Лабораторна робота №2

Тема: Застосування кластерного аналізу в процесі визначення переліку стану об’єкта

Мета: закріпити навички проектування програм для кластерного аналізу опису станів досліджуваних об’єктів.

Хід роботи

Після ознайомлення із теорією кластерного аналізу, було вирішено написати просту веб-сторінку, яка буде містити в собі вхідну таблицю із заповненими даними та проводити обрахунки. В якості мов програмування були обрані **HTML**-розмітка та **JavaScript** логіка. Усі формули були детально розібрані та реалізовані за допомогою методів. Кожен із них був описаний та поміщений у файл calculator.js:

```
// This method calculates the sum of each column
//
// Returns:
// array of sums, e.g. [10, 20, 30] for 3 columns with values [0,5,5], [10,5,5], [10,10,10]
function calculateTheSumOfColumns(inputTable) {
    const columnsAmount = inputTable[0].length
    const allColumnsSums = new Array(columnsAmount).fill(0)

    for (let i = 0; i < inputTable.length; i++) {
        for (let j = 0; j < inputTable[i].length; j++) {
            allColumnsSums[j] += inputTable[i][j]
        }
    }

    return allColumnsSums;
}

// Divides the input table by the columns sums. Formula: Vij = Xij / sum (Xij) [1..n]; n – amount of columns. (formula 11)
//
// Returns:
// 2D array (matrix), Vij.
function divideOnSums(inputTable, columnsSums) {
    const dividedTable = []

    for (let i = 0; i < inputTable.length; i++) {
        const row = []

        for (let j = 0; j < inputTable[i].length; j++) {
            const currentTableValue = inputTable[i][j]
            const currentColumnSum = columnsSums[j]

            row.push(currentTableValue / currentColumnSum)
        }

        dividedTable.push(row)
    }

    return dividedTable;
}
```

Рисунок 1. Приклад методів у файлі calculator.js

Задля виконання функцій перетворення даних у html-розмітку було написано файл ui-utils.js, у якому було написано метод для створення розмітки таблиці із двовимірного масиву, а також метод створення розмітки результату обчислень:

```
// creates a table from given array.
//
// Params:
// title -- title of the table
// columnTitle -- title of the column, e.g. "Attribute", the columns will be "Attribute1", "Attribute2" etc.
// row -- title of the column, e.g. "P", the columns will be "P1", "P2" etc.
// array -- numeric array of table data
//
// Returns:
// HTML code of the table built on given parameters
window.utilsCreateTable = function createTable({ title, array, columnTitle = "Attribute", rowTitle = "P", addCustomRow = false, customRowTitle = "", addCustomColumn = false, custo
    const rowsAmount = array.length;
    if (rowsAmount < 1) return ""
    const columnsAmount = array[0].length;

    var tableHTML =
        "<table id=\"input_table\" class=\"table table-bordered\">\n" +
        "<thead class=\"thead-light\">\n" +
        "<tr><th colspan=\"100%\" style=\"text-align:center;\">" + title + "</th></tr>\n" +
        "<tr>\n<th></th>\n"

    // Add the columns titles to the table
    for (let i = 0; i < columnsAmount; i++) {
        if (addCustomColumn && i == columnsAmount - 1) {
            tableHTML += "<th>" + customColumnTitle + "</th>\n"
        } else {
            tableHTML += "<th>" + columnTitle + (i + 1) + "</th>\n"
        }
    }

    // Create the body
    tableHTML += "</tr>\n</thead>\n<tbody>\n"

    // Add the rows and columns
    for (let i = 0; i < rowsAmount; i++) {
        if (addCustomRow && i == rowsAmount - 1) {
            tableHTML += "<tr>\n<th>" + customRowTitle + "</th>\n"
        } else {
            tableHTML += "<tr>\n<th>" + rowTitle + (i + 1) + "</th>\n"
        }

        for (let j = 0; j < columnsAmount; j++) {
            tableHTML += "<td>" + Math.ceil(array[i][j] * 10000) / 10000 + "</td>\n"
        }

        tableHTML += "</tr>\n"
    }

    // Close the table
    tableHTML += "</tbody>\n</table>\n"
```

Рисунок 2. Метод для створення розмітки таблиці із двовимірного масиву

Для перевірки роботоздатності розробленого веб-застосунку, було виконано кластерний аналіз на прикладах, що були зазначені у теоричному документі до лабораторної роботи. Завдяки цьому можна покроково перевірити правильність виконаних розрахунків.

Кластерний аналіз

Вхідна матриця

	Attribute1	Attribute2	Attribute3	Attribute4
P1	30	0.6	2	5
P2	33	0.6	2.5	5
P3	50	1	2	15
P4	45	0.8	2	10
P5	20	0.2	1	5
P6	25	0.6	1	20

Створити кластери

Рисунок 3. Вхідна матриця тестових даних у розробленому застосунку

Результати обрахунків зображуються, коли користувач натискає на кнопку «Створити кластери». Спочатку виконується обрахунок сум усіх колонок та нормалізація матриці:

Обраховуємо суми колонок:

Вхідна матриця із сумами колонок

	Attribute1	Attribute2	Attribute3	Attribute4
P1	30	0.6	2	5
P2	33	0.6	2.5	5
P3	50	1	2	15
P4	45	0.8	2	10
P5	20	0.2	1	5
P6	25	0.6	1	20
Sum(Xij)	203	3.8	10.5	60

Рисунок 4. Обраховані суми колонок вхідної матриці

На рисунку 5 зображений процес нормалізації вхідної матриці:

Ділимо кожен елемент на суму колонки:

Вхідна матриця поділена на суми колонок				
	Attribute1	Attribute2	Attribute3	Attribute4
P1	0.1478	0.1579	0.1905	0.0834
P2	0.1626	0.1579	0.2381	0.0834
P3	0.2464	0.2632	0.1905	0.25
P4	0.2217	0.2106	0.1905	0.1667
P5	0.0986	0.0527	0.0953	0.0834
P6	0.1232	0.1579	0.0953	0.3334

Нормалізуємо матрицю:

Нормалізована матриця					
	Attribute1	Attribute2	Attribute3	Attribute4	Wij
P1	0.1478	0.1579	0.1905	0.0834	0.5795
P2	0.1626	0.1579	0.2381	0.0834	0.6419
P3	0.2464	0.2632	0.1905	0.25	0.95
P4	0.2217	0.2106	0.1905	0.1667	0.7894
P5	0.0986	0.0527	0.0953	0.0834	0.3298
P6	0.1232	0.1579	0.0953	0.3334	0.7097

Рисунок 5. Нормалізація вхідної матриці для ізотонічної розбивки

Наступним кроком була виконана ізотонічна розбивка на кластери, а її результати були виведені на сторінці. Результат ізотонічної розбивки можна побачити на рисунку 6.

Ізотонічна розбивка

Обраховуємо відстані та мінімальні значення рядків:

Матриця відстаней							
	P1	P2	P3	P4	P5	P6	Min Pi
P1	0	0.0624	0.3705	0.2099	0.2498	0.1302	0.0624
P2	0.0624	0	0.3081	0.1475	0.3122	0.0678	0.0624
P3	0.3705	0.3081	0	0.1606	0.6203	0.2404	0.1606
P4	0.2099	0.1475	0.1606	0	0.4597	0.0798	0.0798
P5	0.2498	0.3122	0.6203	0.4597	0	0.3799	0.2498
P6	0.1302	0.0678	0.2404	0.0798	0.3799	0	0.0678

Критична відстань R (= Max Min Pi) = 0.24976233687667448

Можна виділити наступні кластери:

Кластер 1: P5

Кластер 2: P1,P2,P6,P4,P3

Рисунок 6. Результат ізотонічної розбивки

Далі, за аналогією було виконано ізоморфну розбивку на кластери. Для цього, спочатку потрібно виконати відповідну нормалізацію:

Ізоморфна розбивка

Нормалізуємо матрицю:

Нормалізована матриця (ізоморфна)				
	Attribute1	Attribute2	Attribute3	Attribute4
P1	0.2551	0.2725	0.3287	0.1439
P2	0.2533	0.246	0.371	0.1299
P3	0.2593	0.2771	0.2006	0.2632
P4	0.2809	0.2668	0.2414	0.2112
P5	0.2989	0.1597	0.2889	0.2528
P6	0.1736	0.2226	0.1343	0.4698

Рисунок 7. Нормалізація матриці для ізоморфної розбивки

Результат ізоморфної розбивки можна побачити на рисунку 8:

Обраховуємо відстані та мінімальні значення рядків:

Матриця відстаней (ізоморфна)							
	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**Min Pi**
P1	0	0.0519	0.1753	0.1135	0.1677	0.3914	0.0519
P2	0.0519	0	0.2187	0.1569	0.1772	0.4225	0.0519
P3	0.1753	0.2187	0	0.0703	0.1525	0.2396	0.0703
P4	0.1135	0.1569	0.0703	0	0.1257	0.303	0.0703
P5	0.1677	0.1772	0.1525	0.1257	0	0.3011	0.1257
P6	0.3914	0.4225	0.2396	0.303	0.3011	0	0.2396

Критична відстань R (= Max Min Pi) = 0.2395558483017508

Можна виділити наступні кластери:

Кластер 1: P6

Кластер 2: P1,P2,P3,P4,P5

Рисунок 8. Результат ізоморфної розбивки

Проаналізувавши відповіді побудованої системи та порівнявши їх із обрахунками у теоретичному документі до лабораторної роботи, було зроблено висновок, що усі алгоритми працюють коректно та виконання кластеризації проходить без помилок. Далі було виконано тестування системи на основі заданого варіанту (варіант №2):

2

	X1	X2	X3	X4	X5
P1	15,5	0,2	0,1	105	4,1
P2	12,3	0,15	0,5	55	4,5
P3	11,2	0,18	0,2	75	4,2
P4	17,7	0,17	0,7	100	4,7
P5	19,8	0,22	0,9	50	5,9
P6	12,5	0,16	0,8	65	5,8
P7	12,2	0,17	0,5	80	5,5

Вхідні дані за варіантом зображені на рисунку 9:

Кластерний аналіз

Вхідна матриця					
	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5
P1	15.5	0.2	0.1	105	4.1
P2	12.3	0.15	0.5	55	4.5
P3	11.2	0.18	0.2	75	4.2
P4	17.7	0.17	0.7	100	4.7
P5	19.8	0.22	0.9	50	5.9
P6	12.5	0.16	0.8	65	5.8
P7	12.2	0.17	0.5	80	5.5

Створити кластери

Рисунок 9. Вхідна матриця за варіантом роботи

Обрахунки кластерів для даної матриці зображені на рисунках 10-14:

Обраховуємо суми колонок:

Вхідна матриця із сумами колонок					
	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5
P1	15.5	0.2	0.1	105	4.1
P2	12.3	0.15	0.5	55	4.5
P3	11.2	0.18	0.2	75	4.2
P4	17.7	0.1701	0.7	100	4.7
P5	19.8	0.22	0.9	50	5.9
P6	12.5	0.16	0.8	65	5.8
P7	12.2	0.1701	0.5	80	5.5
Sum(Xij)	101.2	1.25	3.7	530	34.7

Рисунок 10. Обрахунок сум усіх колонок матриці

Ділимо кожен елемент на суму колонки:

Вхідна матриця поділена на суми колонок					
	Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**
P1	0.1532	0.16	0.0271	0.1982	0.1182
P2	0.1216	0.12	0.1352	0.1038	0.1297
P3	0.1107	0.144	0.0541	0.1416	0.1211
P4	0.175	0.136	0.1892	0.1887	0.1355
P5	0.1957	0.176	0.2433	0.0944	0.1701
P6	0.1236	0.128	0.2163	0.1227	0.1672
P7	0.1206	0.136	0.1352	0.151	0.1586

Нормалізуємо матрицю:

Нормалізована матриця						
	Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**	**Wij**
P1	0.1532	0.16	0.0271	0.1982	0.1182	0.6565
P2	0.1216	0.12	0.1352	0.1038	0.1297	0.6102
P3	0.1107	0.144	0.0541	0.1416	0.1211	0.5713
P4	0.175	0.136	0.1892	0.1887	0.1355	0.8243
P5	0.1957	0.176	0.2433	0.0944	0.1701	0.8793
P6	0.1236	0.128	0.2163	0.1227	0.1672	0.7576
P7	0.1206	0.136	0.1352	0.151	0.1586	0.7012

Рисунок 11. Нормалізація матриці

Ізотонічна розбивка

Обраховуємо відстані та мінімальні значення рядків:

Матриця відстаней								
	P1	P2	P3	P4	P5	P6	P7	Min Pi
P1	0	0.0464	0.0852	0.1678	0.2229	0.1011	0.0447	0.0447
P2	0.0464	0	0.0389	0.2141	0.2692	0.1474	0.0911	0.0389
P3	0.0852	0.0389	0	0.253	0.308	0.1863	0.1299	0.0389
P4	0.1678	0.2141	0.253	0	0.0551	0.0667	0.1231	0.0551
P5	0.2229	0.2692	0.308	0.0551	0	0.1218	0.1782	0.0551
P6	0.1011	0.1474	0.1863	0.0667	0.1218	0	0.0564	0.0564
P7	0.0447	0.0911	0.1299	0.1231	0.1782	0.0564	0	0.0447

Критична відстань R (= Max Min Pi) = 0.056389154307309175

Можна виділити наступні кластери:

Кластер 1: P6

Кластер 2: P2,P3,P1,P7,P4,P5

Рисунок 11. Обрахунок відстаней та результат ізотонічної розбивки на кластери

Ізоморфна розбивка

Нормалізуємо матрицю:

Нормалізована матриця (ізоморфна)					
	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5
P1	0.2334	0.2438	0.0412	0.3018	0.18
P2	0.1993	0.1967	0.2215	0.1701	0.2126
P3	0.1938	0.2521	0.0947	0.2478	0.2119
P4	0.2123	0.1651	0.2296	0.229	0.1644
P5	0.2226	0.2002	0.2767	0.1073	0.1934
P6	0.1631	0.169	0.2855	0.1619	0.2207
P7	0.172	0.194	0.1928	0.2153	0.2261

Рисунок 12. Нормалізація матриці для ізоморфної розбивки

Обраховуємо відстані та мінімальні значення рядків:

Матриця відстаней (ізоморфна)								
	P1	P2	P3	P4	P5	P6	P7	Min Pi
P1	0	0.2331	0.0919	0.2184	0.309	0.3024	0.1971	0.0919
P2	0.2331	0	0.1589	0.0839	0.089	0.0794	0.0617	0.0617
P3	0.0919	0.1589	0	0.1696	0.2382	0.2274	0.1214	0.0919
P4	0.2184	0.0839	0.1696	0	0.1386	0.115	0.0884	0.0839
P5	0.309	0.089	0.2382	0.1386	0	0.0912	0.1496	0.089
P6	0.3024	0.0794	0.2274	0.115	0.0912	0	0.1104	0.0794
P7	0.1971	0.0617	0.1214	0.0884	0.1496	0.1104	0	0.0617

Критична відстань R (= Max Min Pi) = 0.09184186367991484

Можна виділити наступні кластери:

Кластер 1: P1,P3

Кластер 2: P2,P7,P6,P4,P5

Рисунок 13. Результат ізоморфної розбивки на кластери

Увесь вихідний код даної системи можна побачити на GitHub репозиторії за посиланням: https://github.com/VadymMytr/ISPPR

Висновок: під час виконання даної лабораторної роботи було ознайомлено із ізотонічною та ізоморфною розбивками на кластери, розібрано відповідні формули, створено систему, яка розбиває задану матрицю на кластері та протестовано її на тестових даних із теоретичної частини, а також на даних за варіантом. Можна стверджувати, що система працює коректно та швидко.