Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ[1]
по дисциплине

«Системы интеллектуальной защиты информации»

Направление подготовки	*01.04.02 «Прикладная информатика»*
Направленность (профиль) образовательной программы	*Математика и информатика в науке и образовании*

Обеспечивающее подразделение
Кафедра ПМ – Прикладная математика

[1] В данном документе представлены типовые оценочные средства. Полный комплект оценочных средств, включающий все варианты заданий (тестов, контрольных работ и др.), предлагаемых обучающемуся, хранится на кафедре в бумажном и электронном виде.

Разработчик ФОС:

Декан ФКТ, к.т.н.		Трещев И.А.
(должность, степень, ученое звание)	(подпись)	(ФИО)

Оценочные материалы по дисциплине рассмотрены и одобрены на заседании кафедры, протокол № _____ от «____» _____ 2022 г.

Заведующий кафедрой ПМ _____ *А.Л. Григорьева*

1 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами образовательной программы

Таблица 1 – Компетенции и индикаторы их достижения

Код и наименование компетенции	Индикаторы достижения	Планируемые результаты обучения по дисциплине
Общепрофессиональные		
ОПК-3 Способен разрабатывать математические модели и проводить их анализ при решении задач в области профессиональной деятельности	ОПК-3.1 Знает основные задачи и области применения методов математического моделирования; особенности объектов моделирования и методики исследования моделей, основные принципы математического моделирования ОПК-3.2 Умеет ставить задачи исследования и оптимизации сложных объектов на основе методов математического моделирования; выявлять общие закономерности исследуемых объектов, выбирать методы исследования математических моделей; строить и исследовать математические модели ОПК-3.3 Владеет методами исследования математических моделей; навыками применения математического аппарата к исследуемым моделям	Знает основные задачи и области применения методов математического моделирования; особенности объектов моделирования и методики исследования моделей, основные принципы математического моделирования Умеет ставить задачи исследования и оптимизации сложных объектов на основе методов математического моделирования; выявлять общие закономерности исследуемых объектов, выбирать методы исследования математических моделей; строить и исследовать математические модели Владеет методами исследования математических моделей; навыками применения математического аппарата к исследуемым моделям
ОПК-4 Способен комбинировать и адаптировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности	ОПК-4.1 Знает содержание ключевых понятий и определений, используемых в теории и практике применения информационных технологий в науке и образовании, информационные ресурсы и базы данных по научно-исследовательской теме, современные и перспективные методы защиты информации ОПК-4.2 Умеет применять прикладное программное обеспечение для решения задач в профессиональной деятельности, науке и образова-	Знает содержание ключевых понятий и определений, используемых в теории и практике применения информационных технологий в науке и образовании, информационные ресурсы и базы данных по научно-исследовательской теме, современные и перспективные методы защиты информации Умеет применять прикладное программное обеспечение для решения задач в профессиональной деятельности, науке и образовании,

	нии, самостоятельно расширять и углублять знания в области информационных технологий ОПК-4.3 Владеет навыками использования прикладного программного обеспечения для решения задач в профессиональной деятельности, науке и образовании, навыками использования интернет-технологий	самостоятельно расширять и углублять знания в области информационных технологий Владеет навыками использования прикладного программного обеспечения для решения задач в профессиональной деятельности, науке и образовании, навыками использования интернет-технологий

Таблица 2 – Паспорт фонда оценочных средств

Контролируемые разделы (темы) дисциплины	**Формируемая компетенция**	**Наименование оценочного средства**	**Показатели оценки**
Классификация и кластеризация **Распознавание образов**	ОПК-3, ОПК-4	Лабораторные работы	Знание основных программных средств, применяемые при автоматизированной разработке ПО Уметь использовать современные технологии программирования и документирования программных комплексов Уметь формулировать требования к создаваемым программным комплексам;
		РГР	Уметь проектировать и разрабатывать ПО Иметь навыки работы с основными технологиями, методами, средствами и навыками выбора, проектирования, реализации, оценки качества и анализа эффективности ПО для решения задач в различных предметных областях

2 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующие процесс формирования компетенций

Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, представлены в виде технологической карты дисци-

плины (таблица 3).

Таблица 3 – Технологическая карта

	Наименование оценочного средства	**Сроки выполнения**	**Шкала оценивания**	**Критерии оценивания**
2 семестр ***Промежуточная аттестация в форме Зачет с оценкой***				
	Лабораторные работы 1-2	В течение семестра	10 баллов/за одну лабораторную работу	10 баллов - студент правильно выполнил практическую работу. Показал отличные знания и умения в рамках освоенного учебного материала. 8 баллов - студент выполнил практическую работу с небольшими неточностями. Показал хорошие знания и умения в рамках освоенного учебного материала. 6 баллов - студент выполнил практическую работу с существенными неточностями. Показал удовлетворительные знания и умения в рамках освоенного учебного материала. 4 баллов - при выполнении практическую работу студент продемонстрировал недостаточный уровень знаний и умений.

	Наименование оценочного средства	Сроки выполнения	Шкала оценивания	Критерии оценивания
				0 баллов – задание не выполнено.
Расчетно-графическая работа		В течение семестра	50	50 баллов - студент правильно выполнил практическую работу. Показал отличные знания и умения в рамках освоенного учебного материала. 30 баллов - студент выполнил практическую работу с небольшими неточностями. Показал хорошие знания и умения в рамках освоенного учебного материала. 20 баллов - студент выполнил практическую работу с существенными неточностями. Показал удовлетворительные знания и умения в рамках освоенного учебного материала. 10 баллов - при выполнении практическую работу студент продемонстрировал недостаточный уровень знаний и умений. 0 баллов – задание не вы-

	Наименование оценочного средства	Сроки выполнения	Шкала оценивания	Критерии оценивания
				полнено.
ИТОГО:		-	70 баллов	-

Критерии оценки результатов обучения по дисциплине:
0 – 64 % от максимально возможной суммы баллов – «неудовлетворительно» (недостаточный уровень для промежуточной аттестации по дисциплине);
65 – 74 % от максимально возможной суммы баллов – «удовлетворительно» (пороговый (минимальный) уровень);
75 – 84 % от максимально возможной суммы баллов – «хорошо» (средний уровень);
85 – 100 % от максимально возможной суммы баллов – «отлично» (высокий (максимальный) уровень)

3 Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующие процесс формирования компетенций в ходе освоения образовательной программы

3.1 Задания для текущего контроля успеваемости

Лабораторная работа 1

Алгоритм kNN или Метод k-ближайших соседей используется для решения задачи классификации. Он относит объекты к классу, которому принадлежит большинство из k его ближайших соседей в многомерном пространстве признаков. Это один из простейших алгоритмов обучения классификационных моделей.

Число k — это количество соседних объектов в пространстве признаков, которые сравниваются с классифицируемым объектом. Иными словами, если k=10, то каждый объект сравнивается с 10-ю соседями. Метод широко применяется в технологиях Data Mining.

В процессе обучения алгоритм просто запоминает все векторы признаков и соответствующие им метки классов. При работе с реальными данными, т.е. наблюдениями, метки класса которых неизвестны, вычисляется расстояние между вектором нового наблюдения и ранее запомненными. Затем выбирается k ближайших к нему векторов, и новый объект относится к классу, которому принадлежит большинство из них.

Выбор параметра k противоречив. С одной стороны, увеличение его значения повышает достоверность классификации, но при этом границы между классами становятся менее четкими. На практике хорошие результаты дают эвристические методы выбора параметра k, например, перекрестная проверка.

Несмотря на свою относительную алгоритмическую простоту, метод показывает хорошие результаты. Главным его недостатком является высокая вычислительная трудоемкость, которая увеличивается квадратично с ростом числа обучающих примеров.

На первом шаге алгоритма задается число k – количество ближайших соседей. Следует задавать значение k > 1, так как при k = 1 алгоритм теряет обобщающую способность. Также не следует задавать значение k слишком большим, потому что в таком случае не будут выявлены многие локальные особенности.

Затем находим k записей с минимальным расстоянием до вектора признаком нового объекта, то есть ищем k соседей.

Для упорядоченных значений атрибутов вычисляется Евклидово расстояние по формуле (1).

$$D_E = \sqrt{\sum_{i}^{n} (x_i - y_i)^2} \quad (1)$$

В данной формуле приняты следующие обозначения:

n – количество параметров;

x_i, y_i … - обозначение параметров объекта.

Далее рассчитывается вероятность принадлежности неизвестной точки к классу путем деления: определяется сколько соседей принадлежит к конкретному классу и делится на k.

При вычислении вероятностей может произойти так, что несколько классов будут иметь равную вероятность. В такой ситуации учитывается также и расстояние до новой записи. Чем меньше расстояние, тем более значимый вклад вносит голос. Голоса за класс находятся по формуле (2).

$$votes(class) = \sum_{i}^{n} \frac{1}{d^2(x_i, y_i)} \quad (2)$$

В формуле (2):

$d^2(x_i, y_i)$ – квадрат расстояния от известной записи из обучающих заданий до неизвестной записи;

Задание:

Пусть система интеллектуальной защиты информации зависит от двух параметров. Реализовать классификацию красных и синих точек (разделение по параметрам)

Для выполнения классификации kNN напишем программу –приложение Windows Forms на языке C#, генерирующее тестовые данные и классифицирующее объект с учетом k ближайших значений. Значение k вводится пользователем

Результатом работы программы является строка определяющая принадлежность объекта к определенному классу.

Рассмотрим принцип работы программы. На рисунке 1 показан внешний вид программы после запуска.

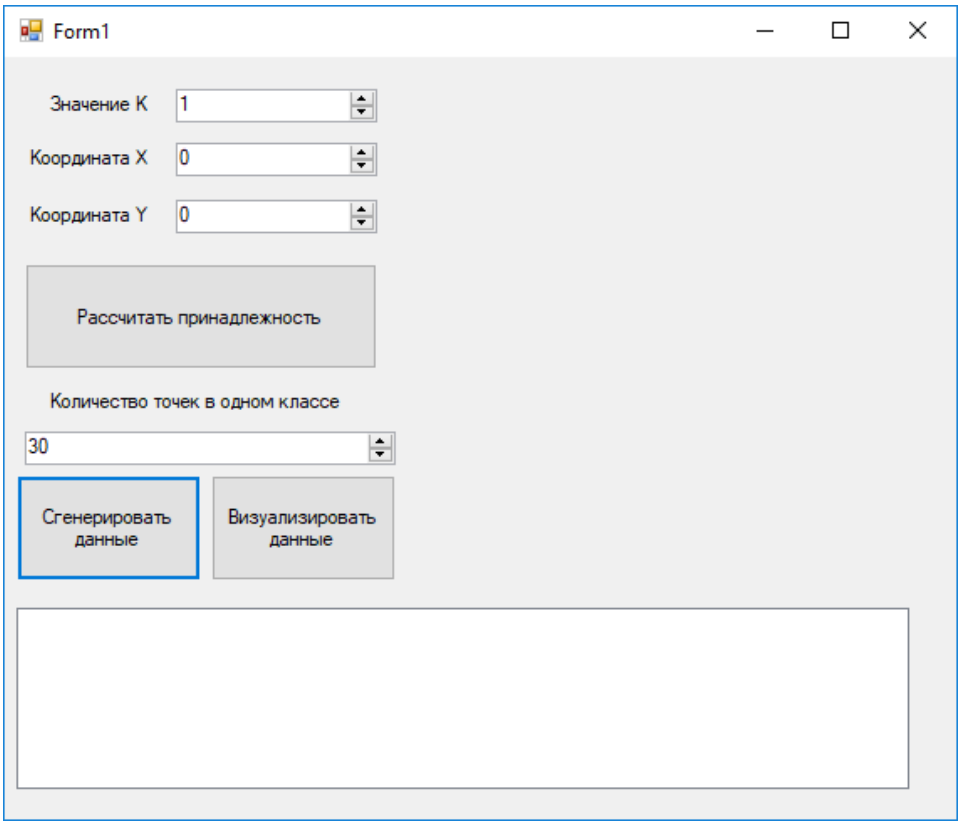

Рисунок 1 – Внешний вид программы

При нажатии на кнопку «Сгенерировать данные» создаются 2 текстовых файла с данными, содержащими координаты точек на плоскости, принадлежащий к первому или второму классу (рисунки 2-3).

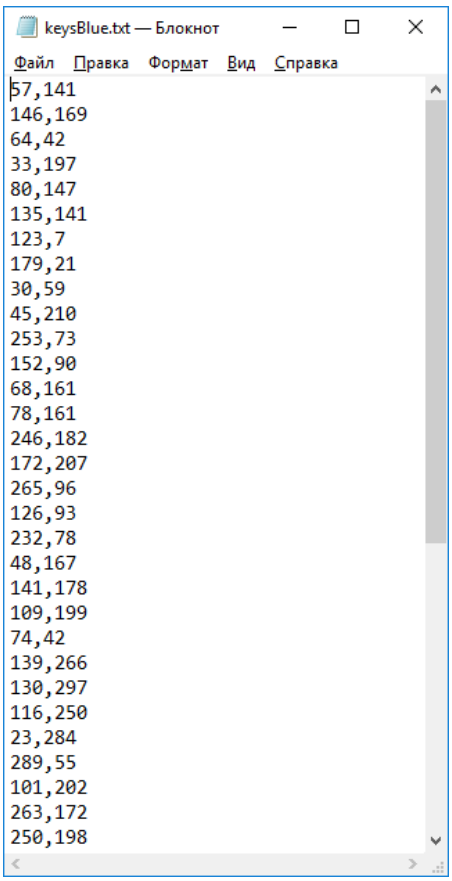

Рисунок 2 – Текстовый файл для синих точек

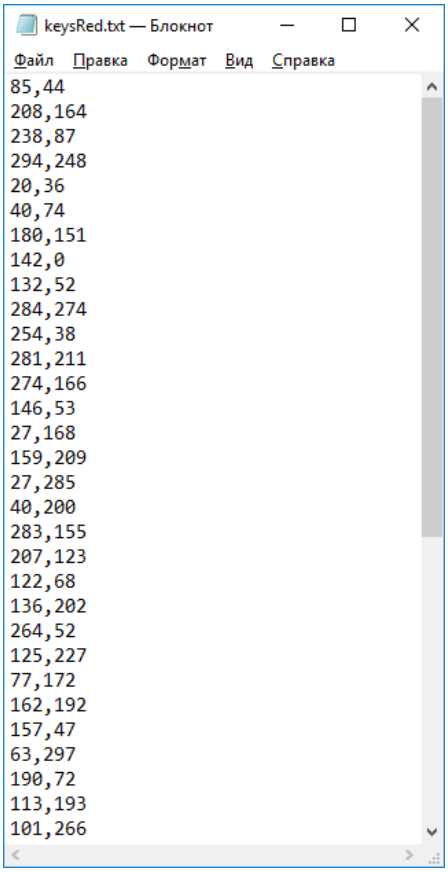

Рисунок 3 – Текстовый файл для красных точек

При нажатии на кнопку «Визуализировать данные» появляется картинка, на которой координаты вышеупомянутых точек окрашены в синий либо красный цвета в зависимости от класса, к которому они принадлежат (рис. 4)

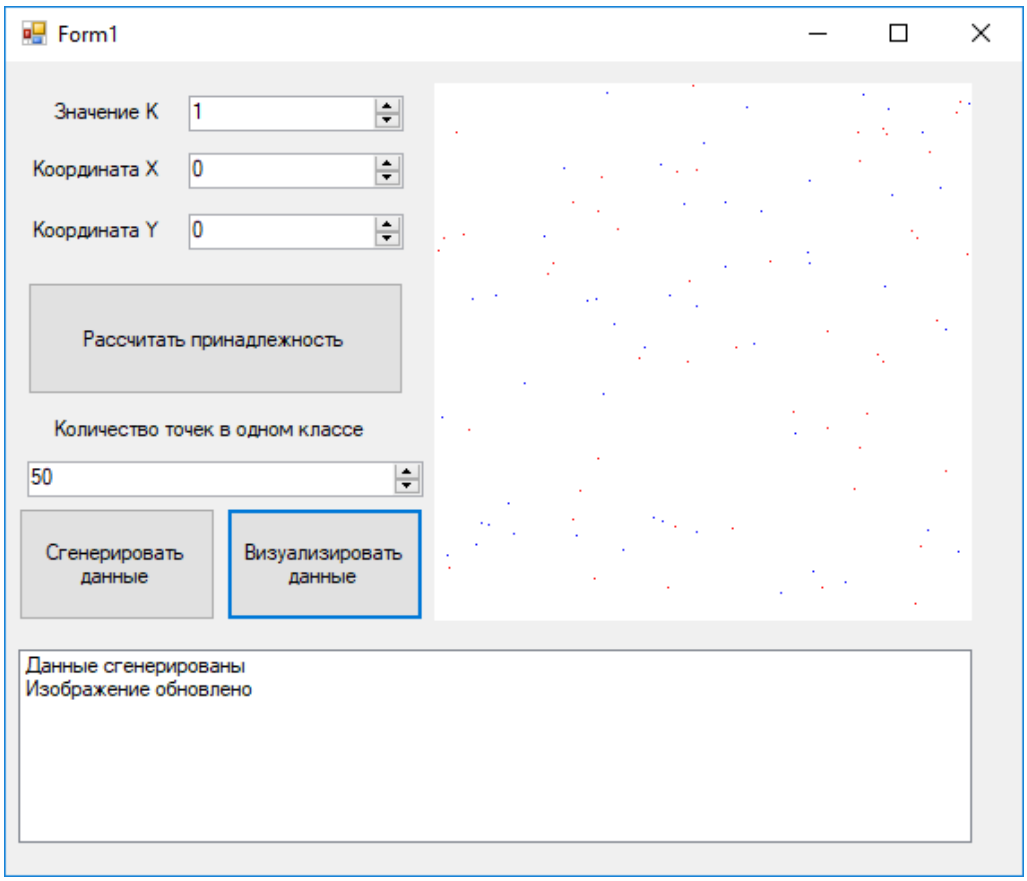

Рисунок 4 – Визуализация данных

Далее пользователь вводит координаты произвольной точки и указывает значение k. При нажатии на кнопку «Рассчитать принадлежность» программа, используя метод kNN, определяет к какому классу цветов относится объект и выдает строку с результатом (рисунок 5). При увеличении значения k возможно изменение принадлежности объекта к определенному классу ввиду более точных расчетов (рисунок 6).

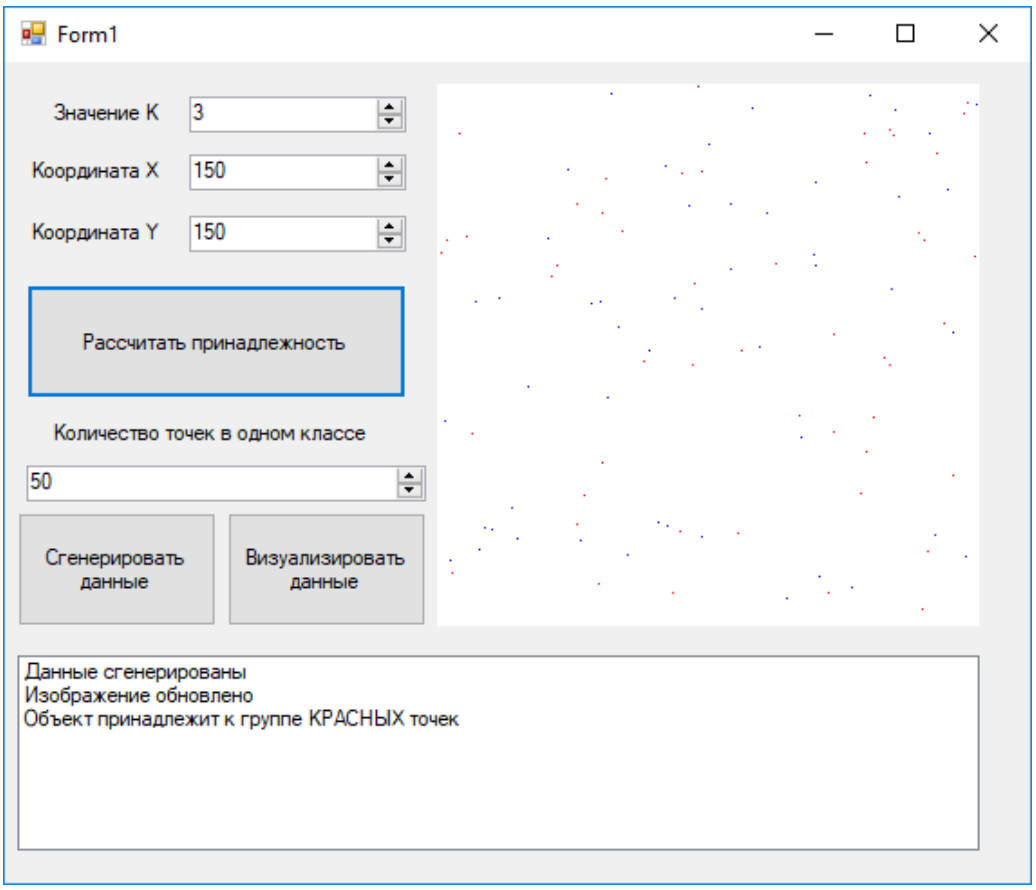

Рисунок 5 – Классификация точки методом kNN (k=3)

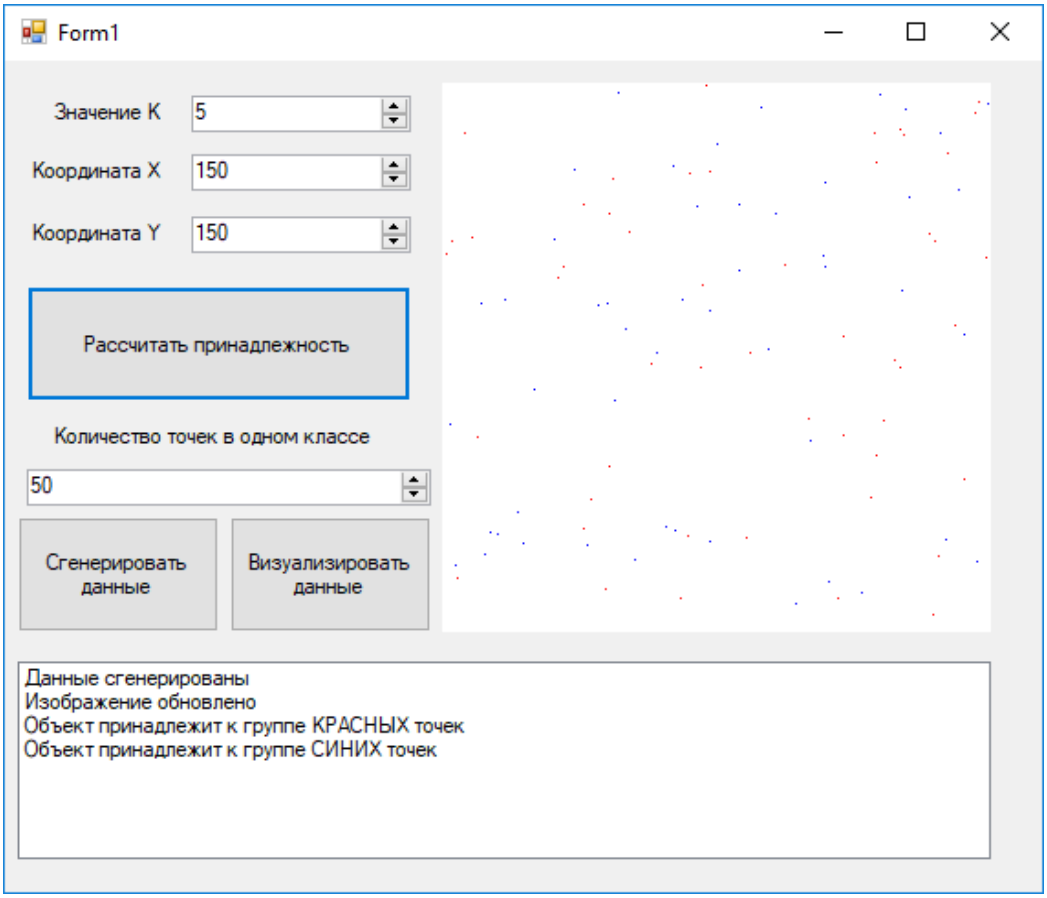

Рисунок 6 – Классификация точки методом kNN (k=5)

Примеры с другими точками и данными приведены на рис. 7-9.

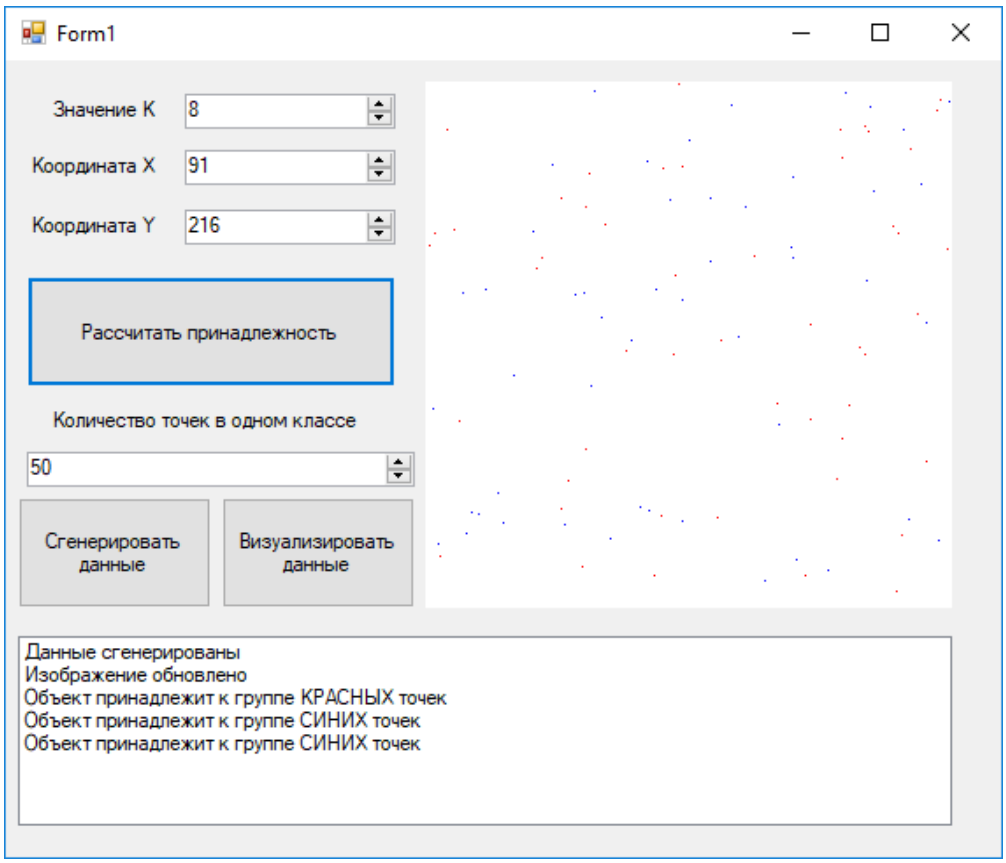

Рисунок 7 – Пример классификации методом kNN

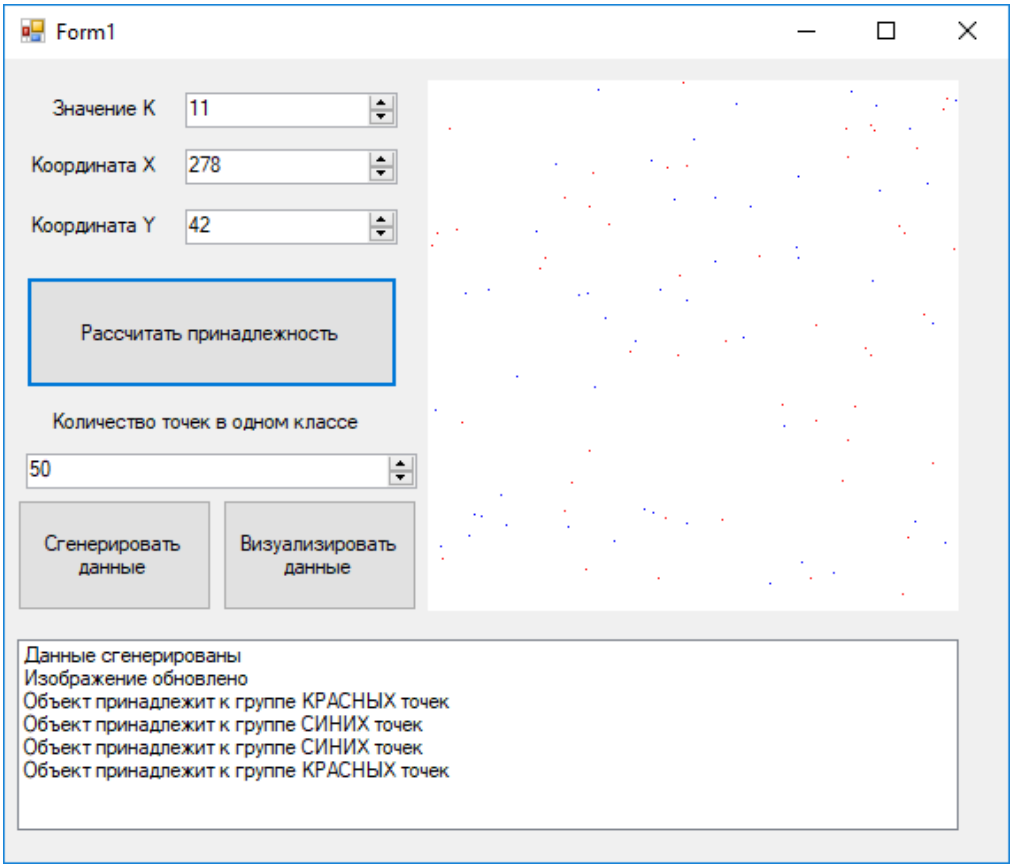

Рисунок 8 – Пример классификации методом kNN

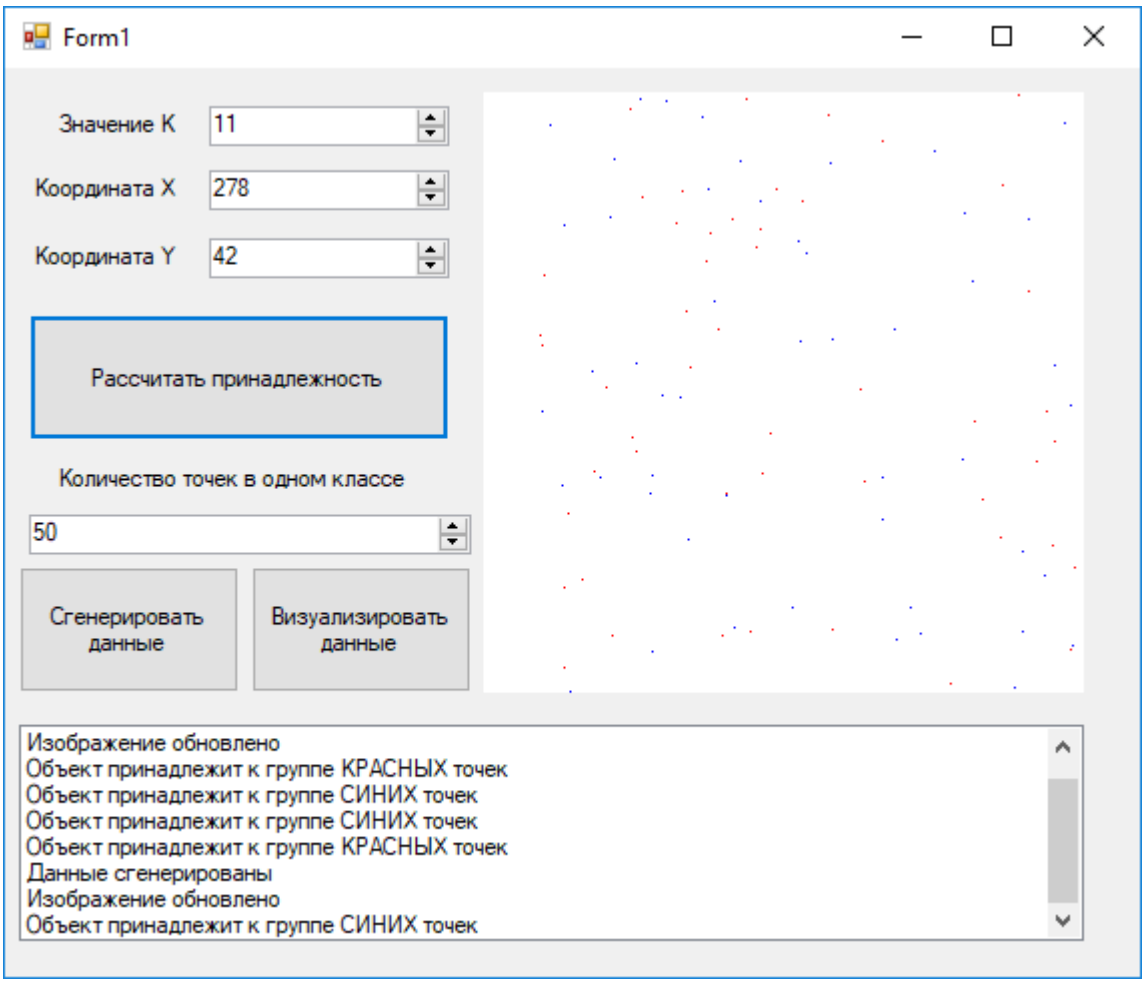

Рисунок 9 – Пример классификации методом kNN

Пример исходного текста на C# разработанного приложения

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Algoritm_kNN
{
public partial class Form1 : Form
{
public int numberOfPoints = 30;
public Form1()
{
InitializeComponent();
}

private void label3_Click(object sender, EventArgs e)
{

}

private void generateButton_Click(object sender, EventArgs e)
{
Random rand = new Random();
```

```
int N1 = 300;
int N2 = 300; //размер плоскости
numberOfPoints = Convert.ToInt32(numericUpDown4.Value); //количество генерируемых значений
for (int m = 0; m < 2; m++) //выполняется дважды для каждого класса значений
{
int[] key1 = new int[numberOfPoints];
int[] key2 = new int[numberOfPoints];
bool check = true;
for (int i = 0; i < numberOfPoints; i++) //генерация значений
{
check = true;
int temp1 = rand.Next(0, N1);
int temp2 = rand.Next(0, N2);
for (int j = 0; j < key1.Length; j++) if (Math.Abs(temp1 - key1[j]) <= 1)
for (int k = 0; k < key2.Length; k++) if (Math.Abs(temp1 - key2[k]) <= 1) { check = false; break; }

if (check)
{
key1[i] = temp1;
key2[i] = temp2;
}
else i--;
}
if (m == 0)
{
using (StreamWriter sw = new StreamWriter(@"D:\fff\keysBlue.txt", false, System.Text.Encoding.Default))
{
for (int i = 0; i < numberOfPoints; i++) //сохранение значений класса СИНИЙ
{
string temp = key1[i] + "," + key2[i];
sw.WriteLine(temp);
}
}
}
else
{
using (StreamWriter sw = new StreamWriter(@"D:\fff\keysRed.txt", false, System.Text.Encoding.Default))
{
for (int i = 0; i < numberOfPoints; i++) //сохранение значений класса КРАСНЫЙ
{
string temp = key1[i] + "," + key2[i];
sw.WriteLine(temp);
}
}
}
}
StatusListBox.Items.Add("Данные сгенерированы\n");
}

private void numericUpDown4_ValueChanged(object sender, EventArgs e)
{

}

private void redrawButton_Click(object sender, EventArgs e)
{
string path;
Bitmap image1 = new Bitmap(@"D:\fff\1.png", true); //загрузка изображения
for (int m = 0; m < 2; m++) //заполнение пустого изображения точками из сгенерированных файлов сначала синего цвета, затем красного
{
if (m == 0) path = @"D:\fff\keysBlue.txt";
```

```
else path = @"D:\fff\keysRed.txt";
using (StreamReader sr = new StreamReader(path, System.Text.Encoding.Default))
{
for (int i = 0; i < numberOfPoints; i++)
{
string temp = sr.ReadLine();
string[] coordinates = temp.Split(',');
int x = Convert.ToInt32(coordinates[0]);
int y = Convert.ToInt32(coordinates[1]);
Color bs = image1.GetPixel(x, y);
if (image1.GetPixel(x, y) == Color.FromArgb(255, 255, 255, 255))
{
if (m == 0) image1.SetPixel(x, y, Color.Blue);
else image1.SetPixel(x, y, Color.Red);
}

}
}
}
image1.Save(@"D:\fff\new.png");
using (var file = new FileStream(@"D:\fff\new.png", FileMode.Open, FileAccess.Read,
FileShare.Inheritable))
{
pictureBox1.Image = Image.FromStream(file); //обновление картинки

}
StatusListBox.Items.Add("Изображение обновлено\n");

}

private void pictureBox1_Click(object sender, EventArgs e)
{

}

private void pictureBox1_Click_1(object sender, EventArgs e)
{

}

private void calculateButton_Click(object sender, EventArgs e)
{
double[] distanceArrayBlue = new double[numberOfPoints];
double[] distanceArrayRed = new double[numberOfPoints]; //инициализация массивов для хранения рас-
считанных расстояний
using (StreamReader sr = new StreamReader(@"D:\fff\keysBlue.txt", System.Text.Encoding.Default))
{
for (int i = 0; i < numberOfPoints; i++) //расчет расстояния от точки до всех объектов принадлежащих
классу СИНИЙ
{
string temp = sr.ReadLine();
string[] coordinates = temp.Split(',');
int x = Convert.ToInt32(coordinates[0]);
int y = Convert.ToInt32(coordinates[1]);
distanceArrayBlue[i] = Math.Sqrt(Math.Pow(Convert.ToInt32(numericUpDownX.Value) - x, 2) +
Math.Pow(Convert.ToInt32(numericUpDownY.Value) - y, 2));

}
double tmp;
for (int i=0; i< numberOfPoints;i++)
{
```

```
for (int j= i+1; j < numberOfPoints; j++) //сортировка массива в порядке возрастания значений
{ (изм.)
•
if (distanceArrayBlue[i] > distanceArrayBlue[j])
{
tmp = distanceArrayBlue[i];
distanceArrayBlue[i] = distanceArrayBlue[j];
distanceArrayBlue[j] = tmp;
}
}
}
}
using (StreamReader sr = new StreamReader(@"D:\fff\keysRed.txt", System.Text.Encoding.Default))
{
for (int i = 0; i < numberOfPoints; i++) //расчет расстояния от точки до всех объектов принадлежащих классу
КРАСНЫЙ
{
string temp = sr.ReadLine();
string[] coordinates = temp.Split(',');
int x = Convert.ToInt32(coordinates[0]);
int y = Convert.ToInt32(coordinates[1]);
distanceArrayRed[i] = Math.Sqrt(Math.Pow(Convert.ToInt32(numericUpDownX.Value) - x, 2) +
Math.Pow(Convert.ToInt32(numericUpDownY.Value) - y, 2));
}
double tmp;
for (int i = 0; i < numberOfPoints; i++)
{
for (int j = i + 1; j < numberOfPoints; j++) //сортировка массива в порядке возрастания значений
{
if (distanceArrayRed[i] > distanceArrayRed[j])
{
tmp = distanceArrayRed[i];
distanceArrayRed[i] = distanceArrayRed[j];
distanceArrayRed[j] = tmp;
}
}
}
}
int K = Convert.ToInt32(numericUpDownK.Value), m = 0, n = 0;
if (distanceArrayBlue[0] == 0 || distanceArrayRed[0] == 0) StatusListBox.Items.Add("Координаты точки совпали
с шаблоном\n"); //проверка на совпадение с уже отмеченными объектами
else
{
while (K != 0) //перебор k ближайших соседей точки сравнением расстояний объектов разных классов до неё
{
if (K == 1 && distanceArrayBlue[m] == distanceArrayRed[n]) break;
else if (distanceArrayBlue[m] > distanceArrayRed[n]) n++;
else if (distanceArrayBlue[m] <= distanceArrayRed[n]) m++;
K--;
}
if (m > n) StatusListBox.Items.Add("Объект принадлежит к группе СИНИХ точек\n");
else if (m < n) StatusListBox.Items.Add("Объект принадлежит к группе КРАСНЫХ точек\n");
else
{
double summBlue = 0, summRed = 0;
for (int i = 0; i < m; i++) summBlue += distanceArrayBlue[i];
for (int i = 0; i < m; i++) summRed += distanceArrayRed[i]; //при равенстве количества ближайших объектов
расчет наименьшей суммы всех расстояний объектов каждого класса до точки
if (summRed > summBlue) StatusListBox.Items.Add("Объект принадлежит к группе СИНИХ точек.\n");
else if (summRed < summBlue) StatusListBox.Items.Add("Объект принадлежит к группе КРАСНЫХ точек.\n");
else StatusListBox.Items.Add("Не удалось определить принадлежность данной точки. Измените значение
K.\n");
```

```
}
}
}
}
}
```

Лабораторная работа 2

Считая что на вход пропускной системы подается массив различных точек(для системы распознавания) реализовать алгоритм k-means для кластеризации.

Алгоритм K-means относится к алгоритмам обучения без учителя. На вход алгоритма подаётся набор точек произвольной размерности. Задача алгоритма – разделить их на заранее известное число кластеров.

Шаги алгоритма:

1) выбирается число кластеров K;
2) выбирается K случайных точек, которые будут служить начальными центрами кластеров;
3) для каждой точки определяется ближайшая центроида, при этом точки, «притянутые» к одной точке из 2-го шага, образуют кластер;
4) вычисляются центроиды – центры тяжести кластеров. Каждый центроид – это вектор, элементы которого представляют собой средние значения соответствующих признаков, вычисленные по всем записям кластера;
5) центр кластера смещается в его центроид, после чего центроид становится новым центром кластера;
6) если на 5 шаге произошло смещение, то алгоритм повторяется с третьего шага. Если же нет, то алгоритм завершил свою работу.

Преимуществом данного алгоритма являются его простота реализации и скорость работы. Недостатком – неопределённость выбора начальных центров кластеров (на шаге 2) и то, что число кластеров должно быть известно заранее.

Описание работы

Реализация алгоритма K-means осуществлена на языке программирования Java с использованием библиотеки графического интерфейса JavaFX.

Исходный код программы состоит из нескольких частей:

1) Main.java – содержит точку входа (приложение А);
2) Controller.java – содержит логику графического интерфейса (приложение Б);
3) kmeans.fxml – содержит разметку графического интерфейса (приложение В);
4) clustering/KmeansAlgorithm.java – содержит код реализации алгоритма K-means (приложение Г);
5) clustering/Point.java – содержит класс многомерной точки, используется в реализации алгоритма K-means (приложение Д);
6) exampleData.txt – пример данных для обучения (приложение Е).

При запуске скомпилированного и собранного проекта откроется окно, изображённое на рисунке 2.1.

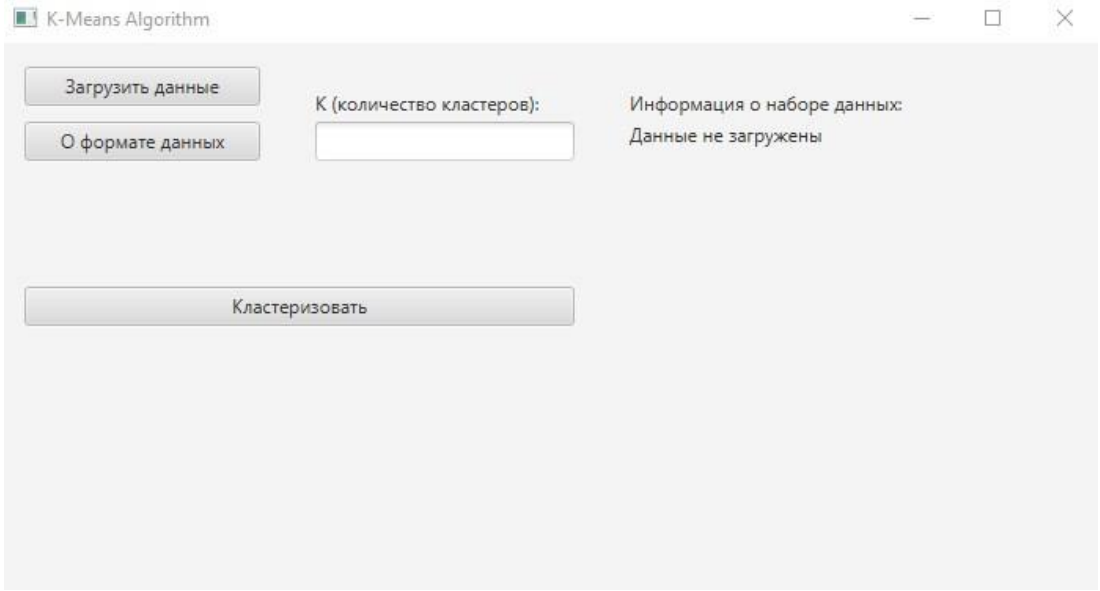

Рисунок 2.1 – Главное окно программы

Далее необходимо ознакомиться с форматом входных данных. Информация об этом доступна по нажатию кнопки «О формате данных». При этом откроется окно, изображённое на рисунке 2.2.

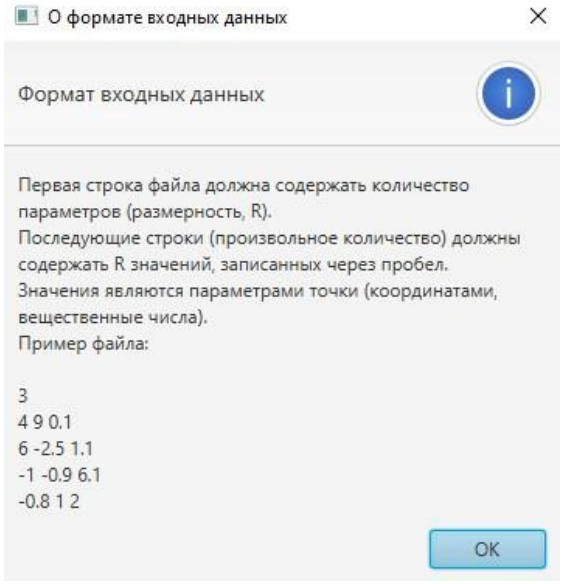

Рисунок 2.2 – Формат входных данных

После ознакомления с форматом входных данных необходимо создать файл, содержащий входные данные (данные для обучения). Пример такого файла приведён в приложении Е.
Далее необходимо загрузить данные. Это производится путём нажатия на кнопку «Загрузить данные». Откроется окно, где требуется выбрать файл входных данных. При невозможности открыть и прочитать файл, пользователь увидит сообщение об ошибке, пример которого изображён на рисунке 2.3.

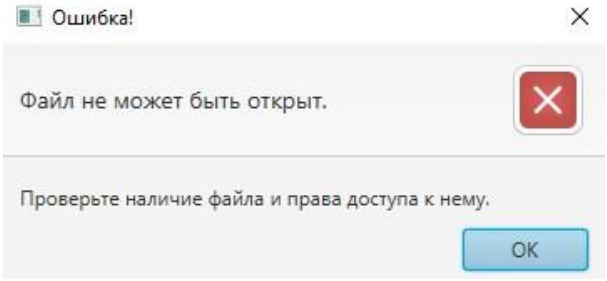

Рисунок 2.3 – Ошибка открытия файла

Если же файл был успешно открыт и прочитан, однако формат данных в файле не соответствует требуемому, пользователь в правой части программы увидит сообщение, представленную на рисунке 2.4.

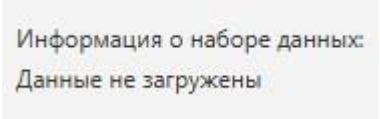

Рисунок 2.4 – Несоответствие формата данных

При успешной загрузке данных в правой части программы будет отображена информация о наборе данных. Её пример приведён на рисунке 2.5.

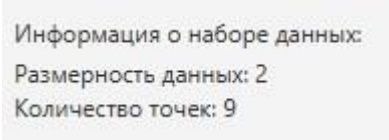

Рисунок 2.5 – Информация о наборе данных

Далее необходимо указать значение K – количество кластеров. Это производится в соответствующем поле вверху окна. При указании некорректного значения K пользователь увидит сообщение об ошибке, пример которого изображён на рисунке 2.6.

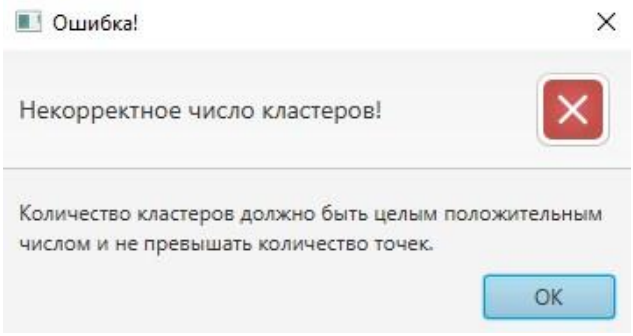

Рисунок 2.6 – Некорректное значение K

После ввода всех необходимых параметров при нажатии кнопки «Кластеризовать» будет запущен метод predictClusters из KmeansAlgorithm.java.
После окончания работы алгоритма будет создан файл output.txt в папке с программой, в котором содержится информация о центрах кластеров и список точек с указанием кластера. Пример файла изображён на рисунке 2.7.

```
Centroids:
Cluster 0: Point(0.3333333333333333 0.3333333333333333)
Cluster 1: Point(10.333333333333334 10.333333333333334)
Cluster 2: Point(-10.333333333333334 -10.333333333333334)

Points:
Point(10.0 10.0): cluster 1
Point(11.0 10.0): cluster 1
Point(10.0 11.0): cluster 1
Point(0.0 0.0): cluster 0
Point(1.0 0.0): cluster 0
Point(0.0 1.0): cluster 0
Point(-10.0 -10.0): cluster 2
Point(-11.0 -10.0): cluster 2
Point(-10.0 -11.0): cluster 2
```

Рисунок 2.7 – Результат работы программы

В некоторых случаях центр кластера может иметь координаты NaN. Это возникает в тех случаях, когда на втором шаге алгоритма точки сгенерировались слишком близко друг к другу, из-за чего одна точка «забирает на себя» точки из выборки, а другая сгенерированная точка перестаёт участвовать в алгоритме из-за того, что к ней не привязана ни одна точка из выборки. Такой случай изображён на рисунке 2.8. Возможность возникновения таких случаев является недостатком алгоритма, поэтому иногда для кластеризации применяют другие алгоритмы, похожие на K-means, например, g-means (gaussianmeans, распределение в кластерах стремится к нормальному).

```
Centroids:
Cluster 0: Point(5.333333333333333 5.333333333333333)
Cluster 1: Point(NaN NaN)
Cluster 2: Point(-10.333333333333334 -10.333333333333334)

Points:
Point(10.0 10.0): cluster 0
Point(11.0 10.0): cluster 0
Point(10.0 11.0): cluster 0
Point(0.0 0.0): cluster 0
Point(1.0 0.0): cluster 0
Point(0.0 1.0): cluster 0
Point(-10.0 -10.0): cluster 2
Point(-11.0 -10.0): cluster 2
Point(-10.0 -11.0): cluster 2
```

Пример разработанного приложения на java.

Main.java

```
package com.sbelousov.kmeans;

import javafx.application.Application; import ja-
vafx.fxml.FXMLLoader; import javafx.scene.Parent; import ja-
vafx.scene.Scene; import javafx.stage.Stage;

public class Main extends Application {

   @Override
   public void start(Stage primaryStage) throws Exception{
      Parent root = FXML-
Loader.load(getClass().getResource("kmeans.fxml"));        primaryStage.setTitle("K-Means Algorithm");
primaryStage.setScene(new Scene(root, 700, 350));        primaryStage.show();
   }

   public static void main(String[] args) {        launch(args);
   }
}
```

Controller.java

```
package com.sbelousov.kmeans;

import com.sbelousov.kmeans.clustering.KmeansAlgorithm; import javafx.fxml.FXML; import
javafx.scene.control.Alert; import javafx.scene.control.Label; import
javafx.scene.control.TextField; import javafx.scene.layout.VBox; import
javafx.stage.FileChooser;

import java.io.File;

public class Controller {

   public KmeansAlgorithm kmeansAlgorithm = new KmeansAlgorithm();

   @FXML public VBox vbox;
   @FXML public Label dataInfoText;
   @FXML public TextField kField;

   private void showAlert(String title, String headerText,
String contentText, Alert.AlertType type) {         Alert alert = new Alert(type);
alert.setTitle(title);         alert.setHeaderText(headerText);
alert.setContentText(contentText);         alert.showAndWait();
   }
   public void onLoadDataButtonClick() {
      FileChooser fileChooser = new FileChooser();
      File             file         =          fileChooser.showOpenDialog(vbox.getScene().getWindow());         if (file ==
null) {            showAlert("Ошибка!",
                "Файл не может быть открыт.",
                "Проверьте наличие файла и права доступа к нему.",
                Alert.AlertType.ERROR);             return;
      }

      try {
         kmeansAlgorithm.parseFile(file);         }
```

```
        catch (NumberFormatException e) {            showAlert("Ошибка!",
                "Некорректный формат файла.",
                "Проверьте соблюдение формата входных данных.",
                Alert.AlertType.ERROR);            return;
        }

        dataInfoText.setText("Размерность данных: " + kmeansAlgorithm.getDimensionNumber() + "\n" +
                "Количество точек: " + kmeansAlgorithm.getSize()
+ "\n");
    }
    public void onAboutDataFormatButtonClick() {        showAlert("О формате входных данных",
            "Формат входных данных",
            "Первая строка файла должна содержать количество параметров (размерность, R).\n" +
                    "Последующие строки (произвольное количество) должны содержать R значений, " +
                    "записанных через пробел.\n" +
                    "Значения являются параметрами точки
(координатами, вещественные числа).\n" +
                    "Пример файла:\n\n" +
                    "3\n" +
                    "4 9 0.1\n" +
                    "6 -2.5 1.1\n" +
                    "-1 -0.9 6.1\n" +
                    "-0.8 1 2",
            Alert.AlertType.INFORMATION);
    }
    public void onComputeButtonClick() {
        // Set number of clusters        try {
            kmeansAlgorithm.setClusters(kField.getText());
        }
        catch (NumberFormatException e) {            showAlert("Ошибка!",
                "Некорректное число кластеров!",
                "Количество кластеров должно быть целым положительным числом " +
                        "и не превышать количество точек.",
                Alert.AlertType.ERROR);            return;
        }

        // Clustering
        kmeansAlgorithm.predictClusters();

        // Output
```

```
        return;
    }
}
```

kmeans.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>

<VBox prefHeight="700.0" prefWidth="350.0" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1"       fx:controller="com.sbelousov.kmeans.Controller" fx:id="vbox">
   <AnchorPane VBox.vgrow="ALWAYS">
      <Button layoutX="15.0" layoutY="15.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="150.0"
            text="Загрузить данные" onAction="#onLoadDataButtonClick"/>
      <Button layoutX="15.0" layoutY="50.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="150.0"
text="О формате данных" onAction="#onAboutData-
FormatButtonClick"/>

      <Button layoutX="15.0" layoutY="155.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="350.0"
text="Кластеризовать" onAction="#onComputeButtonClick"/>

      <Label        layoutX="200.0"              layoutY="30.0"   text="K (количество кластеров):"/>
      <TextField fx:id="kField" layoutX="200.0" layoutY="50.0" prefHeight="25.0" prefWidth="165.0"/>

      <Label layoutX="400.0" layoutY="30.0" text="Информация о наборе данных:"/>
      <Label fx:id="dataInfoText" layoutX="400.0" layoutY="50.0" text="Данные не загружены"/>
   </AnchorPane>
</VBox>
```

KmeansAlgorithm.java

```
package com.sbelousov.kmeans.clustering;

import java.io.File; import java.io.IOException;
import java.nio.file.Files; import ja-
va.nio.file.Path; import java.nio.file.Paths; im-
port java.util.*;
import java.util.stream.Collectors;

public class KmeansAlgorithm {     private int clusters = 2;     pri-
vate int dimensionNumber = 0;
   private final List<Point> points = new ArrayList<>();     private List<Point> centroids = new ArrayList<>();

   public void setClusters(int clusters) {
      if (clusters <= 0 || clusters > this.getSize()) {            throw new NumberFormatException("Incorrect number
K");        }
      this.clusters = clusters;
   }
   public void setClusters(String clusters) {         this.setClusters(Integer.parseInt(clusters));
   }
   public int getSize() {         return points.size();
   }
   public int getDimensionNumber() {         return dimensionNumber;
   }
   public void parseFile(File file) {
      Path path = file.toPath();
      List<String> lines = new ArrayList<>();         try {
         lines = Files.readAllLines(path);
      }
      catch (IOException e) {
         e.printStackTrace();
      }
```

```
        dimensionNumber = Integer.parseInt(lines.get(0));        for (int i = 1; i < lines.size(); i++) {
            List<Double> coordinates = new ArrayList<>();            String[] splitted =
lines.get(i).split("\\s+");            for (int j = 0; j < dimensionNumber; j++) {                coordi-
nates.add(Double.parseDouble(splitted[j]));            }
            points.add(new Point(coordinates));
        }
    }
    public void writeToFile() {
        Path path = Paths.get("output.txt");
        StringBuilder stringBuilder = new StringBuilder();        try {
            stringBuilder.append("Centroids:\n");            for (int i = 0; i < centroids.size(); i++) {                string-
Builder.append("Cluster ").append(i).append(": ").append(centroids.get(i)).append("\n");
            }
                                stringBuilder.append("\nPoints:\n");            for (Point p : points) {                stringBuild-
                                                                                                er.append(p).append(":     cluster
").append(p.getCluster()).append("\n");
            }
            Files.writeString(path, stringBuilder);
        }
        catch (IOException ex) {            ex.printStackTrace();
        }
    }
    private double getMinimalCoordinate() {
        List<Double> coordinates = new ArrayList<>();        for (Point p : points) {
            coordinates.addAll(p.getCoordinates());
        }
        return Collections.min(coordinates);
    }
    private double getMaximalCoordinate() {
        List<Double> coordinates = new ArrayList<>();        for (Point p : points) {
            coordinates.addAll(p.getCoordinates());
        }
        return Collections.max(coordinates);
    }
    public void predictClusters() {
        // Step 0: Determine min and max ranges for choosing random points in next step
        double rangeMin = getMinimalCoordinate();
```

```
    double rangeMax = getMaximalCoordinate();

    // Step 1: Select K random points as cluster centers called centroids
    centroids = new ArrayList<>();        for (int i = 0; i < clusters; i++) {
      List<Double> coordinates = new ArrayList<>();           for (int j = 0; j < dimensionNumber; j++) {
Random random = new Random();
        double coordinate = rangeMin + (rangeMax rangeMin) * random.nextDouble();
        coordinates.add(coordinate);
      }
      Point centroid = new Point(coordinates);           centroids.add(centroid);
    }

    boolean flag;        do {
      flag = false;

      System.out.println("Centroids:");           for (Point c : centroids) {
        System.out.println(c);
      }
      System.out.println();

      // Step 1.9: Save current centroids coordinates
      List<List<Double>> oldCoordinates = new ArrayList<>();
      for (int i = 0; i < centroids.size(); i++) {              oldCoordinates.add(new ArrayList<>());
for (int j = 0; j < dimensionNumber; j++) {                oldCoordi-
nates.get(i).add(centroids.get(i).getCoordinates().get(j));
        }
      }

      // Step 2: Calculate distances from each point to each centroid
      Map<Point, List<Double>> distancesFromPoints = new HashMap<>();
      for (Point centroid : centroids) {              for (Point p : points) {
        if (!distancesFromPoints.containsKey(p)) {                    distancesFromPoints.put(p, new Ar-
rayList<>());
        }
        distancesFromPoints.get(p).add(p.getEuclideanDistance(centroid));
      }
    }
```

```
            // Step 2.5: Assign cluster number
            for (Map.Entry<Point, List<Double>> entry : distancesFromPoints.entrySet()) {
                Point point = entry.getKey();
                double minDistance = Collections.min(entry.getValue());
                int clusterNumber = entry.getValue().indexOf(minDistance);
                point.setCluster(clusterNumber);
            }

            // Step 3: Move centroids
            for (int i = 0; i < centroids.size(); i++) {                int finalI = i;
                List<Point> pointList = points.stream().filter(p
-> p.getCluster() == finalI)
                        .collect(Collectors.toList());
                List<Double> newCoordinates = new ArrayList<>();                for (int j = 0; j < dimensionNumber;
j++) {                        double sum = 0;
                    for (Point p : pointList) {
                        sum += p.getCoordinates().get(j);
                    }
                    newCoordinates.add(sum / pointList.size());
                }
                centroids.get(i).setCoordinates(newCoordinates);            }

            // Step 3.1: If centroids weren't moving that it's the end
            for (int i = 0; i < centroids.size(); i++) {
                if                                                                                (!centroids.get(i).getCoordi-
nates().equals(oldCoordinates.get(i))) {                    flag = true;
break;
                }
            }

        } while (flag);

        writeToFile();
    }
}
```

Point.java

```
package com.sbelousov.kmeans.clustering;

import java.util.ArrayList; import java.util.List;

public class Point {
   private List<Double> coordinates;     private int cluster;

   public List<Double> getCoordinates() {         return coordinates;
   }
   public void setCoordinates(List<Double> coordinates) {         this.coordinates = coordinates;
   }
   public int getCluster() {         return cluster;
   }
   public void setCluster(int cluster) {         this.cluster = cluster;
   }
   public Point(List<Double> coordinates) {         this.coordinates = coordinates;
   }
   public double getEuclideanDistance(Point point) {
      List<Double> items = new ArrayList<>();

      if (this.getCoordinates().size() != point.getCoordinates().size()) {
         throw new IllegalArgumentException("Dimensions of two points are unequal");
      }
      int dimensionsNumber = this.getCoordinates().size();

      for (int i = 0; i < dimensionsNumber; i++) {             double oneCoordinate = this.getCoordinates().get(i);
double anotherCoordinate = point.getCoordinates().get(i);             items.add(Math.pow(oneCoordinate - another-
Coordinate, 2));         }
```

```
        double sum = items.stream().mapToDouble(Double::doubleValue).sum();
        return Math.sqrt(sum);
    }
    public String toString() {
        StringBuilder builder = new StringBuilder();        builder.append("Point(");
        for (Double coordinate : coordinates) {             builder.append(coordinate).append(" ");
        }
        builder.deleteCharAt(builder.length() - 1);        builder.append(")");        return build-
er.toString();
    }
}
```

exampleData.txt

```
2
10 10
11 10
10 11
0 0
1 0
0 1
-10 -10
-11 -10
-10 -11
```

Темы / задания расчетно-графической работы

Считая что пропускная система распознает лица проходящих через нее людей Написать программу, которая будет находить лицо на изображении. Использовать библиотеку Dlib.

1 Обнаружение лиц на изображении

В данной работы будет использоваться dlib и OpenCV для обнаружения лицевых ориентиров на изображении, которые используются для локализации ключевых областей на лице человека:

- Глаза;
- Брови;
- Нос;
- Рот;
- Линия подбородка.

Выявление лицевых ориентиров состоит из двух этапов:

- Локализация лица на изображении;
- Обнаружение значимых областей на лицах в областях интереса.

Существует множество методов обнаружения лиц на изображении.

Нас интересует алгоритм, который определяет контуры лица, т.е. значения точек с координатами (x;y).

Определив на изображении область с лицом человека, можно переходить к шагу 2 – локализация отдельных частей лица.

В методе, который используется в библиотеке dlib применяется следующее:

- Тренировочные задания с маркированными лицевыми ориентирами на изображениях. Эти изображения промаркированы вручную точками (определяющимися координатами (x;y)), окружающими каждую ключевую область лица;
- Вероятности расстояний между парами пикселей на входных изображениях.

Данный обученный детектор из библиотеки dlib использует положение 68 точек, заданных координатами (x;y), соответствующих положению ключевых частей на лице человека.

Индексы точек приводятся на рисунке 1.

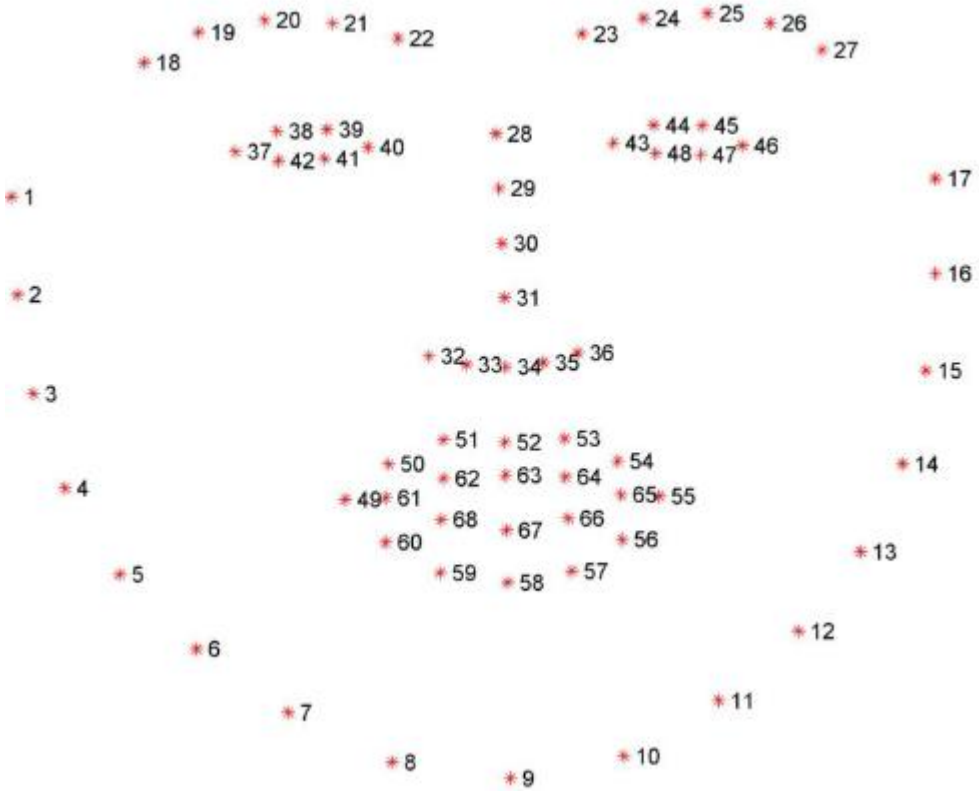

Рисунок 1 – Графическое представление 68 координат лицевых ориентиров

Поняв примерный принцип работы алгоритма, обнаруживающего ориентиры на лице человека, используя готовый обученный заранее детектор ключевых частей лица из библиотеки dlib, можем обнаруживать человеческие лица на изображениях при помощи скрипта на языке Python.

2 Результат работы

Выполним скрипт, который обнаруживает лица на изображениях и маркирует их (рисунок 2).

Контур лица выделяется зеленой рамкой, а части лица по отдельности помечаются красными точками.

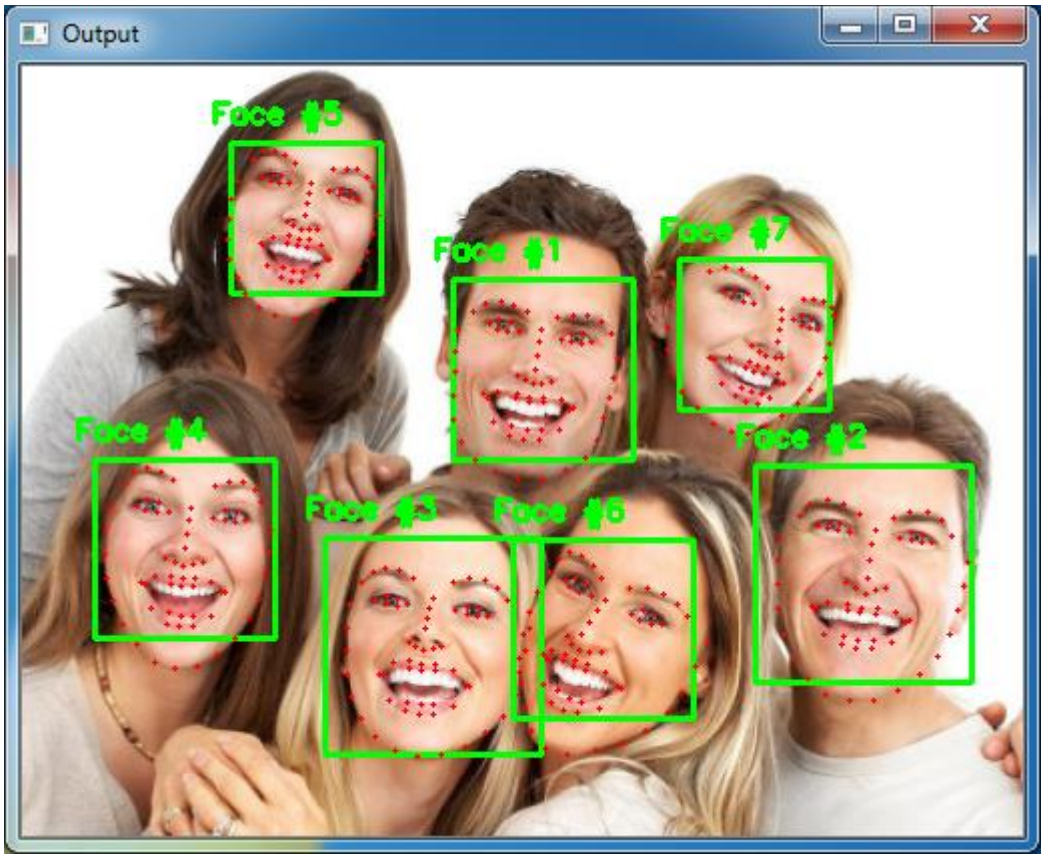

Рисунок 2 – Результат работы скрипта,
обнаруживающего лица на изображении

Как можно увидеть, что губы отмечены у людей, которые открыли рот. Также обнаружено лицо №6, повернутое вбок.

На рисунках 3-8 приводится результат работы скрипта, маркирующего различные части на лице по отдельности.

Для этого в скрипте явно указаны координаты точек, принадлежащих конкретным частям на лицах (листинг 1).

Листинг 1 – Координаты лицевых ориентиров в библиотеке face_utils

```
FACIAL_LANDMARKS_IDXS = OrderedDict([
("mouth", (48, 68)),
("right_eyebrow", (17, 22)),
("left_eyebrow", (22, 27)),
("right_eye", (36, 42)),
("left_eye", (42, 48)),
("nose", (27, 35)),
("jaw", (0, 17))
])
```

Рисунок 3 – Точки, описывающие рот

Рисунок 4 – Точки, описывающие губы

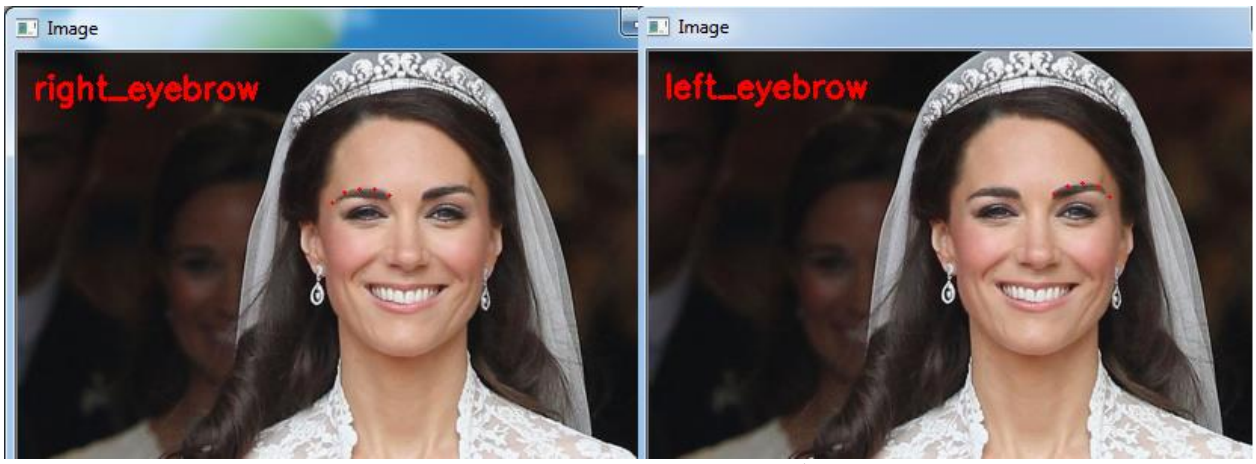

Рисунок 5 – Точки, описывающие брови

Рисунок 6 – Точки, описывающие глаза

Рисунок 7 – Точки, описывающие нос

Рисунок 8 – Точки, описывающие челюсть

На рисунке 9 показан результат работы скрипта, распознающего лицо на видео потоке.

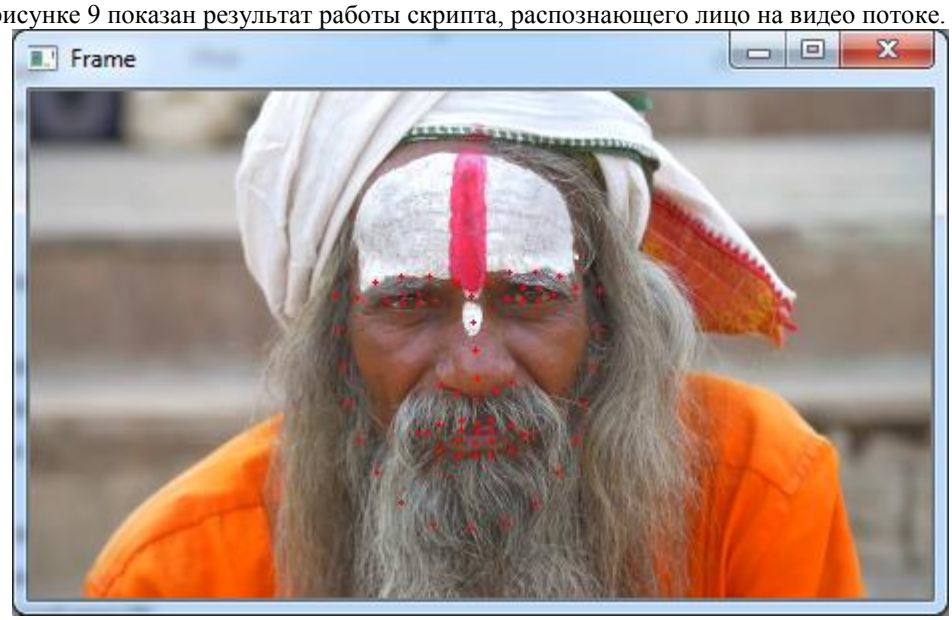

Рисунок 9 – Результат распознавания лица на видео

Исходный код алгоритма распознавания лиц

Исходный код facial-landmarks на языке Python

```
from imutils import face_utils
import numpy as np
import argparse
import imutils
import dlib
import cv2
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
	help="path to facial landmark predictor")
ap.add_argument("-i", "--image", required=True,
	help="path to input image")
args = vars(ap.parse_args())
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
image = cv2.imread(args["image"])
image = imutils.resize(image, width=500)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
rects = detector(gray, 1)
for (i, rect) in enumerate(rects):
	shape = predictor(gray, rect)
	shape = face_utils.shape_to_np(shape)
	 (x, y, w, h) = face_utils.rect_to_bb(rect)
	cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
	cv2.putText(image, "Face #{}".format(i + 1), (x - 10, y - 10),
		cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
	for (x, y) in shape:
		cv2.circle(image, (x, y), 1, (0, 0, 255), -1)
cv2.imshow("Output", image)
cv2.waitKey(0)
```

Исходный код detect-face-parts на языке Python

```
from imutils import face_utils
import numpy as np
import argparse
import imutils
import dlib
import cv2
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
	help="path to facial landmark predictor")
ap.add_argument("-i", "--image", required=True,
	help="path to input image")
args = vars(ap.parse_args())
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
image = cv2.imread(args["image"])
image = imutils.resize(image, width=500)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
rects = detector(gray, 1)
for (i, rect) in enumerate(rects):
	shape = predictor(gray, rect)
	shape = face_utils.shape_to_np(shape)
	for (name, (i, j)) in face_utils.FACIAL_LANDMARKS_IDXS.items():
		clone = image.copy()
		cv2.putText(clone, name, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
			0.7, (0, 0, 255), 2)
		for (x, y) in shape[i:j]:
			cv2.circle(clone, (x, y), 1, (0, 0, 255), -1)
		(x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
		roi = image[y:y + h, x:x + w]
		roi = imutils.resize(roi, width=250, inter=cv2.INTER_CUBIC)
		cv2.imshow("ROI", roi)
		cv2.imshow("Image", clone)
		cv2.waitKey(0)
	output = face_utils.visualize_facial_landmarks(image, shape)
	cv2.imshow("Image", output)
	cv2.waitKey(0)
```

Исходный код real-time-facial-landmarks на языке Python

```
from imutils.video import VideoStream
from imutils import face_utils
import datetime
import argparse
import imutils
import time
import dlib
import cv2
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
	help="path to facial landmark predictor")
ap.add_argument("-v", "--video", required=True,
	help="path to input video file")
args = vars(ap.parse_args())
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
vs = cv2.VideoCapture(args["video"])
while True:
	(grabbed, frame) = vs.read()
	if not grabbed:
		break
	frame = imutils.resize(frame, width=450)
	gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
	rects = detector(gray, 0)
	for rect in rects:
		shape = predictor(gray, rect)
		shape = face_utils.shape_to_np(shape)
		for (x, y) in shape:
			cv2.circle(frame, (x, y), 1, (0, 0, 255), -1)
	cv2.imshow("Frame", frame)
	key = cv2.waitKey(1) & 0xFF
	if key == ord("q"):
		break
vs.release()
cv2.destroyAllWindows()
```