НАДІРЯН Г.О.

КІТ-М120Д

ПОБУДОВА РЕГРЕСІЙНИХ ДІАГНОСТИЧНИХ МОДЕЛЕЙ

Мета роботи: вивчення принципів побудови регресійних діагностичних моделей на підставі таблиці експериментальних даних

Код програми

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

public class Main {
    static  double C[] = new double[4];
    static double BB[][]=new double[4][4];

    static double[] factorF=new double[137];
    static double[] responsK=new double[137];
    static final int d1_size=33;
    static final int d2_size=41;
    static final int d3_size=33;
    static final int d4_size=30;
    static double mas1[][],mas2[][],mas3[][],mas4[][];


    static final int N = 137;
    static final int m = 8;
    static final int m2= 10;


    static double[][] ted2 = new double[N][m2];

    static ArrayList<String> number_of_leukocytes = new ArrayList<>();
    static ArrayList<String> number_of_lymphocytes = new ArrayList<>();
    static ArrayList<String> T_lymphocytes = new ArrayList<>();
    static ArrayList<String> T_helpers = new ArrayList<>();
    static ArrayList<String> T_suppressors_reduce = new ArrayList<>();
    static ArrayList<String> age = new ArrayList<>();
    static ArrayList<String> theophylline_sensiti = new ArrayList<>();
    static ArrayList<String> theophylline_resista = new ArrayList<>();
    static ArrayList<String> B_lymphocytes = new ArrayList<>();


    static void readColumn_all() throws IOException {
        FileReader fr = new FileReader("lab1_v2_all.txt");
        BufferedReader reader = new BufferedReader(fr);
        String line2;
        while ((line2 = reader.readLine()) != null) {
            String[] splited = line2.split("\\s");
            number_of_leukocytes.add(splited[0]);
            number_of_lymphocytes.add(splited[1]);
            T_lymphocytes.add(splited[2]);
```

```
            T_helpers.add(splited[3]);
            T_suppressors_reduce.add(splited[4]);
            theophylline_sensiti.add(splited[5]);
            theophylline_resista.add(splited[6]);
            B_lymphocytes.add(splited[7]);
            age.add(splited[8]);
        }

    }
    static void readTed_all() throws IOException {
        Scanner sc = new Scanner(new BufferedReader(new
FileReader("lab1_v2_all.txt")));
        while (sc.hasNextLine()) {
            for (int i = 0; i < N; i++) {
                String[] line = sc.nextLine().trim().split(" ");
                for (int j = 0; j < m2; j++) {
                    ted2[i][j] = Double.parseDouble(line[j]);
                }

            }
            java.util.Arrays.sort(ted2, new java.util.Comparator<double[]>()
{
                public int compare(double[] a, double[] b) {
                    return Double.compare(a[9], b[9]);
                }
            });
            Arrays.sort(ted2, Comparator.comparingDouble(o -> o[9]));

            for (int i = 0; i < N; i++) {
                for (int j = 0; j < m2; j++) {
                    System.out.print(ted2[i][j] + " ");
                }
                System.out.println("\t");
            }
        }
    }

    public static double[] Gauss(double[][] L, double[] R) {
        int n = R.length;

        for (int p = 0; p < n; p++) {

            int max = p;
            for (int i = p + 1; i < n; i++) {
                if (Math.abs(L[i][p]) > Math.abs(L[max][p])) {
                    max = i;
                }
            }
            double[] temp = L[p]; L[p] = L[max]; L[max] = temp;
            double   t    = R[p]; R[p] = R[max]; R[max] = t;

            for (int i = p + 1; i < n; i++) {
                double alpha = L[i][p] / L[p][p];
                R[i] -= alpha * R[p];
                for (int j = p; j < n; j++) {
                    L[i][j] -= alpha * L[p][j];
                }
            }
        }

        double[] x = new double[n];
        for (int i = n - 1; i >= 0; i--) {
            double sum = 0.0;
            for (int j = i + 1; j < n; j++) {
```

```
                sum += L[i][j] * x[j];
            }
            x[i] = (R[i] - sum) / L[i][i];
        }
        System.out.println("Параметры регрессионной модели:");
        for (int i = 0; i < x.length; i++) {
            System.out.println(x[i]);
        }
        System.out.print("\nY(X) =
("+x[0]+")+("+x[1]+"X)"+"+("+x[2]+"X^2)"+"+("+x[3]+"X^3)\n");
        return x;
    }
    static void calculateXY(ArrayList<String> factor, ArrayList<String>
response){
        for (int i = 0; i < N; i++) {
            factorF[i] = Double.parseDouble(factor.get(i));
            responsK[i]=Double.parseDouble(response.get(i));
        }
        double s1,s2,s3,s4,s5,s6,sy = 0;
        s1 = 0;s2 = 0;s3 = 0;s4 = 0;s5 = 0; s6 = 0;
        for (int i = 0; i <N ; i++) {
            s1 += factorF[i];
            s2 += Math.pow(factorF[i],2);
            s3 += Math.pow(factorF[i],3);
            s4 += Math.pow(factorF[i],4);
            s5 += Math.pow(factorF[i],5);
            s6 += Math.pow(factorF[i],6);
            sy +=responsK[i];
        }

        BB[0][0]=N;
        BB[0][1]=s1;
        BB[0][2]=s2;
        BB[0][3]=s3;

        BB[1][3]=s4;
        BB[2][3]=s5;
        BB[3][3]=s6;


        BB[1][0]=s1;
        BB[2][0]=s2;
        BB[3][0]=s3;
        BB[3][1]=s4;
        BB[3][2]=s5;

        BB[1][1]=s2;
        BB[1][2]=s3;
        BB[2][2]=s4;
        BB[2][1]=s3;
        C[0]=sy;
        for (int i = 0; i <N; i++) {
            C[1]=C[1]+responsK[i]*factorF[i];
            C[2]=C[2]+responsK[i]*factorF[i]*factorF[i];
            C[3]=C[3]+responsK[i]*factorF[i]*factorF[i]*factorF[i];
        }
        System.out.println("\t\t\t\tЛевая часть");
        for (int i = 0; i <4 ; i++) {
            for (int j = 0; j <4 ; j++) {
                System.out.printf("%.1f     ",BB[i][j]);

            }

            System.out.println();
```

```
        }
        System.out.println();
        System.out.println("Правая часть");
        for (int i = 0; i <4 ; i++) {
            System.out.printf("%.1f \n",C[i]);
        }
        System.out.println();

    }

    static void calculateXY(ArrayList<String> factor, ArrayList<String>
response,int N,int i){
        for ( i = 0; i < 137; i++) {
            factorF[i] = Double.parseDouble(factor.get(i));
            responsK[i]=Double.parseDouble(response.get(i));
        }
        double s1,s2,s3,s4,s5,s6,sy = 0;
        s1 = 0;s2 = 0;s3 = 0;s4 = 0;s5 = 0; s6 = 0;
        for ( i = 0; i <N ; i++) {
            s1 += factorF[i];
            s2 += Math.pow(factorF[i],2);
            s3 += Math.pow(factorF[i],3);
            s4 += Math.pow(factorF[i],4);
            s5 += Math.pow(factorF[i],5);
            s6 += Math.pow(factorF[i],6);
            sy +=responsK[i];
        }

     /*   System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);
        System.out.println(s4);
        System.out.println(s5);
        System.out.println(s6);
        System.out.println(sy);*/
        BB[0][0]=N;
        BB[0][1]=s1;
        BB[0][2]=s2;
        BB[0][3]=s3;

        BB[1][3]=s4;
        BB[2][3]=s5;
        BB[3][3]=s6;


        BB[1][0]=s1;
        BB[2][0]=s2;
        BB[3][0]=s3;
        BB[3][1]=s4;
        BB[3][2]=s5;

        BB[1][1]=s2;
        BB[1][2]=s3;
        BB[2][2]=s4;
        BB[2][1]=s3;
        C[0]=sy;
        for ( i = 0; i <N; i++) {
            C[1]=C[1]+responsK[i]*factorF[i];
            C[2]=C[2]+responsK[i]*factorF[i]*factorF[i];
            C[3]=C[3]+responsK[i]*factorF[i]*factorF[i]*factorF[i];
        }
        System.out.println("\t\t\t\tЛевая часть");
        for ( i = 0; i <4 ; i++) {
            for (int j = 0; j <4 ; j++) {
```

```
                System.out.printf("%.1f     ",BB[i][j]);

            }

            System.out.println();
        }
        System.out.println();
        System.out.println("Правая часть");
        for ( i = 0; i <4 ; i++) {
            System.out.printf("%.1f \n",C[i]);
        }
        System.out.println();

    }



    static void sigma(double D[]){
        double regr,delta,sigma;
        double countP = 4;
        delta=0;
        for (int i = 0; i <N; i++) {
            regr =
D[0]+D[1]*factorF[i]+D[2]*factorF[i]*factorF[i]+D[3]*factorF[i]*factorF[i]*fa
ctorF[i];
            delta=delta+Math.pow(responsK[i]-regr,2);
        }

        sigma=delta/(N-countP-1);
        System.out.println("delta: "+delta);
        System.out.println("sigma: "+sigma);
    }
    static void sigma(double D[],int N,int i){
        double regr,delta,sigma;
        double countP = 4;
        delta=0;
        for ( i = 0; i <N; i++) {
            regr =
D[0]+D[1]*factorF[i]+D[2]*factorF[i]*factorF[i]+D[3]*factorF[i]*factorF[i]*fa
ctorF[i];
            delta=delta+Math.pow(responsK[i]-regr,2);
        }

        sigma=delta/(N-countP-1);
        System.out.println("delta: "+delta);
        System.out.println("sigma: "+sigma);
    }
    public static void main(String[] rgs) throws IOException {
        System.out.println("\t\t\t\t\t\tTED");
        readColumn_all();
        readTed_all();
        System.out.println();
        mas1 = new double[N][m2];
        mas2 = new double[N][m2];
        mas3 = new double[N][m2];
        mas4 = new double[N][m2];
        SepareateTable();

    }

    private static void SepareateTable() {

       for (int i = 0; i < 33; i++) {
            for (int j = 0; j < m2; j++) {
```

```
                mas1[i][j]=ted2[i][j];
                System.out.print(ted2[i][j] + " ");
            }
            System.out.println("\t");
        }
System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        for (int i = 33; i < 74; i++) {
            for (int j = 0; j < m2; j++) {
                mas2[i][j]=ted2[i][j];

                System.out.print(ted2[i][j] + " ");
            }
            System.out.println("\t");
        }
System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        for (int i = 74; i < 107; i++) {
            for (int j = 0; j < m2; j++) {
                mas3[i][j]=ted2[i][j];

                System.out.print(ted2[i][j] + " ");
            }
            System.out.println("\t");
        }

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        for (int i = 107; i < N; i++) {
            for (int j = 0; j < m2; j++) {
                mas4[i][j]=ted2[i][j];

                System.out.print(ted2[i][j] + " ");
            }
            System.out.println("\t");
        }nullArr();
        calculateXY(age,number_of_leukocytes,d1_size,0);
        double eq1[]=Gauss(BB,C);
        sigma(eq1,d1_size,0);
        nullArr();
        calculateXY(age,number_of_lymphocytes,d2_size,33);
         double eq2[]=Gauss(BB,C);
        sigma(eq2,d2_size,33);
        nullArr();
        calculateXY(age,number_of_lymphocytes,d3_size,74);
         double eq3[]=Gauss(BB,C);
        sigma(eq3,d3_size,74);
        nullArr();
        calculateXY(age,number_of_lymphocytes,d4_size,107);
         double eq4[]=Gauss(BB,C);
        sigma(eq4,d4_size,107);
        nullArr();
       calculateXY(age,number_of_lymphocytes,137,0);
        double dddd[]=Gauss(BB,C);
        sigma(dddd,137,0);
    }
   static void nullArr(){
        for (int i = 0; i <4 ; i++) {
            for (int j = 0; j <4 ; j++) {
                BB[i][j]=0;
            }
            C[i]=0;
        }
    }
}
```

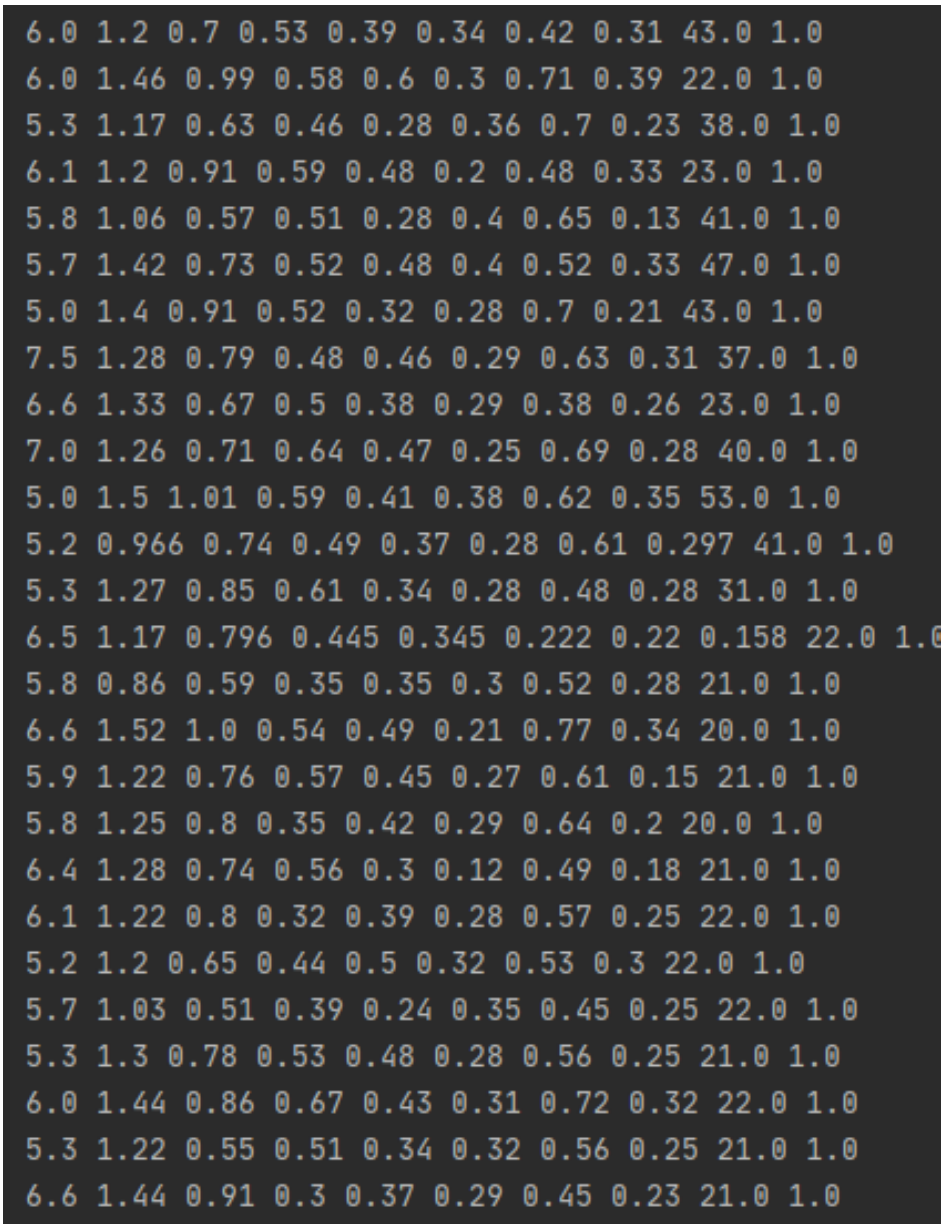

Рисунок 1 –відсортована TED

```
                Левая часть
33,0     1086,0     37438,0     1344300,0
1086,0     37438,0     1344300,0     49974370,0
37438,0     1344300,0     49974370,0     1912466436,0
1344300,0     49974370,0     1912466436,0     74971335778,0

Правая часть
258,1
8536,3
293413,5
10434752,5

Параметры регрессионной модели:
-5.492079972988253
0.34284902057595257
0.017400172182982274
-4.347415334548557E-4

Y(X) = (-5.492079972988253)+(0.34284902057595257X)+(0.017400172182982274X^2)+(-4.347415334548557E-4X^3)
delta: 358.8686289461855
sigma: 12.816736748078053
                Левая часть
41,0     1403,0     50369,0     1885805,0
1403,0     50369,0     1885805,0     73184885,0
50369,0     1885805,0     73184885,0     2927890973,0
1885805,0     73184885,0     2927890973,0     120197445869,0

Правая часть
97,2
3314,0
118548,4
4430103,0

Параметры регрессионной модели:
-19.22925159633525
1.9869090020037088
-0.058446719948914205
5.524796162622337E-4

Y(X) = (-19.22925159633525)+(1.9869090020037088X)+(-0.058446719948914205X^2)+(5.524796162622337E-4X^3)
delta: 21.231454125172874
sigma: 0.5897626145881354
                Левая часть
33,0     1086,0     37438,0     1344300,0
1086,0     37438,0     1344300,0     49974370,0
37438,0     1344300,0     49974370,0     1912466436,0
1344300,0     49974370,0     1912466436,0     74971335778,0

Правая часть
76,5
2484,2
84096,2
2957813,7

Параметры регрессионной модели:
-12.822685492773225
1.3210070887966159
-0.036016652026248666
3.075774316648735E-4

Y(X) = (-12.822685492773225)+(1.3210070887966159X)+(-0.036016652026248666X^2)+(3.075774316648735E-4X^3)
delta: 17.020267454375382
sigma: 0.6078666947991208
                Левая часть
30,0     990,0     34244,0     1234824,0
990,0     34244,0     1234824,0     46140752,0
34244,0     1234824,0     46140752,0     1776219960,0
1234824,0     46140752,0     1776219960,0     70080791144,0

Правая часть
69,6
2251,1
76024,2
2673563,8

Параметры регрессионной модели:
-12.582954515322294
1.3533968320493006
-0.038444383984299974
3.4317866909715835E-4

Y(X) = (-12.582954515322294)+(1.3533968320493006X)+(-0.038444383984299974X^2)+(3.4317866909715835E-4X^3)
delta: 14.66792655190975
sigma: 0.58671706207639
```

Рисунок 1 – Обчислення для кожної з отриманих таблиць. Ліва та права частина СЛАУ, параметри регресійної моделі, рівняння регресійній моделі, показники якості регресійній моделі.

Висновок

Вивчили принципи побудови регресійних діагностичних моделей на підставі таблиці експериментальних даних.