tekushiePriznaki)  
 maksimalnyiIndeks = np.argmax(vyrabotki)  
 if maksimalnyiIndeks == len(ves) - 1:  
 rezultaty.append(f"Персептрон {i + 1}: Дальше")  
 continue  
 else:  
 klassRezultata = indeksSoedineniyaKlassov[i][maksimalnyiIndeks]  
 rezultaty.append(f"Персептрон {i + 1} {klassRezultata}")  
 return rezultaty  
 finalnyiKlass = indeksSoedineniyaKlassov[-1][maksimalnyiIndeks]  
 rezultaty.append(f"Персептрон {len(spisokVesov)} {finalnyiKlass}")  
 return rezultaty  
  
def filtrDannie(udalyaemyeKlassy):  
 novyiDataset = []  
 for klass, attr in vectoryLic:  
 if klass in udalyaemyeKlassy:  
 continue  
 novyiDataset.append([klass, attr])  
 return novyiDataset  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 priznaki = list(map(float, input("Введите вектор данных человека (20 признаков через пробел): ").split()))  
 ispolzuemyeKlassy = []  
 spisokVesov = []  
 pokazGrafiki = input("Отобразить графики (да/нет)? ").strip().lower() == 'да'  
 pokazVesa = input("Вывести веса персептронов (да/нет)? ").strip().lower() == 'да'  
  
 for shag, klassSet in enumerate(klassovyeGruppy):  
 dannyeObucheniya = [d for d in vectoryLic if d[0] not in ispolzuemyeKlassy]  
 vesa = Obuchenie(dannyeObucheniya, klassSet, pokazGrafiki, pokazVesa, f"Шаг {shag + 1}")  
 spisokVesov.append(vesa)  
 ispolzuemyeKlassy += klassSet  
  
 resultaty = kaskadnayaKlassifikaciya(priznaki, spisokVesov, soedinenieKlassov)  
 for resultat in resultaty:  
 print(resultat)  
  
 print("Автор - Москаленко Виталий Александрович УИБ-311")

# **ПРИЛОЖЕНИЕ Б**

#InfoFileLab8.py – дополнительный файл для размещения данных, необходимых в основной программе

vectoryLic = \  
[  
 [0, [29, 64, 94, 73, 70, 82, 46, 46, 84, 70, 66, 97, 74, 29, 42, 42, 41, 61, 31, 33]],  
 [0, [30, 79, 104, 76, 70, 87, 55, 41, 79, 68, 46, 83, 68, 21, 36, 50, 28, 53, 41, 18]],  
 [0, [26, 54, 85, 72, 70, 78, 43, 53, 84, 70, 75, 100, 77, 30, 39, 34, 44, 53, 21, 34]],  
  
 [1, [27, 71, 103, 80, 77, 92, 55, 54, 91, 77, 70, 101, 79, 28, 42, 38, 40, 58, 32, 31]],  
 [1, [28, 78, 107, 78, 74, 93, 58, 48, 87, 72, 53, 90, 72, 22, 34, 46, 28, 54, 40, 23]],  
 [1, [22, 53, 87, 69, 68, 81, 45, 52, 84, 68, 72, 97, 70, 28, 36, 28, 45, 55, 23, 38]],  
  
 [2, [30, 66, 94, 73, 73, 82, 48, 48, 82, 71, 65, 92, 71, 29, 42, 43, 42, 65, 34, 33]],  
 [2, [31, 86, 108, 81, 76, 92, 63, 47, 83, 75, 50, 85, 74, 24, 37, 52, 29, 52, 41, 15]],  
 [2, [22, 53, 88, 76, 73, 83, 46, 58, 89, 74, 82, 105, 79, 31, 36, 27, 49, 48, 15, 39]],  
  
 [3, [30, 71, 101, 81, 76, 88, 53, 53, 87, 75, 72, 99, 78, 30, 43, 37, 40, 55, 29, 30]],  
 [3, [31, 81, 107, 82, 75, 90, 57, 42, 82, 71, 48, 86, 73, 22, 36, 51, 30, 45, 37, 14]],  
 [3, [25, 55, 91, 76, 74, 85, 46, 58, 91, 75, 81, 107, 82, 31, 39, 30, 46, 48, 18, 35]],  
  
 [4, [28, 66, 93, 72, 68, 81, 49, 48, 79, 66, 65, 90, 69, 27, 41, 35, 36, 54, 30, 28]],  
 [4, [28, 76, 98, 73, 70, 84, 55, 42, 77, 67, 46, 79, 66, 21, 34, 47, 26, 50, 38, 17]],  
 [4, [23, 49, 80, 68, 67, 75, 42, 56, 82, 69, 78, 97, 73, 28, 39, 26, 47, 48, 15, 36]],  
  
 [5, [34, 78, 114, 88, 87, 102, 58, 58, 102, 88, 79, 115, 89, 35, 48, 49, 51, 74, 38, 40]],  
 [5, [36, 92, 121, 92, 86, 102, 65, 46, 90, 85, 51, 91, 82, 27, 43, 58, 41, 58, 45, 15]],  
 [5, [26, 55, 96, 81, 81, 92, 47, 63, 102, 85, 89, 120, 90, 35, 41, 36, 57, 59, 19, 45]],  
  
 [6, [29, 74, 103, 79, 75, 92, 57, 57, 92, 77, 75, 106, 82, 29, 43, 37, 38, 59, 32, 33]],  
 [6, [32, 87, 110, 83, 77, 93, 63, 53, 88, 77, 62, 97, 81, 25, 38, 49, 27, 57, 40, 21]],  
 [6, [4, 10, 13, 12, 12, 12, 8, 9, 12, 11, 11, 14, 12, 4, 6, 5, 5, 6, 3, 3]],  
  
 [7, [29, 68, 95, 75, 71, 82, 50, 51, 81, 69, 70, 94, 73, 29, 42, 37, 39, 60, 31, 32]],  
 [7, [28, 77, 103, 74, 68, 88, 56, 42, 80, 68, 47, 84, 68, 21, 33, 45, 27, 51, 38, 20]],  
 [7, [23, 52, 83, 68, 67, 76, 44, 55, 82, 67, 77, 97, 71, 29, 37, 28, 47, 57, 21, 39]],  
  
 [8, [26, 64, 90, 71, 69, 79, 47, 48, 79, 68, 64, 89, 69, 26, 43, 40, 41, 61, 31, 32]],  
 [8, [26, 75, 106, 75, 72, 93, 56, 43, 87, 68, 47, 89, 67, 19, 33, 49, 26, 52, 43, 23]],  
 [8, [23, 52, 85, 69, 68, 78, 43, 55, 84, 69, 74, 97, 71, 27, 39, 29, 49, 58, 22, 40]],  
  
 [9, [28, 75, 102, 78, 73, 89, 57, 55, 87, 72, 72, 99, 77, 27, 42, 34, 34, 56, 32, 30]],  
 [9, [29, 81, 104, 79, 72, 87, 59, 48, 81, 69, 55, 86, 73, 20, 33, 44, 22, 45, 37, 15]],  
 [9, [24, 63, 94, 76, 72, 85, 52, 58, 87, 71, 78, 101, 76, 28, 41, 28, 42, 54, 24, 35]],  
  
 [10, [27, 64, 85, 66, 61, 72, 46, 42, 70, 62, 58, 81, 65, 26, 38, 34, 33, 51, 27, 26]],  
 [10, [27, 68, 86, 66, 59, 70, 47, 35, 63, 60, 43, 69, 61, 22, 30, 38, 26, 42, 30, 13]],  
 [10, [20, 51, 69, 56, 49, 57, 34, 29, 55, 49, 39, 62, 54, 18, 27, 28, 23, 27, 19, 11]],  
  
 [11, [23, 59, 81, 66, 63, 72, 47, 47, 71, 62, 59, 80, 64, 22, 32, 28, 29, 44, 23, 24]],  
 [11, [25, 67, 87, 69, 63, 74, 49, 39, 68, 61, 46, 73, 64, 19, 29, 37, 23, 37, 28, 12]],  
 [11, [22, 55, 80, 67, 64, 72, 45, 50, 75, 64, 65, 86, 68, 24, 32, 25, 33, 43, 19, 27]],  
  
 [12, [30, 68, 98, 81, 82, 88, 51, 53, 90, 82, 70, 100, 83, 30, 44, 46, 45, 61, 31, 30]],  
 [12, [3, 9, 11, 9, 9, 10, 7, 7, 10, 9, 10, 12, 9, 4, 6, 4, 5, 8, 3, 4]],  
 [12, [3, 8, 11, 9, 9, 9, 6, 6, 9, 9, 8, 11, 9, 3, 5, 4, 4, 5, 3, 3]],  
  
 [13, [30, 70, 97, 75, 73, 85, 52, 50, 84, 75, 67, 94, 76, 29, 44, 42, 42, 64, 34, 32]],  
 [13, [31, 79, 104, 79, 75, 88, 57, 44, 80, 73, 50, 83, 71, 24, 36, 49, 32, 54, 40, 17]],  
 [13, [25, 56, 87, 75, 75, 81, 47, 59, 87, 77, 80, 102, 81, 30, 43, 34, 47, 56, 21, 36]],  
  
 [14, [35, 79, 109, 85, 82, 94, 57, 56, 92, 83, 75, 104, 83, 34, 47, 46, 48, 73, 38, 37]],  
 [14, [8, 17, 28, 21, 23, 27, 14, 17, 29, 23, 22, 33, 24, 9, 12, 13, 14, 20, 9, 13]],  
 [14, [33, 74, 112, 96, 91, 100, 59, 70, 106, 94, 98, 126, 101, 38, 47, 38, 56, 63, 24, 42]],  
  
]  
  
klassovyeGruppy = [[0, 2, 4, 8, 11, -1], [3, 5, 7, 12, -1], [1, 6, 10, -1], [9, 13, 14]]  
  
soedinenieKlassov = [  
 ['Мельник', 'Сафронов', 'Скобкин', 'Егор', 'Мария', 'Дальше'],  
 ['Саша', 'Стрельников', 'Валерия', 'Скорский', 'Дальше'],  
 ['Нартов', 'Андрей', 'Любасова', 'Дальше'],  
 ['Ксения', 'Фачкин', 'Циганаш'],  
 ]