Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня
магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Міняйло Данила Дмитровича*
	(ПІБ)
академічної групи	*121М-20-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Розробка та дослідження асамблевих алгоритмів кластеризації даних*

Д.Д. Міняйло

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний				
економічний	*Проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*Доц. Приходченко С.Д.*			

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 21 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-20-1* *Міняйло Данилу Дмитровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка та дослідження асамблевих алгоритмів кластеризації даних*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від ___.___.2021 р. № _____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процеси розробки та дослідження асамблевих алгоритмів кластеризації даних.

Предмет досліджень – використання програмних засобів для дослідження асамблевих алгоритмів кластеризації даних.

Методи дослідження: нейромережеві методи, використання існуючих програмних пакетів для кластеризації, розробка програмного забезпечення реалізації асамблевих алгоритмів кластеризації даних.

Мета роботи – розробити та реалізувати асамблеві алгоритми кластеризації даних, перевірити ефективність застосованих методів.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що запропоновано у випадку із ансамблем на основі матриці схожості, що використовує метод Даве-Сена, були отримані більш вищі бали, які навіть вищі, ніж ті, що були у розбиттів до входження у ансамбль. Варіант цього ансамблю, що кластеризує дані ієрархічним алгоритмом має більш кращі результати тільки за індексом Фолка-Меллоу. У будь-кому разі можна скласти висновок, що використовуючи ансамбль вдалося отримати більш якісне розбиття.

Практична цінність результатів полягає у тому, що в результаті проведеного дослідження було спроектовано алгоритм для дослідження асамблевих алгоритмів кластеризації даних.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Для тестування розробленого програмного забезпечення, порівняльного аналізу методів кластеризації та дослідження ансамблевих алгоритмів провести аналіз різних наборів штучних та реальних даних з різноманітною кластерною структурою. Якість отриманих результатів оцінити відносними та зовнішніми (для даних з відомою кластерною структурою) критеріями.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз джерел та постановка задачі	12.09.2021 - 30.09.2021
Побудова математичних моделей та розробка алгоритму	01.10.2021 - 31.10.2021
Розробка та тестування програмного забезпечення для реалізації асамблевих алгоритмів кластеризації даних, перевірка ефективності застосованих методів	01.11.2021 - 30.12.2021

Завдання видав _____ _____
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Міняйло Д.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2021 р.*

Термін подання кваліфікаційної роботи до ЕК *20.01.2021*

РЕФЕРАТ

Пояснювальна записка: ___ стор., ___ рис., ___ таблиці, ___ додатка, ___ джерел.

Об’єкт досліджень – процеси розробки та дослідження асамблевих алгоритмів
кластеризації даних.

Предмет досліджень – використання програмних засобів для дослідження асамблевих алгоритмів кластеризації даних.

Методи дослідження: нейромережеві методи, використання існуючих програмних пакетів для кластеризації, розробка програмного забезпечення реалізації асамблевих алгоритмів кластеризації даних.

Мета роботи – розробити та реалізувати асамблеві алгоритми кластеризації даних, перевірити ефективність застосованих методів.

Новизна запропонованих рішень визначається тим, що запропоновано у випадку із ансамблем на основі матриці схожості, що використовує метод Дave-Сена, були отримані більш вищі бали, які навіть вищі, ніж ті, що були у розбиттів до входження у ансамбль. Варіант цього ансамблю, що кластеризує дані ієрархічним алгоритмом має більш кращі результати тільки за індексом Фолка-Меллоу.

Практична цінність результатів полягає у тому, що в результаті проведеного дослідження було спроектовано алгоритм для дослідження асамблевих алгоритмів кластеризації даних.

У розділі «Економіка» проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПЗ і тривалості його розробки, а також проведені маркетингові дослідження ринку збуту створеного програмного продукту.

Список ключових слів: НЕЙРОННІ МЕРЕЖІ, АСАМБЛЕВИЙ НЕЙРОМЕРЕЖЕВИЙ ПІДХІД, НЕЧІТКА КЛАСТЕРИЗАЦІЯ

ABSTRACT

Explanatory note: ___ pages, ___ figures, ___ tables, ___ appendices, ___ sources.

The object of research is the processes of development and research of assembly algorithms data clustering.

Subject of research the use of software for the study of assembly algorithms for data clustering.

Research methods: neural network methods, use of existing software packages for clustering, development of software for the implementation of assembly algorithms for data clustering.

The purpose of the work - to develop and implement assembly algorithms for data clustering, to test the effectiveness of applied methods.

The novelty of the proposed solutions is determined by the fact that proposed in the case of an ensemble based on a similarity matrix using the Dave-Sen method, higher scores were obtained, which are even higher than those of the breakups before joining the ensemble. A variant of this ensemble that clusters data by a hierarchical algorithm has better results only in the Folk-Mellow index. In any case, we can conclude that using the ensemble managed to get a better breakdown.

The practical value of the results is that as a result of the study an algorithm was designed to study the assembly algorithms of data clustering.

In the section "Economics" calculations of the complexity of software development, the cost of creating software and the duration of its development, as well as marketing research of the market for the software product.

List of key words: NEURAL NETWORKS, ASSEMBLY NEURAL NETWORK APPROACH, FUZZY CLUSTERIZATION

ЗМІСТ

ВСТУП

Кластерний аналіз (англ. Data clustering) – задача розбиття заданої вибірки об'єктів на підмножини, що називаються кластерами, за певним критерієм. При цьому кожен кластер повинен складатися зі схожих об'єктів, а об'єкти різних кластерів – істотно відрізнятися один від одного. Завдання кластеризації відноситься до статистичної обробки, а також до широкого класу завдань навчання без вчителя.

Кластерний аналіз даних є актуальною задачею, що знайшла застосування у багатьох сферах діяльності людини. У комп'ютерних науках кластеризація використовується для розпізнавання об'єктів, у пошукових системах – для групування сайтів за змістом, що дає більш релевантну відповідь на запит користувача, у медицині – для групування хворих зі схожим діагнозом у групи для подальшого лікування. У біології кластеризація дозволяє розбити великі набори генів на підгрупи, що полегшує аналіз їх взаємозв'язків. Також кластеризація широко використовується у психології, соціології, антропології, маркетингу, економіці тощо.

Нечітка кластеризація розглядає приналежність об’єкта одразу багатьом кластерам, що покращує рівень гнучкості необхідний для дослідження невизначеності, що присутня реальним даним різноманітних прикладних областей.

Ансамблеві методи розроблені для покращення надійності та точності алгоритмів кластеризації, а також для можливості обробки даних зі складною кластерною структурою. Застосування ансамблів алгоритмів у кластерному аналізі є досить актуальним напрямом досліджень, оскільки на основі даного підходу може бути вирішено багато задач, таких як підвищення точності та стійкості результатів, зменшення простору ознак, кластеризація різнотипних даних, розпаралелювання обчислень та ін. Ансамблі нечіткої кластеризації

дозволяють поєднати гнучкість теорії нечіткої логіки та надійність ансамблевої агрегації.

Метою кваліфікаційної роботи є створення програмного забезпечення кластеризації даних та дослідження чітких, нечітких та ансамблевих методів кластерного аналізу.

Кваліфікаційна робота складається з трьох розділів. У першому розділі було розглянуто алгоритмічну основу, а саме описано методи чіткої та нечіткої кластеризації, відносні та зовнішні критерії оцінки якості роботи та ансамблеві алгоритми кластеризації даних. У другому розділі описано розроблене у процесі виконання роботи програмне забезпечення, його функціонал та інтерфейс. Третій розділ містить у собі практичну реалізацію розробленого програмного забезпечення та аналіз результатів отриманих у процесі його роботи.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ТА АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДАНИХ

1.1 Огляд проблеми

Нехай існує множина $X = \{x_1, x_2, \dots, x_n\}$, що містить у собі n об'єктів. Кожний з цих об'єктів має m властивостей. Об'єкти $X = \{x_1, x_2, \dots, x_n\}$ разом із властивостями можна зобразити у вигляді матриці розмірності nxm:

$$X = \begin{Bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{Bmatrix}, \quad (1.1)$$

де x_{ij} – це j-та властивість і-го об'єкта.

Оскільки кожен об'єкт має m властивостей, то його можна зобразити, як точку m-вимірного простору, а всю множину об'єктів $X = \{x_1, x_2, \dots, x_n\}$ — як сукупність n точок m-вимірного простору.

Якщо задати деяку метрику відстані між об'єктами, то можна побудувати матрицю відстаней. Оскільки кількість об'єктів дорівнює n, то розмірність матриці буде nxn.

$$d = \begin{Bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \dots & \dots & \dots & \dots \\ d_{n1} & d_{n2} & \dots & d_{nn} \end{Bmatrix}, \quad (1.2)$$

де d_{ij} – відстань між об'єктами і та j.

Усі об'єкти множини $X = \{x_1, x_2, \dots, x_n\}$ можна класифікувати певним чином у деяку кількість класів. Процес розбиття множини $X = \{x_1, x_2, \dots, x_n\}$ на однорідні за певним формальним критерієм подібності групи (кластери)

називається кластеризація. Основною властивістю цих груп є те, що об'єкти, які належать одному кластеру, подібніші між собою, ніж об’єкти з різних кластерів.)

У випадку інтерпретації об’єктів $X = \{x_1, x_2, \dots, x_n\}$, як точок m-вимірного простору, такою спільною рисою буде геометрична відстань. Тобто об’єкти, що відносяться до одного кластера будуть на відносно меншій відстані один від одного.

Методи кластерного аналізу можна поділити на чіткі та нечіткі. У випадку чіткої кластеризації кожен об’єкт множини $X = \{x_1, x_2, \dots, x_n\}$ буде відноситись тільки до одного кластера із усіх. Для зображення приналежності об’єкта до певного кластеру використовується матриця розбиття:

$$U = \begin{Bmatrix} u_{11} & u_{12} & \dots & u_{1k} \\ u_{21} & u_{22} & \dots & u_{2k} \\ \dots & \dots & \dots & \dots \\ u_{n1} & u_{n2} & \dots & u_{nk} \end{Bmatrix}, \qquad (1.3)$$

де u_{ik} – відношення і-го об’єкта до k-го кластеру, при $u_{ik} \in \{0, 1\}$.

У випадку нечіткої кластеризації кожен об’єкт $X = \{x_1, x_2, \dots, x_n\}$ відноситься до усіх кластерів одночасно, але із різним ступенем вірогідності. Тобто у матриці розбиття значення u_{ik} буде приймати значення із інтервалу [0, 1], а не з дискретної множини {0, 1}, як у випадку чіткої кластеризації.

Після аналізу результатів нечіткої кластеризації іноді виникає необхідність перейти до чітких кластерів, також така потреба є при візуалізації результатів. Оскільки кожен об’єкт даних відноситься у певній мірі до кожного кластеру, то буде неможливо зобразити їх на діаграмі. Для вирішення цієї проблеми потрібно кожен об’єкт віднести тільки до одного кластера одночасно. Тобто необхідно кожній нечіткій множині поставити у відповідність певне число. Цей процес називається дефазифікація. У кваліфікаційній роботі в якості дефазифікації був використаний метод максимумів. Суть цього методу полягає у наступному: кожен об’єкт буде відноситись до того кластеру, до якого має більшу ступінь

відношення. Тобто із вектора розбиття, що характеризує відношення об’єкта до кожного кластера $U_i = (u_{i1}, u_{i2}, ..., u_{ik})$, де i – об’єкт вхідних даних, k – кількість кластерів, потрібно знайти максимальне значення M = мах(u_{ij}), j = 1...k. Після знаходження максимального елементу u_{ij} об’єкт можна віднести до j-го кластера. Цей алгоритм потрібно застосувати до кожного об’єкту вхідних даних.

1.2. Методи нечіткої кластеризації

1.2.1. Метод C-MEANS

Параметри методу [1, 4]:

1) C – кількість нечітких кластерів $C \geq 2$;
2) N – кількість досліджуваних об'єктів;
3) P – кількість атрибутів у вхідних даних;
4) d – функція відстані між об’єктами;
5) t – кількість ітерацій;
6) ε – умова зупинки алгоритму $\varepsilon > 0$;
7) γ – ступінь нечіткості кластеризації $1 \leq \gamma \leq \infty$.

Алгоритм методу:

1. Випадковим чином задати матрицю розбиття $P_{C\times N} \in [U_{li}],\ 0 \leq U_{li} \leq 1,\ \sum_{i=1}^{N} U_{li} > 0$, таким чином отримуємо початкове розбиття $P_{(0)} = \left(M_{(0)}^{1}, \dots, M_{(0)}^{K}\right)$ на C нечітких кластерів, $t = 1$.

2. Визначити центри нечітких кластерів наступним чином:

$$O_l = \frac{1}{\sum_{i=1}^{N}(U_{li})^{\gamma}} * \sum_{i=1}^{N}((U_{li})^{\gamma} X_i),\quad l = 1, \dots, C$$

$$O_l = \left(O_{l1}, O_{l2}, \dots, O_{lj}\right),\qquad j = 1, \dots, N$$

$$O_{lj} = \frac{1}{\sum_{i=1}^{N}(U_{li})^{\gamma}} * \sum_{i=1}^{N}\left((U_{li})^{\gamma} X_{ij}\right), \qquad l = 1, \dots, C, \qquad j = 1, \dots, P$$

3. Визначити відстань між елементами наступним чином:

$$d(X_i, O_l) = \sqrt{\sum_{j=1}^{P}\left(X_{ij} - O_{lj}\right)^2}, \qquad i, j = 1, \dots, N, \qquad l = 1, \dots, C$$

4. Обчислити нову матрицю приналежності наступним чином:

$$U_{li} = \left(\sum_{k=1}^{C}\left(\frac{d(X_i, O_l)}{d(X_i, O_k)}\right)^{\frac{2}{\gamma-1}}\right)^{-1}, \qquad l, k = 1, \dots, C, \qquad i = 1, \dots, N$$

5. Якщо $|Q(P_t) - Q(P_{t-1})| < \varepsilon$, то $P = P_{(t)}$, то закінчити кластеризацію, інакше $t = t + 1$ і перейти до пункту 2).

1.2.2 Метод Густаффсона-Кесселя

Параметри алгоритму [3, 4]:

1) C – кількість нечітких кластерів $C \geq 2$;
2) N – кількість досліджуваних об'єктів;
3) P – кількість атрибутів у вхідних даних;
4) d – функція відстані між об’єктами;
5) t – кількість ітерацій;
6) ε – умова зупинки алгоритму $\varepsilon > 0$;
7) γ – ступінь нечіткості кластеризації $1 \leq \gamma \leq \infty$.

Алгоритм методу:

1. Випадковим чином задати матрицю розбиття $P_{C\times N} \in [U_{li}],\ 0 \leq U_{li} \leq 1\ \sum_{i=1}^{N} U_{li} > 0$, таким чином отримуємо початкове розбиття $P_{(0)} = \left(M_{(0)}^{1}, \dots, M_{(0)}^{K}\right)$ на C нечітких кластерів, $t = 1$.

2. Визначити центри нечітких кластерів наступним чином:

$$O_l = \frac{1}{\sum_{i=1}^{N}(U_{li})^{\gamma}} * \sum_{i=1}^{N}((U_{li})^{\gamma} X_i), \quad l = 1, \dots, C$$

$$O_l = \left(O_{l1}, O_{l2}, \dots, O_{lj}\right), \qquad j = 1, \dots, N$$

$$O_{lj} = \frac{1}{\sum_{i=1}^{N}(U_{li})^{\gamma}} * \sum_{i=1}^{N}\left((U_{li})^{\gamma} X_{ij}\right), \qquad l = 1, \dots, C, \qquad j = 1, \dots, P$$

3. Визначити коваріаційну матрицю кожного кластера наступним чином:

$$F_l = \frac{1}{\sum_{i=1}^{N}(U_{li})^{\gamma}} * \sum_{i=1}^{N}\left((U_{li})^{\gamma}(X_i - O_l)^T(X_i - O_l)\right), \qquad i = 1, \dots, N, l = 1, \dots, C$$

4. Визначити відстань між елементами наступним чином:

$$d(X_i, O_l) = \sqrt{(X_i - O_l)\left[det(F_l)^{\frac{1}{p}} F_l^{-1}\right](X_i - O_l)^T}, \qquad i = 1, \dots, N, l = 1, \dots, C$$

5. Обчислити нову матрицю приналежності наступним чином:

$$U_{li} = \left(\sum_{k=1}^{C}\left(\frac{d(X_i, O_l)}{d(X_i, O_k)}\right)^{\frac{2}{\gamma-1}}\right)^{-1}, \qquad l, k = 1, \dots, C, \qquad i = 1, \dots, N$$

6. Якщо $|Q(P_t) - Q(P_{t-1})| < \varepsilon$, та закінчити кластеризацію, інакше $t = t + 1$ і перейти до пункту 2).

1.2.3 Метод Даве-Сена

Параметри алгоритму [6]:

1) C – кількість нечітких кластерів $C \geq 2$;
2) N – кількість досліджуваних об'єктів;
3) d – функція відстані між об’єктами;
4) t – кількість ітерацій;
5) ε – умова зупинки алгоритму $\varepsilon > 0$;

6) γ – ступінь нечіткості кластеризації $1 \le \gamma \le \infty$.

Алгоритм методу:

1. Випадковим чином задати матрицю розбиття $P_{C\times N} \in [U_{li}],\ 0 \le U_{li} \le 1,\ \sum_{i=1}^{N} U_{li} > 0$, таким чином отримуємо початкове розбиття $P_{(0)} = \left(M_{(0)}^{1}, \dots, M_{(0)}^{K}\right)$ на С нечітких кластерів, $t = 1$.

2. Обчислити значення матриці прототипів $V_{(b)}$ наступним чином:

$$V_{li} = \frac{\gamma \sum_{j=1}^{N} \left(U_{lj}^{\gamma} d\left(X_i, X_j\right)\right)}{\sum_{j=1}^{N} U_{lj}^{\gamma}} - \frac{\gamma \sum_{h=1}^{N} \sum_{j=1}^{N} \left(U_{lj}^{\gamma} U_{lh}^{\gamma} d\left(X_J, X_h\right)\right)}{2\left(\sum_{j=1}^{N} U_{lj}^{\gamma}\right)^{2}}$$

$$l = 1, \dots, C, \qquad i, j, h = 1, \dots, N$$

Для всіх $l = 1, \dots, C$ з використанням нових обчислених значень приналежності U_{lj} або U_{lh}, якщо $j < i$ або $h < i$ у відповідності із пунктом 4), і попередніх значень приналежності U_{lj} або U_{lh} при $j \ge i$ або $h \ge i$.

3. Здійснити перерахунок значень приналежності U_{lj} або U_{lh} наступним чином:

$$U_{li} = \frac{\left(\frac{1}{V_{li}}\right)^{\frac{1}{\gamma-1}}}{\sum_{h=1}^{C} \left(\frac{1}{V_{hi}}\right)^{\frac{1}{\gamma-1}}} h = 1, \dots, C$$

4. Якщо $i < N$, то $i = i + 1$ та здійснити перехід до пункту 3).

5. Якщо $|Q(P_t) - Q(P_{t-1})| < \varepsilon$, та закінчити кластеризацію, інакше $t = t + 1$ і перейти до пункту 2).

1.3. Методи чіткої кластеризації

1.3.1. Метод k-середніх

Метод К-середніх є найбільш вживаним алгоритмом розділової кластеризації. Вирізняється простотою реалізації, наочністю та швидкістю

роботи, що дозволяє його застосування для дуже великих за обсягом вибірок. Головний недолік полягає у тому, що метод дуже чутливий до початкового вибору центрів кластерів та їх кількості.

Алгоритм методу [4, 6]:

1. Серед досліджуваних об'єктів $x_i, i=\overline{1,N}$ обираємо деяким чином (випадково, найвіддаленіші, перші) K еталонних $x_i^e, i=\overline{1,K}$, та вважаємо їх рівними центрам окремих кластерів $C_i=(c_{i1},c_{i2},...,c_{ip}),\ C_i=x_i^e,\ \ N_i=1,\ \ i=\overline{1,K}$, K – кількість кластерів – вхідний параметр алгоритму.

2. Кожен об'єкт $x_i, i=\overline{1,N}$ відносимо до кластеру g_l: $d(x_i,C_l)=\min\limits_{j=1,K} d(x_i,C_j),\ N_l=N_l+1.$

3. Обчислюємо нові центри кластерів: $C_i=\frac{1}{N_i}\sum\limits_{j=1}^{N_i} x_j,\ c_{ih}=\frac{1}{N_i}\sum\limits_{j=1}^{N_i} x_{jh},\ h=\overline{1,p}$

4. Повторюємо кроки 2–3 доки центри не стабілізуються або кількість ітерацій не перевищить задану.

1.3.2 Метод DBSCAN

Параметри алгоритму [6]:

- ε – окіл у якому повинний знаходитись об'єкт, щоб потрапити у кластер;
- MinP – мінімальна кількість об'єктів для формування кластеру.

Алгоритм методу:

1. Задаємо параметри алгоритму ε та $\min P$. Усі точки досліджуваної траєкторії розглядаємо як множину некластеризованих точок S.

2. З множини S обираємо довільну точку r та обчислюємо відстані $d_{ri},\ i\in S$

3. Знаходимо $countP$ – кількість точок, що містяться у ε-околі точки r, тобто таких, що $d_{ri}<\varepsilon$,

- якщо $countP < \min P$, то позначаємо точку r шумовою і переходимо до п.2,
- інакше з точки r та її ε-околу формуємо кластер та переходимо до п.4.

4. Для кожної точки, що належить кластеру знаходимо точки, що належать її ε-околу та додаємо до кластера.

5. Усі точки сформованого кластера виключаємо з множини S.

6. Якщо у множині S залишились нешумові об'єкти, переходимо до п.2, інакше завершуємо кластеризацію.

1.3.3. Методи ієрархічної агломеративної кластеризації

Перед використанням ієрархічного алгоритму кластеризації [8] потрібно ввести поняття відстані між кластерами. Нехай існує два кластери $S_1 = \{X_i^{(1)},\ i = 1, \dots, N_1\}$ та $S_2 = \left\{X_i^{(2)},\ i = 1, \dots, N_2\right\}$. Відстань $D(S_1,\ S_2)$ між цими об'єктами можна визначати:

1) як відстань найближчого сусіда, що дорівнює відстані між найближчими об'єктами кластерів

$$D(S_1,\ S_2) = min_{\substack{i_1 = 1,\dots,N_1;\\ i_2 = 1,\dots,N_2}} d(X_{i_1}^{(1)},\ X_{i_2}^{(2)})$$

2) відстань найвіддаленішого сусіда – відстань між найвіддаленішими об'єктами кластерів

$$D(S_1,\ S_2) = max_{\substack{i_1 = 1,\dots,N_1;\\ i_2 = 1,\dots,N_2}} d(X_{i_1}^{(1)},\ X_{i_2}^{(2)})$$

3) середню зважену відстань, яка дорівнює середньому значенню попарних відстаней між об'єктами різних кластерів

$$D(S_1,\ S_2) = \frac{1}{N_1 N_2} \sum_{i_1 = 1}^{N_1} \sum_{i_2 = 1}^{N_2} d(X_{i_1}^{(1)},\ X_{i_2}^{(2)})$$

4) середню незважену відстань, яка відрізняється від попередньої тим, що в ній не враховується розміри кластерів

$$D(S_1,\ S_2) = \frac{1}{4} \sum_{i_1 = 1}^{N_1} \sum_{i_2 = 1}^{N_2} d(X_{i_1}^{(1)},\ X_{i_2}^{(2)})$$

5) медіанна відстань

$$D(S_1,\ S_2) = \frac{1}{2}\ d(Me^{(1)},\ Me^{(2)})\ ,$$

де $Me^{(l)} = (me_1^{(l)},\ me_2^{(l)},\ \ldots,\ me_p^{(l)})$ – об'єкт із медіанними значеннями ознак у l-му кластері, $l = 1, 2$; значення $me_j^{(l)}$, j = 1, …, p визначають за відсортованою вибіркою $\{x_{1,j}^{(l)},\ x_{2,j}^{(l)},\ \ldots,\ x_{N_l,j}^{(l)}\}$ за формулою

$$me_j^{(l)} = \begin{cases} x_{(N_l+1)/2,j}^{(l)} & \text{коли } N_l - \text{непарне;} \\ \frac{1}{2}\left(x_{\frac{N_l}{2},j}^{(l)} + x_{\frac{N_l}{2}+1,j}^{(l)}\right), & \text{коли } N_l - \text{парне;} \end{cases}$$

6) відстань між центрами , яка дорівнює відстані між об'єктами із середніми значеннями усіх ознак $D(S_1,\ S_2) = d(\overline{X^{(1)}}, \overline{X^{(2)}})$,

де $\overline{X^{(l)}} = \left(\overline{x_1^{(l)}}, \overline{x_2^{(l)}}, \ldots, \overline{x_p^{(l)}}\right)$, $\overline{x_j^{(l)}} = \frac{1}{N_l}\sum_{i=1}^{N_l} x_{i,j}^{(l)}$, $l = 1, 2$

7) відстань Уорда

$$D(S_1,\ S_2) = \frac{N_1 N_2}{N_1 + N_2} d^2(\overline{X^{(1)}}, \overline{X^{(2)}})$$

Алгоритм методу:

1. Кожен об'єкт вважається окремим кластером, що містить тільки один елемент. Розраховується відстань між всіма утвореними кластерами, а результати заносяться у матрицю відстаней, тобто

$$D = (D_{ij} = d(X_i, X_j);\ i,j = 1, \ldots, N)$$

2. У матриці відстаней D шукаємо мінімальне значення $D(S_i, S_j)$. Кластери S_i та S_j об'єднують в один кластер S_{i+j}. Якщо мінімальних значень буде декілька, то потрібно обрати перше із них.

3. Із матриці D вилучаються рядки i та j, а також стовпчики i та j, тобто ті рядки та стовпчики, що містили відстані від кластерів S_i та S_j до усіх інших кластерів. Після цього до матриці відстаней додають один стовпчик та один рядок, що містять відстані від кластеру S_{i+j} до інших кластерів.

4. Повертаємося до пункту 2 доки не залишиться тільки один кластер.

Відстань від нового кластера до інших можна обчислювати не за означенням, а використовуючи вже відомі відстані за формулою Ланса-Уільямса [3].

$$D(S_{l+h}, S_m) = \alpha_l D(S_l, S_m) + \alpha_h D(S_h, S_m) + \beta D(S_l, S_h) + \gamma |D(S_l, S_m) - D(S_h, S_m)|, \text{ де } \alpha_l, \alpha_h, \beta, \gamma - \text{числові параметри.}$$

Різні значення параметрів α_l, α_h, β, γ відповідають різним способам підрахунку відстані між кластерами та породжують різні види агломеративних ієрархічних методів:

1) Найближчого сусіда (одного зв'язку):

$$\alpha_l = 0{,}5,\ \alpha_h = 0{,}5,\ \beta = 0,\ \gamma = -0{,}5$$

2) Найвіддаленішого сусіда (повного зв'язку):

$$\alpha_l = 0{,}5,\ \alpha_h = 0{,}5,\ \beta = 0,\ \gamma = 0{,}5$$

3) Середнього зваженого зв'язку:

$$\alpha_l = \frac{N_l}{N_l + N_h},\ \alpha_h = \frac{N_h}{N_l + N_h},\ \beta = 0,\ \gamma = 0$$

4) Простого середнього зв'язку:

$$\alpha_l = 0{,}5,\ \alpha_h = 0{,}5,\ \beta = 0,\ \gamma = 0$$

5) Медіанного зв'язку:

$$\alpha_l = 0{,}5,\ \alpha_h = 0{,}5,\ \beta = -0{,}25,\ \gamma = 0$$

6) Центроїдний:

$$\alpha_l = \frac{N_l}{N_l + N_h},\ \alpha_h = \frac{N_h}{N_l + N_h},\ \beta = -\alpha_l \alpha_h = -\frac{N_l N_h}{(N_l + N_h)^2},\ \gamma = 0$$

7) Уорда:

$$\alpha_l = \frac{N_m + N_l}{N_m + N_l + N_h},\ \alpha_h = \frac{N_m + N_h}{N_m + N_l + N_h},\ \beta = \frac{N_m}{N_m + N_l + N_h},\ \gamma = 0,$$

де N_m – кількість об'єктів у кластері S_m, відстань до якого обчислюють.

1.4. Постановка задачі

У даній кваліфікаційній роботі необхідно вирішити наступні задачі:

1. Розробити програмне забезпечення кластерного аналізу даних, ядро якого складають чіткі (K-means, DBSCAN, ієрархічні агломеративні), нечіткі (K-means, Густаффсона-Кесселя, Даве-Сена) та ансамблеві (Педрича класичний та модифікований, на основі матриці подібності у двох варіантах) методи.
2. Реалізувати оцінювання якості отриманих результатів відносними та зовнішніми критеріями.
3. Для візуалізації отриманих результатів передбачити наявність діаграми розсіювання, дендрограми та таблиць.
4. За допомогою розробленого програмного забезпечення провести дослідження впливу ансамблевих методів на стійкість та якість результатів кластеризації різних наборів штучних та реальних даних.

РОЗДІЛ 2
АСАМБЛЕВІ АЛГОРИТМИ КЛАСТЕРИЗАЦІЇ ДАНИХ

2.1. Оцінка якості результатів кластеризації

На даний момент серед великої кількості підходів до розв’язання задачі кластеризації не існує універсальних методів. Кожен підхід має свої переваги та недоліки. Результат досить сильно залежить від структури досліджуваних даних, вибору системи ознак, мір близькості, способів формалізації уявлень про схожість об’єктів та кластерів. Випадковий необґрунтований вибір методу може призвести до того, що отримане ним розбиття буде зовсім відмінним від природної, притаманної досліджуваним даним, кластерної структури. Тому актуальними проблемами кластерного аналізу є оцінювання результатів для пошуку розбиття, що найкраще відповідає структурі даних.

Однозначне визначення та кількісні характеристики якісної кластерної структури на даний момент відсутні в літературі. Існує лише інтуїтивне поняття того, що об’єкти всередині кластерів повинні розташовуватися якомога ближче один до одного, а самі кластери мають бути значно відокремленими.

Існує три підходи дослідження якості результатів кластеризації. Перший ґрунтується на зовнішніх критеріях, тобто оцінюються результати алгоритму кластеризації на основі заздалегідь визначеної структури, яка накладається на набір даних і відображає наші припущення. Другий підхід заснований на внутрішніх критеріях. Можна оцінювати результати алгоритму кластеризації в термінах величин, які пов’язані з векторами даних та матрицею близькості. Третій підхід використовує відносні критерії. Основна ідея полягає в оцінці структури кластеризації, порівнюючи її з іншими кластеризаційними схемами, отриманими іншими алгоритмами або при різних значеннях параметрів одного методу.

2.1.1. Відносні критерії якості нечіткої кластеризації

- Критерій Рубенса [4, 5]

$$Q_R(P) = \frac{C\left[\sum_{l=1}^{C}\sum_{i=1}^{N} U_{li}^2\right] - N}{N(C-1)} \to max, \qquad l = 1, \dots, C, \qquad i = 1, \dots, N$$

де C – кількість нечітких кластерів $C \geq 2$, N – кількість досліджуваних об'єктів, U – кінцева матриця приналежності нечіткої кластеризації.

Той метод кластеризації у якого буде найбільше значення критерію буде вважатися найбільш вдалим для даного набору даних.

- Критерій Хіє-Бені [4, 7]

$$Q_{XB}(P) = \frac{\sum_{l=1}^{C}\sum_{j=1}^{N}\left(U_{lj}^{\gamma}\left(X_j - O_l\right)^2\right)}{N\min\limits_{l,h}(O_l - O_h)^2} \to min\,, \qquad l, h = 0, \dots, C, \qquad j = 1, \dots, N$$

де C – кількість нечітких кластерів $C \geq 2$, N – кількість досліджуваних об'єктів, U – кінцева матриця приналежності нечіткої кластеризації, O – центри нечітких кластерів.

Той метод кластеризації у якого буде найменше значення критерію буде вважатися найбільш вдалим для даного набору даних.

2.1.2. Відносні критерії якості чіткої кластеризації

- Індекс Калінського-Гарабача [9]

$$Q_{CH}(G) = \frac{B/(K-1)}{W/(N-K)} \to \max,$$

$$B = \sum_{i=1}^{K} N_i \left\|C_i - C\right\|^2, \;\; W = \sum_{i=1}^{K}\sum_{j=1}^{N_i}\left\|u_j^{(i)} - C_i\right\|^2.$$

де К – кількість кластерів $К \geq 2$, N – кількість досліджуваних об'єктів, $C_i = (\bar{u}_1^{(i)}, \bar{u}_2^{(i)}, \ldots, \bar{u}_T^{(i)})$ – центр *i*-го кластера, $C = (\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_T)$ – центр усієї множини об'єктів.

Той метод кластеризації у якого буде найбільше значення критерію буде вважатися найбільш вдалим для даного набору даних.

- Індекс Беджека-Данна [10]

$$Q_{BD}(G) = \frac{\min\limits_{i,j \in 1,K,\, i \neq j} \{\delta(g_i, g_j)\}}{\max\limits_{h \in 1,K} \{\Delta(g_h)\}} \to \max,$$

$$\delta(g_i, g_j) = \frac{1}{N_i \cdot N_j} \sum_{l=1}^{N_i} \sum_{m=1}^{N_j} d(u_l^{(i)}, u_m^{(j)}), \quad \Delta(g_h) = \frac{2}{N_h} \sum_{i=1}^{N_h} d(u_i^{(h)}, C_h),$$

де К – кількість нечітких кластерів $К \geq 2$, N – кількість досліджуваних об'єктів, $C_i = (\bar{u}_1^{(i)}, \bar{u}_2^{(i)}, \ldots, \bar{u}_T^{(i)})$ – центр *i*-го кластера.

Той метод кластеризації у якого буде найбільше значення критерію буде вважатися найбільш вдалим для даного набору даних.

2.1.3. Зовнішні критерії якості

У випадку, коли відома істинна кластерна структура $\tilde{G} = \{\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_K\}$, можна застосовувати зовнішні критерії для оцінки якості результатів:

- Індекс Ренда [4]: $R(G, \tilde{G}) = \dfrac{SS + DD}{SS + SD + DS + DD} \to \max$,
- Індекс Жаккарда [4]: $J(G, \tilde{G}) = \dfrac{SS}{SS + SD + DS} \to \max$,
- Індекс Фолка-Меллоу [4]: $FM(G, \tilde{G}) = \sqrt{\dfrac{SS}{SS + SD} \cdot \dfrac{SS}{SS + DS}} \to \max$,

де SS – кількість пар об’єктів, що належать одному і тому ж кластеру як у G, так і у $\tilde{G}$, SD – кількість пар об’єктів, що належать одному кластеру у G та різним кластерам у $\tilde{G}$, DS – кількість пар об’єктів, що належать різним кластерам у G та одному кластеру у $\tilde{G}$, DD – кількість пар об’єктів, що належать різним кластерам як у G, так і у $\tilde{G}$.

2.2. Ансамблеві алгоритми кластеризації даних

Оскільки не існує ні універсального методу кластеризації, ні однозначного способу оцінки якості отриманих результатів, завжди є ризик, що отримане розбиття не буде відповідати істинній кластерній структурі досліджуваних даних. Кожен алгоритм має свої переваги та недоліки. Один і той же метод, що обробив різні набори даних, може повернути результати, що матимуть різній індекс якості. Тобто будь-які початкові дані не можна оброблювати будь-яким методом. На результати може впливати структура даних, уявлення про схожість об’єктів, вибір мір близькості. Виникає питання, як отримати розбиття, що максимально відповідає структурі даних. Для вирішення цього питання можна побудувати ансамбль алгоритмів на основі деякого набору розбиттів. Ансамблеві методи розроблені для покращення надійності та точності алгоритмів кластеризації, а також для можливості обробки даних зі складною кластерною структурою.

У загальному вигляді ідея використання ансамблю алгоритмів кластеризації має наступний вигляд:

1. Отримання окремих розбиттів (різними методами або при різних значеннях параметрів)
2. Групування результатів
3. Визначення узагальненого розв’язку.

2.2.1. Алгоритм на основі побудови узагальненої матриці подібності

Для застосування цього методу попередньо отримуємо набір чітких розбиттів, на основі якого буде побудовано узагальнений ансамблевий розв'язок. У разі нечітких методів слід застосувати дефазифікацію.

Алгоритм методу [11, 12]:

1. Створюємо матрицю $S = \{s_{ij}\}$, $i,j = 1, \ldots, N$ та ініціалізуємо всі її елементи нулями: $s_{ij} = 0$, $i,j = 1, \ldots, N$
2. По черзі розглядаємо розбиття із набору кластеризацій $G_t : t = 1, \ldots, T$. Якщо у t-му розбитті i-й та j-й об'єкти належать одному кластеру, то до значення s_{ij} додаємо одиницю: $s_{ij} = s_{ij} + 1$
3. Зводимо всі елементи матриці S до одиничної шкали: $s_{ij} = \frac{s_{ij}}{T}$, $i, j = 1, \ldots, N$. Тепер всі елементи матриці S мають значення від 0 до 1
4. Перетворюємо елементи матриці S за формулою: $s_{ij} = 1 - s_{ij}$, $i, j = 1, \ldots, N$
5. Отримуємо узагальнюючий розв'язок. Щоб отримати фінальне розбиття $G' = \{g_1, \ldots, g_K'\}$ потрібно до матриці S застосувати методи кластеризації, що приймають матрицю відстаней у якості вхідних даних. Це можуть бути як методи чіткої кластеризації, наприклад ієрархічні, так і нечіткі, наприклад Даве-Сена.

2.2.2. Алгоритм Педрича

Цей ансамблевий алгоритм агрегує результати нечітких розбиттів отриманих попередньо різними методами або при різних значеннях параметрів.

Алгоритм методу [2, 4]:

1. Нехай $b = 1$, $l = 1$

2. Порівнюємо l-й рядок матриці розбиття P_{c*n}^{b} з рядками матриці P_{c*n}^{b+1} та шукається такий індекс l_0, $l_0 = 1, \ldots, c$, що відстань між l-м рядком матриці P_{c*n}^{b} та l_0-м рядком P_{c*n}^{b+1} є мінімальною.
3. Попередній крок виконується для усіх l, $l = 1, \ldots, c$, що дозволяє побудувати функцію i = i(l), яка показує відповідність між рядками матриць розбиття P_{c*n}^{b} та P_{c*n}^{b+1}
4. Обчислюються середні значення функцій приналежності відповідних кластерів на основі яких будується узагальнена матриця розбиття
5. Якщо b < K, то до b додаємо одиницю: b = b + 1 та переходимо на крок 2, виконується заміщення матриці P_{c*n}^{b} обчисленою середньою матрицею розбиття. При b = K процес зупиняємо.

2.2.3. Модифікований алгоритм Педрича

Модифікований алгоритм Педрича будує середню матрицю з урахуванням критерію якості розбиття, тобто більш якісне розбиття сильніше вплине на результат середньої матриці. Такий підхід дозволяє отримати більш якісний розв'язок, ніж у випадку з класичним алгоритмом Педрича, хоча він і потребує додаткових обчислень.

Алгоритм методу [2, 4]:

1. Нехай b = 1, $l = 1$
2. Порівнюємо l-й рядок матриці розбиття P_{c*n}^{b} з рядками матриці P_{c*n}^{b+1} та шукається такий індекс l_0, $l_0 = 1, \ldots, c$, що відстань між l-м рядком матриці P_{c*n}^{b} та l_0-м рядком P_{c*n}^{b+1} є мінімальною.
3. Попередній крок виконується для усіх l, $l = 1, \ldots, c$, що дозволяє побудувати функцію $i = i(l)$, яка показує відповідність між рядками матриць розбиття P_{c*n}^{b} *та* P_{c*n}^{b+1}
4. Розраховується значення критерію якості для обох розбиттів: R(P_{c*n}^{b}) *та* R(P_{c*n}^{b+1})

5. Обчислюється значення функції приналежності відповідних кластерів за формулою $\frac{(l\ *\mathrm{R}(P_{c*n}^{b})+\mathrm{i}(l)*\mathrm{R}(P_{c*n}^{b+1}))}{P_{c*n}^{b}+P_{c*n}^{b+1}}$ на основі яких будується узагальнена матриця розбиття
6. Якщо b < K, то до b додаємо одиницю: b = b + 1 та переходимо на крок 2, виконується заміщення матриці P_{c*n}^{b} обчисленою середньою матрицею розбиття. При b = K процес зупиняємо.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ

3.1. Опис класів розробленого програмного забезпечення

Для розробки програми була обрана мова програмування Python. У якості середи розробки – PyCharm. Для створення графічного інтерфейсу були використані бібліотеки PyQT5, які дозволили створити більш візуально приємний інтерфейс, ніж у випадку з використанням бібліотек TKinter.

Програмне забезпечення містить наступні класи:

1. Клас main – основний клас програми.
2. Клас GustaffsonKessel – клас, що містить методи для реалізації алгоритму Густаффсона-Кесселя.
3. Клас KMeans – клас, що містить методи для реалізації алгоритму KMeans.
4. Клас IerarhicAnsemble – клас, що оброблює та візуалізує результати роботи ансамблевого алгоритму на основі матриці схожості ієрархічним методом.
5. Клас DaveSenaAnsemble – клас, що оброблює та візуалізує результати роботи ансамблевого алгоритму на основі матриці схожості методом Даве-Сена.
6. Клас Ierarhic – клас, що містить методи для реалізації ієрархічного алгоритму.
7. Клас GUI – клас, що відповідає за побудову графічного інтерфейсу.
8. Клас DaveSena – клас, що містить методи для реалізації алгоритму Даве-Сена.

9. Клас CMeans – клас, що містить методи для реалізації алгоритму CMeans.

Розглянемо більш детально вміст кожного з класів:

Клас CMeans містить у собі наступні методи:

def __init__(self, data, countOfClusters, epsilon) – конструктор класу.

def startRandom(self, data, countOfClusters) – метод для заповнення матриці розбиття випадковими даними.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод, що разраховує центри кластерів.

def countDistanceBitweenDataAndCenters(self, data, centersOfClusters, countOfClusters) – метод, що знаходить відстань від об’єкта до центра кластера.

def recountMatrix(self, matrixOldValue, data, distanceBitweenDataAndCenters, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

Клас GustaffsonKessel містить методи:

def __init__(self, countOfClusters, data, epsilon) – конструктор класу.

def startRandom(self, data, countOfClusters) – метод, що заповнює матрицю розбиття випадковими даними.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод для разрахунку центрів кластерів.

def recountMatrix(self, matrixOldValue, data, distance, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

def countDistance(self, data, centersOfClusters) – метод, що знаходить відстань між об’єктами.

def multiplyMatrix(self, firstMatrix, secondMatrix) – метод для знаходження добутку двох матриць.

def covariation(self, matrix, data, centersOfClusters, distance, countOfClusters) – метод, що обчислює матрицю коваріації.

Клас KMeans містить наступні методи:

def __init__(self, data, countOfClusters, epsilon) – конструктор класу.

def countCentersOfClusters(self, data, matrix, centersOfClusters) – метод для разрахунку центрів кластерів.

def countDistance(self, data, centersOfClusters, countOfClusters) – метод, що знаходить відстань між об’єктами.

def recountMatrix(self, matrixOldValue, data, distance, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

Клас DaveSena містить у собі методи:

def __init__(self, data, countOfClusters, epsilon) – конструктор класу.

def recountMatrix(self, matrixOldValue, data, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод для разрахунку центрів кластерів.

def startRandom(self, data, countOfClusters) – метод, що заповнює матрицю розбиття випадковими даними.

def multiplyMatrix(self, firstMatrix, secondMatrix) – метод для знаходження добутку двох матриць.

def countPrototipe(self, data, matrix, oldMatrix) – метод, що перераховує матрицю прототипів.

def countDistance(self, object_1, object_2) – метод, що знаходить відстань між об’єктами.

Клас Ierarhic містить методи:

def __init__(self, data, countOfClusters, alfa_1, alfa_2, beta, hamma) – конструктор класу.

def unity(self, beforeLevelOfTree, cluster_1, cluster_2) – метод для об’єднання двох кластерів.

def findMinDistance(self, distanceBitweenClusters) – метод для пошуку мінімального значеня.

def startValueOfDistanceBitweenClusters(self, data) – метод, що обчислює початкові відстані між кластерами.

def calcDistance(self, object_1, object_2) – метод для підрахунку відстані між об’єктами.

def recountDistanceBitweenClusters(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для підрахунку відстані між кластерами.

def modificationAndRecountMatrix(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для модифікації та перерахунку матриці відстаней.

def countCenterOfCluster(self, objectsOfCluster) – метод, що розраховує центри кластерів.

Клас DaveSenaAnsemble містить у собі наступні методи:

def __init__(self, data, countOfClusters, epsilon, startDistance) – конструктор класу.

def recountMatrix(self, matrixOldValue, data, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод для разрахунку центрів кластерів.

def startRandom(self, data, countOfClusters) – метод, що заповнює матрицю розбиття випадковими даними.

def multiplyMatrix(self, firstMatrix, secondMatrix) – метод для знаходження добутку двох матриць.

def countPrototipe(self, data, matrix, oldMatrix) – метод, що перераховує матрицю прототипів.

def countStartValueOfPrototipe(self, data, matrix, oldMatrix, startDistance) – метод для розрахунку початкових значені матриці прототипів.

def countDistance(self, object_1, object_2) – метод, що знаходить відстань між об’єктами.

Клас IerarhicAnsemble – містить методи:

def __init__(self, data, countOfClusters, alfa_1, alfa_2, beta, hamma, dist_a) – конструктор класу.

def unity(self, beforeLevelOfTree, cluster_1, cluster_2) – метод для об’єднання двох кластерів.

def findMinDistance(self, distanceBitweenClusters) – метод для пошуку мінімального значеня.

def startValueOfDistanceBitweenClusters(self, data) – початкові відстані між кластерами.

def calcDistance(self, object_1, object_2) – метод, що підраховує відстані між об’єктами.

def recountDistanceBitweenClusters(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для підрахунку відстані між кластерами.

def modificationAndRecountMatrix(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для модифікації та перерахунку матриці відстаней.

def countCenterOfCluster(self, objectsOfCluster) – метод, що розраховує центри кластерів.

Клас main містить у собі наступні методи:

def __init__(self) – конструктор класу.

def ansamblePedrich(self) – метод ансамблевої кластеризації на основі методу Педрича.

def ansambleClassic(self) – метод класичної ансамблевої кластеризації.

def choseAnsemble(self) – метод вибору ансамблевої кластеризації.

def ansamble(self) – метод ансамблевої кластеризації.

def saveToFile(self) – метод збереження даних у файл.

def plot(self, X, Y) – метод побудови діаграми розсіювання.

def openFile(self) – метод зчитування даних з файлу.

def openSomeFiles(self) – метод зчитування даних з декількох файлів.

def choseClusterization(self) – метод вибору алгоритму кластеризації.

def DaveSena(self, data, countOfClusters, epsilon) – метод для кластеризації даних алгоритмом Даве-Сена.

def Ierarhic(self, data, countOfClusters) – метод для кластеризації даних ієрархічним алгоритмом.

def dendrogram(self, tree, forDendrogram) – метод для побудови дендрограми.

def indexes(self) – метод розрахунку індексу Ренда, індексу Жаккарда та індексу Фолка-Меллоу.

def CMeans(self, countOfClusters) – метод кластеризації даних алгоритмом CMeans.

def DBSCAN(self, esp, minCount) – метод для кластеризації даних алгоритмом DBSCAN.

def KMeans(self, countOfClusters) – метод кластеризації даних алгоритмом KMeans.

def GustaffsonKessel(self, countOfClusters) – метод для кластеризації даних алгоритмом Густаффона-Кесселя.

def HieBeni(self, data, centersOfClusters, matrix) – метод розрахунку індексу якості Хіє-Бені.

def Rubens(self, matrix) – метод розрахунку індексу якості Рубенса.

def KalinskiyGarabach(self, data, centersOfClusters, labels, countOfClusters) – метод розрахунку індексу якості Калінського-Гарабача.

def BedjacDann(self, labels, countOfClusters, data, centersOfClusters) – метод розрахунку індексу якості Беджека-Данна.

3.2. Функції інтерфейсу користувача

Приклад інтерфейсу зображений на рисунку 3.1.

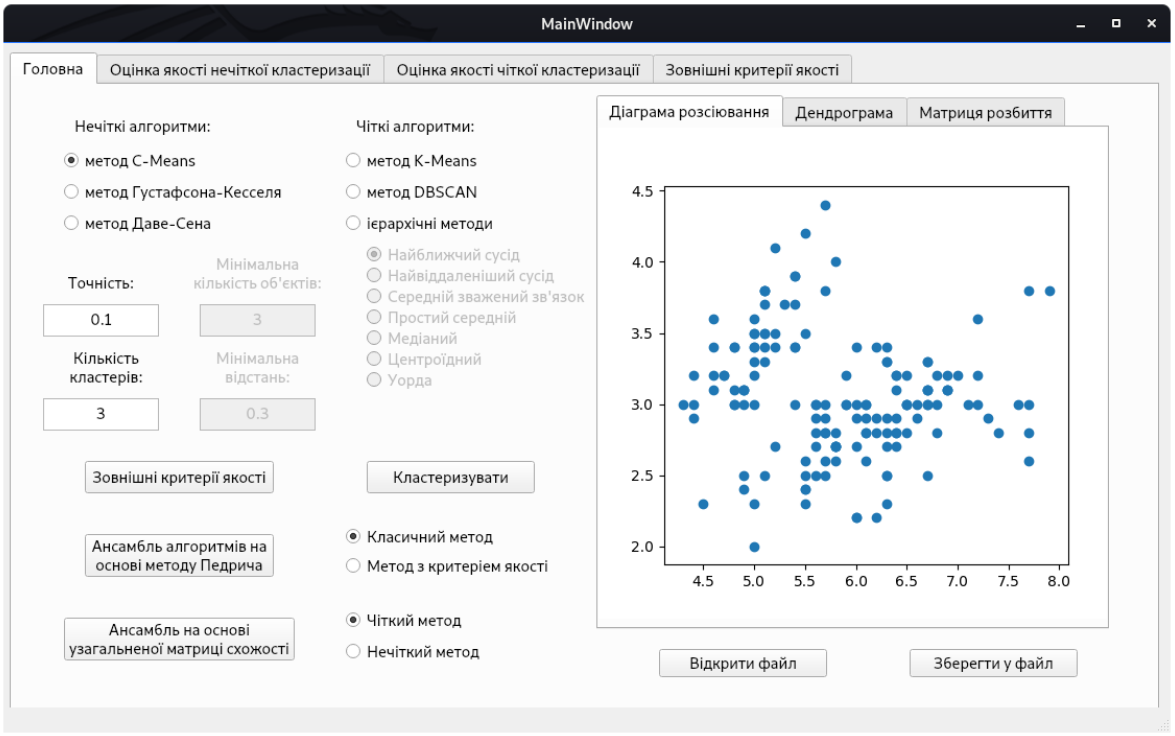


Рис. 3.1. Приклад інтерфейсу програми

Інтерфейс складається з чотирьох вкладинок. На рисунку 3.1 зображена перша вкладника. На ній розміщені наступні елементи керування:

- кнопка «Відкрити файл» – необхідна для зчитування початкових даних з файлу;
- кнопка «Вивід у файл» – виводить кластеризовані дані у файл;
- кнопка «Кластеризувати» – кластеризує дані та виводить їх на графік;
- радіогрупа «Нечіткі алгоритми», яка містить нечіткі алгоритми кластеризації: C-Means, Густафсона-Кесселя та Даве-Сена;
- радіогрупа «Чіткі алгоритми», яка містить чіткі алгоритми кластеризації: DBSCAN, K-Means та ієрархічний метод. Ієрархічний алгоритм має і свою радіогрупу для вибору способу обчислення відстані між кластерами.
- вікно «Точність» – приймає значення необхідної точності;

- вікно «Кількість кластерів» – приймає кількість кластерів;
- вікно «Мінімальна кількість об’єктів» – приймає мінімальну кількість об’єктів для утворення кластеру у методі DBSCAN;
- вікно «Мінімальна відстань» – приймає мінімальну відстань між об’єктами для методу DBSCAN;
- кнопка «Ансамбль алгоритмів на основі методу Педрича» – реалізує ансамбль алгоритмів на основі методу Педрича;
- радіогрупа для вибору різновиду методу Педрича: класичний метод та метод з критерієм якості;
- кнопка «Ансамбль на основі форм матриці схожості» – реалізує ансамбль на основі форм матриці схожості;
- радіогрупа для вибору типу ансамблевого алгоритму на основі матриці схожості: нечіткий (із застосуванням методу Даве-Сена) та чіткий (на базі ієрархічного методу);
- кнопка «Зовнішні критерії якості» – реалізує оцінку якості зовнішніми критеріями.

Також на цій вкладниці знаходиться діаграма розсіювання початкових даних та результатів кластеризації, дендрограма (рисунок 3.2) та таблиця з матрицею результатів нечіткого розбиття (рисунок 3.3).

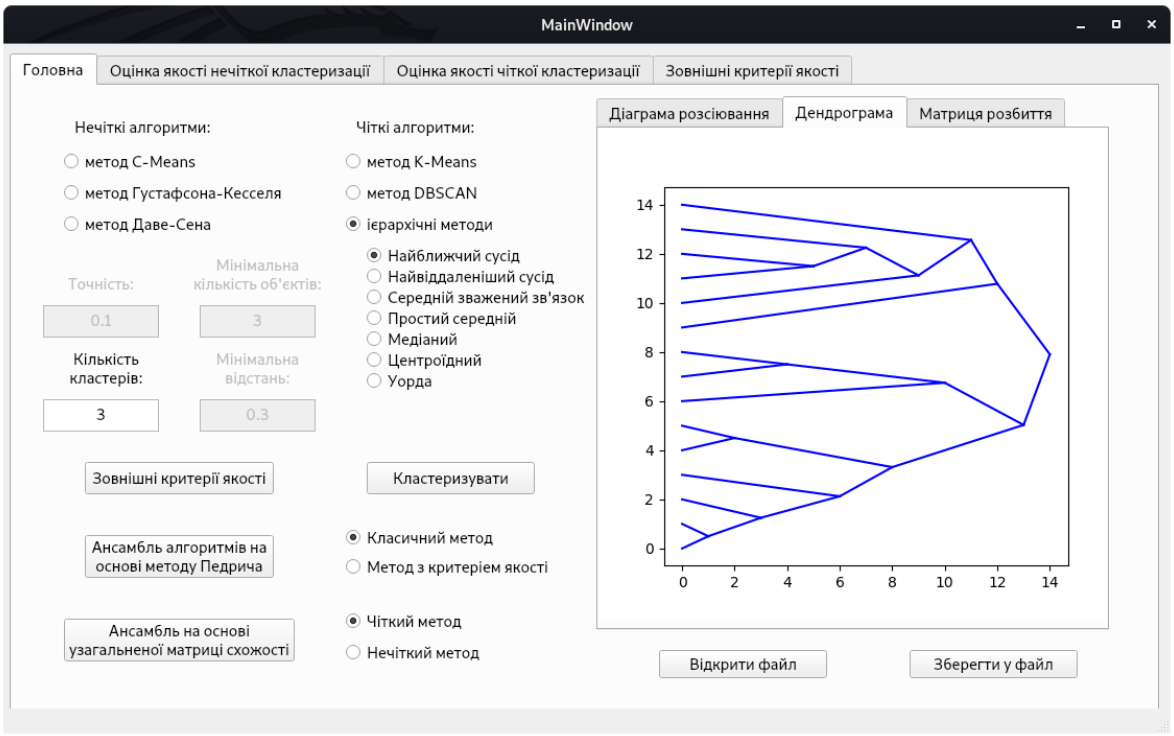
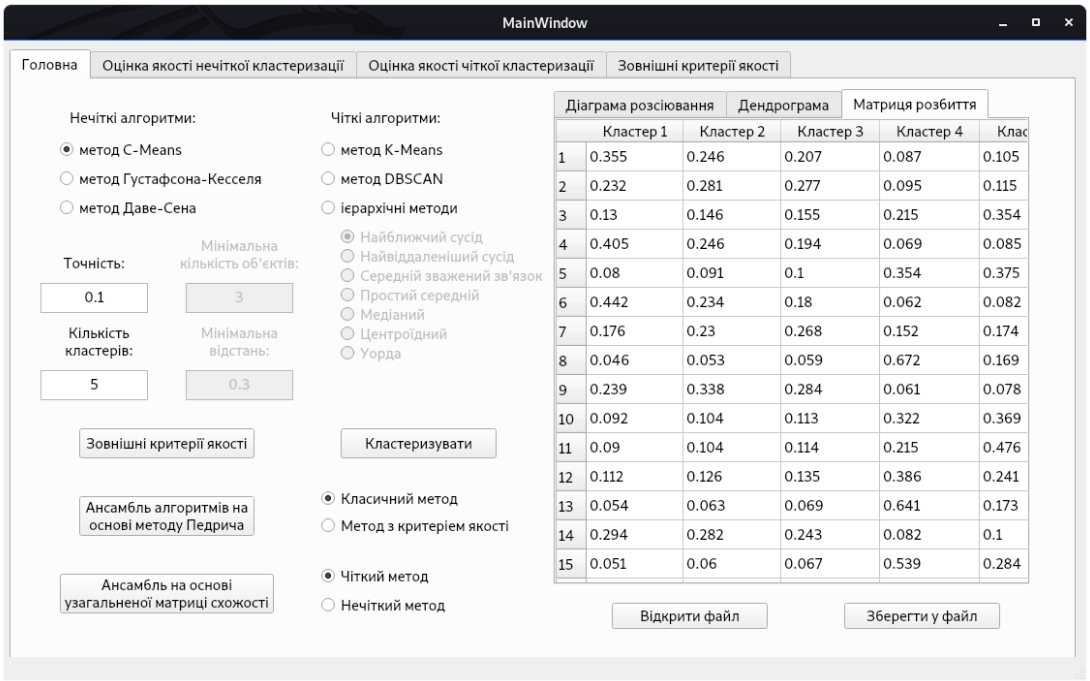


Рис. 3.2. Результат ієрархічного методу у вигляді дендрограми



	Кластер 1	Кластер 2	Кластер 3	Кластер 4	Клас
1	0.355	0.246	0.207	0.087	0.105
2	0.232	0.281	0.277	0.095	0.115
3	0.13	0.146	0.155	0.215	0.354
4	0.405	0.246	0.194	0.069	0.085
5	0.08	0.091	0.1	0.354	0.375
6	0.442	0.234	0.18	0.062	0.082
7	0.176	0.23	0.268	0.152	0.174
8	0.046	0.053	0.059	0.672	0.169
9	0.239	0.338	0.284	0.061	0.078
10	0.092	0.104	0.113	0.322	0.369
11	0.09	0.104	0.114	0.215	0.476
12	0.112	0.126	0.135	0.386	0.241
13	0.054	0.063	0.069	0.641	0.173
14	0.294	0.282	0.243	0.082	0.1
15	0.051	0.06	0.067	0.539	0.284

Рис. 3.3. Результат нечіткого методу у вигляді матриці розбиття

На рисунку 3.4 показано результат роботи програми після кластеризації даних.

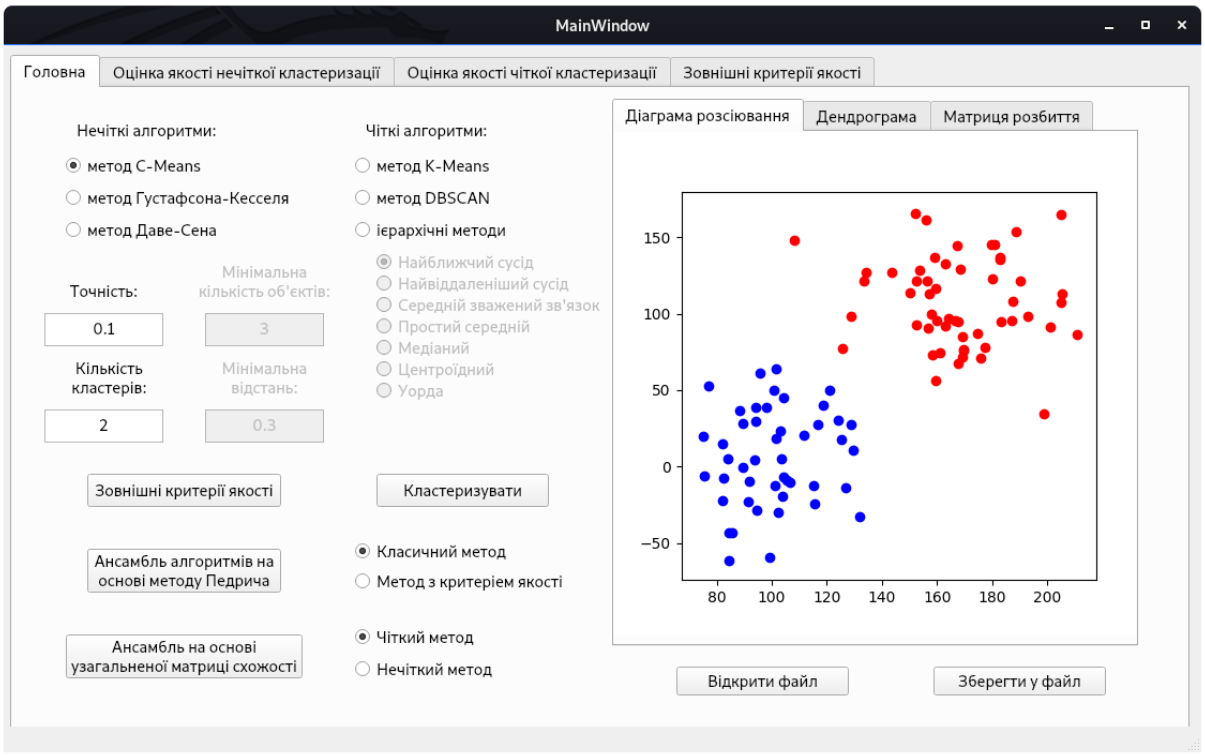
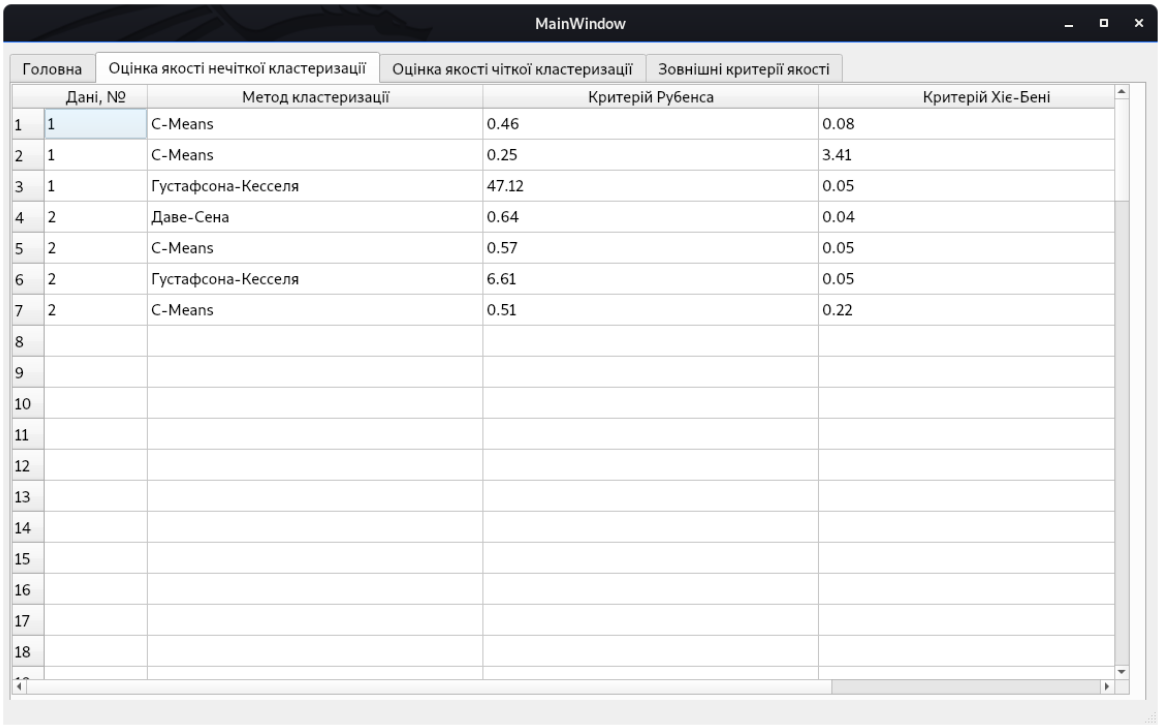


Рис. 3.4. Результат кластеризації даних

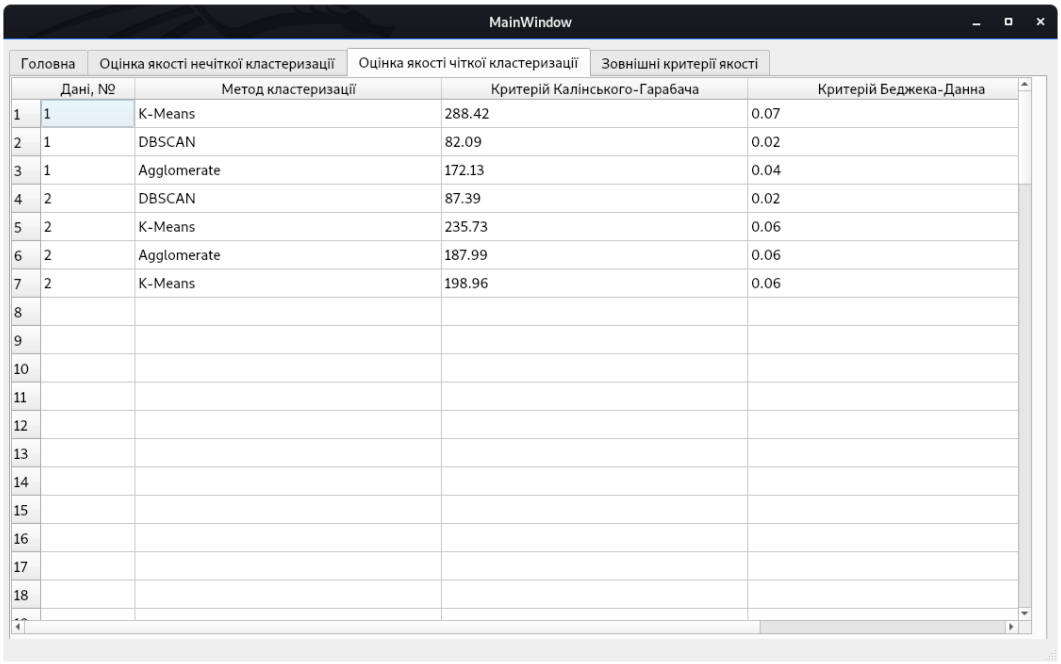
На рисунку 3.5 показана друга вкладинка з оцінками якості результатів роботи нечітких методів за критеріями Рубенса та Хіє-Бені.



	Дані, №	Метод кластеризації	Критерій Рубенса	Критерій Хіє-Бені
1	1	C-Means	0.46	0.08
2	1	C-Means	0.25	3.41
3	1	Густафсона-Кесселя	47.12	0.05
4	2	Даве-Сена	0.64	0.04
5	2	C-Means	0.57	0.05
6	2	Густафсона-Кесселя	6.61	0.05
7	2	C-Means	0.51	0.22
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

Рис. 3.5. Таблиця оцінки якості результатів нечітких методів

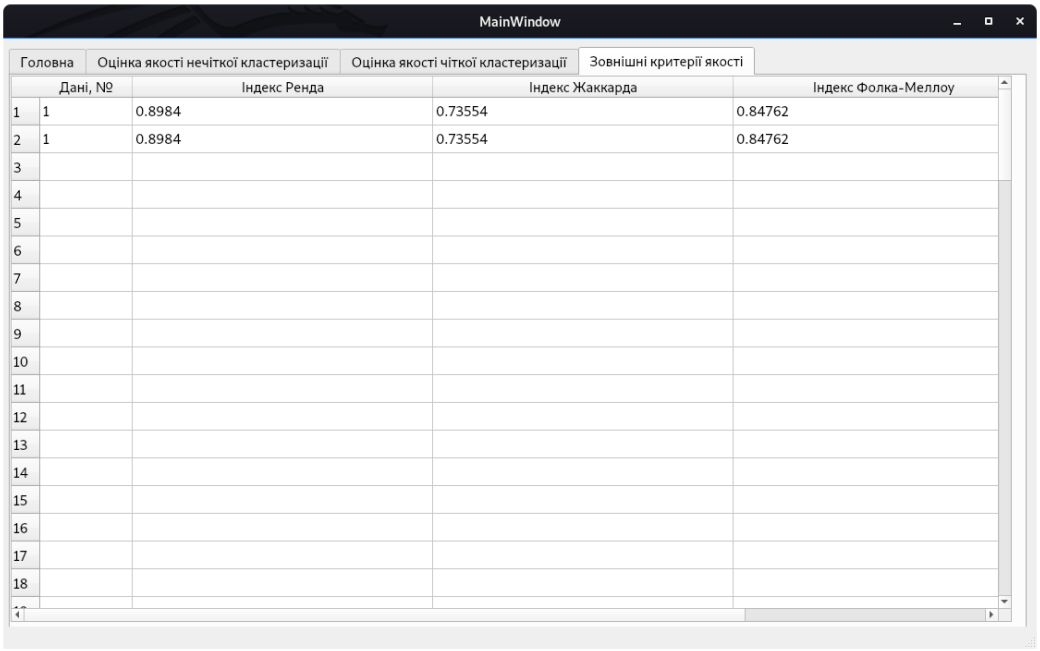
На рисунку 3.6 показана третя вкладинка з оцінками якості результатів чітких методів за критеріями Калінського-Гарабача та Беджека-Данна.



	Дані, №	Метод кластеризації	Критерій Калінського-Гарабача	Критерій Беджека-Данна
1	1	K-Means	288.42	0.07
2	1	DBSCAN	82.09	0.02
3	1	Agglomerate	172.13	0.04
4	2	DBSCAN	87.39	0.02
5	2	K-Means	235.73	0.06
6	2	Agglomerate	187.99	0.06
7	2	K-Means	198.96	0.06
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

Рис. 3.6. Таблиця оцінки якості результатів чітких методів

На рисунку 3.7 показана четверта вкладинка, яка містить таблицю результатів роботи зовнішніх критеріїв якості.



	Дані, №	Індекс Ренда	Індекс Жаккарда	Індекс Фолка-Меллоу
1	1	0.8984	0.73554	0.84762
2	1	0.8984	0.73554	0.84762
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

Рис. 3.7. Таблиця оцінки якості результатів зовнішніми критеріями

3.3. Дослідження якості результатів кластеризації на основі відносних критеріїв.

Розглянемо результати роботи усіх реалізованих методів на наборі даних, представленому на рисунку 3.8

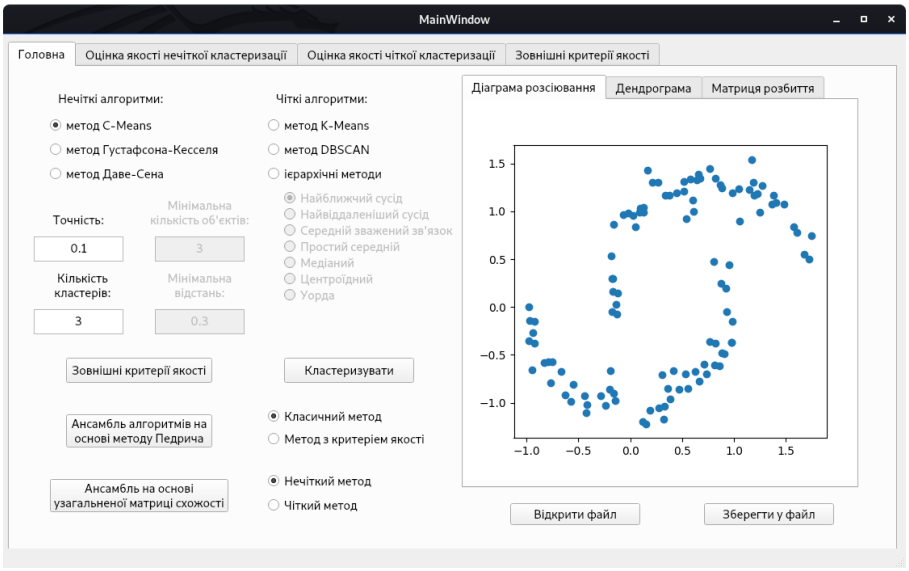


Рис. 3.8. Початковий вигляд даних

Почнемо аналіз з нечітких методів: Даве-Сена, Густаффсона-Кесселя, C-Means. Оберемо кількість кластерів – 3 та точність обчислень – 0,1. Результати роботи методів після дефазифікації зображені у вигляді діаграми розсіювання на рисунках 3.9 – 3.11 відповідно.

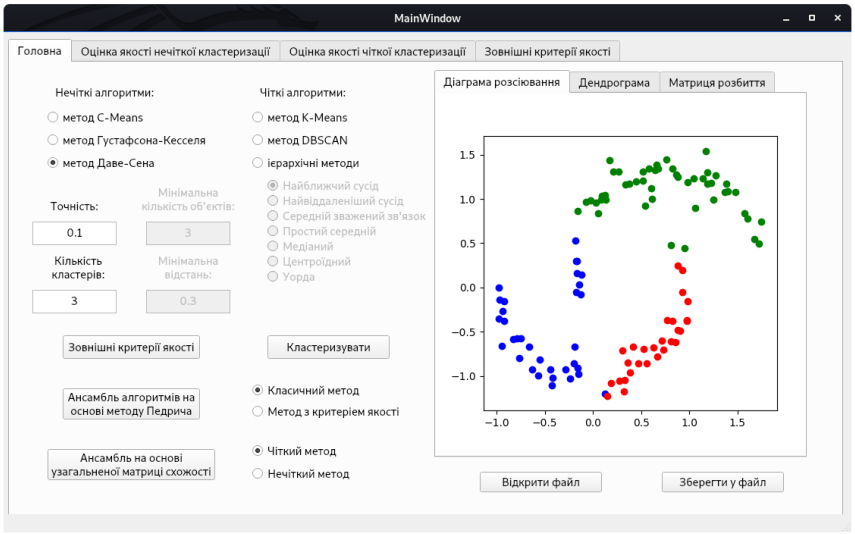


Рис. 3.9. Дефазифіковані результати метода Даве-Сена

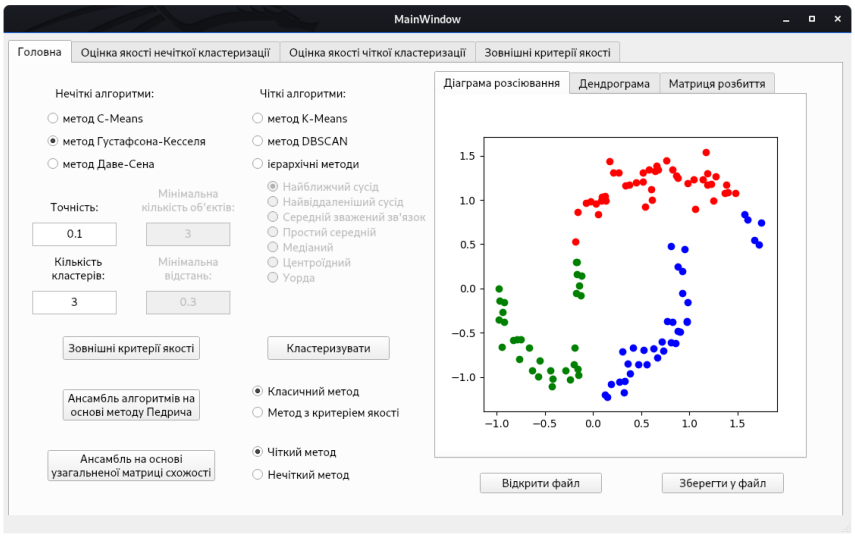


Рис. 3.10. Дефазифіковані результати метода Густаффсона-Кесселя

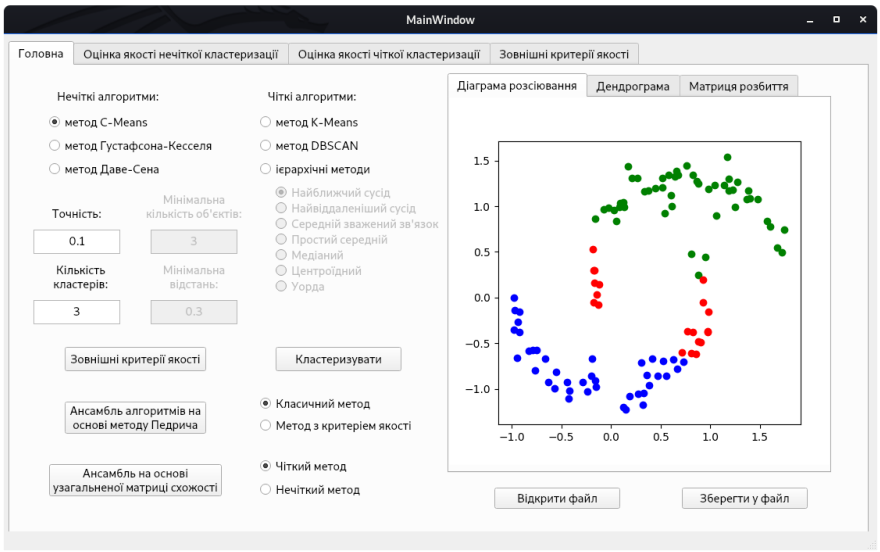
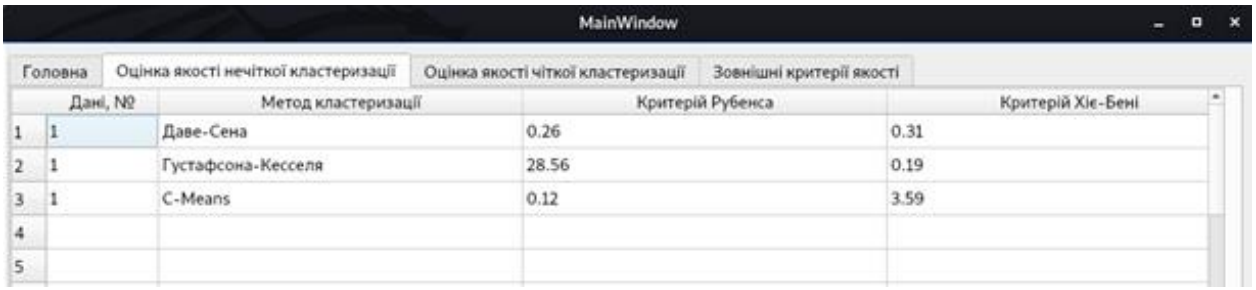


Рис. 3.11. Дефазифіковані результати метода C-Means

На рисунках можна побачити, що алгоритми Даве-Сена, Густаффсона-Кусселя та C-Means розбили вхідні дані по-різному, але жодному з них не вдалося отримати кластерну структуру, що є візуально наглядною. Щоб кількісно оцінити результати роботи кожного алгоритму потрібно використати певний критерій. Для цього на другій вкладниці програми розміщено таблицю з критеріями якості нечітких методів: Рубенса та Хіє-Бені. Цю таблицю можна побачити на рисунку 3.12.

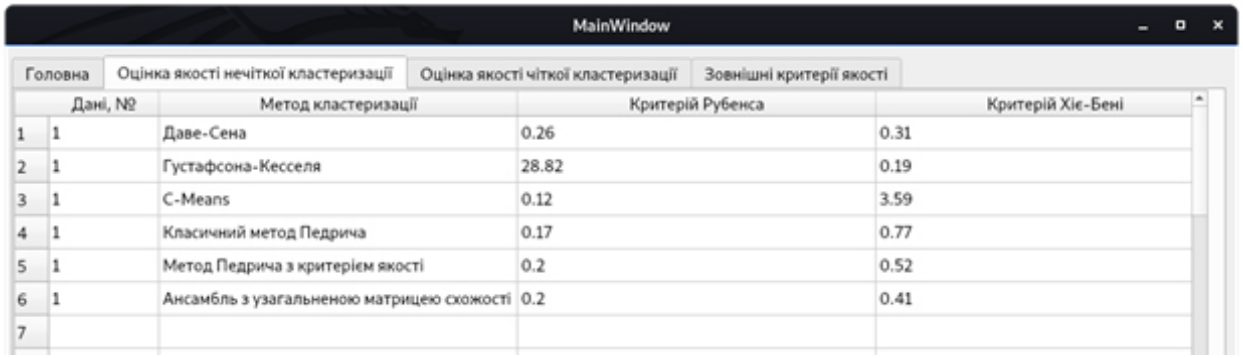


	Дані, №	Метод кластеризації	Критерій Рубенса	Критерій Хіє-Бені
1	1	Даве-Сена	0.26	0.31
2	1	Густафсона-Кесселя	28.56	0.19
3	1	C-Means	0.12	3.59
4				
5				

Рис. 3.12. Таблиця з критеріями якості Рубенса та Хіє-Бені

З таблиці видно, що за обома критеріями метод Густаффсона-Кесселя повернув найбільш якісний результат, а метод C-Means розбив вхідні дані найменш якісно.

Оскільки усі методи кластеризації повернули різні розбиття, то виникає питання, як отримати більш стійкий результат. Для його отримання можна застосувати ансамбль алгоритмів. Для нечітких методів у роботі реалізовано метод Педрича у класичному вигляді та його модифікацію, а також метод на основі матриці подібності, який потребує попередньої дефазифікації результатів індивідуальних розв’язків, що входять у ансамбль, але результат представляє у нечіткому вигляді. Результати наведено на рисунку 3.13.



	Дані, №	Метод кластеризації	Критерій Рубенса	Критерій Хіє-Бені
1	1	Даве-Сена	0.26	0.31
2	1	Густафсона-Кесселя	28.82	0.19
3	1	C-Means	0.12	3.59
4	1	Класичний метод Педрича	0.17	0.77
5	1	Метод Педрича з критерієм якості	0.2	0.52
6	1	Ансамбль з узагальненою матрицею схожості	0.2	0.41
7				

Рис. 3.13. Порівняльний аналіз ансамблевих методів

У таблиці результатів можна побачити, що ансамбль алгоритмів на основі матриці схожості за обома критеріями якості кращий ніж класичний метод Педрича. При порівнянні його з модифікованим методом Педрича отримуємо таке ж саме розбиття за критерієм Рубенса та більш якісне за критерієм Хіє-Бені. Модифікований алгоритм Педрича має більш високі оцінки критеріїв якості, ніж його класичний варіант. Отже, можна зробити висновок, що більш раціональним

є використання саме модифікованого методу. На рисунках 3.14 –.3.16 наведено діаграми розсіювання дефазифікованих результатів розглянутих ансамблевих методів.

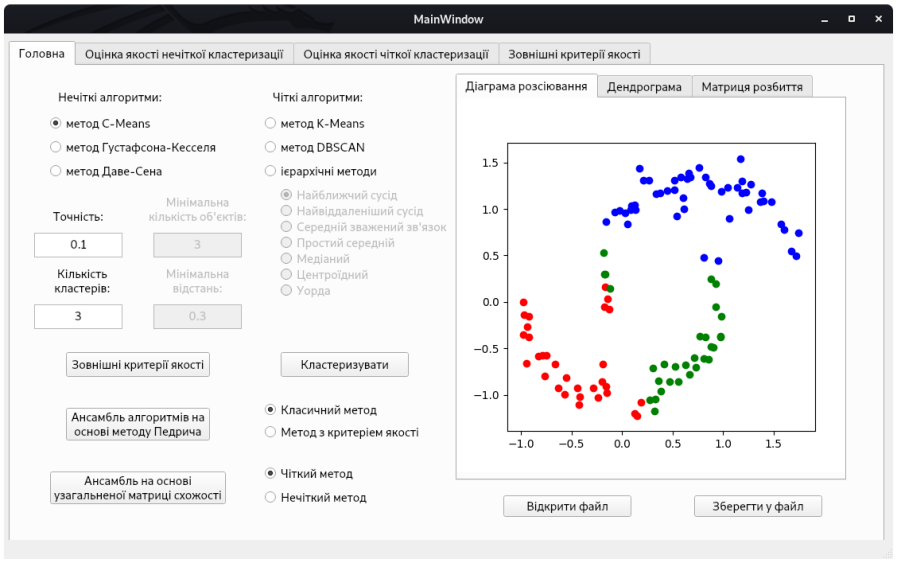


Рис. 3.14. Результат класичного ансамблевого метода Педрича

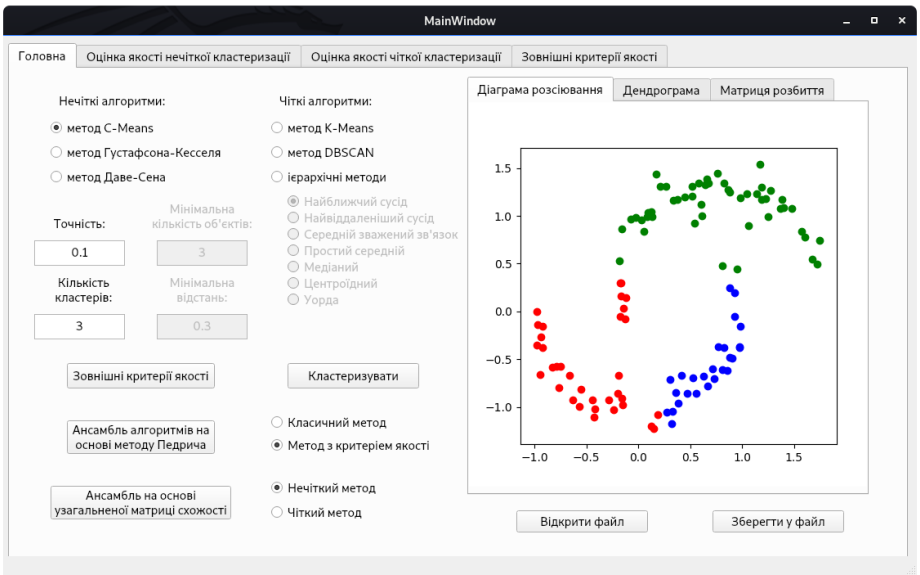


Рис. 3.15. Результат модифікованого ансамблевого метода Педрича

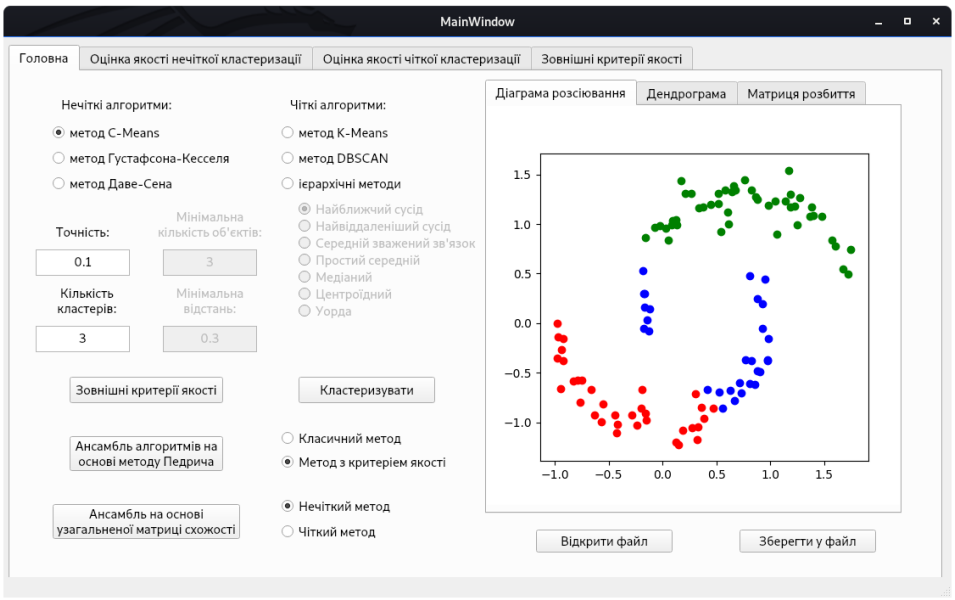


Рис. 3.16. Результат ансамблю алгоритмів на основі матриці схожості

Тепер обробимо вхідні дані за допомогою чітких методів кластеризації. Оберемо алгоритм K-Means, кількість кластерів залишимо рівною трьом, точність обчислень 0,1.

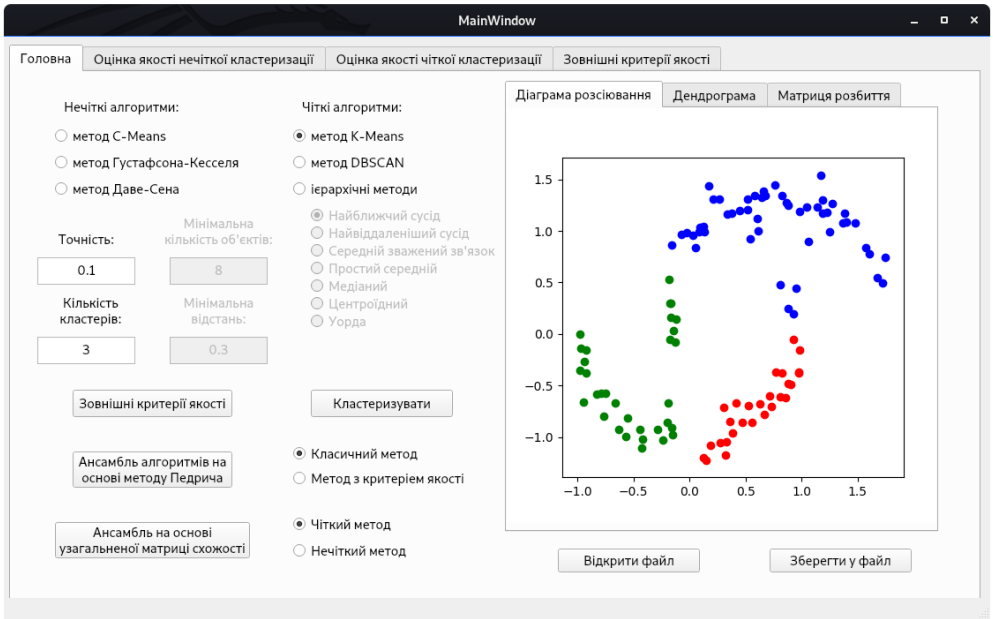


Рис. 3.17. Результат кластеризації алгоритмом K-Means

Застосуємо тепер методи ієрархічної агломеративної кластеризації, які відрізняються між собою способом обчислення відстаней між кластерами. У роботі реалізовано 7 різних варіантів: найближчого сусіда (рис. 3.18),

найвіддаленішого сусіда (рис. 3.19), середнього зваженого зв’язку, простого середнього, медіанного, центроїдного та Уорда (рис. 3.20).

Рис. 3.18. Результат ієрархічної кластеризації найближчого сусіда

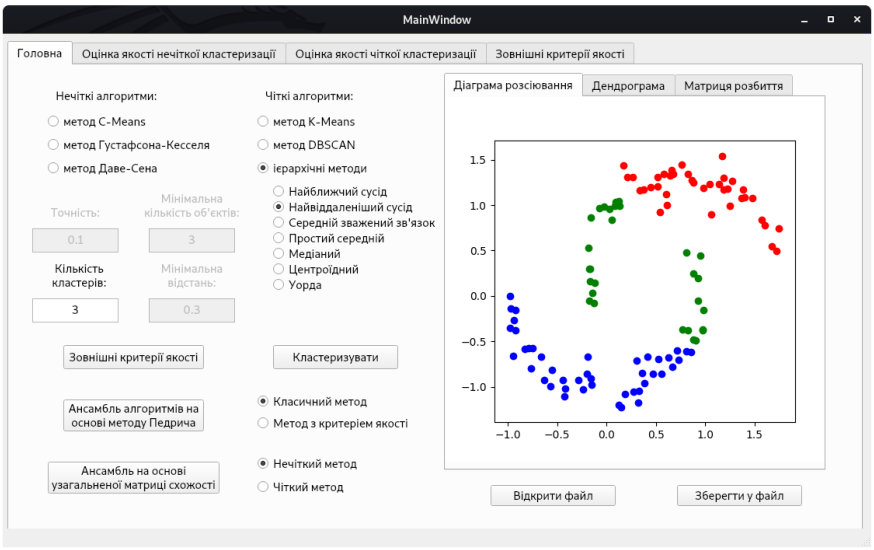


Рис. 3.19. Результат ієрархічної кластеризації найвіддаленішого сусіда

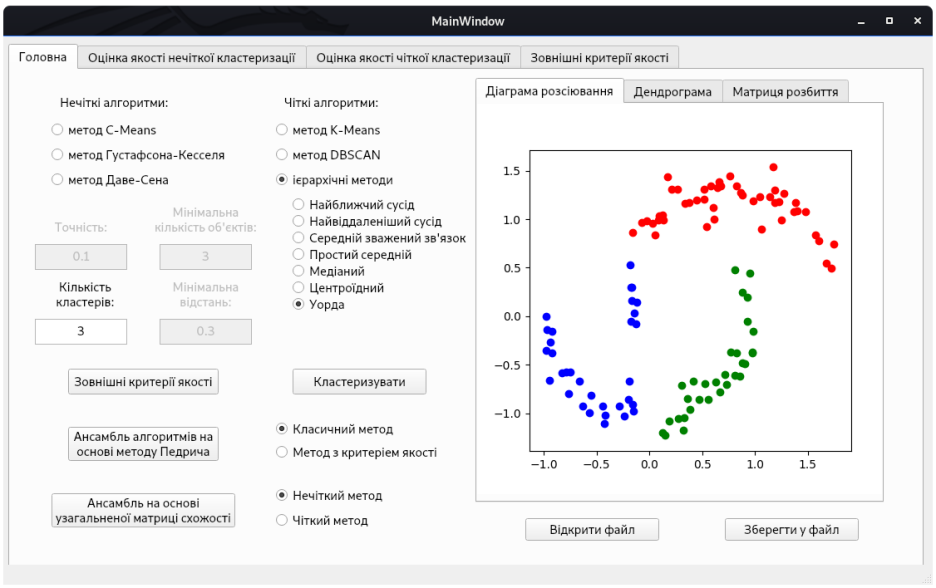


Рис. 3.20. Результат ієрархічної кластеризації Уорда

Тепер оберемо алгоритм DBSCAN. Особливістю цього алгоритму є виділення шуму, тобто об’єкти, що помічені, як шум, не потраплять до жодного кластеру. Для подальшої побудови ансамблю алгоритмів параметри потрібно підібрати таким чином, щоб алгоритм не виділив шум і усі об’єкти потрапили до якогось кластеру. Для цього набору даних такими параметрами будуть: мінімальна кількість об’єктів – 3, мінімальна відстань – 0,33.

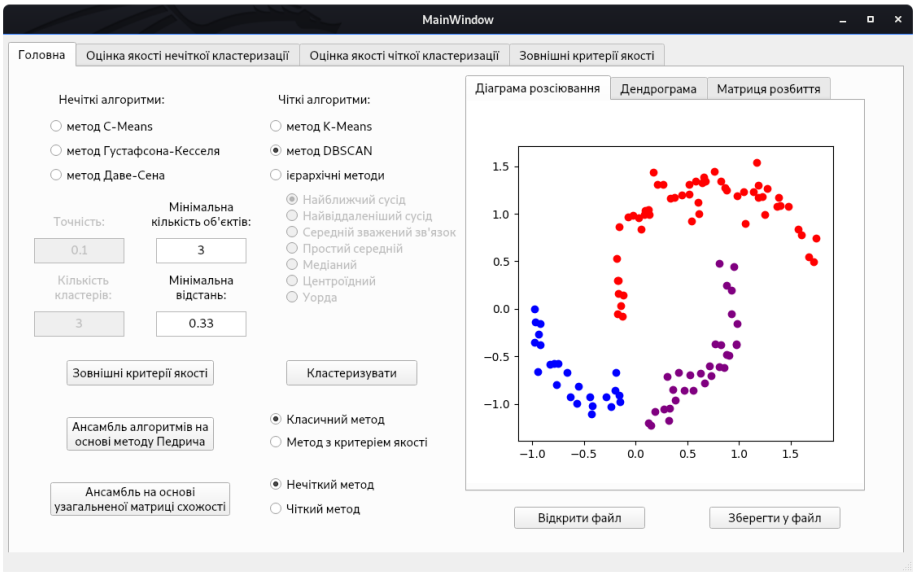
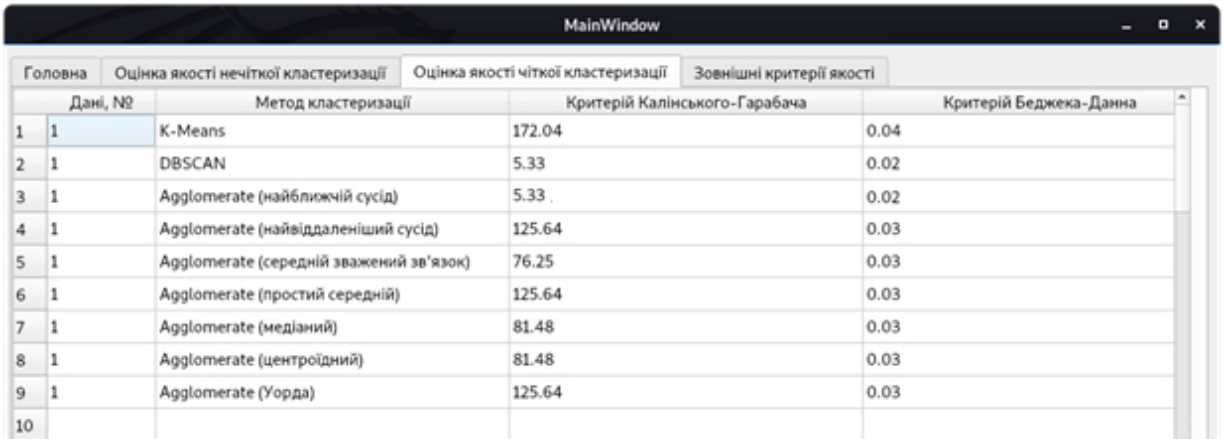


Рис. 3.21. Результат роботи алгоритму DBSCAN

Як і у випадку з нечіткими алгоритмами, знову були отримані різні розбиття. Щоб оцінити якість цих розбиттів застосуємо критеріїв якості чітких методів: Калінського-Гарабача та Беджека-Данна. Вкладинка з таблицею показана на рисунку 3.22.



	Дані, №	Метод кластеризації	Критерій Калінського-Гарабача	Критерій Беджека-Данна
1	1	K-Means	172.04	0.04
2	1	DBSCAN	5.33	0.02
3	1	Agglomerate (найближчій сусід)	5.33	0.02
4	1	Agglomerate (найвіддаленіший сусід)	125.64	0.03
5	1	Agglomerate (середній зважений зв'язок)	76.25	0.03
6	1	Agglomerate (простий середній)	125.64	0.03
7	1	Agglomerate (медіаний)	81.48	0.03
8	1	Agglomerate (центроїдний)	81.48	0.03
9	1	Agglomerate (Уорда)	125.64	0.03
10				

Рис. 3.22. Таблиця з критеріями якості

Із результатів таблиці за обома критеріями можна зробити висновок, що найбільш якісно виконав кластеризацію метод K-Means, а алгоритм DBSCAN та ієрархічний найближчого сусіда відпрацювали найменш якісно на цьому наборі вхідних даних. Проте візуалізація результатів свідчить про протилежне. Така ситуація можлива через те, що критерії найкращим розбиттям вважають таке, яке складається з найбільш компактних кластерів, що найбільш відокремлені один від одного. У цій же ситуації ми маємо справу з ланцюжковими кластерами. Тому можна зробити висновок, що результатам відносних критеріїв не можна завжди довіряти, або слід розглядати значно більшу їх кількість та різноманітність. Це підкреслює актуальність ансамблевого підходу, адже працюючи з реальними багатовимірними даними дуже часто не можна візуально оцінити, яке розбиття є кращим, тому щоб не обрати помилково найгірший результат, краще отримати більш стійкий узагальнений.

3.4. Дослідження якості результатів кластеризації на основі зовнішніх критеріїв

Відносні критерії якості дозволяють оцінювати та порівнювати розбиття отримані різними алгоритмами, проте вони не показують, наскільки результати близькі до істиної кластерної структури, притаманної досліджуваним даним. Щоб порівняти деяке розбиття із еталоном використовують зовнішні критерії якості, які повертають певний індекс, за яким і можна оцінити якість кластеризації відносно істинної структури. У якості таких критеріїв будемо використовувати індекс Ренда, індекс Жаккарда та індекс Фолка-Меллоу.

Розглянемо набір реальних даних «Iris» [15]. Обробимо його почергово всіма алгоритмами, а розбиття передамо зовнішнім критеріям якості. Результати їх роботи показано у таблиці 3.1.

Як вже відомо, усі три критерії повертають максимальне значення для розбиття, що найбільше відповідає істинній структурі даних. У таблиці 3.1 можна побачити, що за індексом Ренда найбільш якісним є розбиття алгоритму Даве-Сена. Найгірший результат, за тим же критерієм, повернув ієрархічний алгоритм середнього зваженого зв’язку. Тепер розглянемо індекс Жаккарда. Знову алгоритм Даве-Сена розбив дані найбільш якісно, на відміну від ієрархічного методу, який використовував алгоритм середнього зваженого зв’язку. Перейдемо до індексу Фолка-Меллоу. Найбільший індекс знову у алгоритму Даве-Сена, а найменший знову у ієрархічного методу, що отримував відстань на основі алгоритму середнього зваженого зв’язку. Можна зробити висновок, що результати роботи методу Даве-Сена найбільше відповідають істинній кластерній структурі.

Таблиця 3.1

Оцінка якості результатів методів на даних «Iris»

Алгоритм кластеризації	Індекс Ренда	Індекс Жаккарда	Індекс Фолка-Меллоу
C-Means	0.8984	0.73554	0.84762
Густаффсона-Кесселя	0.85253	0.64246	0.78241
Даве-Сена	0.89947	0.73868	0.84971
K-Means	0.88053	0.70093	0.82432
Ієрархічний (найближчій сусід)	0.83333	0.60106	0.75083
Ієрархічний (найвіддаленіший сусід)	0.82898	0. 59237	0. 74401
Ієрархічний (середній зважений зв’язок)	0.78098	0.53684	0.70103
Ієрархічний (простий середній)	0.83333	0. 60106	0. 75083
Ієрархічний (медіанний)	0.83333	0. 60106	0. 75083
Ієрархічний (центроїдний)	0. 83333	0. 60106	0. 75083
Ієрархічний (Уорда)	0.82898	0.59237	0.74401

Тепер побудуємо ансамблі алгоритмів на основі матриці схожості. Також застосуємо ансамблевий метод Педрича та його модифікацію з урахуванням критерію якості. Оскільки серед усіх методів кластеризації ієрархічний алгоритм середнього зваженого зв’язку отримав найнижчі індекси якості, то для побудови ансамблю його використовувати не будемо. Серед нечітких методів найменший індекс якості має метод Густаффсона-Кесселя, тому не будемо його додавати до ансамблів на основі методу Педрича. Результати роботи ансамблів наведено у таблиці 3.2.

Таблиця 3.2

Оцінка якості ансамблевих методів на даних «Iris»

Ансамбль алгоритмів	Індекс Ренда	індекс Жаккарда	індекс Фолка-Меллоу
Класичний метод Педрича	0.91298	0.7691	0.86948
Модифікований метод Педрича	0.91298	0.7691	0.86948
На основі матриці схожості (нечіткий)	0.91298	0.7691	0.86948
На основі матриці схожості (чіткий)	0.83333	0.60106	0.75083

У таблиці 3.2 видно, що обидва ансамблі на основі методу Педрича та ансамбль на основі матриці схожості, результати якого були оброблені алгоритмом Даве-Сена, повернули одні і ті ж самі значення. Всі три індекси є більшими, ніж індекси відповідних критеріїв у таблиці 3.1. Щодо другого варіанту ансамблю, який базується на основі матриці схожості, то він має гірші

індекси за усіма трьома критеріями, але навіть він розбив дані краще ніж найбільш неякісний метод, що входив до цього ансамблю. Можна зробити висновок, що відкидання найменш якісного розбиття сприяє побудові більш якісного ансамблю.

Оберемо інший набір даних, що має назву «Seeds» [16]. Кластеризуємо його всіма алгоритмами, як і в попередньому випадку, а результати роботи передамо до зовнішніх критеріїв якості. Індекси якості показано у таблиці 3.3.

Таблиця 3.3

Оцінка якості результатів методів на даних «Seeds»

Алгоритм кластеризації	Індекс Ренда	індекс Жаккарда	індекс Фолка-Меллоу
C-Means	0.87696	0.69086	0.81721
Густаффсона-Кесселя	0.73347	0.45954	0.63142
Даве-Сена	0.87016	0.67503	0.806
K-Means	0.87497	0.68535	0.81331
Ієрархічний (найближчій сусід)	0.40644	0.30963	0.5179
Ієрархічний (найвіддаленіший сусід)	0.4302	0.32196	0. 53144
Ієрархічний (середній зважений зв’язок)	0.4302	0. 32196	0. 53144
Ієрархічний (простий середній)	0.4302	0. 32196	0. 53144

Продовження таблиці 3.3

Алгоритм кластеризації	Індекс Ренда	індекс Жаккарда	індекс Фолка-Меллоу
Ієрархічний (медіанний)	0.4302	0. 32196	0. 53144
Ієрархічний (центроїдний)	0.4302	0. 32196	0. 53144
Ієрархічний (Уорда)	0.4302	0. 32196	0. 53144

Із результатів таблиці 3.3 видно, що метод C-Means отримав найбільший індекс за критерієм Ренда. Найменшу оцінку отримав ієрархічний алгоритм найближчого сусіда. C-Means також найкраще розбив дані і за індексом Жаккарда, вищеназваний варіант ієрархічного методу отримав найнижче значення і за цим критерієм. Щодо індексу Фолка-Меллоу, то тут тенденція зберігається і C-Means та ієрархічний метод мають вище та нижче значення відповідно.

Побудуємо ансамбль алгоритмів і для цього набору даних. Серед нечітких методів алгоритм Густаффсона-Кесселя отримав найнижчий індекс, тому у побудові ансамблю його використовувати не будемо. У випадку з ансамблем на основі матриці схожості, відкинемо одну із модифікацій ієрархічного алгоритму, що відраховує відстань між кластерами методом найближчого сусіда. Результати роботи ансамблю можна побачити у таблиці 3.4.

Таблиця 3.4

Оцінка якості ансамблевих методів на даних «Seeds»

Ансамбль алгоритмів	Індекс Ренда	індекс Жаккарда	індекс Фолка-Меллоу
Класичний метод Педрича	0.87197	0.67979	0.80939
Модифікований метод Педрича	0.87678	0.68966	0.81635
На основі матриці схожості (нечіткий)	0.84054	0.61907	0.76482
На основі матриці схожості (чіткий)	0.4302	0.32196	0.53144

За даними таблиці 3.4 можна помітити, що метод Педрича з урахуванням критерію якості має вищий індекс одразу за трьома зовнішніми критеріями порівнюючи із його класичною реалізацією. Тепер розглянемо ансамбль на основі матриці подібностей із обробкою результатів алгоритмом Даве-Сена. Видно, що усі три індекси набули приблизно середнє значення усіх відповідних індексів із таблиці 3.3. Тобто, результат роботи ансамблю алгоритмів хоч і має менший індекс, ніж найкраще розбиття, але він набагато кращий за найгірше. У випадку, якщо результати вищеназваного алгоритму оброблює ієрархічний метод, то розбиття стає набагато гіршим. Але якщо порівняти ці індекси якості із відповідними індексами у алгоритмів, що входили у ансамбль, то можна побачити, що ці значення мають одразу 6 методів, тобто більшість усіх алгоритмів. Саме тому ансамбль і прийняв їх значення. У будь-якому разі ансамбль на основі матриці схожості, що оброблює результати за допомогою

методу Даве-Сена, повернув якісніше розбиття, ніж той його різновид, що використовує ієрархічний алгоритм. Тепер візьмемо третій набір даних, який називається «Glass Identification Database» [17]. Індекси якості розбиттів показано у таблиці 3.5.

Таблиця 3.5

Оцінка якості результатів методів на даних «Glass»

Алгоритм кластеризації	Індекс Ренда	Індекс Жаккарда	Індекс Фолка-Меллоу
C-Means	0.66547	0.3165	0.48907
Густаффсона-Кесселя	0.49113	0.30845	0.52889
Даве-Сена	0.63757	0.32031	0.50141
K-Means	0.57704	0.34419	0.55684
Ієрархічний (найближчій сусід)	0.28837	0.26905	0.51776
Ієрархічний (найвіддаленіший сусід)	0.28837	0. 26905	0. 51776
Ієрархічний (середній зважений зв’язок)	0.5138	0. 3175	0. 53643
Ієрархічний (простий середній)	0. 28837	0. 26905	0. 51776
Ієрархічний (медіанний)	0. 28837	0. 26905	0. 51776

Продовження таблиці 3.5

Алгоритм кластеризації	Індекс Ренда	Індекс Жаккарда	Індекс Фолка-Меллоу
Ієрархічний (центроїдний)	0. 28837	0. 26905	0. 51776
Ієрархічний (Уорда)	0. 28837	0. 26905	0. 51776

Спочатку розглянемо індекс Ренда. На таблиці 3.5 можна побачити, що найвище значення отримав метод C-Means, а найнижче – одразу 6 методів, а саме усі модифікації ієрархічного алгоритму, окрім того, що розраховує відстань між кластерами за методом середнього зваженого зв’язку. Проте у випадку із індексом Жаккарда найбільший бал отримав алгоритм K-Means, а найгірший результат знову отримали вже названі варіанти ієрархічного методу. Алгоритм K-Means отримав кращий результат і за індексом Фолка-Меллоу, а найнижчий індекс має метод C-Means, хоча за індексом Ренда його розбиття є найкращим.

Для цього набору розбиттів також побудуємо ансамбль методів кластеризацій. За результатами роботи зовнішніх критеріїв якості на нечітких алгоритмах не можна виділити кращий та гірший. Метод, що отримав найнижчій бал за одним критерієм, отримав найвищий індекс за іншим. Тому додамо до ансамблю одразу усі нечіткі методи. Ансамбль на основі матриці схожості буде містити усі нечіткі методи, алгоритм K-Means та ієрархічний метод, який обчислює відстань на основі алгоритму середнього зваженого зв’язку. Результати показано у таблиці 3.6.

Таблиця 3.6

Оцінка якості ансамблевих методів на даних «Glass»

Ансамбль алгоритмів	Індекс Ренда	Індекс Жаккарда	Індекс Фолка-Меллоу
Класичний метод Педрича	0.67023	0.31985	0.49248
Модифікований метод Педрича	0.66547	0.3165	0.48907
На основі матриці схожості (нечіткий)	0.67447	0.28724	0.44875
На основі матриці схожості (чіткий)	0.56669	0.32508	0.53071

Розглянемо індекси зображені у таблиці 3.6. Видно, що класичний метод Педрича має більші індекси, ніж його варіант, що враховував критерії якості. Можна зробити висновок, що не завжди класичний варіант алгоритму Педрича є гіршим його модифікованого алгоритму. Хоча на інших наборах даних були отримані протилежні результати. Ансамблевий алгоритм на основі матриці схожості, що оброблює результати методом Даве-Сена, отримав найгірші результати за індексами Жаккарда та Фолка-Меллоу, але отримав найвищий бал за індексом Ренда. У будь-якому разі цей ансамбль розбив вхідні дані більш якісно, якщо порівнювати його із початковими розбиттями. Тепер розглянемо цей же ансамбль, але з побудовою кластерів ієрархічним методом. У таблиці можна побачити, що він створив більш краще розбиття, ніж його інший варіант, одразу за двома критеріями, а саме за індексами Жаккарда та Фолка-Меллоу. Хоча із алгоритмів, що входили у ансамбль є ті, що розбили дані більш якісно,

ніж сам ансамбль, то все одно індекси зовнішніх критеріїв якості ансамблю є кращими, ніж відповідні значення цих критеріїв у більшості методів. Із цього можна зробити висновок, що використання ансамблів дозволяє, як мінімум запобігти отриманню найгіршого розбиття, якщо не отримати найкраще.

Розглянемо набір даних «Wine Recognition Data» [18]. Розіб'ємо його на кластери різними алгоритмами. Індекси якості зовнішніх критеріїв показано у таблиці 3.7.

Таблиця 3.7

Оцінка якості результатів методів на даних «Wine»

Алгоритм кластеризації	Індекс Ренда	Індекс Жаккарда	Індекс Фолка-Меллоу
C-Means	0.71733	0.41685	0.58843
Густаффсона-Кесселя	0.43366	0.31091	0.50964
Даве-Сена	0.72428	0.42314	0.59466
K-Means	0.72024	0.41883	0.59039
Ієрархічний (найближчій сусід)	0.62827	0.42969	0.62368
Ієрархічний (найвіддаленіший сусід)	0. 62827	0. 42969	0. 62368
Ієрархічний (середній зважений зв'язок)	0. 62827	0. 42969	0. 62368

Продовження таблиці 3.7

Алгоритм кластеризації	Індекс Ренда	Індекс Жаккарда	Індекс Фолка-Меллоу
Ієрархічний (простий середній)	0. 62827	0. 42969	0. 62368
Ієрархічний (медіанний)	0. 62827	0. 42969	0. 62368
Ієрархічний (центроїдний)	0. 62827	0. 42969	0. 62368
Ієрархічний (Уорда)	0. 62827	0. 42969	0. 62368

За результатами таблиці 3.7 можна побачити, що алгоритм Густаффсона-Кесселя має найнижчі індекси за всіма зовнішніми критеріями. Одразу усі модифікації ієрархічного методу отримали найвищі бали за індексами Жаккарда та Фолка-Меллоу. У випадку із критерієм Ренда, то краще розбиття має метод Даве-Сена.

На основі отриманої інформації побудуємо ансамбль методів. Алгоритм Густаффсона-Кесселя відкидаємо, через низькі індекси якості. До ансамблю на основі методу Педрича будуть додані алгоритми Даве-Сена та C-Means, а до ансамблю на основі матриці схожості додамо, ще і усі варіанти ієрархічного методу та алгоритм K-Means. Результати роботи ансамблів кластеризації можна побачити у таблиці 3.8.

Таблиця 3.8

Оцінка якості ансамблевих методів на даних «Wine»

Ансамбль алгоритмів	індекс Ренда	індекс Жаккарда	індекс Фолка-Меллоу
Класичний метод Педрича	0.71885	0.41625	0.58782
Модифікований метод Педрича	0. 71885	0. 41625	0. 58782
На основі матриці схожості (нечіткий)	0.73173	0.43371	0.60503
На основі матриці схожості (чіткий)	0.62827	0.42969	0.62368

За результатами таблиці видно, що обидва варіанти методу Педрича повернули однакові індекси за усіма критеріями, тобто урахування відносного критерію якості не завжди дає більш кращий результат. У випадку із ансамблем на основі матриці схожості, що використовує метод Даве-Сена, були отримані більш вищі бали, які навіть вищі, ніж ті, що були у розбиттів до входження у ансамбль. Варіант цього ансамблю, що кластеризує дані ієрархічним алгоритмом має більш кращі результати тільки за індексом Фолка-Меллоу. У будь-кому разі можна скласти висновок, що використовуючи ансамбль вдалося отримати більш якісне розбиття.

РОЗДІЛ 4
ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Розрахунок трудомісткості і вартості розробки програмного продукту

Вхідні дані:

- ✓ передбачуване число операторів – 2800
- ✓ коефіцієнт складності програми – 1,6
- ✓ коефіцієнт корекції програми в ході її розробки – 0,07
- ✓ часовая заработная плата программиста, грн/г – 120,0

У процесі створення ПЗ нормування праці ускладнено в силу творчого характеру праці програміста. Тому, трудомісткість розробки ПЗ розраховується на основі системи моделей з різною точністю оцінки.

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ чол.-г} \qquad (4.1)$$

де t_u – затрати праці на дослідження алгоритму розв'язання задачі, чол.-г;

t_a – затрати праці на розробку блок-схеми алгоритму, чол.-г;

t_n – затрати праці на програмування по готовій блок–схемі, чол.-г;

$t_{отл}$ – затрати праці на налагодження програми на ЕОМ, чол.-г;

$t_д$ – затрати праці на підготовку документації по завданню, чол.-г.

Ці затрати праці визначаються через умовне число операторів при розробці ПЗ, в число яких входять ті оператори , які необхідно написати в процесі роботи над програмою з урахуванням можливих уточнень у постановці завдання і вдосконалення алгоритму .

Умовне число операторів в програмі обчислюється за формулою:

$$Q = qC(1 + p), \qquad (4.2)$$

де q - передбачуване число операторів q = 2100 ;

c – коефіцієнт складності програми c = 1,6;

p − коефіцієнт кореляції програми в ході ее розробки p = 0,07.

Q = 2800 * 1,6 (1+ 0,07) = 3595

Затрати праці на вивчення опису завдання $t_и$ визначається з урахуванням уточнення опису та кваліфікації програміста.

$$t_u = \frac{QB}{(75..85)K} = \frac{3595*1,3}{77*1,2} = 50,58 \quad \text{чол.-год.,} \qquad (4.3)$$

де В - коефіцієнт збільшення затрат праці внаслідок недостатнього опису завдання:

$$B = 1,2. . . . 1,5;$$

К – коефіцієнт кваліфікації програміста, який визначається залежно від стажу роботи за даною спеціальністю. Він становить при стажі роботи, роки:

до 2 – 0,8;

від 2 до 3 – 1,0;

від 3 до 5 - 1,1 . . . 1,2;

від 5 до 7 - 1,3 . . . 1,4;

вище 7 – 1,5 . . . 1,6.

Затрати праці на розробку алгоритма для рішення задачі:

$$t_a = \frac{Q}{(20\ldots25)K} = \frac{3595}{22*1,2} = 136,17 \quad \text{чол.-год.} \qquad (4.4)$$

Витрати на складання програми по готовій блок -схемі:

$$t_n = \frac{Q}{(20..25)K} = \frac{3595}{22*1,2} = 136,17 \quad \text{чол.-год.} \qquad (4.5)$$

Затрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4..5)K} = \frac{3595}{4*1,2} = 748,96 \text{ чол.-год.} \qquad (4.6)$$

$$t_{отл}^{K} = 1,5t_{отл} = 1,5*748,96 = 1123,44 \text{ чол.-год.} \qquad (4.7)$$

Витрати на підготовку документації:

$$t_д = t_{др} + t_{до} \quad \text{чол.-год.,} \qquad (4.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів:

$$t_{др} = \frac{Q}{(15 \dots 20)K} = \frac{3595}{17 * 1,2} = 206,61 \text{ чол.-год.} \quad (4.9)$$

$t_{до}$ – трудомісткість редагування, друку та оформлення документації:

$$t_{до} = 0,75 t_{др} = 0,75 * 206,61 = 154,96 \text{ чол.-год.} \quad (4.10)$$

$t_{д} = t_{др} + t_{до} = 206,61 + 154,96 = 361,57$ чол.-год.

У підсумку отримуємо, що трудомісткість розробки ПЗ становить:

t = 50,58+136,17+136,17+1123,44+361,57 = 1807,93 чол.-год.

4.2 Затрати на створення програмного забезпечення

Витрати на створення ПО (Кпо) включають витрати на заробітну плату виконавців програми (Зз/п), визначену множенням сумарної трудомісткості розробки ПО (t) на середню зарплату з нарахуваннями програміста і вартості машинного часу, необхідного для відладки програми на ЕОМ (Змв), визначеною виходячи з вартості 1-го машинного години конкретного типу ЕОМ, і витрат машинного часу на налагодження.

$$K_{ПО} = З_{ЗП} + З_{МВ} \text{ , грн.} \quad (4.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * C_{ПР} = 1807,93 * 120,0 = 216951,6 \text{ грн.}, \quad (4.12)$$

де t - загальна трудомісткість, чол.-г.;

$C_{пр}$ - середня годинна заробітна плата програміста, грн./год;

$C_{пр}$ =50,0 грн./год.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} * C_{мч} = 1123,44 * 1,2 = 1348,13 \text{ грн.}, \quad (4.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.;

$C_{мч}$ – вартість машинного часу ЕОМ, грн./год.

Кпо = 216951,6 + 1348,13 = 218299,73 грн. (4.14)

Визначені таким чином витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУТП . Очікуваний період розробки ПЗ:

$$T = \frac{t}{B_k \bullet F_p} \text{ міс.,} \qquad (4.15)$$

де B_k – кількість виконавців;

F_p – місячний фонд робочого часу (при 40-ка годинному робочому тиждні F_p=176 годин).

$$T = \frac{1807,93}{1 * 176} = 10 \text{ місяців.}$$

Таким чином, період розробки програми складе приблизно 10 місяців.

методу.

4.3 Маркетингові дослідження ринку

Кластерний аналіз даних є актуальною задачею, що знайшла застосування у багатьох сферах діяльності людини. У комп'ютерних науках кластеризація використовується для розпізнавання об'єктів, у пошукових системах – для групування сайтів за змістом, що дає більш релевантну відповідь на запит користувача, у медицині – для групування хворих зі схожим діагнозом у групи для подальшого лікування. У біології кластеризація дозволяє розбити великі набори генів на підгрупи, що полегшує аналіз їх взаємозв'язків. Також кластеризація широко використовується у психології, соціології, антропології, маркетингу, економіці тощо.

Нечітка кластеризація розглядає приналежність об’єкта одразу багатьом кластерам, що покращує рівень гнучкості необхідний для дослідження невизначеності, що присутня реальним даним різноманітних прикладних областей.

Ансамблеві методи розроблені для покращення надійності та точності алгоритмів кластеризації, а також для можливості обробки даних зі складною кластерною структурою. Застосування ансамблів алгоритмів у кластерному аналізі є досить актуальним напрямом досліджень, оскільки на основі даного підходу може бути вирішено багато задач, таких як підвищення точності та стійкості результатів, зменшення простору ознак, кластеризація різнотипних даних, розпаралелювання обчислень та ін. Ансамблі нечіткої кластеризації дозволяють поєднати гнучкість теорії нечіткої логіки та надійність ансамблевої агрегації.

Для тестування розробленого програмного забезпечення, порівняльного аналізу методів кластеризації та дослідження ансамблевих алгоритмів було проведено аналіз різних наборів штучних та реальних даних з різноманітною кластерною структурою. Якість отриманих результатів оцінювалася відносними та зовнішніми (для даних з відомою кластерною структурою) критеріями.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи створено програмне забезпечення кластерного аналізу даних, ядро якого складають чіткі (K-means, DBSCAN, ієрархічні агломеративні), нечіткі (K-means, Густаффсона-Кесселя, Даве-Сена) та ансамблеві (Педрича класичний та модифікований, на основі матриці подібності у двох варіантах) методи. Реалізовано оцінювання якості отриманих результатів відносними та зовнішніми критеріями. Для візуалізації отриманих результатів було передбачено наявність діаграми розсіювання, дендрограми та таблиць. За допомогою розробленого програмного забезпечення було проведено дослідження впливу ансамблевих методів на стійкість та якість результатів кластеризації різних наборів штучних та реальних даних.

Для тестування розробленого програмного забезпечення, порівняльного аналізу методів кластеризації та дослідження ансамблевих алгоритмів було проведено аналіз різних наборів штучних та реальних даних з різноманітною кластерною структурою. Якість отриманих результатів оцінювалася відносними та зовнішніми (для даних з відомою кластерною структурою) критеріями.

Було розроблено програмне забезпечення кластерного аналізу даних, яке реалізує:

1. Чіткі методи: K-means, DBSCAN, ієрархічні агломеративні.
2. Нечіткі методи: C-means, Густаффсона-Кесселя, Даве-Сена.
3. Ансамблеві алгоритми: Педрича класичний та модифікований, на основі матриці подібності у двох варіантах.
4. Відносні критерії оцінювання якості отриманих результатів: критерії Рубенса та Хіє-Бені для нечітких методів, індекси Калінського-Гарабача та Беджека-Данна для чітких методів.
5. Зовнішні критерії оцінювання: індекси Ренда, Жаккарда та Фолка-Меллоу.

Для візуалізації отриманих результатів було передбачено наявність діаграми розсіювання, дендрограми та таблиць.

Також за допомогою розробленого програмного забезпечення було проведено дослідження впливу ансамблевих методів на стійкість та якість результатів кластеризації різних наборів штучних та реальних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ckerman M. Measures of Clustering Quality: A Working Set of Axioms for Clustering / B.-D. Shai, M. Ackerman // Proceedings of NIPS Conference, 2008. – P. 121–128.
2. Ensemble of Clustering Algorithms for Large Datasets / I. A. Pestunov, V. B. Berikov, E. A. Kulikova, S. A. Rylov // Optoelectronics, Instrumentation and Data Processing. – 2011. – Vol. 47, No. 3. – P. 245–252.
3. Fern X. Z. Solving cluster ensemble problems by bipartite graph partitioning / X. Z. Fern, C. E. Brodley // Proceedings of the 21 st International Conference on Machine Learning, Canada, 2004. – P. 47.
4. Iris Dataset Set. URL: https://archive.ics.uci.edu/ml/datasets/iris (дата звернення: 4.12.2019)
5. Seeds Data Set. URL: https://archive.ics.uci.edu/ml/datasets/seeds (дата звернення: 4.10.2021)
6. Glass Identification Data Set. URL: https://archive.ics.uci.edu/ml/datasets/glass+identification (дата звернення: 4.10.2021)
7. Wine Data Set. URL: https://archive.ics.uci.edu/ml/datasets/wine (дата звернення: 4.10.2021)
8. M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, TagProp: Discriminative metric learning in nearest neighbor models for image auto-annotation, Kyoto, Japan, 2009. – 6с.
9. W. Zhou, H. Li, Q. Tian, Recent Advance in Content-based Image Retrieval: A Literature Survey, Fellow, 2017. – 31с.
10. A. Makadia, V. Pavlovic, S. Kumar, New Baseline for Image Annotation, Piscataway, 2015. – 10с.
11. P. Kingma, J. Lei Ba, Adam: a method for stochastic optimization, Toronto, 2015. – 3с.

12. J. Fu, Y. Rui, Advances in deep learning approaches for image tagging, Cambridge, 2017. – 1c.
13. Y. Gong, Y. Jia, T. K. Leung, Deep Convolutional Ranking for Multilabel Image Annotation, Fellow , 2014. – 31c.
14. Frahim J. Securing the Internet of Things: A Proposed Framework / J. Frahim // Cisco White Paper.- 2015.
15. Aujla GS Data Offloading in 5G-Enabled Software-Defined Vehicular Networks: A Stackelberg Game-Based Approach / GS Aujla // IEEE Commun. Mag.- vol. 55, no. 7.- July 2017.
16. Peng M. Energy-Efficient Resource Assignment and Power Allocation in Heterogeneous Cloud Radio Access Networks / M. Peng // IEEE Transactions on Vehicular Technology.- vol. 64, no. 11.- Nov. 2015.- P. 5275-5287.
17. Gonzales D. Cloud-trust - A Security Assessment Model for Infrastructure as a Service (Iaas) Clouds / D. Gonzales // IEEE Transactions on Cloud Compu-ting.- vol. 5, no. 3.- July-Sept. 1, 2017.- P. 523-536.
18. John W. Research Directions in Network Service Chaining / W. John // 2013 IEEE SDN for Future Networks and Services.- Nov. 2013.- p. 1-7.
19. Automatic speech recognition and speech variability: A review / M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi, C. Wellekens // Speech Communication, Elsevier. – 2007. – №49. – P. 763-786.
20. V.V.R. Vegesna. Prosody modification for speech recognition in emotionally mismatched conditions / V.V.R. Vegesna, K. Gurugubelli, A.K. Vuppala // International Journal of Speech Technology, 3. – 2018. – P. 521-532.
21. Harpreet Kaur. Prosody Modification of its Output Speech Signal / Harpreet Kaur, Parminder Singh // International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE). – 2014. – №5. – P. 1056–1059.

22. Krothapalli Sreenivasa Rao. Real Time Prosody Modification / Krothapalli Sreenivasa Rao // Journal of Signal and Information Processing. – 2010. – №1. – P. 50-62.
23. Fast Prosody Modification using Instants of Significant Excitation / S. R. M. Prasanna, D. Govind, K. S. Rao, B. Yegnanarayana // Speech Prosody. – 2010.
24. Synthesis of Emotional Speech by Prosody Modification of Vowel Segments of Neutral Speech / Md Shah Fahad, Shreya Singh, Shruti Gupta, Akshay Deepak and Abhinav // Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE). – 2019. – P. 49-54
25. D. Joshi. Speech Emotion Recognition: A Review / D. Joshi, M. B. Zalte // IOSR Journal of Electronics and Communication Engineering (IOSR – JECE). – 2013. – №4. – P. 34–37.

Додаток А

Лістинг програми

Для розробки програми була обрана мова програмування Python. У якості середи розробки – PyCharm. Для створення графічного інтерфейсу були використані бібліотеки PyQT5, які дозволили створити більш візуально приємний інтерфейс, ніж у випадку з використанням бібліотек TKinter.

Програмне забезпечення містить наступні класи:

1. Клас main – основний клас програми.
2. Клас GustaffsonKessel – клас, що містить методи для реалізації алгоритму Густаффсона-Кесселя.
3. Клас KMeans – клас, що містить методи для реалізації алгоритму KMeans.
4. Клас IerarhicAnsemble – клас, що оброблює та візуалізує результати роботи ансамблевого алгоритму на основі матриці схожості ієрархічним методом.
5. Клас DaveSenaAnsemble – клас, що оброблює та візуалізує результати роботи ансамблевого алгоритму на основі матриці схожості методом Даве-Сена.
6. Клас Ierarhic – клас, що містить методи для реалізації ієрархічного алгоритму.
7. Клас GUI – клас, що відповідає за побудову графічного інтерфейсу.
8. Клас DaveSena – клас, що містить методи для реалізації алгоритму Даве-Сена.
9. Клас CMeans – клас, що містить методи для реалізації алгоритму CMeans.

Розглянемо більш детально вміст кожного з класів:

Клас CMeans містить у собі наступні методи:

def __init__(self, data, countOfClusters, epsilon) – конструктор класу.

def startRandom(self, data, countOfClusters) – метод для заповнення матриці розбиття випадковими даними.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод, що разраховує центри кластерів.

def countDistanceBitweenDataAndCenters(self, data, centersOfClusters, countOfClusters) – метод, що знаходить відстань від об’єкта до центра кластера.

def recountMatrix(self, matrixOldValue, data, distanceBitweenDataAndCenters, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

Клас GustaffsonKessel містить методи:

def __init__(self, countOfClusters, data, epsilon) – конструктор класу.

def startRandom(self, data, countOfClusters) – метод, що заповнює матрицю розбиття випадковими даними.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод для разрахунку центрів кластерів.

def recountMatrix(self, matrixOldValue, data, distance, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

def countDistance(self, data, centersOfClusters) – метод, що знаходить відстань між об’єктами.

def multiplyMatrix(self, firstMatrix, secondMatrix) – метод для знаходження добутку двох матриць.

def covariation(self, matrix, data, centersOfClusters, distance, countOfClusters) – метод, що обчислює матрицю коваріації.

Клас KMeans містить наступні методи:

def __init__(self, data, countOfClusters, epsilon) – конструктор класу.

def countCentersOfClusters(self, data, matrix, centersOfClusters) – метод для разрахунку центрів кластерів.

def countDistance(self, data, centersOfClusters, countOfClusters) – метод, що знаходить відстань між об’єктами.

def recountMatrix(self, matrixOldValue, data, distance, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

Клас DaveSena містить у собі методи:

def __init__(self, data, countOfClusters, epsilon) – конструктор класу.

def recountMatrix(self, matrixOldValue, data, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод для разрахунку центрів кластерів.

def startRandom(self, data, countOfClusters) – метод, що заповнює матрицю розбиття випадковими даними.

def multiplyMatrix(self, firstMatrix, secondMatrix) – метод для знаходження добутку двох матриць.

def countPrototipe(self, data, matrix, oldMatrix) – метод, що перераховує матрицю прототипів.

def countDistance(self, object_1, object_2) – метод, що знаходить відстань між об’єктами.

Клас Ierarhic містить методи:

def __init__(self, data, countOfClusters, alfa_1, alfa_2, beta, hamma) – конструктор класу.

def unity(self, beforeLevelOfTree, cluster_1, cluster_2) – метод для об’єднання двох кластерів.

def findMinDistance(self, distanceBitweenClusters) – метод для пошуку мінімального значеня.

def startValueOfDistanceBitweenClusters(self, data) – метод, що обчислює початкові відстані між кластерами.

def calcDistance(self, object_1, object_2) – метод для підрахунку відстані між об’єктами.

def recountDistanceBitweenClusters(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для підрахунку відстані між кластерами.

def modificationAndRecountMatrix(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для модифікації та перерахунку матриці відстаней.

def countCenterOfCluster(self, objectsOfCluster) – метод, що розраховує центри кластерів.

Клас DaveSenaAnsemble містить у собі наступні методи:

def __init__(self, data, countOfClusters, epsilon, startDistance) – конструктор класу.

def recountMatrix(self, matrixOldValue, data, countOfClusters, epsilon) – метод для перерахунку матриці розбиття.

def countCentersOfClusters(self, data, matrix, countOfClusters) – метод для разрахунку центрів кластерів.

def startRandom(self, data, countOfClusters) – метод, що заповнює матрицю розбиття випадковими даними.

def multiplyMatrix(self, firstMatrix, secondMatrix) – метод для знаходження добутку двох матриць.

def countPrototipe(self, data, matrix, oldMatrix) – метод, що перераховує матрицю прототипів.

def countStartValueOfPrototipe(self, data, matrix, oldMatrix, startDistance) – метод для розрахунку початкових значені матриці прототипів.

def countDistance(self, object_1, object_2) – метод, що знаходить відстань між об’єктами.

Клас IerarhicAnsemble – містить методи:

def __init__(self, data, countOfClusters, alfa_1, alfa_2, beta, hamma, dist_a) – конструктор класу.

def unity(self, beforeLevelOfTree, cluster_1, cluster_2) – метод для об’єднання двох кластерів.

def findMinDistance(self, distanceBitweenClusters) – метод для пошуку мінімального значеня.

def startValueOfDistanceBitweenClusters(self, data) – початкові відстані між кластерами.

def calcDistance(self, object_1, object_2) – метод, що підраховує відстані між об’єктами.

def recountDistanceBitweenClusters(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для підрахунку відстані між кластерами.

def modificationAndRecountMatrix(self, distanceBitweenClusters, firstCluster, secondCluster) – метод для модифікації та перерахунку матриці відстаней.

def countCenterOfCluster(self, objectsOfCluster) – метод, що розраховує центри кластерів.

Клас main містить у собі наступні методи:

def __init__(self) – конструктор класу.

def ansamblePedrich(self) – метод ансамблевої кластеризації на основі методу Педрича.

def ansambleClassic(self) – метод класичної ансамблевої кластеризації.

def choseAnsemble(self) – метод вибору ансамблевої кластеризації.

def ansamble(self) – метод ансамблевої кластеризації.

def saveToFile(self) – метод збереження даних у файл.

def plot(self, X, Y) – метод побудови діаграми розсіювання.

def openFile(self) – метод зчитування даних з файлу.

def openSomeFiles(self) – метод зчитування даних з декількох файлів.

def choseClusterization(self) – метод вибору алгоритму кластеризації.

def DaveSena(self, data, countOfClusters, epsilon) – метод для кластеризації даних алгоритмом Даве-Сена.

def Ierarhic(self, data, countOfClusters) – метод для кластеризації даних ієрархічним алгоритмом.

def dendrogram(self, tree, forDendrogram) – метод для побудови дендрограми.

def indexes(self) – метод розрахунку індексу Ренда, індексу Жаккарда та індексу Фолка-Меллоу.

def CMeans(self, countOfClusters) – метод кластеризації даних алгоритмом CMeans.

def DBSCAN(self, esp, minCount) – метод для кластеризації даних алгоритмом DBSCAN.

def KMeans(self, countOfClusters) – метод кластеризації даних алгоритмом KMeans.

def GustaffsonKessel(self, countOfClusters) – метод для кластеризації даних алгоритмом Густаффона-Кесселя.

def HieBeni(self, data, centersOfClusters, matrix) – метод розрахунку індексу якості Хіє-Бені.

def Rubens(self, matrix) – метод розрахунку індексу якості Рубенса.

def KalinskiyGarabach(self, data, centersOfClusters, labels, countOfClusters) – метод розрахунку індексу якості Калінського-Гарабача.

def BedjacDann(self, labels, countOfClusters, data, centersOfClusters) – метод розрахунку індексу якості Беджека-Данна.

Додаток Б

ВІДГУК

керівника економічного розділу

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Керівника економічної частини Професора Вагонової О.Г.
(прізвище, ім'я, по батькові, вчене звання)

на магістерську роботу

Студента • ІІ курсу групи 121м-20-1 Міняйло Данила Дмитровича
(прізвище, ім'я, по батькові)

На тему: Розробка та дослідження ансамблевих алгоритмів кластеризації даних.

.

«___»_____2022 р. _____
(підпис)

Додаток Д

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Міняйло.doc	Пояснювальна записка кваліфікаційної роботи. Документ Word.
Міняйло.pdf	Пояснювальна записка кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація.ppt	Презентація роботи