СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“

Факултет по математика и информатика

Катедра „Компютърна информатика“

Дисциплина: СОЗ (3ти курс ИС, зимен семестър 2020/2021)

ЗАДАНИЕ ЗА ДОМАШНА РАБОТА №3

Зорница Николаева Димитрова, фак. №71843

Реализация:

Използвани данни – data.csv

1. Individual.java

```java
public class Individual {
    private final int x;
    private final int y;
    private final int classification;

    public Individual(int x, int y, int classification) {
        this.x = x <= 0 ? 1000 : x;
        this.y = y < 0 ? 2 : y;
        this.classification = classification;
    }

    public int getX() { return x; }

    public int getY() { return y; }

    public int getClassification() { return classification; }

    public double euclDistance(Individual rhs) {
        double xDist = x - rhs.x;
        double yDist = y - rhs.y;
        return Math.sqrt(xDist * xDist + yDist * yDist);
    }
}
```

Съдържа точките на коорд. система.

Трите променливи са константни, защото след като конструктора ги сетне, не ги променяме повече.

В този клас се смята и евклидовото разстояние.

2. KNN.java

```
public class KNN {

    private static final int k = 3;
    private static final ArrayList<Individual> individuals = new ArrayList<>();
    private static final Map<String, Integer> map = new HashMap<>() {{
        put("Rarely", 0);
        put("Sometimes", 1);
        put("Often", 2);
        put("Very often", 3);
    }};
```

k – броя индивиди, които гледам, при изчисляване на вота на мажоритарност.

individuals – задава контейнер за всички вече класифицирани индивиди.

map – задава изображение от string към int(на всяка честота съпоставям число).

2.1. read()

```
private static void read() {
    try {
        BufferedReader fin;

        String file = "data.csv";
        fin = new BufferedReader(new FileReader(file));
        fin.readLine();

        String line = "";
        while ((line = fin.readLine()) != null) {
            String[] currentLine = line.split( regex: ";");


            if (currentLine.length > 0) {

                individuals.add(new Individual(
                        Integer.parseInt(currentLine[0]),
                        map.get(currentLine[1].strip()),
                        Integer.parseInt(currentLine[2])
                ));
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Пропускам първия ред(хедърите) от файла, защото не ни трябва и в стринг записвам другите стойности, като от тях правя индивиди и ги добавям в individuals.

Разпределям вече класифицираните индивиди върху Декартовата коорд. система.

2.2. predict()

```
private static int predict(int amount, String frequency) {

    Individual toClassify = new Individual(amount, map.get(frequency.strip()), classification: 0);

    PriorityQueue<Individual> allDistances = new PriorityQueue<>((i1, i2) -> {
        double distanceI1 = toClassify.euclDistance(i1);
        double distanceI2 = toClassify.euclDistance(i2);

        if (distanceI1 < distanceI2) {
            return -1;
        } else if (distanceI1 == distanceI2) {
            return 0;
        }
        return 1;
    });

    allDistances.addAll(individuals);

    ArrayList<Integer> classifications = new ArrayList<>();


    for (int i = 0; i < k; i++) {
        classifications.add(Objects.requireNonNull(allDistances.poll()).getClassification());
    }
```

Създавам инидивид, който трябва да класифицирам.

В приоритетната опашка се съдържат всички индивиди, започвайки от най-близкия.

Във цикъла добавям класификациите на първите к най-близки индивиди(вадя индивиди от опашката и взимам класификациите им).

```
    int counterZero = 0;
    int counterOne = 0;

    for (int i : classifications) {
        switch (i) {
            case 0:
                ++counterZero;
                break;
            case 1:
                ++counterOne;
                break;
        }
    }

    return counterZero > counterOne ? 0 : 1;
}
```

Минавам през класификациите и броя нулите и единиците. Ако нулите са повече връщам нула, иначе едно.

2.3. Test

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

    System.out.println(predict( amount: 859,  frequency: "Rarely") == 0); // 0 (725, 925, 952)
    System.out.println(predict( amount: 550,  frequency: "Often") == 0); // 0
    System.out.println(predict( amount: 1254,  frequency: "Rarely") == 1); // 1
    System.out.println(predict( amount: 525,  frequency: "Sometimes") == 0); // 0
    System.out.println(predict( amount: 724,  frequency: "Very often") == 0); // 0
    System.out.println(predict( amount: 1477,  frequency: "Rarely") == 1); // 1
    System.out.println(predict( amount: 975,  frequency: "Rarely") == 0); // 0
    System.out.println(predict( amount: 1148,  frequency: "Rarely") == 0); // 0 (1213, 1213, 1215)
    System.out.println(predict( amount: 1315,  frequency: "Sometimes") == 1); // 1
    System.out.println(predict( amount: 869,  frequency: "Very often") == 0); // 0
    System.out.println(predict( amount: 1400,  frequency: "Very often") == 1); // 1
    System.out.println(predict( amount: 1200,  frequency: "Often") == 0); // 0 (1213, 1213, 1215)
    System.out.println(predict( amount: 1000,  frequency: "Often") == 0); // 0 (978, 982, 1071)
    System.out.println(predict( amount: 960,  frequency: "Sometimes") == 1); // 1 (952, 959, 978)
}
```

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.3\lib\idea_rt.jar=25880:C:\Prog
true
true
true
true
true
true
true
true
true
true
true
true
true
true
```