Надірян Г.О.

КІТ-М120Д

# 1. ПОПЕРЕДНЯ ОБРОБКА ЕКСПЕРИМЕНТАЛЬНИХ ДАНИХ

**Мета роботи:** вивчення принципів побудови таблиці експериментальних даних (ТЕД) типу «об'єкт-ознака» і методів їх попередньої обробки.

**Індивідуальне завдання.**

| | |
|---|---|
| Ім’я файла | ksmd_v2 |
| Міра зв'язку | Коефіцієнт рангової кореляції Кендалла. |
| Перетворення ознак | Виконати центрування і нормування ознак на «одиничний куб »за наступним виразом |
| Міра відстані | Відстань Чебишева |

```
6.0 1.2 0.7 0.53 0.39 0.34 0.42 0.31
6.0 1.46 0.99 0.58 0.6 0.3 0.71 0.39
7.9 2.45 1.29 1.35 0.71 0.78 0.86 0.39
12.2 3.02 2.23 1.42 0.91 0.61 2.1 0.23
5.3 1.17 0.63 0.46 0.28 0.36 0.7 0.23
6.1 1.2 0.91 0.59 0.48 0.2 0.48 0.33
5.8 1.06 0.57 0.51 0.28 0.4 0.65 0.13
4.6 2.85 1.27 1.06 0.46 0.02 1.07 0.33
6.7 2.45 1.62 1.07 0.55 0.78 1.32 0.55
```

Рисунок 1 – зчитування з файлу ТЕД

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1,000000 | 0,522692 | 0,575446 | 0,668054 | 0,426220 | 0,590241 | 0,403805 | -0,508153 |
| 0,522692 | 1,000000 | 0,777288 | 0,582859 | 0,546717 | 0,162897 | 0,645388 | 0,124217 |
| 0,575446 | 0,777288 | 1,000000 | 0,635787 | 0,649644 | 0,316216 | 0,701181 | 0,033181 |
| 0,668054 | 0,582859 | 0,635787 | 1,000000 | 0,484245 | 0,481900 | 0,534166 | -0,072062 |
| 0,426220 | 0,546717 | 0,649644 | 0,484245 | 1,000000 | 0,200706 | 0,597188 | 0,010791 |
| 0,590241 | 0,162897 | 0,316216 | 0,481900 | 0,200706 | 1,000000 | 0,217169 | -0,276990 |
| 0,403805 | 0,645388 | 0,701181 | 0,534166 | 0,597188 | 0,217169 | 1,000000 | 0,112572 |
| -0,508153 | 0,124217 | 0,033181 | -0,072062 | 0,010791 | -0,276990 | 0,112572 | 1,000000 |

Рисунок 2 – Матриця коефіцієнтів кореляції Пірсона

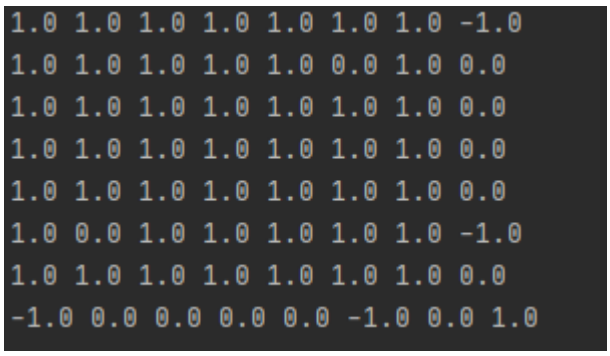


Рисунок 3 - Матриця значимості за критерієм Стьюдента

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1,000000 | 0,239802 | 0,311936 | 0,424431 | 0,233362 | 0,545728 | 0,212752 | -0,262344 |
| 0,249678 | 1,000000 | 0,620653 | 0,443753 | 0,411335 | 0,108630 | 0,478746 | 0,142980 |
| 0,313010 | 0,621726 | 1,000000 | 0,504079 | 0,481967 | 0,211249 | 0,540790 | 0,087591 |
| 0,426578 | 0,438171 | 0,506011 | 1,000000 | 0,405539 | 0,305711 | 0,421211 | 0,053027 |
| 0,232933 | 0,425719 | 0,479176 | 0,401889 | 1,000000 | 0,167024 | 0,468441 | 0,126020 |
| 0,543581 | 0,095105 | 0,204809 | 0,303349 | 0,151782 | 1,000000 | 0,147059 | -0,149850 |
| 0,213611 | 0,491198 | 0,540361 | 0,434951 | 0,480249 | 0,156934 | 1,000000 | 0,149420 |
| -0,253757 | 0,157149 | 0,104981 | 0,066337 | 0,131816 | -0,137827 | 0,164019 | 1,000000 |

Рисунок 4 - Коефіцієнт рангової кореляції Кендалла.

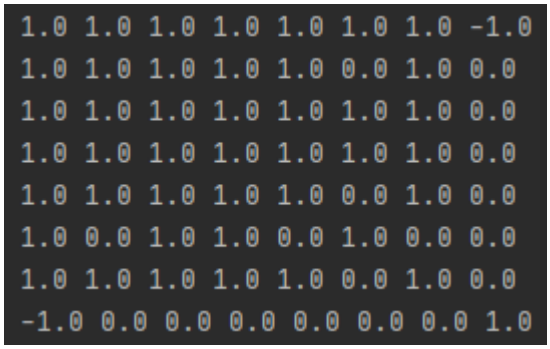


Рисунок 5 - Матриця значимості за критерієм Стьюдента

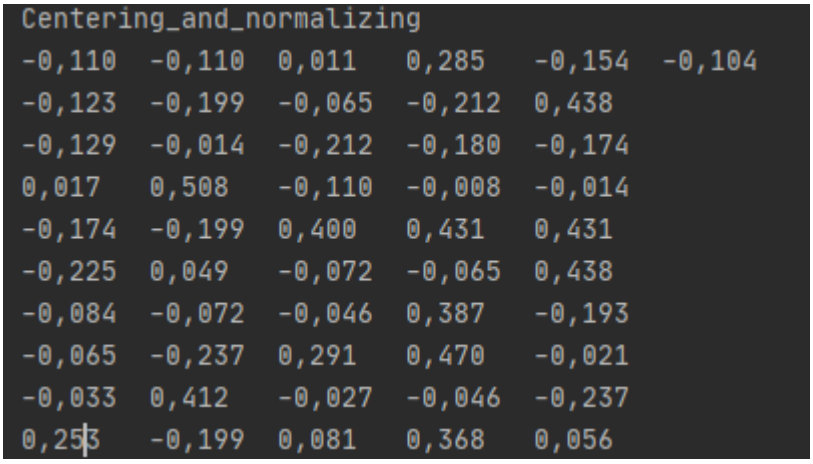


Рисунок 6 - центрування і нормування ознак на «одиничний куб »

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0,000000 | 9,180000 | 9,970000 | 10,780000 | 11,290000 | 11,590000 | 10,100000 | 11,970000 |
| 9,180000 | 0,000000 | 1,580000 | 1,790000 | 2,390000 | 2,830000 | 1,780000 | 2,790000 |
| 9,970000 | 1,580000 | 0,000000 | 0,810000 | 1,320000 | 1,620000 | 0,430000 | 2,000000 |
| 10,780000 | 1,790000 | 0,810000 | 0,000000 | 0,640000 | 1,040000 | 0,680000 | 1,190000 |
| 11,290000 | 2,390000 | 1,320000 | 0,640000 | 0,000000 | 0,440000 | 1,190000 | 0,680000 |
| 11,590000 | 2,830000 | 1,620000 | 1,040000 | 0,440000 | 0,000000 | 1,490000 | 0,390000 |
| 10,100000 | 1,780000 | 0,430000 | 0,680000 | 1,190000 | 1,490000 | 0,000000 | 1,870000 |
| 11,970000 | 2,790000 | 2,000000 | 1,190000 | 0,680000 | 0,390000 | 1,870000 | 0,000000 |

Рисунок 7 - Відстань Чебишева

## Фрагмент коду програми

Фрагмент коду програми

```
public class Main {


    static double avgDevition(ArrayList<String> x_ar) {
        double avg = 0;
        double x[] = new double[137];
        for (int i = 0; i < N; i++) {
            x[i] = Double.parseDouble(x_ar.get(i));
        }
        for (int i = 0; i < N; i++) {
            avg += x[i] / N;
        }
        System.out.println(avg);

        return avg;
    }

    static void readColumn() throws IOException {
        FileReader fr = new FileReader("lab1_v2.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]);
        }

    }

    static void ReadTed() throws IOException {
        Scanner sc = new Scanner(new BufferedReader(new
FileReader("lab1_v2.txt")));
        int rows = 137;
        int columns = 8;

        while (sc.hasNextLine()) {
```

```
            for (int i = 0; i < N; i++) {
                String[] line = sc.nextLine().trim().split(" ");
                for (int j = 0; j < m; j++) {
                    ted[i][j] = Double.parseDouble(line[j]);
                    System.out.print(ted[i][j] + " ");
                }
                System.out.println("\t");
            }

        }
    }

    static void Calc_m() {
        for (int i = 0; i < 137; i++) {
            for (int j = 0; j < m; j++) {
                mx += ted[i][j];

            }
        }
        mx = mx / N / m;
        System.out.print("mx = " + mx + "");
        System.out.print("Геометрический центр облака данных = " + mx + "");
    }

    static void Calc_s() {
        double s = 0;

        for (int i = 0; i < 137; i++) {
            for (int j = 0; j < m; j++) {
                s = s + Math.pow(ted[i][j] - mx, 2);
            }

        }
        System.out.println("s:" + s);
        s = s / Math.sqrt(s / (N * m - 1));
        // System.out.println();
        // System.out.println("s = "+s);
        System.out.println("Среднеквадратичное отклонение облака данных = " +
s);
    }

    static void Calc_R() {
        R = Math.abs(ted[0][1] - mx);
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < m; j++) {
                if (R < Math.abs(ted[i][j]) - mx) {
                    R = Math.abs(ted[i][j] - mx);
                }
            }
        }
        System.out.println(R);
    }

    static void Calc_mj() {
        for (int j = 1; j < m; j++) {
            mjx[j] = 0;
            for (int i = 0; i < N; i++) {
                mjx[j] = mjx[j] + ted[i][j];
            }
            mjx[j] = mjx[j] / N;
        }

    }
```

```
    static double Pirson(ArrayList<String> al_xk, ArrayList<String> al_xj) {
        double xk[] = new double[137];
        double xj[] = new double[137];
        double sk = 0;
        double sj = 0;
        double skj = 0;
        double sk2 = 0;
        double sj2 = 0;
        for (int i = 0; i < N; i++) {
            xk[i] = Double.parseDouble(al_xk.get(i));
            xj[i] = Double.parseDouble(al_xj.get(i));
        }
        for (int i = 0; i < N; i++) {
            sk = sk + xk[i];
            sj = sj + xj[i];
            skj = skj + xk[i] * xj[i];
            sk2 = sk2 + Math.pow(xk[i], 2);
            sj2 = sj2 + Math.pow(xj[i], 2);
        }
        //  System.out.println();
        tau = (N * skj - sk * sj) / Math.sqrt((N * sk2 - Math.pow(sk, 2)) *
(N * sj2 - Math.pow(sj, 2)));
        System.out.printf("%4f\t", tau);

        //  System.out.println(tau+" ");
        return tau;
    }

    static void MySortList(double x[], double y[], int N) {
        double a;
        double f = 1.0;

        for (int j = 0; j < N - 1; j++) {
            for (int i = 0; i < N - 1; i++) {
                if (x[i] > x[i + 1]) {
                    a = x[i];
                    x[i] = x[i + 1];
                    x[i + 1] = a;
                    a = y[i];
                    y[i] = y[i + 1];
                    y[i + 1] = a;
                    f = 0;
                }
            }
        }
        for (int i = 0; i < N; i++) {
            xk_kendall[i] = x[i];
        }
        for (int i = 0; i < N; i++) {
            xj_kendall[i] = y[i];

        }
    }

    static void MySortList_rangk_k(double x[], double y[], int N) {

        double a;
        double f = 1.0;

        for (int j = 0; j < N - 1; j++) {
            for (int i = 0; i < N - 1; i++) {
                if (x[i] > x[i + 1]) {
                    a = x[i];
                    x[i] = x[i + 1];
```

```
                    x[i + 1] = a;
                    a = y[i];
                    y[i] = y[i + 1];
                    y[i + 1] = a;
                    f = 0;
                }
            }
        }

        for (int i = 0; i < N; i++) {
            xk_kendall[i] = x[i];
        }
        for (int i = 0; i < N; i++) {
            rangk_kendall[i] = y[i];
        }
    }

    public static double[] Calc_rang(double x[], int N) {
        double rang[] = new double[137];
        int kol = 0;
        for (int i = 0; i < N; i++) {
            if (x[i] == x[i + 1]) {
                kol++;
            } else {
                for (int j = i - kol; j <= i; j++) {
                    rang[j] = i - kol / 2;
                }
                kol = 0;
            }
        }
        for (int j = N - kol; j < N; j++) {
            rang[j] = N - kol / 2;
        }
        return rang;
    }

    private static double Calc_V(double[] rang, int N) {
        int kol = 0;
        double V = 0;
        double ti = 1;
        for (int i = 0; i < N - 1; i++) {
            if (rang[i] == rang[i + 1]) {
                kol++;
            } else {
                V = V + Math.pow(ti, 2) - ti;
                ti = 1;
            }

        }
        V = (V + Math.pow(ti, 2) - ti) / 2;
        return V;
    }

    private static void Calc_PQ(double rangk[]) {
        P = 0;
        Q = 0;
        for (int i = 0; i < N; i++) {
            for (int k = i; k < N; k++) {
                if (rangk[k] >= rangk[i]) {
                    P++;
                } else if (rangk[k] < rangk[i]) {
                    Q++;
                }
            }
```

```
        }


    }

    static double Kendall(ArrayList<String> al_xk, ArrayList<String> al_xj) {

        for (int i = 0; i < N; i++) {
            xk_kendall[i] = Double.parseDouble(al_xk.get(i));
            xj_kendall[i] = Double.parseDouble(al_xj.get(i));
        }
        MySortList(xk_kendall, xj_kendall, N);
        double rangk[] = Calc_rang(xk_kendall, N);

        double Vk = Calc_V(rangk, N);
        MySortList_rangk_k(xj_kendall, rangk, N);

        double rangj[] = Calc_rang(xj_kendall, N);
        double Vj = Calc_V(rangj, N);
        Calc_PQ(rangk_kendall);
        double a = (Math.pow(N, 2) - N) / 2;

        double r = (P - Q) / (Math.sqrt((a - Vk) * (a - Vj)));
        System.out.printf("%4f\t", r);
        return r;
    }


    static void callKandall() {

        arrayListsKandall[0] = number_of_leukocytes;
        arrayListsKandall[1] = number_of_lymphocytes;
        arrayListsKandall[2] = T_lymphocytes;
        arrayListsKandall[3] = T_helpers;
        arrayListsKandall[4] = T_suppressors_reduce;
        arrayListsKandall[5] = theophylline_sensiti;
        arrayListsKandall[6] = theophylline_resista;
        arrayListsKandall[7] = B_lymphocytes;
        int count=0;

        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                arrayKendall[count]=Kendall(arrayListsKandall[i],
arrayListsKandall[j]);
                count++;
            }

            System.out.println("\t");

        }

    static void callPirson() {
        arrayListsPirson[0] = number_of_leukocytes;
        arrayListsPirson[1] = number_of_lymphocytes;
        arrayListsPirson[2] = T_lymphocytes;
        arrayListsPirson[3] = T_helpers;
        arrayListsPirson[4] = T_suppressors_reduce;
        arrayListsPirson[5] = theophylline_sensiti;
        arrayListsPirson[6] = theophylline_resista;
        arrayListsPirson[7] = B_lymphocytes;
        int count=0;
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
```

```
            arrayPirson[count]=Pirson(arrayListsPirson[i],
arrayListsPirson[j]);
            count++;
            }
            System.out.println("\t");
        }
    private static double Calc_T(double[] rang, int N) {
        double T = 0.0;
        double ti = 1;
        int kol = 0;
        for (int i = 0; i < N - 1; i++) {
            if (rang[i] == rang[i + 1]) {
                kol++;
            } else {
                T = T + Math.pow(ti, 3) - ti;
                ti = 1;
            }
        }
        T = (T + Math.pow(ti, 3)) / 12;
        return T;
    }

    static double[][] TablS_D(ArrayList<String> al_xk, ArrayList<String>
al_xj) {
        double xk_[] = new double[N];
        double xj_[] = new double[N];
        for (int i = 0; i < N; i++) {
            xk_[i] = Double.parseDouble(al_xk.get(i));
            xj_[i] = Double.parseDouble(al_xj.get(i));
        }
        double tabl[][] = new double[N][N];
        tabl[0][0] = 0;
        tabl[0][1] = 0;
        tabl[1][0] = 0;
        tabl[1][1] = 0;
        double j, k = 0;
        for (int i = 0; i < N; i++) {
            j = xj_[i] + 1;
            k = xk_[i] + 1;
            tabl[(int) j][(int) k]++;
            System.out.println("j:" + j + " k:" + k);
        }
        return tabl;
    }

    static double[][] TablS_K(ArrayList<String> al_xk, ArrayList<String>
al_xj) {
        double xk[] = new double[N];
        double xj[] = new double[N];
        for (int i = 0; i < N; i++) {
            xk[i] = Double.parseDouble(al_xk.get(i));
            xj[i] = Double.parseDouble(al_xj.get(i));
        }
        double tabl[][] = new double[N][N];
        tabl[0][0] = 0;
        tabl[0][1] = 0;
        tabl[1][0] = 0;
        tabl[1][1] = 0;
        double mj = 0;
        double mk = 0;
        for (int i = 0; i < N; i++) {
            mj = mj + xj[i];
            mk = mk + xk[i];
        }
```

```
        mj = mj / N;
        mk = mk / N;
        double j = 0;
        double k = 0;
        for (int i = 0; i < N; i++) {
            if (xj[i] < mj) {
                j = 1;
            } else {
                j = 2;
            }
            if (xk[i] < mk) {
                k = 1;
            } else {
                k = 2;
            }
            tabl[(int) j][(int) k]++;
            System.out.println("j:" + j + " k:" + k);

        }
        return tabl;
    }

    static void maxmink(ArrayList<String> al_xk, int N, double max, double
min) {
        double x[] = new double[N];
        for (int i = 0; i < N; i++) {
            x[i] = Double.parseDouble(al_xk.get(i));
        }
        max = x[1];
        min = x[1];
        for (int i = 0; i < N; i++) {
            if (x[i] < min) {
                min = x[i];
            } else if (x[i] > max) {
                max = x[i];
            }
            maxk = max;
            mink = min;

        }
    }

    static void maxminj(ArrayList<String> al_xk, int N, double max, double
min) {
        double x[] = new double[N];
        for (int i = 0; i < N; i++) {
            x[i] = Double.parseDouble(al_xk.get(i));
        }
        max = x[1];
        min = x[1];
        for (int i = 0; i < N; i++) {
            if (x[i] < min) {
                min = x[i];
            } else if (x[i] > max) {
                max = x[i];
            }
            maxj = max;
            minj = min;
        }
    }

    static double[][] TablS(ArrayList<String> al_xj, ArrayList<String> al_xk,
int N) {
        double xk[] = new double[N];
```

```
        double xj[] = new double[N];

        maxminj(al_xj, N, maxj, minj);
        maxmink(al_xk, N, maxk, mink);
        for (int i = 0; i < N; i++) {
            xk[i] = Double.parseDouble(al_xk.get(i));
            xj[i] = Double.parseDouble(al_xj.get(i));
        }
        double j = 0;
        double k = 0;
        int L = 137;
        int P = 8;
        double tabl[][] = new double[N][N];
        for (int i = 0; i < L; i++) {
            for (j = 0; j < P; j++) {
                tabl[i][(int) j] = 0;
            }
        }
        int s = 0;
        double dj = (maxj - minj) / L;
        double dk = (maxk - mink) / P;
        for (int i = 0; i < N; i++) {
            for (s = 0; s < L; s++) {
                if ((minj + (s - 1) * dj < xj[i] && xj[i] >= minj + s * dj)
|| (xj[i] == minj) && s == 1) {
                    j = s;
                    for (s = 0; s < P; s++) {
                        if ((mink + (s - 1) * dk < xk[i] && xk[i] >= mink + s
* dk) || (xk[i] == mink && s == 1)) {
                            k = s;
                        }
                    }
                }
            }
           /* j++;
            k++;*/
            System.out.println(j + " " + k);
        }

        return tabl;
    }

    static double[][] TablSS(ArrayList<String> al_xj, ArrayList<String>
al_xk, int N) {
        double xk[] = new double[N];
        double xj[] = new double[N];
        for (int i = 0; i < N; i++) {
            xk[i] = Double.parseDouble(al_xk.get(i));
            xj[i] = Double.parseDouble(al_xj.get(i));
        }
        double j = 0;
        double k = 0;
        double tabl[][] = new double[N][m];
        for (int i = 0; i < N; i++) {
            for (j = 0; j < m; j++) {
                tabl[(int) j][(int) k] = 0;
            }
            for (int l = 0; l < N; l++) {
                j = (int) xj[i];
                k = (int) xk[i];
                tabl[(int) j][(int) k] += 1;
                System.out.println(tabl[(int) j][(int) k]);
            }
        }
```

```
        System.out.println("\ni j ");

        for (int i = 0; i < N; i++) {
            for (int l = 0; l < m; l++) {
                System.out.print(tabl[i][l] + " ");
            }
            System.out.print("\t");

        }

        return tabl;
    }

    static double Dch(ArrayList<String> al_xk, ArrayList<String> al_xj, int
m) {
        double a[] = new double[137];
        double b[] = new double[137];
        for (int i = 0; i < N; i++) {
            a[i] = Double.parseDouble(al_xk.get(i));
            b[i] = Double.parseDouble(al_xj.get(i));
        }
        double d = Math.abs(a[0] - b[0]);
        for (int k = 0; k < m; k++) {
            if (d < Math.abs(a[k] - b[k])) {
                d = Math.abs(a[k] - b[k]);
            }
            //System.out.print(d + "\t");

        }           System.out.printf("%4f\t", d);

        return d;
    }
    static void CallDch(){

     arrayListsDch[0]=number_of_leukocytes;
     arrayListsDch[1]=number_of_lymphocytes;
     arrayListsDch[2]=T_lymphocytes;
     arrayListsDch[3]=T_helpers;
     arrayListsDch[4]=T_suppressors_reduce;
     arrayListsDch[5]=theophylline_sensiti;
     arrayListsDch[6]=theophylline_resista;
     arrayListsDch[7]=B_lymphocytes;
        int count=0;
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                arrayDch[count]=Dch(arrayListsDch[i], arrayListsDch[j],m);
                count++;
            }
            System.out.println("\t");
        }

    }
    static void Centering_and_normalizing(ArrayList<String> al_xk) {
        double x[] = new double[137];
        double AVG = avgDevition(al_xk);
        Calc R();
        for (int i = 0; i < N; i++) {
            x[i] = Double.parseDouble(al_xk.get(i));
        }


        double[] standart = new double[N];
        System.out.println("Centering_and_normalizing");
        for (int i = 0; i < N; i++) {
```

```
            standart[i] = (x[i] - AVG) / R;
            System.out.printf("%.3f\t", standart[i]);
            if (i % 5 == 0 && i != 0) {
                System.out.println();
            }
        }


    }

    static void student_cof_Pir() {
        double t = 1.9777;
        double kr = 0;
        double C[][] = new double[8][8];
        double oneToDim[][] = new double[8][8];
        int c = 0;
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                oneToDim[i][j] = arrayPirson[c];
                c++;
            }

        }

        for (int j = 0; j < 8; j++) {
            for (int k = j; k < 8; k++) {
                if (j != k) {
                    kr = Math.abs(oneToDim[j][k]) * Math.sqrt(N - 2) /
Math.sqrt(1 - Math.pow(oneToDim[j][k], 2));
                    //System.out.println("\nkr:" + kr);
                }
                if (kr > t) {
                    C[j][k] = 1;
                    C[k][j] = 1;
                } else {
                    C[j][k] = 0;
                    C[k][j] = 0;

                }
                if (C[j][k] == 1 && oneToDim[j][k] < 0) {
                    C[j][k] = -1;
                    C[k][j] = -1;
                }
                if (j == k)
                    C[j][k] = 1;
            }
        }
        for (int i = 0; i <8 ; i++) {
            for (int j = 0; j <8 ; j++) {
                System.out.print(C[i][j]+" ");
            }
            System.out.println("\t");
        }
    }
   static void student_cof(double arrayNameAuth[]) {
       double t = 1.9777;
       double kr = 0;
       double C[][] = new double[8][8];
       double oneToDim[][] = new double[8][8];
       int c = 0;
       for (int i = 0; i < 8; i++) {
           for (int j = 0; j < 8; j++) {
               oneToDim[i][j] = arrayNameAuth[c];
               c++;
```

```
            }

        }

        for (int j = 0; j < 8; j++) {
            for (int k = j; k < 8; k++) {
                if (j != k) {
                    kr = Math.abs(oneToDim[j][k]) * Math.sqrt(N - 2) /
Math.sqrt(1 - Math.pow(oneToDim[j][k], 2));
                    //System.out.println("\nkr:" + kr);
                }
                if (kr > t) {
                    C[j][k] = 1;
                    C[k][j] = 1;
                } else {
                    C[j][k] = 0;
                    C[k][j] = 0;

                }
                if (C[j][k] == 1 && oneToDim[j][k] < 0) {
                    C[j][k] = -1;
                    C[k][j] = -1;
                }
                if (j == k)
                    C[j][k] = 1;
            }
        }
        for (int i = 0; i <8 ; i++) {
            for (int j = 0; j <8 ; j++) {
                System.out.print(C[i][j]+" ");
            }
            System.out.println("\t");
        }
    }


    public static void main(String[] rgs) throws IOException {
        ReadTed();
        readColumn();

        callPirson();
        System.out.println("\n");
        callKandall();
        student_cof(arrayPirson);
        System.out.println();
        student_cof(arrayKendall);
        CallDch();

    }
}
```

## Висновок

Вивчив принципи побудови таблиці експериментальних даних (ТЕД) типу «об'єкт-ознака» і методів їх попередньої обробки.