МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М. В. ЛОМОНОСОВА
ФИЛИАЛ МОСКОВСКОГО ГОСУДАРСТВЕННОГО
УНИВЕРСИТЕТА ИМЕНИ М. В. ЛОМОНОСОВА В ГОРОДЕ
СЕВАСТОПОЛЕ

Факультет «Компьютерной математики»
Направление подготовки «Прикладная математика и информатика»
01.03.02 (бакалавр)

ОТЧЕТ

по работе по курсу

«Основы проектирования интеллектуальных систем»

Работу выполнил:
студент группы ПМ-401
Хаметов Марк Владимирович

Руководитель:
к.т.н., доцент
Скаковская А.Н.

Севастополь, 2023

Постановка задачи

Пусть дано входное математическое описание классификатора в виде обучающей матрицы целых значений яркости изображений отпечатков пальцев человека $\left\| y_{m,i}^{(j)} \right\|, m = \overline{1,M}, i = \overline{1,N}, j = \overline{1,n}$, где M=4, N, n - количество классов распознавания (изображений), признаков распознавания и испытаний соответственно. На этапе обучения необходимо найти оптимальное в информационном смысле разбиения пространства признаков распознавания на классы распознавания и на этапе экзамена по результатам ограниченного числа испытаний в режиме функционирования системы распознавания получить высоко достоверное решение о принадлежности вектора реализации образа к некоторому классу с априорно определенного конечного алфавита классов распознавания $\{X_m^o\}$. Выполнить вычисление критерия функциональной эффективности по Шеннону.

Требования к оформлению

1. Титульный лист
2. Постановка задачи
3. Теоретические сведения
4. Программа

5. Результаты имитационного моделирования разработанной системы:

1. Форма, содержащая изображения, обучающие матрицы, бинарные матрицы, эталонные векторы, значения допусков и уровня селекции;

2. Форма, содержащая:

• таблицу вычисленных значений КФЭ, точностных характеристик с выделенной рабочей областью;

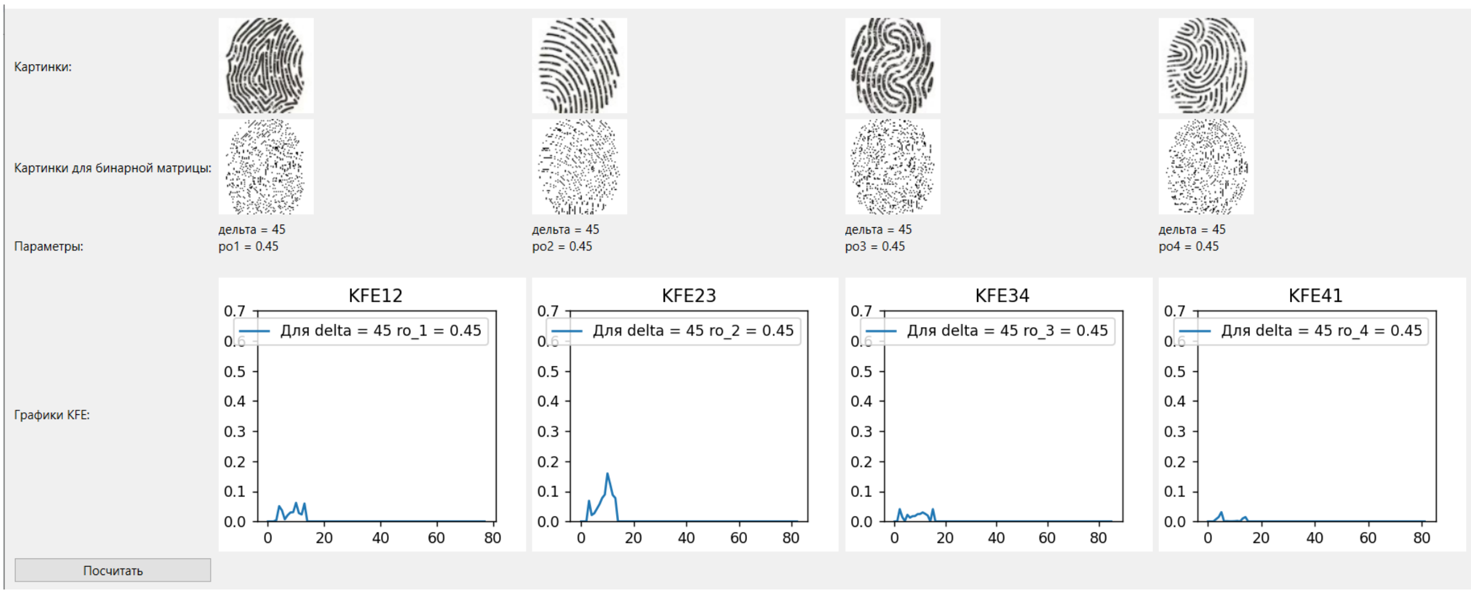
• графики зависимости радиусов контейнеров от КФЭ с выделенной рабочей областью на промежутке от нуля до межцентрового кодового расстояния;

3. Форма, содержащая значения функции принадлежности и результаты классификации (не менее 20 испытаний).

4. Выводы

Результаты имитационного моделирования разработанной системы

Для запуска программы без оптимизации параметров были получены следующие результаты:



Значения эталонного вектора были получены:

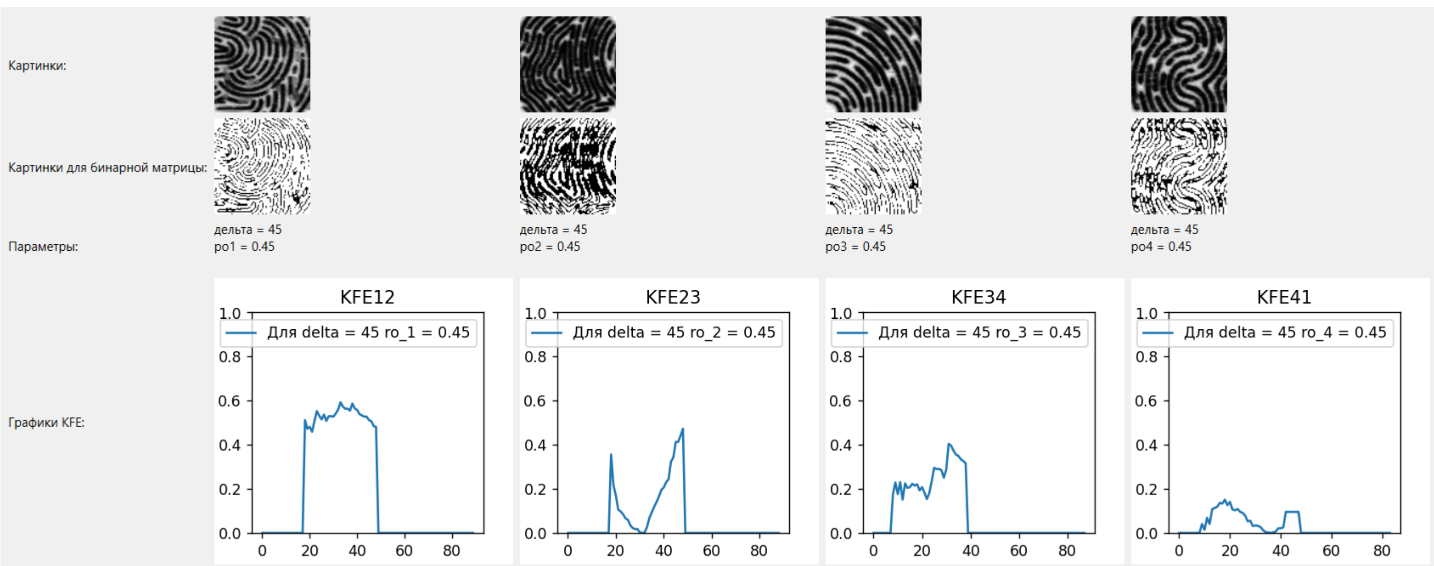
EV1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

EV2 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

EV3 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

EV4 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Векторы получились нулевые, что неудотвлетворяет решению задачи. Поэтому было принято решение использовать другой набор изображений.



EV1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

EV2 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]

EV3 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

EV4 = [0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

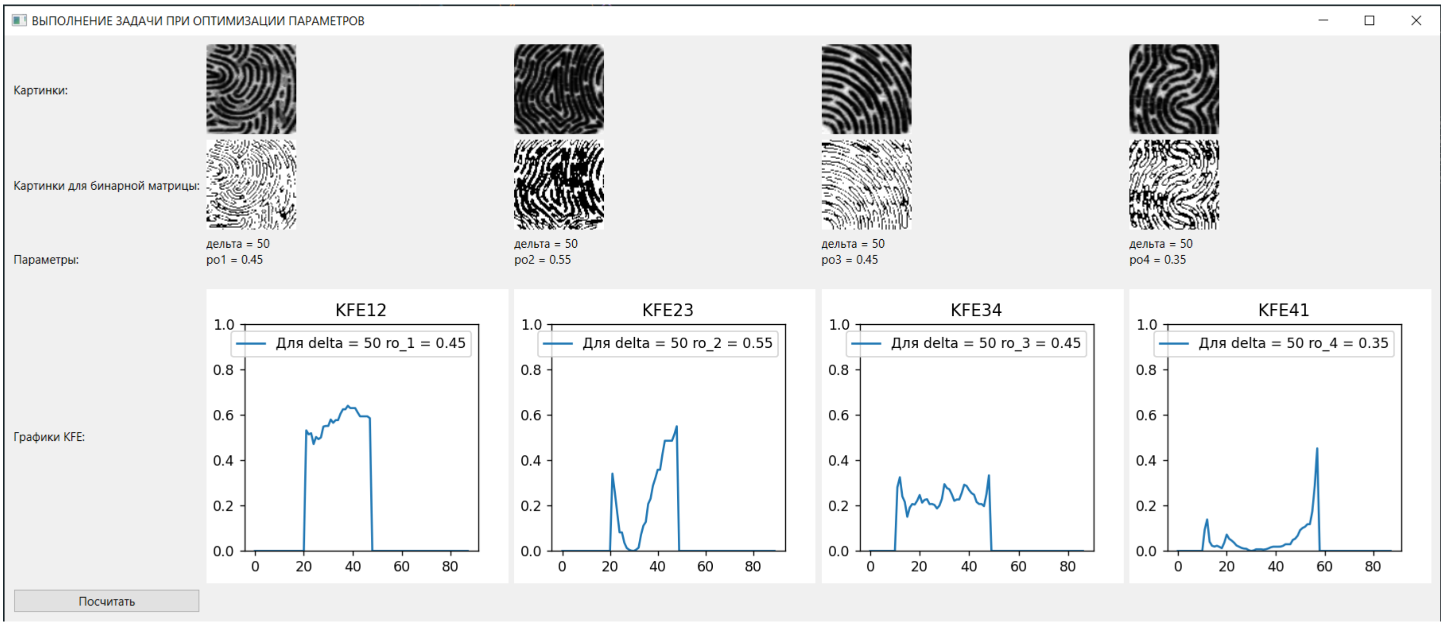
Все вектора получились ненулевыми, поэтому этот набор изображений было принято оставить. Проведем оптимизацию параметров.

Результаты экзаминации до оптимизации параметров:

Выбрана реализация 56 из класса 4	Выбрана реализация 71 из класса 2
Функция принадлежности 1 классу 0.375	Функция принадлежности 1 классу -0.02083333333333326
Функция принадлежности 2 классу -1.5	Функция принадлежности 2 классу -1.2222222222222223
Функция принадлежности 3 классу 0.032258064516129004	Функция принадлежности 3 классу -0.5806451612903225
Функция принадлежности 4 классу -0.4444444444444444	Функция принадлежности 4 классу -1.3888888888888888
Объект принадлежит 1 классу	Объект принадлежит 1 классу
Выбрана реализация 30 из класса 3	Выбрана реализация 47 из класса 2
Функция принадлежности 1 классу 0.6875	Функция принадлежности 1 классу 0.08333333333333337
Функция принадлежности 2 классу -1.7777777777777777	Функция принадлежности 2 классу -1.0555555555555554
Функция принадлежности 3 классу 0.5161290322580645	Функция принадлежности 3 классу -0.4193548387096775
Функция принадлежности 4 классу -0.16666666666666674	Функция принадлежности 4 классу -1.0
Объект принадлежит 1 классу	Объект принадлежит 1 классу
Выбрана реализация 17 из класса 2	Выбрана реализация 21 из класса 3
Функция принадлежности 1 классу -0.08333333333333326	Функция принадлежности 1 классу 0.6666666666666667
Функция принадлежности 2 классу -0.8333333333333333	Функция принадлежности 2 классу -2.388888888888889
Функция принадлежности 3 классу -0.6774193548387097	Функция принадлежности 3 классу 0.4838709677419355
Функция принадлежности 4 классу -1.8888888888888888	Функция принадлежности 4 классу -0.22222222222222232
Объект принадлежит 1 классу	Объект принадлежит 1 классу
Выбрана реализация 44 из класса 3	Выбрана реализация 10 из класса 3
Функция принадлежности 1 классу 0.875	Функция принадлежности 1 классу 0.7708333333333334
Функция принадлежности 2 классу -2.1666666666666665	Функция принадлежности 2 классу -1.7777777777777777
Функция принадлежности 3 классу 0.8064516129032258	Функция принадлежности 3 классу 0.6451612903225806
Функция принадлежности 4 классу 0.0	Функция принадлежности 4 классу -0.2777777777777777
Объект принадлежит 1 классу	Объект принадлежит 1 классу
Выбрана реализация 89 из класса 2	Выбрана реализация 86 из класса 2
Функция принадлежности 1 классу 0.47916666666666663	Функция принадлежности 1 классу 0.375
Функция принадлежности 2 классу -1.5555555555555554	Функция принадлежности 2 классу -1.6111111111111112
Функция принадлежности 3 классу 0.19354838709677424	Функция принадлежности 3 классу 0.032258064516129004
Функция принадлежности 4 классу -0.5	Функция принадлежности 4 классу -0.4444444444444444
Объект принадлежит 1 классу	Объект принадлежит 1 классу

Для параметров без оптимизации из 100 запусков было успешно предсказано 22.

Далее была проведена оптимизация параметров. Были получены следующие результаты.



EV1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

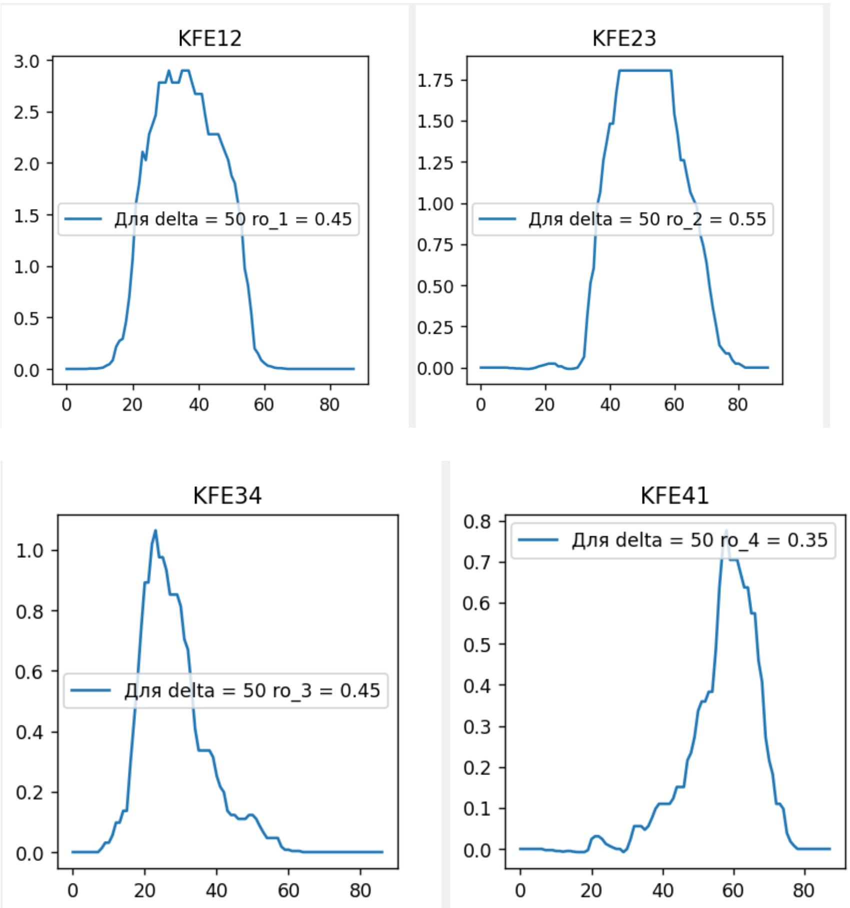
EV2 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0]

EV3 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

EV4 = [0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0]

Полученные параметры: delta = 50, po1 = 0.45, po2 = 0.55, po3 = 0.45, po4 = 0.35.

Графики результатов для кфэ Кульбака: (Построены графики разной высоты)



Далее был проведен этап экзамена. Ниже представлен каждый десятый тест из ста:

Выбрана реализация 8 из класса 1 Функция принадлежности 1 классу 0.6382978723404256 Функция принадлежности 2 классу -0.02083333333333326 Функция принадлежности 3 классу -0.5 Функция принадлежности 4 классу 0.0 Объект принадлежит 1 классу ==========	Выбрана реализация 2 из класса 1 Функция принадлежности 1 классу 0.6808510638297872 Функция принадлежности 2 классу -0.02083333333333326 Функция принадлежности 3 классу -0.33333333333333326 Функция принадлежности 4 классу -0.10526315789473695 Объект принадлежит 1 классу ==========
Выбрана реализация 32 из класса 4 Функция принадлежности 1 классу 0.7446808510638299 Функция принадлежности 2 классу -0.16666666666666674 Функция принадлежности 3 классу -0.08333333333333326 Функция принадлежности 4 классу 0.07894736842105265 Объект принадлежит 1 классу ==========	Выбрана реализация 62 из класса 4 Функция принадлежности 1 классу 0.46808510638297873 Функция принадлежности 2 классу 0.0625 Функция принадлежности 3 классу -1.1666666666666665 Функция принадлежности 4 классу -0.21052631578947367 Объект принадлежит 1 классу ==========
Выбрана реализация 8 из класса 1 Функция принадлежности 1 классу 0.6382978723404256 Функция принадлежности 2 классу -0.02083333333333326 Функция принадлежности 3 классу -0.5 Функция принадлежности 4 классу 0.0 Объект принадлежит 1 классу ==========	Выбрана реализация 47 из класса 2 Функция принадлежности 1 классу -0.12765957446808507 Функция принадлежности 2 классу 0.35416666666666663 Функция принадлежности 3 классу -3.333333333333333 Функция принадлежности 4 классу 0.21052631578947367 Объект принадлежит 2 классу ==========
Выбрана реализация 63 из класса 4 Функция принадлежности 1 классу 0.5106382978723405 Функция принадлежности 2 классу 0.02083333333333337 Функция принадлежности 3 классу -1.0 Функция принадлежности 4 классу -0.05263157894736836 Объект принадлежит 1 классу ==========	Выбрана реализация 7 из класса 1 Функция принадлежности 1 классу 0.5957446808510638 Функция принадлежности 2 классу -0.0625 Функция принадлежности 3 классу -0.6666666666666667 Функция принадлежности 4 классу -0.05263157894736836 Объект принадлежит 1 классу ==========
Выбрана реализация 65 из класса 1 Функция принадлежности 1 классу 0.6595744680851063 Функция принадлежности 2 классу -0.20833333333333326 Функция принадлежности 3 классу -0.5833333333333333 Функция принадлежности 4 классу 0.13157894736842102 Объект принадлежит 1 классу	Выбрана реализация 11 из класса 1 Функция принадлежности 1 классу 0.7234042553191489 Функция принадлежности 2 классу -0.10416666666666674 Функция принадлежности 3 классу -0.33333333333333326 Функция принадлежности 4 классу 0.1578947368421053 Объект принадлежит 1 классу

Из 100 запусков было успешно предсказано 56.

Вывод

Была разработана интеллектуальная система для распознавания классов схожих изображений отпечатков пальцев. Для имитационного моделирования системы было найдено оптимальное разбиение пространства признаков распознавания на классы распознавания. Найдены эталонные векторы и разработаны обучающие матрицы. Далее было получено решение о принадлежности распознаваемого вектора реализации к некоторому классу. Были построены графики зависимости радиусов контейнеров от КФЭ с выделенной рабочей областью. Проведен экзамен для проверки результатов классификации.

В процессе выполнения программной реализации сделаны выводы:

1. Наличие «белых уголков» на изображении уменьшает количество строк, которые могут стать признаком распознавания. Это происходит из-за того что для фиксированных значений дельта и ро строки с белыми помехами не относящимися к информативной части изображения всегда обращаются в ноль, а строки без этих помех разбиваются нормально.

   В связи с этим выводом было принято использовать сегменты отпечатков пальцев.
2. Изображения разной насыщенности, то есть такие изображения бинарные матрицы которых при одинаковых параметрах имеют значительное различие в количестве черных пикселей, обладают существенным различием в количестве признаков распознавания в эквивалентных векторах.

   Таким образом, при запуске алгоритма без оптимизации параметров изображения разной насыщенности сильно различаются, а одинаковой насыщенности схожи.
3. Оптимизация параметров помогает исправить проблему из второго вывода.

```
from PIL import Image
import numpy as np

import sys
from PyQt6.QtGui import QPixmap, QAction, QColor, QPainter,
QTransform
from PyQt6.QtWidgets import QApplication, QMainWindow,
QPushButton, QWidget, QStackedLayout,QLineEdit, QGridLayout,
QLabel,QTableWidget, QTableWidgetItem, QFileDialog,QTableView
from PyQt6.QtCore import Qt,QAbstractTableModel

import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt5agg import
FigureCanvasQTAgg as FigureCanvas
from torch import true_divide


class MainWindow(QMainWindow):
    def __init__(self,myname):
        super().__init__()
        self.myname = myname
        self.IMAGESIZE = 90
        self.setWindowTitle("ВЫПОЛНЕНИЕ ЗАДАЧИ ПРИ
ОПТИМИЗАЦИИ ПАРАМЕТРОВ")

        central_widget = QWidget()
        self.setCentralWidget(central_widget)
        self.layout = QGridLayout()

        label = QLabel("Картинки:")
        self.layout.addWidget(label,0, 0)
        label = QLabel("Картинки для бинарной матрицы:")
        self.layout.addWidget(label,1, 0)
        label = QLabel("Параметры:  ")
        self.layout.addWidget(label,2, 0)
        label = QLabel("Графики KFE:")
        self.layout.addWidget(label,3, 0)
        self.button = QPushButton("Посчитать")
        self.button.clicked.connect(self.main_calculation)
        self.layout.addWidget(self.button ,4, 0)

        self.picture_out1bin = QLabel()
        self.layout.addWidget(self.picture_out1bin,1, 1)
        self.picture_out2bin = QLabel()
        self.layout.addWidget(self.picture_out2bin,1, 2)
        self.picture_out3bin = QLabel()
        self.layout.addWidget(self.picture_out3bin,1, 3)
        self.picture_out4bin = QLabel()
        self.layout.addWidget(self.picture_out4bin,1, 4)

        self.txt_out1par = QLabel()
        self.str_out1par = ""
        self.layout.addWidget(self.txt_out1par,2, 1)
        self.txt_out2par = QLabel()
        self.str_out2par = ""
        self.layout.addWidget(self.txt_out2par,2, 2)
        self.txt_out3par = QLabel()
        self.str_out3par = ""
        self.layout.addWidget(self.txt_out3par,2, 3)
        self.txt_out4par = QLabel()
        self.str_out4par = ""
        self.layout.addWidget(self.txt_out4par,2, 4)

        self.holdmatrixobjectsforme = []
        for j in range(1,5):
            label = QLabel()
            pixmap = QPixmap(f'{j}.png')
            image_1 = Image.open(f'{j}.png')
            image_1 = image_1.convert("L")
            matrix1 = np.asarray(np.array(image_1))
            self.holdmatrixobjectsforme.append(matrix1)
            label.setPixmap(pixmap)
            self.layout.addWidget(label,0, j)

        print(self.holdmatrixobjectsforme)

        central_widget.setLayout(self.layout)


    def main_calculation(self):
        np.set_printoptions(threshold=np.inf)
        bestkfe = -1
        last_cfe = 0
        for delta in range(45,46,5):
            weshouldbreakthings = False
            weshouldconsiderbreaking = False

            self.matrixbin1 = self.holdmatrixobjectsforme[0]
            for i,e in enumerate(self.holdmatrixobjectsforme[0]):
                #print(e)
                avrg = np.mean(e)
                avrg1= avrg - delta
                #print(avrg, avrg1)
                for ii,ee in enumerate(e):
                    if ee <= avrg and ee >= avrg1:
                        self.matrixbin1[i][ii] = 1
                    else:
                        self.matrixbin1[i][ii] = 0

            self.matrixbin2 = self.holdmatrixobjectsforme[1]
            for i,e in enumerate(self.holdmatrixobjectsforme[1]):
                avrg = np.mean(e)
                avrg1= avrg - delta
                for ii,ee in enumerate(e):
                    if ee <= avrg and ee >= avrg1:
                        self.matrixbin2[i][ii] = 1
                    else:
                        self.matrixbin2[i][ii] = 0

            self.matrixbin3 = self.holdmatrixobjectsforme[2]
            for i,e in enumerate(self.holdmatrixobjectsforme[2]):
                avrg = np.mean(e)
                avrg1= avrg - delta
                for ii,ee in enumerate(e):
                    if ee <= avrg and ee >= avrg1:
                        self.matrixbin3[i][ii] = 1
                    else:
                        self.matrixbin3[i][ii] = 0

            self.matrixbin4 = self.holdmatrixobjectsforme[3]
            for i,e in enumerate(self.holdmatrixobjectsforme[3]):
                avrg = np.mean(e)
                avrg1= avrg - delta
                for ii,ee in enumerate(e):
                    if ee <= avrg and ee >= avrg1:
                        self.matrixbin4[i][ii] = 1
                    else:
                        self.matrixbin4[i][ii] = 0

            for ro_1 in range(45,51,10):
                nro_1 = ro_1/100
                if weshouldbreakthings:
                    continue

                self.EV1 = []
                for i,e in enumerate(self.matrixbin1):
                    tempcount = 0
                    for ii,ee in enumerate(e):
                        if ee == 1:
                            tempcount+=1
                    if tempcount>= (len(self.matrixbin1[0])*nro_1):
                        self.EV1.append( 1 )
                    else:
                        self.EV1.append( 0 )


                for ro_2 in range(45,51,10):
                    nro_2 = ro_2/100
                    if weshouldbreakthings:
                        continue

                    self.EV2 = []
                    for i,e in enumerate(self.matrixbin2):
                        tempcount = 0
                        for ii,ee in enumerate(e):
                            if ee == 1:
                                tempcount+=1
                        if tempcount>= (len(self.matrixbin2[0])*nro_2):
                            self.EV2.append( 1 )
                        else:
                            self.EV2.append( 0 )


                    for ro_3 in range(45,46,10):
                        nro_3 = ro_3/100
                        if weshouldbreakthings:
                            continue

                        self.EV3 = []
                        for i,e in enumerate(self.matrixbin3):
                            tempcount = 0
                            for ii,ee in enumerate(e):
                                if ee == 1:
                                    tempcount+=1
                            if tempcount>= (len(self.matrixbin3[0])*nro_3):
                                self.EV3.append( 1 )
                            else:
                                self.EV3.append( 0 )

                        for ro_4 in range(45,46,10):
                            nro_4 = ro_4/100
                            if ro_4>self.IMAGESIZE:
                                weshouldbreakthings = True
                                break
                            if ro_1>self.IMAGESIZE:
                                weshouldbreakthings = True
                                break
                            if ro_2>self.IMAGESIZE:
                                weshouldbreakthings = True
                                break
                            if ro_3>self.IMAGESIZE:
                                weshouldbreakthings = True
                                break
                            if weshouldbreakthings:
                                break

                            self.EV4 = []
                            for i,e in enumerate(self.matrixbin4):
                                tempcount = 0
                                for ii,ee in enumerate(e):
                                    if ee == 1:
                                        tempcount+=1
                                if tempcount>= (len(self.matrixbin4[0])*nro_4):
                                    self.EV4.append( 1 )
                                else:
                                    self.EV4.append( 0 )

                            print(delta,ro_1,ro_2,ro_3,ro_4)

                            if check(self.EV4):
                                continue

                            self.SK1_short    = []
                            self.pkrf1_strout = ""

                            list_append_me = []
                            for i,e in enumerate(self.EV1):
                                counter_append_me = 0
                                for ee in self.matrixbin1[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK1_short.append(list_append_me)

                            list_append_me = []
                            for i,e in enumerate(self.EV1):
                                counter_append_me = 0
                                for ee in self.matrixbin2[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK1_short.append(list_append_me)

                            list_append_me = []
                            for i,e in enumerate(self.EV1):
                                counter_append_me = 0
                                for ee in self.matrixbin3[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK1_short.append(list_append_me)

                            list_append_me = []
                            for i,e in enumerate(self.EV1):
                                counter_append_me = 0
                                for ee in self.matrixbin4[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK1_short.append(list_append_me)

                            self.SK2_short    = []

                            list_append_me = []
                            for i,e in enumerate(self.EV2):
                                counter_append_me = 0
                                for ee in self.matrixbin1[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK2_short.append(list_append_me)

                            list_append_me = []
                            for i,e in enumerate(self.EV2):
                                counter_append_me = 0
                                for ee in self.matrixbin2[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK2_short.append(list_append_me)

                            list_append_me = []
                            for i,e in enumerate(self.EV2):
                                counter_append_me = 0
                                for ee in self.matrixbin3[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK2_short.append(list_append_me)

                            list_append_me = []
                            for i,e in enumerate(self.EV2):
                                counter_append_me = 0
                                for ee in self.matrixbin4[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK2_short.append(list_append_me)

                            self.SK3_short    = []

                            list_append_me = []
                            for i,e in enumerate(self.EV3):
                                counter_append_me = 0
                                for ee in self.matrixbin1[i]:
                                    if e!=ee:
                                        counter_append_me += 1
                                list_append_me.append(counter_append_me)
                            self.SK3_short.append(list_append_me)

                            list_append_me = []
                            for i,e in enumerate(self.EV3):
                                counter_append_me = 0
                                for ee in self.matrixbin2[i]:
```

```
                    if e!=ee:
                        counter_append_me += 1
                list_append_me.append(counter_append_me)
            self.SK3_short.append(list_append_me)

            list_append_me = []
            for i,e in enumerate(self.EV3):
                counter_append_me = 0
                for ee in self.matrixbin3[i]:
                    if e!=ee:
                        counter_append_me += 1
                list_append_me.append(counter_append_me)
            self.SK3_short.append(list_append_me)

            list_append_me = []
            for i,e in enumerate(self.EV3):
                counter_append_me = 0
                for ee in self.matrixbin4[i]:
                    if e!=ee:
                        counter_append_me += 1
                list_append_me.append(counter_append_me)
            self.SK3_short.append(list_append_me)

            self.SK4_short    = []
            self.pkrf1_strout = ""

            list_append_me = []
            for i,e in enumerate(self.EV4):
                counter_append_me = 0
                for ee in self.matrixbin1[i]:
                    if e!=ee:
                        counter_append_me += 1
                list_append_me.append(counter_append_me)
            self.SK4_short.append(list_append_me)

            list_append_me = []
            for i,e in enumerate(self.EV4):
                counter_append_me = 0
                for ee in self.matrixbin2[i]:
                    if e!=ee:
                        counter_append_me += 1
                list_append_me.append(counter_append_me)
            self.SK4_short.append(list_append_me)

            list_append_me = []
            for i,e in enumerate(self.EV4):
                counter_append_me = 0
                for ee in self.matrixbin3[i]:
                    if e!=ee:
                        counter_append_me += 1
                list_append_me.append(counter_append_me)
            self.SK4_short.append(list_append_me)

            list_append_me = []
            for i,e in enumerate(self.EV4):
                counter_append_me = 0
                for ee in self.matrixbin4[i]:
                    if e!=ee:
                        counter_append_me += 1
                list_append_me.append(counter_append_me)
            self.SK4_short.append(list_append_me)

            self.rasstoyanie12 = 0
            for i,e in enumerate(self.SK1_short[0]):
                if e != self.SK1_short[1][i]:
                    self.rasstoyanie12 += 1
            self.rasstoyanie13 = 0
            for i,e in enumerate(self.SK1_short[0]):
                if e != self.SK1_short[2][i]:
                    self.rasstoyanie13 += 1

            self.rasstoyanie21 = 0
            for i,e in enumerate(self.SK2_short[1]):
                if e != self.SK2_short[0][i]:
                    self.rasstoyanie21 += 1
            self.rasstoyanie23 = 0
            for i,e in enumerate(self.SK2_short[1]):
                if e != self.SK2_short[2][i]:
                    self.rasstoyanie23 += 1

            self.rasstoyanie31 = 0
            for i,e in enumerate(self.SK3_short[2]):
                if e != self.SK3_short[0][i]:
                    self.rasstoyanie31 += 1

            self.rasstoyanie34 = 0
            for i,e in enumerate(self.SK3_short[2]):
                if e != self.SK3_short[3][i]:
                    self.rasstoyanie34 += 1

            self.rasstoyanie41 = 0
            for i,e in enumerate(self.SK4_short[3]):
                if e != self.SK4_short[0][i]:
                    self.rasstoyanie41 += 1

            KFE12_Shennon, KFE12_Kulbak =
cnt_kfe(self.rasstoyanie12, self.SK1_short[0],
self.SK1_short[1],self.IMAGESIZE)
            bestkfe1, bestrad1 = bestkferad(KFE12_Shennon ,
KFE12_Shennon , min( self.rasstoyanie12,  self.rasstoyanie12))

            KFE23_Shennon, KFE23_Kulbak =
cnt_kfe(self.rasstoyanie23, self.SK2_short[1],
self.SK2_short[2],self.IMAGESIZE)
            bestkfe2, bestrad2 = bestkferad(KFE23_Shennon,
KFE23_Shennon, min(self.rasstoyanie23, self.rasstoyanie23))

            KFE34_Shennon, KFE34_Kulbak =
cnt_kfe(self.rasstoyanie34, self.SK3_short[2],
self.SK3_short[3],self.IMAGESIZE)
            bestkfe3, bestrad3 = bestkferad(KFE34_Shennon ,
KFE34_Shennon , min( self.rasstoyanie34,  self.rasstoyanie34))

            KFE41_Shennon, KFE41_Kulbak =
cnt_kfe(self.rasstoyanie41, self.SK4_short[3],
self.SK4_short[0],self.IMAGESIZE)
            bestkfe4, bestrad4 = bestkferad(KFE41_Shennon ,
KFE41_Shennon , min( self.rasstoyanie41,  self.rasstoyanie41))

            boolfirsteverloop = False
            cur_kfe=bestkfe1+bestkfe2+bestkfe3+bestkfe4
            print(cur_kfe)
            if cur_kfe == last_cfe:
                print("brr")
                if  weshouldconsiderbreaking:
                    print("br")
                    break
                weshouldconsiderbreaking = True
            else:
                weshouldconsiderbreaking = False
            last_cfe = cur_kfe
            if (cur_kfe>bestkfe):
                bestkfe=cur_kfe
                sbestdelta=delta
                sbestro1=ro_1
                sbestro2=ro_2
                sbestro3=ro_3
                sbestro4=ro_4
                BESTKFE12_Shennon, BESTKFE12_Kulbak =
KFE12_Shennon, KFE12_Kulbak
                BESTKFE23_Shennon, BESTKFE23_Kulbak =
KFE23_Shennon, KFE23_Kulbak
                BESTKFE34_Shennon, BESTKFE34_Kulbak =
KFE34_Shennon, KFE34_Kulbak
                BESTKFE41_Shennon, BESTKFE41_Kulbak =
KFE41_Shennon, KFE41_Kulbak
                self.bestev1 = self.EV1
                self.bestev2 = self.EV2
                self.bestev3 = self.EV3
                self.bestev4 = self.EV4

        delta=sbestdelta
        ro1=sbestro1/100
        ro2=sbestro2/100
        ro3=sbestro3/100
        ro4=sbestro4/100
        print("Конец первого этапа оптимизации")
        print(f"Лучшее значение дельта = {delta}\n")
        print(f"Лучшее значение ро 1 {ro1}\n")
        print(f"Лучшее значение ро 2 {ro2}\n")
        print(f"Лучшее значение ро 3 {ro3}\n")
        print(f"Лучшее значение ро 4 {ro4}\n")
        self.str_out1par = self.str_out1par+f"дельта = {delta}\npo1 =
{ro1}\n"
        self.txt_out1par.setText(self.str_out1par)
        self.str_out2par = self.str_out2par+f"дельта = {delta}\npo2 =
{ro2}\n"
        self.txt_out2par.setText(self.str_out2par)
        self.str_out3par = self.str_out3par+f"дельта = {delta}\npo3 =
{ro3}\n"
        self.txt_out3par.setText(self.str_out3par)
        self.str_out4par = self.str_out4par+f"дельта = {delta}\npo4 =
{ro4}\n"
        self.txt_out4par.setText(self.str_out4par)
        print(f"Лучшее значение суммы кфе Шеннона KFE
{bestkfe}\n")
        print(f"EV1 = {self.bestev1}\nEV2 = {self.bestev2}\nEV3 =
{self.bestev3}\nEV4 = {self.bestev4}\n")


        pixmap2 = QPixmap(self.IMAGESIZE, self.IMAGESIZE)
        painter = QPainter(pixmap2)
        for i in range(self.IMAGESIZE):
            for j in range(self.IMAGESIZE):
                color2 = QColor(0,0,0)
                if  self.matrixbin1[i][j] == 0:
                    color2 = QColor(255, 255, 255)
                painter.setPen(color2)
                painter.setBrush(QColor(255, 255, 255))
                painter.drawRect(j, i, 1, 1)
        painter.end()
        pixmap2.save("output.png")
        self.picture_out1bin.setPixmap(pixmap2)

        pixmap2 = QPixmap(self.IMAGESIZE, self.IMAGESIZE)
        painter = QPainter(pixmap2)
        for i in range(self.IMAGESIZE):
            for j in range(self.IMAGESIZE):
                color2 = QColor(0,0,0)
                if  self.matrixbin2[i][j] == 0:
                    color2 = QColor(255, 255, 255)
                painter.setPen(color2)
                painter.setBrush(QColor(255, 255, 255))
                painter.drawRect(j, i, 1, 1)
        painter.end()
        pixmap2.save("output.png")
        self.picture_out2bin.setPixmap(pixmap2)

        pixmap2 = QPixmap(self.IMAGESIZE, self.IMAGESIZE)
        painter = QPainter(pixmap2)
        for i in range(self.IMAGESIZE):
            for j in range(self.IMAGESIZE):
                color2 = QColor(0,0,0)
                if  self.matrixbin3[i][j] == 0:
                    color2 = QColor(255, 255, 255)
                painter.setPen(color2)
                painter.setBrush(QColor(255, 255, 255))
                painter.drawRect(j, i, 1, 1)
        painter.end()
        pixmap2.save("output.png")
        self.picture_out3bin.setPixmap(pixmap2)

        pixmap2 = QPixmap(self.IMAGESIZE, self.IMAGESIZE)
        painter = QPainter(pixmap2)
        for i in range(self.IMAGESIZE):
            for j in range(self.IMAGESIZE):
                color2 = QColor(0,0,0)
                if  self.matrixbin4[i][j] == 0:
                    color2 = QColor(255, 255, 255)
                painter.setPen(color2)
                painter.setBrush(QColor(255, 255, 255))
                painter.drawRect(j, i, 1, 1)
        painter.end()
        pixmap2.save("output.png")
        self.picture_out4bin.setPixmap(pixmap2)

        fig1, ax1 = plt.subplots()
#       ax1.plot(BESTKFE12_Kulbak,label=f'Для delta = {delta} ro_1
= {ro1}')
        ax1.plot(BESTKFE12_Shennon,label=f'Для delta = {delta} ro_1
= {ro1}')
        plt.ylim(0, 1)
        plt.legend()
        plt.title("KFE12")
        fig1.savefig('111.png')
        canvas = FigureCanvas(fig1)
        self.layout.addWidget(canvas,3, 1)

        fig1, ax1 = plt.subplots()
#       ax1.plot(BESTKFE23_Kulbak,label=f'Для delta = {delta} ro_2
= {ro2}')
        ax1.plot(BESTKFE23_Shennon,label=f'Для delta = {delta} ro_2
= {ro2}')
        plt.ylim(0, 1)
        plt.legend()
        plt.title("KFE23")
        fig1.savefig('222.png')
        canvas = FigureCanvas(fig1)
        self.layout.addWidget(canvas,3, 2)

        fig1, ax1 = plt.subplots()
#       ax1.plot(BESTKFE34_Kulbak,label=f'Для delta = {delta} ro_3
= {ro3}')
        ax1.plot(BESTKFE34_Shennon,label=f'Для delta = {delta} ro_3
= {ro3}')
        plt.ylim(0, 1)
        plt.legend()
        plt.title("KFE34")
        fig1.savefig('333.png')
        canvas = FigureCanvas(fig1)
        self.layout.addWidget(canvas,3, 3)

        fig1, ax1 = plt.subplots()
#       ax1.plot(BESTKFE41_Kulbak,label=f'Для delta = {delta} ro_4
= {ro4}')
        ax1.plot(BESTKFE41_Shennon,label=f'Для delta = {delta} ro_4
= {ro4}')
        plt.ylim(0, 1)
        plt.legend()
        plt.title("KFE41")
        fig1.savefig('444.png')
        canvas = FigureCanvas(fig1)
        self.layout.addWidget(canvas,3, 4)


print(self.rasstoyanie12,self.rasstoyanie21,self.rasstoyanie31,self.rasst
oyanie41)
        r1s=vychislkfe_class(self.rasstoyanie12, self.rasstoyanie21,
self.SK1_short[0], self.SK1_short[1], self.SK2_short[1],
self.SK2_short[0], 1, 2,
3,BESTKFE12_Shennon,BESTKFE23_Shennon)
        r2s=vychislkfe_class(self.rasstoyanie21, self.rasstoyanie12,
self.SK2_short[1], self.SK2_short[0], self.SK1_short[0],
self.SK1_short[1], 2, 1,
3,BESTKFE23_Shennon,BESTKFE34_Shennon)
        r3s=vychislkfe_class(self.rasstoyanie31, self.rasstoyanie12,
self.SK3_short[2], self.SK3_short[0], self.SK1_short[0],
self.SK1_short[1], 3, 1,
2,BESTKFE34_Shennon,BESTKFE41_Shennon)
        r4s=vychislkfe_class(self.rasstoyanie41, self.rasstoyanie12,
self.SK4_short[3], self.SK4_short[0], self.SK1_short[0],
self.SK1_short[1], 4, 1,
2,BESTKFE41_Shennon,BESTKFE12_Shennon)
```

```
    import random
    trials=100
    correct=0
    for t in range(trials):
        fromcl=random.choice([1,2,3,4])
        real=random.randrange(0,90)
        pred=0
        if t%10==0: print("Выбрана реализация",real,"из класса",
fromcl)
        if fromcl==1:
            test_string=self.matrixbin1[real]#self.IMAGESIZE
        if fromcl==2:
            test_string=self.matrixbin2[real]
        if fromcl==3:
            test_string=self.matrixbin3[real]
        if fromcl==4:
            test_string=self.matrixbin4[real]
        d1=0
        d2=0
        d3=0
        d4=0
        for j in range(80):
            if test_string[j]!=self.EV1[j]:
                d1+=1
            if test_string[j]!=self.EV2[j]:
                d2+=1
            if test_string[j]!=self.EV3[j]:
                d3+=1
            if test_string[j]!=self.EV4[j]:
                d4+=1
        try:
            mu1=1-d1/r1s
        except:
            mu1=1-d1/1
        try:
            mu2=1-d2/r2s
        except:
            mu2=1-d2/1
        try:
            mu3=1-d3/r3s
        except:
            mu3=1-d3/1
        try:
            mu4=1-d4/r4s
        except:
            mu4=1-d4/1
        if t%10==0: print("Функция принадлежности 1
классу",mu1)
        if t%10==0: print("Функция принадлежности 2
классу",mu2)
        if t%10==0: print("Функция принадлежности 3
классу",mu3)
        if t%10==0: print("Функция принадлежности 4
классу",mu4)
        c=0.5
        mx=max(mu1,mu2,mu3,mu4)
        cont=0
        if mx==mu1: cont+=1
        if mx==mu2: cont+=1
        if mx==mu3: cont+=1
        if mx==mu4: cont+=1
        elif mx==mu1:
            pred=1
            if t%10==0: print("Объект принадлежит 1 классу")
        elif mx==mu2:
            pred=2
            if t%10==0: print("Объект принадлежит 2 классу")
        elif mx==mu3:
            pred=3
            if t%10==0: print("Объект принадлежит 3 классу")
        elif mx==mu4:
            pred=4
            if t%10==0: print("Объект принадлежит 4 классу")
        if pred==fromcl:
            correct+=1
        if t%10==0: print("============================")
    #print("Успешно предсказано",correct,"из", trials, ", доля
верных =", correct/trials)
    print(f"Из {trials} запусков было успешно предсказано
{correct}.")
```

```
def vychislkfe_class(dist1, dist2, sk1_1, sk1_2, sk2_1, sk2_2, our,
another1, another2,kfe1,kfe2):
  k1=[dist1]
  k2=[dist1]
  k3=[dist1]
  k4=[dist1]
  alpha=[dist1]
  beta=[dist1]
  D1=[dist1]
  D2=[dist1]
  KFE1_Shennon=kfe1
  orange=[dist1]
  blue=[dist1]
  for i in range(dist1):
    for j in range(90):
      if sk1_1[j]<=i:
        k1[i]+=1
      else: k2[i]+=1
      if sk1_2[j]<=i:
        k3[i]+=1
      else: k4[i]+=1
    alpha[i]=k2[i]/90
    beta[i]=k3[i]/90
    D1[i]=k1[i]/90
    D2[i]=k4[i]/90
    k1=[0]*dist2
    k2=[0]*dist2
    k3=[0]*dist2
    k4=[0]*dist2
    alpha=[0]*dist2
    beta=[0]*dist2
    D1=[0]*dist2
    D2=[0]*dist2
    KFE2_Shennon=kfe2
    orange=[0]*dist2
    blue=[0]*dist2
    for i in range(dist2):
      for j in range(90):
        if sk2_1[j]<=i:
          k1[i]+=1
        else: k2[i]+=1
        if sk2_2[j]<=i:
          k3[i]+=1
        else: k4[i]+=1
      alpha[i]=k2[i]/90
      beta[i]=k3[i]/90
      D1[i]=k1[i]/90
      D2[i]=k4[i]/90
    maxkfe_Shennon=0
    maxkfe_Kulbak=0
    optrshan=0
    optrkul=0
    print(KFE1_Shennon,dist1, dist2)
    for i in range(min(dist1, dist2)):
      try:
        if (KFE1_Shennon[i]+KFE2_Shennon[i]>maxkfe_Shennon):
          maxkfe_Shennon=KFE1_Shennon[i]+KFE2_Shennon[i]
          optrshan=i
      except:
        pass
    print(optrshan)
    return optrshan

def cnt_kfe(dist, sk_1, sk_2,imagesize):
    k1=[0]*dist
    k2=[0]*dist
    k3=[0]*dist
    k4=[0]*dist
    alpha=[0]*dist
    beta=[0]*dist
    D1=[0]*dist
    D2=[0]*dist
    KFE_Kulbak=[0]*dist
    KFE_Shennon=[0]*dist
    for i in range(dist):
      for j in range(imagesize):
        if sk_1[j]<=i:
          k1[i]+=1
        else: k2[i]+=1
        if sk_2[j]<=i:
          k3[i]+=1
        else: k4[i]+=1
      alpha[i]=k2[i]/imagesize
      beta[i]=k3[i]/imagesize
      D1[i]=k1[i]/imagesize
      D2[i]=k4[i]/imagesize
      if ((k1[i]!=0) and (k2[i]!=0) and (k3[i]!=0) and (k4[i]!=0)):

KFE_Shennon[i]=1+1/2*(k1[i]/(k1[i]+k3[i])*np.log2(k1[i]/(k1[i]+k3
[i])) + k2[i]/(k2[i]+k4[i])*np.log2(k2[i]/(k2[i]+k4[i])) +
k3[i]/(k1[i]+k3[i])*np.log2(k3[i]/(k1[i]+k3[i])) +
k4[i]/(k2[i]+k4[i])*np.log2(k4[i]/(k2[i]+k4[i])))
      else:
        KFE_Shennon[i]=0

KFE_Kulbak[i]=1/imagesize*np.log2((200+(10**(-5))-k2[i]-k3[i])/(k
2[i]+k3[i]+(10**(-5))))*(imagesize-k2[i]-k3[i])
    return (KFE_Shennon, KFE_Kulbak)
```

```
def bestkferad(KFE1, KFE2, dist):
    maxkfe=0
    bestrad=0
    for i in range(dist):
        if (KFE1[i]+KFE2[i]>maxkfe):
            maxkfe=KFE1[i]+KFE2[i]
            bestrad=i
    return (maxkfe, bestrad)

def check(list):
    return all(i == list[0] for i in list)

app = QApplication(sys.argv)
window = MainWindow("window")
window.show()
app.exec()
```