Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

**АНАЛІЗ МЕТОДІВ КЛАСТЕРИЗАЦІЇ**
**НА ОСНОВІ ЩІЛЬНОСТІ РОЗПОДІЛУ ДАНИХ**
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Авлякулов Т.Е.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Авлякулову Тимуру Елбарсовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз методів кластеризації на основі щільності розподілу даних

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, документація використаних бібліотек.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд кластеризації, розбір основних компонентів.
2. Аналіз методів кластеризації.
3. Програмна реалізація методів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність методів кластеризації, постановка задачі, тестові дані.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| Консультант з дотримання діючих стандартів та норм | Доцент Творошенко І.С. | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № з/п | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання на кваліфікаційну роботу | 10.04.2023 | |
| 2 | Аналіз завдання, підбір літератури | 11.04.23-17.04.23 | |
| 3 | Аналіз літератури з досліджуваної проблеми | 18.04.23-20.04.23 | |
| 4 | Аналіз методів | 21.04.23-30.04.23 | |
| 5 | Програмна реалізація методів | 01.05.23-14.05.23 | |
| 6 | Програмна реалізація | 15.05.23-23.05.23 | |
| 7 | Оформлення пояснювальної записки | 24.05.23-26.05.23 | |
| 8 | Перевірка на плагіат | 27.05.23 | |
| 9 | Рецензування | 28.05.23 | |
| 10 | Підготовка презентації та доповіді | 29.05.23-30.05.23 | |
| 11 | Занесення роботи в електронний архів | 31.05.23 | |
| 12 | Попередній захист кваліфікаційної роботи | 06.06.23 | |

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шафроненко А.Ю._____
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 61 с., 1 табл., 13 рис., 31 джерело.

ІЄРАРХІЧНІ МЕТОДИ КЛАСТЕРИЗАЦІЇ, МЕТОДИ НА ОСНОВІ ЦЕНТРІВ, *K*-СЕРЕДНІХ, *K*-МЕДІАН, DBSCAN.

Об’єктом роботи є дослідження методів кластеризації.

Метою роботи є дослідження різних методів кластеризації та їх використання для групування схожих об’єктів у кластери з вихідних даних.

Дослідження методів кластеризації може допомогти вибрати найбільш ефективний метод для конкретного завдання, яке потрібно вирішити. Крім того, дослідження може допомогти зрозуміти, як працюють різні методи, та їх переваги та недоліки.

У результаті роботи здійснена програмна реалізація різних методів кластеризації даних за різними ознаками.

HIERARCHICAL CLUSTERING METHODS, CENTER-BASED METHODS, *K*-MEANS, *K*-MEDIAN, DBSCAN.

The object of the work is the study of various clustering methods.

The purpose of the work is to study various clustering methods and their use to group similar objects into clusters from the source data.

The study of clustering methods can help to choose the most effective method for a specific task to be solved. In addition, research can help to understand how different methods work, and their advantages and disadvantages.

As a result of the work, software implementation of various methods of data clustering according to various characteristics is carried out.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE – Integrated Drive Electronics (інтегроване середовище розробки)

ВВП – внутрішній валовий продукт

ARI – Adjusted Rand Index (скоригований ранд індекс)

AMI – Adaptive Mutual Information (адаптивна взаємоінформація)

# ВСТУП

Кластеризація є важливим інструментом аналізу даних, який дозволяє групувати схожі об’єкти в кластери на основі подібності між ними. Один з підходів до кластеризації полягає в використанні методів на основі щільності розподілу даних. Ці методи засновані на припущенні, що точки, що знаходяться в тих же областях в просторі ознак, належать до одного кластера.

Методи на основі щільності розподілу даних можуть використовувати різні функції щільності, такі як функції Гаусса або Кронекера, для визначення областей високої щільності даних, які можна вважати кластерами. Ці методи можуть допомогти виявити структуру даних, навіть якщо вона не явно виражена, і знайти схожість між об’єктами.

Проте, методи на основі щільності розподілу даних також мають свої недоліки, такі як чутливість до вибірки даних та складнощі з вибором оптимальних значень параметрів. Тому важливо ретельно досліджувати ці методи та їх використання в конкретних завданнях.

У даній роботі будуть розглянуті різні методи кластеризації на основі щільності розподілу даних, їх особливості та ефективність на різних вибірках даних. Також буде розглянуто приклади використання цих методів на реальних даних та їх порівняння з іншими методами кластеризації. Результатом роботи буде вибір найбільш ефективного методу кластеризації на основі щільності розподілу даних для конкретного завдання та його використання в практиці.

# 1 ОГЛЯД КЛАСТЕРИЗАЦІЇ

Кластеризація – це метод машинного навчання, який використовується для групування точок даних за їхніми схожими властивостями. При використанні алгоритму кластеризації, кожна точка даних класифікується у конкретну групу залежно від її подібності до інших точок у наборі. Поняття «подібності» визначається як обернена метрика відстані між точками даних, де коротша відстань вказує на більшу подібність між об’єктами. Існує багато метрик подібності, таких як евклідова відстань, які використовуються для кластеризації даних [1-3].

Існують чотири основні групи методів кластеризації, включаючи техніки оптимізації зверху вниз, знизу вгору та аналітичну оптимізацію, а також ієрархічні методи центроїд ного розподілу, розподілів очікування максимізації та густини. Також можна використовувати жорстку або м’яку кластеризацію, яка відноситься до бінарних або нечітких відношень відповідно. Остання група кластеризації визначає відмінність між перекриттям проти не пересікаючи груп розділів загалом, заснована на природі кластерних відносин [4]. У процесі кластеризації алгоритм намагається знайти групи об'єктів, які подібні між собою і відрізняються від об'єктів інших груп. Кластери можуть бути сформовані засновані на різних критеріях, таких як відстань, схожість або інші метрики, які вимірюють ступінь подібності між об'єктами.

Важливо зазначити, що кластеризація може бути складною задачею, особливо у випадку високої розмірності даних, з якими часто доводиться працювати в більшості реальних застосувань [5]. Стандартні методи кластеризації можуть бути не ефективними у таких випадках.

## 1.1 Класифікація кластеризації за типом

Класифікація кластеризації включає різні підходи до групування даних за схожістю та інших характеристик. В залежності від обраного підходу можна отримати різні кількості та типи кластерів. Кластеризація є важливим інструментом для розв’язання задач аналізу даних, виявлення шаблонів та структури даних, класифікації та візуалізації даних. У залежності від конкретного завдання можна обирати різні методи кластеризації, які можуть відрізнятися за складністю, часом роботи та точністю. Загалом, кластеризацію можна розділити на дві підгрупи [5].

### 1.1.1 Жорстка кластеризація

Жорстка кластеризація – це метод кластеризації, в якому кожен об’єкт даних належить лише до одного кластера, без будь-якої перетину між кластерами. Таким чином, жорстка кластеризація забезпечує чітке приналежність об’єктів до конкретного кластера.

Для жорсткої кластеризації використовують різні алгоритми, які базуються на певних критеріях. Один з найпоширеніших алгоритмів жорсткої кластеризації – *k*-means. Цей алгоритм розподіляє об’єкти даних на *k* кластерів, де *k* – це попередньо встановлене число кластерів. Алгоритм *k*-means робить ітеративні кроки для знаходження центрів кластерів та приналежності об’єктів до кластерів, на основі відстаней між об’єктами та центрами кластерів [6-8].

Однією з переваг жорсткої кластеризації є її простота та швидкість обробки даних. Також перевага жорсткої кластеризації є те, що вона дає чітку інтерпретацію результатів і зручну структуру для подальшого аналізу даних. Крім того, жорстка кластеризація є досить ефективною у випадках, коли

кластери мають чітку межу та значення параметрів точок даних розподілені нормально.

Однак, жорстка кластеризація має свої обмеження. Наприклад, вона не може розподіляти об’єкти, що належать до декількох кластерів, тому що кожен об’єкт має бути призначений лише одному кластеру. Крім того, вибір кількості кластерів перед запуском жорсткої кластеризації може бути складною задачею, а невірно вибраний кількість кластерів може призвести до невірних результатів [9, 10].

### 1.1.2 М’яка кластеризація (нечітка кластеризація)

М’яка кластеризація – це метод групування об’єктів, в якому кожен об’єкт може належати до декількох кластерів з різними ступенями належності. Це відрізняється від жорсткої кластеризації, де кожен об’єкт може належати тільки до одного кластеру. М’яка кластеризація використовується для групування об’єктів, які можуть мати схожі характеристики, але можуть відрізнятися в різних аспектах.

Один з найбільш відомих методів м’якої кластеризації – це Fuzzy C-Means (FCM). В FCM кожен об’єкт може належати до кожного кластеру з різним ступенем належності, який визначається за допомогою функції належності. Функція належності визначає, наскільки даний об’єкт належить до кожного кластеру. FCM використовується для групування даних, які мають суттєву перекриття між кластерами, наприклад, коли деякі об’єкти можуть належати до двох кластерів одночасно.

М’яка кластеризація має свої переваги та недоліки. Перевагою є те, що вона дозволяє використовувати більше інформації з даних та дає можливість краще відображати сутність даних. Однак, вона може бути більш складною в обчисленнях та вимагати більше обчислювальних ресурсів [10-14].

У загальному, м’яка кластеризація є корисним методом групування даних, коли потрібно більш детально розділити об’єкти в кластерах та врахувати можливість перекриття між ними [15].

## 1.2 Щільність розподілу даних

У контексті кластеризації, щільність розподілу даних відображає, які області простору ознак мають більшу концентрацію даних, а які – меншу. Це дає можливість відрізняти один кластер від іншого.

Зазвичай, щільність розподілу даних можна визначити шляхом використання алгоритмів оцінки щільності, таких як ядерна оцінка щільності (Kernel Density Estimation). Ці алгоритми побудовують гладку функцію щільності на основі набору даних, яка дозволяє оцінити ймовірність того, що випадкова точка належить до певного кластера.

Після отримання функції щільності розподілу можна використовувати різні методи для виявлення кластерів в даних, такі як алгоритми знаходження локальних екстремумів (наприклад, алгоритм DBSCAN), або методи візуалізації даних з використанням графіків щільності (наприклад, графіки розсіювання з контурними лініями щільності).

Щільність розподілу даних може бути корисним інструментом для кластеризації великих обсягів даних, де немає можливості використовувати інші методи кластеризації, наприклад, методи, які вимагають розбиття даних на підмножини або побудови матриці відстаней між точками.

Щільність розподілу може мати велике значення при визначенні кластерів та визначенні границь між ними. Загалом, в кластеризації можуть спостерігатися такі типи щільності розподілу:

– однорідна щільність точки даних в кластері розташовані досить близько одна до одної, і відстань між ними невелика. Це означає, що кластер

має високу щільність. В такому випадку, точки в кластері зазвичай належать до одного і того ж самого підкласу або мають схожі властивості;

– розріджена щільність точки даних в кластері розташовані досить далеко одна від одної, і відстань між ними значна. Це означає, що кластер має низьку щільність. Точки в такому кластері можуть представляти різні підкласи або мати різні властивості;

– змінна щільність розподілу може варіюватися в різних областях простору даних. Це означає, що деякі області можуть мати високу щільність, тоді як інші можуть мати низьку щільність. Такі варіації щільності можуть вказувати на наявність різних кластерів або підкласів у даних.

Аналіз щільності розподілу даних може бути корисним для визначення кластерів, вибору оптимальної кількості кластерів або виявлення аномалій у наборі даних. Це може бути досягнуто за допомогою різних методів, таких як оцінка щільності розподілу, графічне візуалізація або використання алгоритмів кластеризації, які враховують щільність розподілу даних при прийнятті рішень про кластеризацію [16].

## 1.3 Характеристики ефективності алгоритмів кластеризації

Характеристики ефективності алгоритмів кластеризації визначають, наскільки добре вони виконують свою основну мету – групування подібних об'єктів разом і розділення різних груп. Існує два типи метрик якості – зовнішні та внутрішні. Зовнішні метрики використовують інформацію про правильне розбиття на кластери, тоді як внутрішні метрики не залежать від зовнішньої інформації та оцінюють якість кластеризації лише на основі набору даних. Зазвичай оптимальну кількість кластерів визначають з використанням внутрішніх метрик [17, 18].

### 1.3.1 Adjusted Rand Index (ARI)

Вважається, що маємо інформацію про правильні мітки об’єктів. Цей показник не залежить від самого значення міток, а лише від способу поділу вибірки на кластери. Припустимо, що $n$ – кількість об’єктів у вибірці. Позначимо $a$ – кількість пар об’єктів з однаковими мітками, що знаходяться в одному кластері, і $b$ – кількість пар об’єктів з різними мітками, що знаходяться в різних кластерах. Тоді Rand Index це

$$RI = \frac{2(a+b)}{n(n-1)}. \quad (1.1)$$

Тобто це частка об’єктів, для яких ці розбиття (вихідне і отримане в результаті кластеризації) «узгоджені». Rand Index (RI) висловлює схожість двох різних кластеризації однієї і тієї ж вибірки [19]. Щоб цей індекс давав значення близькі до нуля для випадкових кластеризації при будь-якому n і числі кластерів, необхідно унормувати його. Так визначається Adjusted Rand Index:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]}. \quad (1.2)$$

Цей захід симетричний, не залежить від значень і перестановок міток. Таким чином, даний індекс є мірою відстані між різними розбивками вибірки. ARI приймає значення в діапазоні [- 1, 1]. Негативні значення відповідають «незалежним» розбиття на кластери, значення, близькі до нуля, – випадковим розбиття, і позитивні значення свідчать про те, що два розбиття схожі (збігаються при ARI = 1).

Плюси:

– інтерпретування нескоригований індекс Ренда пропорційний кількості пар вибірок, мітки яких однакові в обох labels_predi / labels_true або розрізняються в обох;

– якщо випадковим чином присвоюємо мітки об’єктам, то скоригований бал індексу *Rand* буде наближатися до 0,0 незалежно від значень n_clusters та n_samples (що не відноситься, наприклад, до нескоригованого індексу *Rand* або міри *V*;

– обмежений діапазон більш низькі значення вказують на різні маркування, аналогічні кластери мають високий (скоригований або нескоригований) індекс *Rand*, 1,0 – це оцінка ідеального відповідності. Діапазон оцінок становить [0, 1] для нескоректована індексу *Rand* і [-1, 1] для скоригованого індексу *Rand*;

– не робиться ніяких припущень про структуру кластера: (скоригований або нескоригований) індекс *Rand* може використовуватися для порівняння всіх видів алгоритмів кластеризації і може використовуватися для порівняння алгоритмів кластеризації, таких як *k*-середніх, який передбачає ізотропні форми краплі з результатами спектрального аналізу. алгоритми кластеризації, які можуть знайти кластер зі «складеними» формами [20-23].

Мінуси:

– незважаючи на інерцію, для (скоригованого або нескоригованого) індексу Ренда потрібні знання основних класів істинності, що майже ніколи не є доступними на практиці або вимагають ручного призначення анотаторами - людьми (як у випадках контрольованого навчання). Однак (скоригований або нескоригований) індекс Ренда також може бути корисним в суто неконтрольованих налаштуваннях як будівельний блок для консенсусного індексу, що може використовуватися для вибору моделі кластеризації;

– нескоригований індекс *Rand* часто близько до 1,0 навіть якщо кластеризації істотно розрізняються. Це можна зрозуміти, інтерпретуючи індекс Ренда як точність маркування пар елементів, отриману в результаті кластеризації на практиці часто існує більшість пар елементів, яким

присвоюється different мітка пари як при прогнозованій, так і при базовій кластеризації істинності, що призводить до високої частка парних міток, які згодні, що згодом призводить до високої оцінки.

### 1.3.2 Ентропія

Ці заходи формально визначаються за допомогою функцій ентропії та умовної ентропії, які дозволяють розглядати розбиття вибірки як дискретні розподіли:

$$h = 1 - \frac{H(C|K)}{H(C)}\,,\ c = 1 - \frac{H(K|C)}{H(K)}\,, \tag{1.3}$$

де $K$ – результат кластеризації;

$C$ – справжнє розбиття вибірки на класи.

Таким чином, $h$ вимірює, наскільки кожен кластер складається з об’єктів одного класу, а $c$ – наскільки об’єкти одного класу відносяться до одного кластеру. Ці заходи не є симетричними. Обидві величини приймають значення в діапазоні [0, 1] , і великі значення відповідають більш точної кластеризації. Ці заходи не є нормалізованими, як ARI або AMI і тому залежать від числа кластерів [24]. Випадкова кластеризація не даватиме нульові показники при великому числі класів і малому числі об’єктів. У цих випадках краще використовувати ARI. Однак при числі об’єктів більше 1000 і числі кластерів менше 10 дана проблема не так явно виражена і може бути проігнорована. Для обліку обох величин $h$ і $c$ одночасно вводиться $V$ – заходи, як їх середнє гармонійне.

$$v = 2\frac{hc}{h+c}\,. \tag{1.4}$$

Переваги:

– обмежені бали 0 – це настільки погано, наскільки це можливо, 1 – ідеальний бал;

– інтуїтивна інтерпретація: кластеризацію з поганою *V*-мірою можна якісно проаналізувати з точки зору однорідності та повноти, щоб краще зрозуміти, які «помилки» допускаються при завданні;

– не робиться ніяких припущень про структуру кластера може використовуватися для порівняння алгоритмів кластеризації, таких як *k*-середнє, яке передбачає ізотропні форми краплі, з результатами алгоритмів спектральної кластеризації, які можуть знаходити кластер зі «складеними» формами.

Мінуси очевидні. Раніше введені метрики не нормалізовані щодо випадкової маркування: це означає, що в залежності від кількості вибірок, кластерів і основних класів істинності повністю випадкова маркування не завжди буде давати однакові значення для однорідності, повноти і, отже, *v*-заходи [25].

Ентропія вимірює невизначеність випадкової змінної або набору даних. Чим більше ентропія, тим більше невизначеність. Коли всі можливі значення змінної рівномірно розподілені, ентропія досягає максимального значення. На іншому боці, коли всі значення змінної однакові, ентропія досягає мінімального значення.

У контексті кластеризації, ентропія використовується для оцінки якості кластеризації або чистоти кластерів. Чим менше ентропія, тим краще кластеризація, оскільки це свідчить про більшу однорідність кластерів. Іншими словами, чисті кластери мають низьку ентропію, оскільки вони мають меншу невизначеність або змінність.

## 1.4 Постановка задачі

Задача кластеризації – це по факту задача класифікації, бо в обох випадках діляться об’єкти на основі їх подібності між собою, але у випадку кластеризації приналежність навчальних об’єктів будь-яким класам не задається. Така задача – загальна, тому для її розв’язання використовуються різні підходи. Алгоритми побудови кластерів можуть дуже відрізнятися у підходах до того, що відносити в один кластер і як їх взагалі ефективніше шукати.

Кластери можна утворювати ґрунтуючись на відстані між ними, на щільності ділянок у просторі даних, інтервалах або на конкретних статистичних розподілах. Це все залежать від конкретного набору даних та мети використання результатів. Кластерний аналіз не є автоматизованим, це скоріше ітераційний процес, тому що часто доводиться змінювати метод опрацювання даних та параметри моделі, поки не буде отримано з результат з заданими властивостями.

Розв’язок неоднозначний, і на це є кілька причин. По-перше, не існує найкращого критерію якості кластеризації. Відомий цілий ряд досить ефективних критеріїв, а також ряд алгоритмів, які не мають чітко вираженого критерію, але все одно здійснюють досить якісну кластеризацію по побудові. Всі вони можуть давати різні результати.

По-друге, число кластерів, як правило, не відомо заздалегідь і встановлюється відповідно до деякого суб’єктивного критерія. По-третє, результат кластеризації істотно залежить від метрики $\rho$, вибір якої, як правило, також суб’єктивний і визначається спеціалістом.

Формальна постановки задач:

Маємо вибірку $X\ell = \{x1, \ldots, x\ell\} \subset X$ і функцію відстані між об’єктами $\rho(x, x')$. Треба розбити вибірку на підмножини, які не будуть перетинатися і щоб кожен кластер складався з об’єктів, близьких по метриці ρ, а об’єкти

різних кластерів істотно відрізняються. При цьому кожному об'єкту $x \in X$ $\ell$ приписується мітка (номер) кластера $y_i$.

Алгоритм кластеризації – це функція $a$: $X \to Y$, що будь-якому об'єкту $x \in X$ ставить у відповідність мітку кластера $y \in Y$. Множина міток $Y$ в деяких випадках відома заздалегідь, однак частіше завдання полягає в тому, щоб визначити оптимальне число кластерів з точки зору того чи іншого критерію якості кластеризації. Задача групування набору об'єктів полягає в тому, що об'єкти в одному кластері більш схожі один на одного, ніж об'єкти в інших кластерах. Подібність – це буквально кількість, яка собою відображає міцність взаємозв'язку між двома об'єктами. Кластеризація використовується в основному для отримання даних, а також в інших областях, таких як машинне навчання, розпізнавання образів, аналіз зображень, пошук інформації, біоінформатика, стиснення даних і комп'ютерна графіка [5-8].

Рішенням задачі кластерного аналізу є розбиття, що задовольняє деякому критерію оптимальності. Цей критерій може являти собою деякий функціонал, що виражає рівні бажаності різних розбиття і угруповань, який називають цільовою функцією.

Рішення завдання кластеризації принципово неоднозначно, і тому є кілька причин:

– не існує однозначно найкращого критерію якості кластеризації. Відомий цілий ряд евристичних критеріїв, а також ряд алгоритмів, які не мають чітко вираженого критерію, але здійснюють досить розумну кластеризації «з побудови». Всі вони можуть давати різні результати. Отже, для визначення якості кластеризації потрібно експерт предметної області, який би міг оцінити осмисленість виділення кластерів;

– число кластерів, як правило, невідомо заздалегідь і встановлюється відповідно до деякого суб'єктивним критерієм. Це справедливо тільки для методів дискримінації, так як в методах кластеризації виділення кластерів йде за рахунок формалізованого підходу на основі заходів близькості;

– результат кластеризації істотно залежить від метрики, вибір якої, як правило, також суб’єктивний і визначається експертом. Але є ряд рекомендацій щодо вибору заходів близькості для різних завдань.

Один з простих прикладів кластеризації може бути групування фруктів за їхніми ознаками. Наприклад, якщо мається множину фруктів, таких як яблука, груші, банани та апельсини, є можливість застосувати алгоритм кластеризації, щоб розділити їх на підгрупи на основі їхньої подібності.

Одним з можливих підходів може бути визначення подібності між фруктами на основі їх ваги, розміру та кольору шкірки. Наприклад, яблука та груші можуть бути груповані разом, оскільки вони мають подібний розмір та форму, а банани та апельсини можуть бути розміщені в інші групи, оскільки вони мають інші характеристики.

Цей приклад демонструє, як кластеризація може бути використана для групування об’єктів на основі їхніх спільних характеристик, що дозволяє легше розуміти та аналізувати великі множини даних.

Кластеризація використовується у багатьох галузях, таких як машинне навчання, комп’ютерне зорове сприйняття, обробка природних мов, біоінформатика та інші. Наприклад, кластеризація може бути використана для групування користувачів на основі їхнього поведінки на вебсайті, для класифікації зображень на основі їхніх ознак [9].

Задача кластеризації може бути вирішена за допомогою різних алгоритмів, які базуються на різних підходах до оцінки подібності між об’єктами. Вибір конкретного алгоритму залежить від властивостей даних та мети кластеризації.

Об’єктом роботи є дослідження різних методів кластеризації.

Метою роботи є аналіз методів, на основі щільності розподілу даних.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів кластеризації;

– розробити алгоритм методів кластеризації на вибірці даних.

# 2 МЕТОДИ РОЗВ'ЯЗКУ ЗАДАЧІ КЛАСТЕРИЗАЦІЇ

## 2.1 Density-Based Spatial Clustering of Applications with Noise

Алгоритм DBSCAN розглядає кластери як області високої щільності, розділені областями низької щільності. Завдяки цьому досить загальному погляду, кластери, знайдені DBSCAN, можуть мати будь-яку форму, на відміну від *k*-середніх, які припускають, що кластери мають опуклу форму. Центральним компонентом DBSCAN є концепція кернових проб, тобто проб, що знаходяться в зонах високої щільності. Таким чином, кластер – це набір основних зразків, кожен з яких знаходиться близько один до одного (вимірюється деякою мірою відстані), і набір неосновних зразків, які близькі до основного зразка (але самі по собі не є основними зразками). Існує два параметри алгоритму, min_samples і eps, які формально визначають, що мається на увазі, коли говоримо про щільність. Вищі min_samples або менші eps вказують на вищу щільність, необхідну для формування кластера.

Більш формально, визначається базова вибірку як вибірку в наборі даних, що існує min_samples інших зразків на відстані eps, які визначаються як сусіди основної вибірки. Це говорить нам про те, що зразок ядра знаходиться в щільній області векторного простору. Кластер – це набір основних зразків, які можна побудувати шляхом рекурсивного взяття основного зразка, пошуку всіх його сусідів, які є основними зразками, пошуку всіх їхніх сусідів, які є основними зразками, і так далі. Кластер також має набір неосновних зразків, які є зразками, які є сусідами основного зразка в кластері, але самі не є основними зразками. Інтуїтивно зрозуміло, що ці зразки знаходяться на периферії кластера.

Будь-який базовий зразок за визначенням є частиною кластера. Алгоритм вважає будь-який зразок, який не є основним зразком і знаходиться на відстані принаймні eps від будь-якого основного зразка.

У той час як параметр min_samples головним чином контролює, наскільки алгоритм толерантний до шуму (на галасливих і великих наборах даних може бути бажаним збільшити цей параметр), параметр eps має вирішальне значення для правильного вибору для набору даних і функції відстані, і зазвичай його не можна залишати. за значенням за замовчуванням. Він контролює локальне оточення точок. Якщо вибрати занадто малий розмір, більшість даних взагалі не буде кластеризовано (і позначено як -1 для «шуму»). Якщо вибрано надто великий, це призводить до об’єднання близьких кластерів в один кластер і, зрештою, весь набір даних повертається як один кластер. Деякі евристики для вибору цього параметра обговорювалися в літературі, наприклад, на основі коліна на графіку відстаней найближчого сусіда (як обговорюється в посиланнях нижче) [26].

На зображені (рис. 2.1) нижче колір вказує на членство в кластері, а великі кола вказують на основні зразки, знайдені алгоритмом. Менші кола – це неосновні зразки, які все ще є частиною кластера. Крім того, викиди позначені чорними точками внизу.

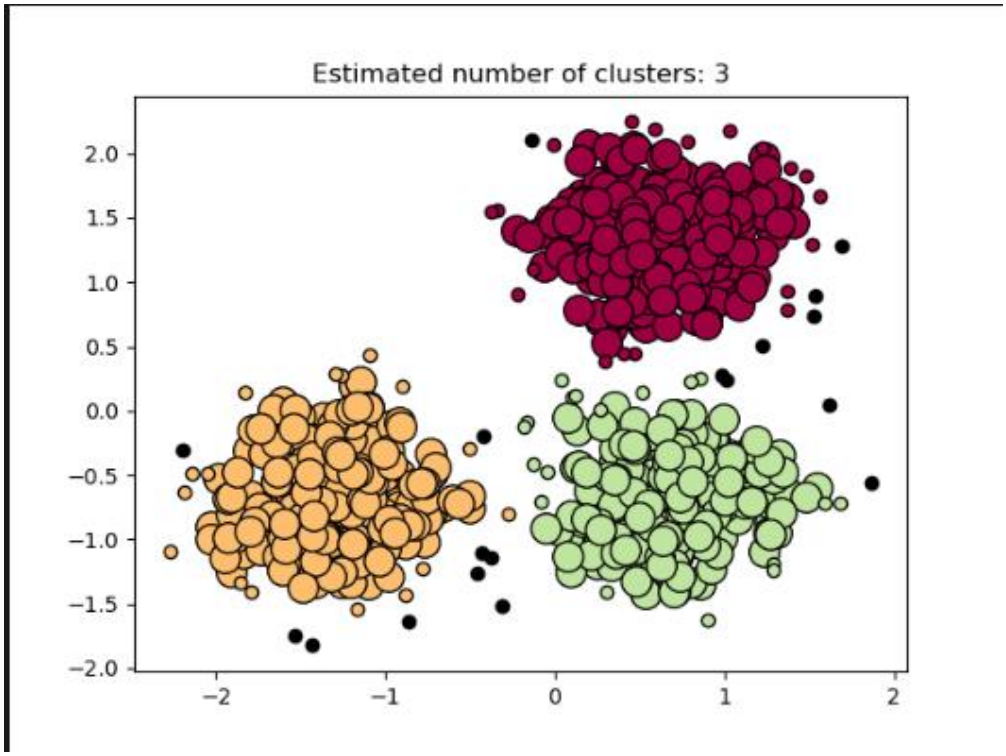



Рисунок 2.1 – Розподіл кластерів

## 2.2 OPTICS (Ordering Points To Identify the Clustering Structure)

Алгоритм OPTICS багато в чому схожий з алгоритмом DBSCAN, і його можна вважати узагальненням DBSCAN, яке зменшує вимогу eps з одного значення до діапазону значень. Ключова відмінність між DBSCAN і OPTICS полягає в тому, що алгоритм OPTICS будує графік досяжності, який призначає кожному зразку як відстань досяжності, так і місце в атрибуті впорядкування кластера; ці два атрибути призначаються під час встановлення моделі та використовуються для визначення членства в кластері. Якщо OPTICS запускається зі значенням inf за замовчуванням, встановленим для max_eps, тоді вилучення кластерів у стилі DBSCAN може виконуватися неодноразово в лінійному часі для будь-якого значення eps за допомогою методу cluster_optics_dbscan. Встановлення нижчого значення max_eps призведе до скорочення часу виконання, і його можна розглядати як максимальний радіус сусідства від кожної точки для пошуку інших потенційно доступних точок.

Відстані досяжності використовуються для вимірювання схожості між об’єктами і групування їх у кластери. Вони допомагають визначити, наскільки близькі або віддалені об’єкти один від одного.

Одним з найпоширеніших показників відстані досяжності є відстань Евкліда. Вона обчислюється як евклідова відстань між двома точками в n-вимірному просторі. Це може бути використано для вимірювання відстані між об’єктами на основі їх координат.

Відстані досяжності (рис. 2.2), створені OPTICS, дозволяють виділяти кластери зі змінною щільністю в одному наборі даних. Як показано на наведеному вище графіку, поєднання відстаней досяжності та впорядкування набору даних створює графік досяжності, де щільність точок представлена на осі Y, а точки впорядковані таким чином, що найближчі точки є суміжними. «Розрізання» графіка досяжності за одним значенням дає результати, подібні до DBSCAN; усі точки над «зрізом» класифікуються як шум, і кожен раз, коли є перерва при читанні зліва направо, означає новий кластер.

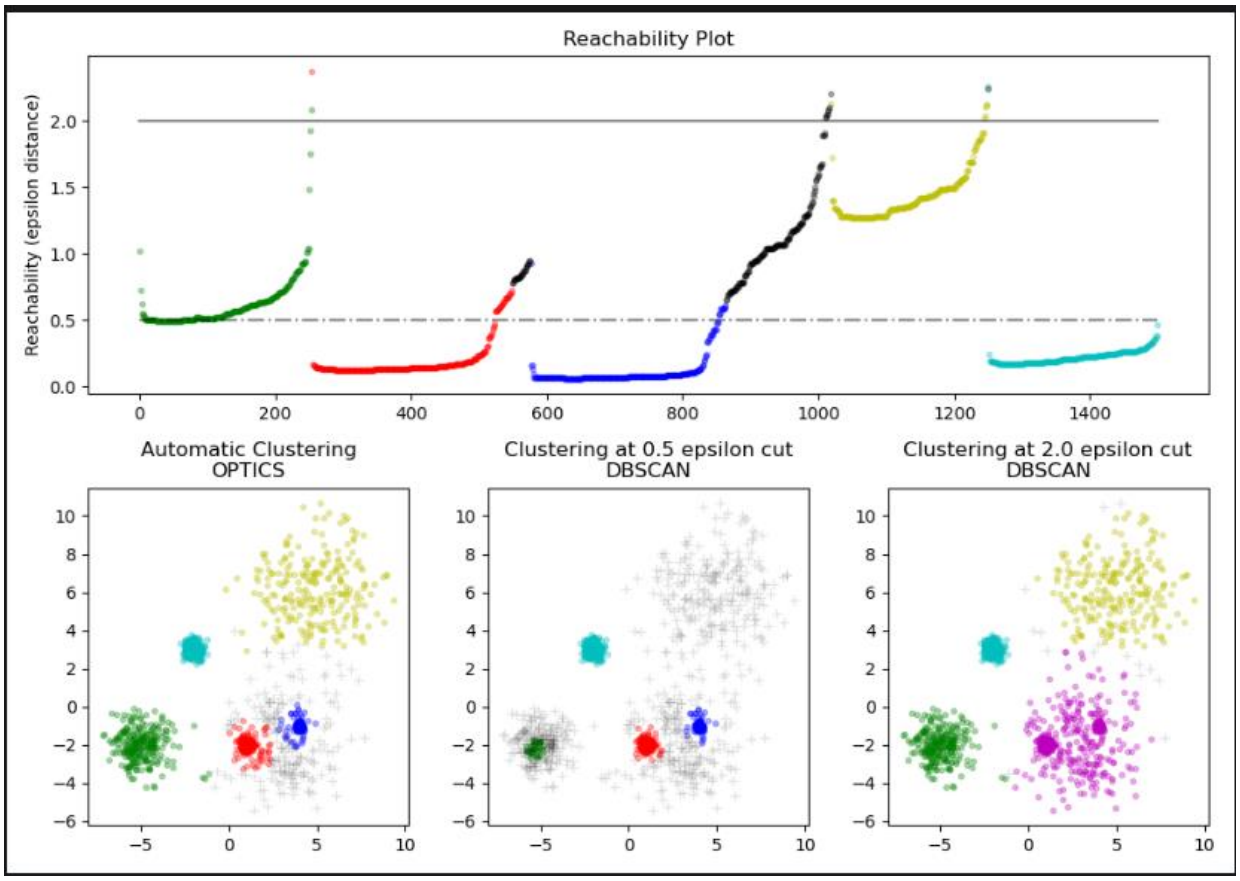



Рисунок 2.2 – Графік досяжності кластерів

Стандартне виділення кластерів за допомогою OPTICS розглядає круті схили на графіку, щоб знайти кластери, і користувач може визначити, що вважається крутим схилом, використовуючи параметр *xi*. Основна ідея полягає в тому, щоб встановити порядок обробки точок в наборі даних на основі їх густини і відстані між ними. Він враховує як густину точок, так і їх відстань одна від одної при визначенні кластерів. Існують також інші можливості для аналізу на самому графіку, такі як генерування ієрархічних представлень даних за допомогою дендрограм графіка досяжності, а також доступ до ієрархії кластерів, виявлених алгоритмом, через параметр cluster_hierarchy_. Діаграма вище була позначена кольором, щоб кольори кластерів у плоскому просторі відповідали кластерам лінійних сегментів графіка досяжності. Зауважте, що синій і червоний кластери є суміжними на графіку досяжності та можуть бути ієрархічно представлені як діти більшого батьківського кластера.

## 2.3 DENCLUE (Density-Based Clustering Algorithm for Applications with Noise)

Denclue – це алгоритм кластеризації, який використовує функцію щільності для визначення кластерів у багатовимірних просторах.

Основна ідея алгоритму Denclue полягає в тому, щоб моделювати розподіл густини даних у багатовимірному просторі за допомогою ядерної функції. Для кожного об’єкта даних алгоритм обчислює ваговий коефіцієнт, заснований на значенні ядерної функції для цього об’єкта та його найближчих сусідів. Ці вагові коефіцієнти потім використовуються визначення центроїдів кластерів і присвоєння кожного об’єкта до найближчого кластеру.

Алгоритм Denclue має ряд переваг у порівнянні з іншими алгоритмами кластеризації, такими як *k*-середній або ієрархічний кластерний аналіз. Наприклад, Denclue може виявляти кластери довільної форми і вимагає завдання числа кластерів заздалегідь.

Однак, Denclue може бути чутливий до шуму даних і може вимагати більш тривалого часу обчислень, особливо для великих наборів даних [27].

## 2.4 HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise)

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) – це алгоритм кластеризації, який базується на підході, що називається Density-Based Spatial Clustering of Applications with Noise (DBSCAN). HDBSCAN є покращеною версією алгоритму DBSCAN і зазвичай використовується для кластеризації даних зі складною структурою або з деякою кількістю шумових даних.

HDBSCAN заснований на ієрархічному підході до кластеризації, де дані розділяються на кластери на основі їх щільності. Алгоритм працює наступним чином:

Крок 1. Обчислення щільності точок: спочатку визначається щільність кожної точки даних, яка визначається як кількість сусідніх точок в заданому радіусі.

Крок 2. Вибір мінімальної щільності: HDBSCAN використовує мінімальну щільність, яка визначає, які точки даних будуть включені до кластерів. Ця мінімальна щільність вибирається автоматично за допомогою статистичних методів.

Крок 3. Обчислення кінцевих кластерів: кластери формуються на основі щільності даних та їх геометрії. HDBSCAN використовує оптимальне дерево кластерів для забезпечення максимальної стійкості кластерів.

Крок 4. Відокремлення шуму: алгоритм відокремлює шумові точки, які не можуть бути віднесені до жодного з кластерів.

Однією з основних переваг HDBSCAN є його здатність автоматично визначати кількість кластерів та відділяти шумові дані. Крім того, HDBSCAN може кластеризувати дані зі складними структурами, такими як вузькі пасма даних або точки, що знаходяться близько до інших кластерів. Це дозволяє отримати більш точні та стійкі результати кластеризації.

Ще одна перевага HDBSCAN полягає в його ефективності. Алгоритм може працювати з дуже великими обсягами даних та дозволяє швидко знаходити оптимальні кластери.

Однак, HDBSCAN має деякі недоліки. Наприклад, якщо щільність точок в кластері неоднакова, можуть виникнути проблеми зі стабільністю кластерів. Крім того, алгоритм може працювати не так ефективно з даними, що містять багато шумових точок.

У загальному, HDBSCAN є потужним алгоритмом кластеризації, який здатен знаходити оптимальні кластери в складних даних. З його допомогою

можна ефективно аналізувати великі обсяги даних та отримувати точні результати кластеризації [27].

## 2.5 Mean Shift Clustering

Mean shift є алгоритмом кластеризації, який може знаходити кластери в даних, не залежно від їх форми та розміру. Цей алгоритм є ненадмірно простим та досить ефективним, особливо коли маємо справу з невеликими наборами даних.

Принцип роботи алгоритму полягає в тому, що кожна точка в даних розглядається як центр мас. Для кожної точки встановлюється вікно, яке представляє деяку область навколо цієї точки. Потім, для кожної точки, обчислюється середнє значення всіх точок, що знаходяться в межах цього вікна. Потім центр вікна переміщується до отриманого середнього значення. Процес повторюється до тих пір, поки центр вікна не досягне локального максимуму густини точок. Таким чином, кластери визначаються як згущені області в даних, де центр вікна стабілізується.

Однією з переваг Mean shift є його здатність працювати з даними різної форми та розміру. Крім того, алгоритм не потребує визначення числа кластерів, що потрібно знайти, що є важливою перевагою, особливо коли маємо справу з невідомими даними.

Недоліком Mean shift є те, що він може бути вразливим до шуму в даних, особливо якщо шумові точки знаходяться близько до центрів кластерів. Крім того, алгоритм може бути повільним для великих наборів даних.

У загальному, Mean shift є простим та ефективним алгоритмом кластеризації, який може знаходити кластери в даних різної форми та розміру. Однак, він може бути вразливим до шуму в даних та може вимагати багато обчислювальних ресурсів для обробки великих наборів даних. Також,

алгоритм може видавати результати, які залежать від початкових умов та розмірів вікон.

Існує декілька підходів до покращення Mean shift. Один з них полягає в використанні адаптивних вікон, які можуть змінювати свій розмір в залежності від густини точок. Це дозволяє зменшити вплив шуму та збільшити ефективність алгоритму.

Інший підхід – використання зменшення розмірності даних перед застосуванням алгоритму Mean shift. Це може зменшити обчислювальні витрати та покращити точність результатів.

Кластеризація MeanShift спрямована на виявлення кулястих об’єктів в плавній щільності вибірки. Це алгоритм, що базується на центроїдах, який працює, оновлюючи кандидатів на центроїди як середнє значення точок у певній області. Після цього кандидати фільтруються на етапі пост-процесингу, щоб уникнути близьких дублікатів та створити кінцевий набір центроїдів.

Позиція кандидатів на центроїди ітеративно коригується за допомогою техніки, яку називають підйому на гірку, яка знаходить локальні максимуми оціненої щільності ймовірності. Заданий кандидат на центроїд для ітерації оновлюється згідно з наступним рівнянням:

$$x^{t+1} = x^t + m(x^t). \tag{2.1}$$

Маємо вектор зсуву середнього значення, який обчислюється для кожного центроїда та показує в напрямку області з максимальним зростанням щільності точок. Щоб обчислити вектор, потрібно визначити околицю вибірок, які знаходяться на відстані від центроїда. Потім обчислюється, ефективність оновлення центроїду, як середнє значення вибірок, що знаходяться в його околиці.

Алгоритм автоматично встановлює кількість кластерів, замість того, щоб сподіватися на параметр, який визначає розмір області, через яку проводиться пошук. Цей параметр можна встановити вручну, але його також

можна оцінити за допомогою функції , яка викликається, якщо ширина смуги не встановлена.

Алгоритм не є високопродуктивним, оскільки він вимагає кількох пошуків найближчих сусідів під час виконання алгоритму. Алгоритм гарантовано збігається, проте він зупинятиметься після того, як зміна центроїдів стане незначною [28].

Позначення нового прикладу виконується шляхом знаходження найближчого центроїда для заданого прикладу (рис. 2.3).

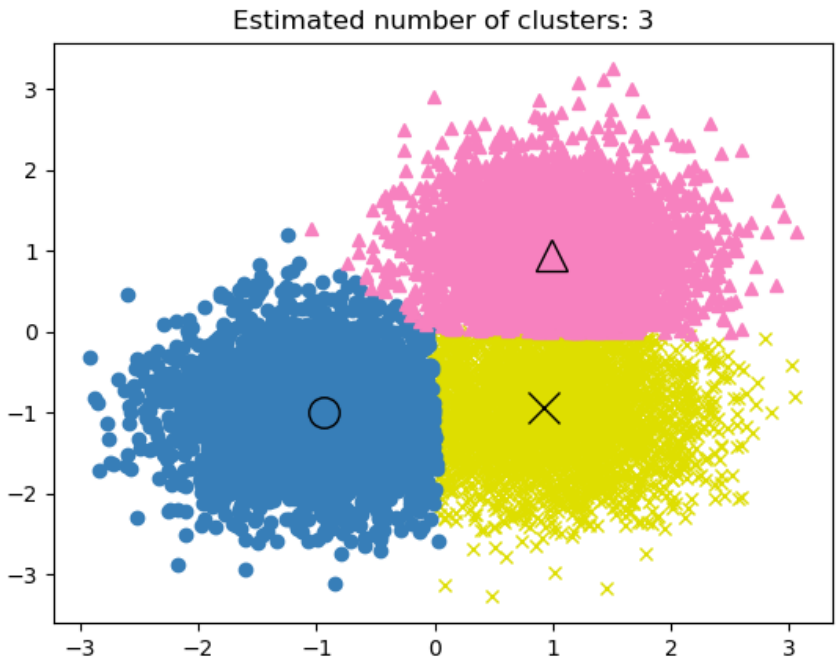



Рисунок 2.3 – Приклад роботи Mean Shift

Узагальнюючи, Mean shift є ефективним та простим алгоритмом кластеризації, який може знаходити кластери в даних різної форми та розміру. Однак, він має свої обмеження, такі як вразливість до шуму та вимогу до обчислювальних ресурсів. Врахування цих факторів та застосування відповідних підходів може покращити ефективність алгоритму. Його ефективність залежить від правильного вибору параметрів та оптимального управління обчислювальними ресурсами, але він може бути використаний в

багатьох областях, включаючи комп’ютерне зору, обробку зображень, розпізнавання образів та аналіз даних [29].

## 2.6 ST-DBSCAN (Spatial-Temporal Density-Based Clustering of Applications with Noise)

ST-DBSCAN (Spatial-Temporal Density-Based Spatial Clustering of Applications with Noise) є розширенням алгоритму DBSCAN для кластеризації просторових-часових даних, таких як дані GPS або телематики.

Основна ідея ST-DBSCAN полягає в тому, щоб розділити простір і час на дві різні виміри, тобто кластеризувати точки одночасно за їх просторовим та часовим положенням. Алгоритм використовує дві ключові метрики: Eps (радіус) та MinPts (мінімальна кількість точок), які також використовуються в DBSCAN.

ST-DBSCAN працює наступним чином:

– обчислюється просторовий відстань між точками;

– обчислюється часова відстань між точками;

– якщо обидві відстані менші за Eps, то точки вважаються сусідніми;

– формуються кластери шляхом об’єднання сусідніх точок в групи. Точки вважаються як частина кластеру, якщо вони мають не менше MinPts сусідів;

– якщо точка не є частиною жодного кластеру та не має достатньо багато сусідів, то вона вважається шумом.

ST-DBSCAN має кілька переваг порівняно з іншими методами кластеризації просторових-часових даних. Він може обробляти дані з нерівномірним і нелінійним розподілом, враховуючи їх просторові та часові властивості. Крім того, ST-DBSCAN відносно легко використовувати, тому є популярним методом для кластеризації просторових-часових даних.

Алгоритм ST-DBSCAN вимагає чотирьох параметрів *Eps1, Eps2, MinPts* і $\Delta\epsilon$ через розширення. *Eps1* є параметром відстані для просторових атрибутів

(широти та довготи). *Eps2* – це параметр відстані для непросторових атрибутів. Для *Eps1* і *Eps2* можна використовувати такі показники відстані, як Евклідова, Манхеттенська або Мінковського. *MinPts* – це мінімальна кількість точок на відстані *Eps1* і *Eps2* від точки. Якщо регіон щільний, він повинен містити більше балів, ніж значення *MinPts*. У представлена проста евристика, ефективна в багатьох випадках для визначення параметрів *Eps* і *MinPts*. Евристика передбачає $MinPts \approx ln(n)$, де $n$ – це розмір бази даних, а *Eps* потрібно вибирати залежно від значення *MinPts*. Першим кроком евристичного методу є визначення відстаней до *k*-найближчих сусідів для кожного об'єкта, де *k* дорівнює *MinPts*. Потім ці значення *k*-відстаней слід відсортувати в порядку спадання. Потім потрібно визначити порогову точку, яка є першою «долиною» відсортованого графа. Для цього потрібно обрати *Eps* менше, ніж відстань, визначена першою долиною. Останній параметр $\Delta\epsilon$ використовується для запобігання виявлення об'єднаних кластерів через невеликі відмінності в непросторових значеннях сусідніх місць.

Алгоритм починається з першої точки *p* в базі даних *D* і отримує всі точки, досяжні за щільністю з *p* відносно *Eps1* і *Eps2*. Якщо *p* є основним об'єктом, формується кластер. Якщо p є граничним об'єктом, жодна точка не може бути досягнута за щільністю з p, і алгоритм відвідує наступну точку бази даних. Процес повторюється, поки не будуть оброблені всі точки.

Як показано на, алгоритм починається з першої точки в базі даних *D*. Після обробки цієї точки він вибирає наступну точку в *D*. Якщо вибраний об'єкт не належить до жодного кластера, викликається функція Retrieve_Neighbors. Виклик Retrieve_Neighbors(*object, Eps1, Eps2*) повертає об'єкти, відстань яких до вибраного об'єкта менша за параметри *Eps1* і *Eps2*. Іншими словами, функція Retrieve_Neighbors отримує всі об'єкти, досяжні за щільністю з вибраного об'єкта щодо *Eps1, Eps2* і *MinPts*. Набір результатів формує Eps-Neighborhood вибраного об'єкта. Retrieve_Neighbours (*object, Eps1, Eps2*) дорівнює перетину Retrieve_Neighbours (*object, Eps1)* і Retrieve_Neighbours(*object, Eps2*). Якщо загальна кількість повернених точок

у Eps-Neighborhood менша за вхід *MinPts*, об’єкту призначається шум. Це означає, що вибрана точка не має достатньо сусідів для кластеризації. Точки, які були позначені як шумові, можуть бути змінені пізніше, якщо вони не доступні безпосередньо для щільності, але вони доступні для щільності з іншої точки бази даних. Це відбувається для граничних точок кластера.

## 2.7 CLIQUE (Clustering In Quest)

CLIQUE – це алгоритм кластеризації підпростору на основі щільності та сітки. Отже, давайте спочатку подивимося, що таке метод кластеризації на основі сітки та щільності.

Техніка кластеризації на основі сітки: у методах на основі сітки простір екземпляра поділено на структуру сітки. Потім застосовуються методи кластеризації з використанням клітинок сітки замість окремих точок даних як базових одиниць.

Техніка кластеризації на основі щільності: у методах на основі щільності кластер – це максимальний набір з’єднаних щільних одиниць у підпросторі.

Алгоритм CLIQUE використовує техніку на основі щільності та сітки, тобто алгоритм кластеризації підпростору, і знаходить кластер, використовуючи порогове значення щільності та кількість сіток як вхідні параметри. Він спеціально розроблений для роботи з наборами даних із великою кількістю вимірів. Алгоритм CLIQUE дуже масштабований щодо значення записів і кількості вимірів у наборі даних, оскільки він заснований на сітці та ефективно використовує властивість Apriori.

Апріорний підхід стверджує, що якщо *X*-вимірна одиниця є щільною, то всі її проекції в *X*-1 – вимірному просторі також є щільними.

Це означає, що щільні області в даному підпросторі повинні створювати щільні області при проектуванні на низько вимірний підпростір. CLIQUE

обмежує пошук високо вимірних щільних комірок перетином щільних комірок у підпросторі, оскільки CLIQUE використовує апріорні властивості.

Алгоритм CLIQUE спочатку розбиває простір даних на сітки. Це робиться шляхом поділу кожного виміру на рівні проміжки, які називаються одиницями. Після цього він визначає щільні одиниці. Одиниця вважається щільною, якщо точки даних у ній перевищують порогове значення.

Коли алгоритм знаходить щільні комірки вздовж одного виміру, алгоритм намагається знайти щільні комірки вздовж двох вимірів, і він працює, доки не будуть знайдені всі щільні комірки вздовж усього виміру.

Після знаходження всіх щільних комірок у всіх вимірах алгоритм переходить до пошуку найбільшого набору («кластеру») з'єднаних щільних комірок. Нарешті, алгоритм CLIQUE генерує мінімальний опис кластера. Потім кластери генеруються з усіх щільних підпросторів за допомогою апріорного підходу.

CLIQUE – це алгоритм кластеризації підпростору, який перевершує *K*-means, DBSCAN і Farthest First як за часом виконання, так і за точністю.

CLIQUE може знаходити кластери будь-якої форми та будь-яку кількість кластерів у будь-якій кількості вимірів, де кількість не визначається параметром.

Один з найпростіших методів, можливість інтерпретації результатів.

Основним недоліком алгоритму CLIQUE є те, що якщо розмір комірки не підходить для набору дуже високих значень, то буде виконано занадто багато оцінок і не вдасться знайти правильний кластер [29].

## 2.8 BIRCH

Даний алгоритм Будується дерево під назвою Clustering Feature Tree (CFT) за наведеними даними. Дані, по суті, стиснуті з втратами до набору Вузли функції кластеризації (CF Nodes). Вузли CF мають ряд підкластери, які

називаються підкластерами функції кластеризації (CF Subclusters) і ці CF підкластери, розташовані в нетермінальних CF вузлах може мати вузли CF як діти.

Підкластери CF містять необхідну інформацію для кластеризації, яка запобігає необхідність утримувати в пам’яті всі вхідні дані. Ця інформація включає:

– кількість вибірок у підкластері;

– лінійна сума – це *n*-вимірний вектор, що містить суму всіх вибірок;

– квадратна сума – це сума квадрата L2 норми всіх вибірок;

– центроїди використовується уникнути перерахунку лінійної суми / n_samples;

– квадратна норма центроїдів.

Алгоритм BIRCH має два параметри, поріг і коефіцієнт розгалуження. Коефіцієнт розгалуження обмежує кількість підкластерів у вузлі та поріг обмежує відстань між вхідним зразком і існуючим підкластерів.

Цей алгоритм можна розглядати як екземпляр або метод зменшення даних, оскільки він зводить вхідні дані до набору підкластерів, які виходять безпосередньо з листя CFT. Ці зменшені дані можуть бути додатково оброблені шляхом годування Це в глобальний кластер. Цей глобальний кластер може бути встановлений за допомогою . Якщо встановлено значення Немає, то підкластери з листя знаходяться безпосередньо Читайте, інакше глобальний крок кластеризації позначає ці підкластери в глобальні Кластери (мітки) і вибірки зіставляються з глобальною міткою найближчого підкластера.

Опис алгоритму – новий зразок вставляється в корінь CF Tree, який є вузлом CF. Потім вона зливається з підкластером кореня, який має найменший радіус після злиття, обмежений пороговими і коефіцієнтними умовами розгалуження. Якщо підкластер має будь-який дочірній вузол, то це робиться багаторазово, поки він не досягне лист. Після знаходження найближчої

підкластери в листі, властивості цього Підкластер і батьківські підкластери рекурсивно оновлюються.

Якщо радіус підкластера отриманий шляхом злиття нової вибірки і найближчий підкластер більше квадрата порога і якщо кількість підкластерів більше коефіцієнта розгалуження, тоді простір тимчасово виділено для цього новий зразок. Взяті два найдальших підкластера і Підкластери діляться на дві групи виходячи з відстані між цими підкластерами.

Якщо цей розділений вузол має батьківський підкластер і є місце Для нового підкластера тоді батько розбивається на два. Якщо місця немає, Потім цей вузол знову розщеплюється на два і процес продовжується рекурсивно, поки не досягне кореня [29].

## 2.9 DSAP (Density Sensitive Affinity Propagation)

Density Sensitive Affinity Propagation (DSAP) - це алгоритм кластеризації, який є модифікацією базового алгоритму Affinity Propagation. Він був розроблений з метою покращення ефективності та точності кластеризації в областях з високою густини даних.

Основна ідея DSAP полягає в тому, щоб враховувати густину даних при визначенні кількості кластерів та розташування центрів кластерів. DSAP знаходить кластери шляхом визначення «відстані» між елементами даних, які вимірюються як міра схожості між ними.

У DSAP на відміну від базового алгоритму Affinity Propagation, використовується параметр плотності, який визначається як кількість точок даних, розташованих у заданому радіусі навколо кожної точки даних. Цей параметр дозволяє DSAP виявляти густинні області в даних та розміщувати центри кластерів в цих областях.

DSAP також використовує параметр ваги, який відображає силу взаємодії між точками даних. Цей параметр можна налаштувати залежно від особливостей даних та задачі кластеризації.

Узагальнюючи, DSAP є ефективним та точним алгоритмом кластеризації, який використовує плотність та вагу даних для визначення кількості кластерів та розташування їх центрів. DSAP може бути застосований в багатьох областях, де необхідна точна та швидка кластеризація даних з високою густиністю.

## 2.10 ROSE (Rank-Order-based Similarity Evaluation for High-dimensional Clustering)

Rank-Order-based Similarity Evaluation for High-dimensional Clustering (ROSEC) – це метод кластеризації, який був розроблений для роботи з високовимірними даними. ROSEC використовує ранжування для визначення схожості між точками даних та визначення їх приналежності до кластерів.

Одна з проблем, пов’язаних з кластеризацією високовимірних даних, полягає в тому, що традиційні міри схожості, такі як Евклідова відстань, можуть бути недостатньо ефективними в цих випадках. ROSEC вирішує цю проблему шляхом використання рангової міри схожості, яка базується на порівнянні рангів замість значень.

У ROSEC спочатку проводиться ранжування значень кожного атрибуту в даних. Далі, для кожної пари точок даних визначається рангова відстань між ними, що відображає відносну відстань між рангами їх атрибутів. Ці рангові відстані потім використовуються для визначення схожості між точками даних та їх приналежності до кластерів.

Оцінка алгоритму ROSE включатиме в себе порівняння прогнозованих міток кластерів з справжніми мітками (якщо вони доступні), обчислення метрик оцінки та аналіз результатів. При виборі алгоритму кластеризації та

налаштуванні параметрів варто враховувати особливості даних та конкретну задачу, щоб отримати оптимальні результати.

Основна ідея алгоритму ROSE полягає у визначенні рангової схожості між об’єктами у просторі ознак. За допомогою рангової схожості об’єктів, алгоритм ROSE формує граф схожості, де вершини представляють об’єкти, а ребра відображають схожість між ними. Далі, алгоритм використовує інформацію про граф для виявлення кластерів.

ROSEC також використовує алгоритм кластеризації Affinity Propagation (AP), який використовує інформацію про схожість та силу взаємодії між точками даних для визначення кластерів. AP розподіляє точки даних на кластери, використовуючи дві міри – відстань та «відповідальність». Відстань відображає відносну відстань між точками даних, а «відповідальність» відображає силу взаємодії між точками даних.

ROSEC також використовує потужний алгоритм кластеризації AP, який враховує схожість та силу взаємодії між точками даних.

Основною перевагою ROSEC є його ефективність та здатність обробляти великі набори даних з багатьма атрибутами. Крім того, ROSEC дозволяє отримувати більш точні результати кластеризації в порівнянні з традиційними методами, особливо при роботі з високовимірними даними.

Незважаючи на свої переваги, ROSEC має деякі обмеження. Наприклад, він може працювати не так добре з даними, що мають складні залежності між атрибутами, а також з даними, що мають нерівномірний розподіл.

Узагальнюючи, ROSEC є потужним методом кластеризації для високовимірних даних, що базується на ранговій мірі схожості та алгоритмі кластеризації AP. Він має свої переваги та обмеження, тому вибір методу кластеризації повинен базуватися на конкретних потребах та характеристиках набору даних [30].

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи було розглянуто алгоритми кластерищзації на основі щільності розподілу даних. Для реалізації даних алгоритмів було обрано мову Python . Це обумовлено тим, що Python має ряд таких переваг:

– багата екосистема бібліотек. Python має широкий набір бібліотек, таких як scikit-learn, NumPy, SciPy і pandas, які надають потужні інструменти для кластеризації та обробки даних. Ці бібліотеки мають високий рівень оптимізації, що дозволяє ефективно виконувати різні алгоритми кластеризації на великих наборах даних;

– простота використання. Python є досить простою мовою програмування, що робить його доступним для широкого кола користувачів. Він має зрозумілий синтаксис і велику кількість прикладів та документації, що допомагає легко реалізувати алгоритми кластеризації;

– візуалізація результатів. Python має багато бібліотек для візуалізації даних, таких як Matplotlib і Seaborn. Це дозволяє зручно візуалізувати результати кластеризації, що сприяє кращому розумінню даних і виявленню закономірностей;

– можливість інтеграції. Python легко інтегрується з іншими мовами програмування та інструментами аналізу даних. Наприклад, ви можете використовувати Python для попередньої обробки даних, а потім передавати їх до інших інструментів аналізу даних, таких як R чи MATLAB;

– розширені можливості машинного навчання. Python має потужні бібліотеки, такі як scikit-learn, які надають реалізації різних алгоритмів машинного навчання, включаючи методи кластеризації. Це дозволяє використовувати широкий спектр алгоритмів кластеризації, таких як K-

середніх, ієрархічна кластеризація, спектральна кластеризація та багато інших. Ці бібліотеки надають готові реалізації алгоритмів з різними налаштуваннями, що спрощує їх використання та експериментування з різними підходами.

Крім того, Python є популярною мовою серед спільноти дослідників та практикуючих аналітиків даних. Це означає, що ви можете легко знайти підтримку, поради та приклади коду в Інтернеті або в спеціалізованих форумах, що допоможе вам вирішувати проблеми та вдосконалювати свої навички в кластеризації.

Загалом, використання Python для кластеризації дозволяє отримати потужні інструменти, широкий вибір алгоритмів, зручну візуалізацію та доступність до розширених можливостей машинного навчання. Все це робить Python популярним вибором для виконання завдань кластеризації в різних сферах, включаючи аналіз даних, машинне навчання, обробку зображень та багато інших. Він пропонує потужні бібліотеки та інструменти, які допомагають в обробці, візуалізації, моделюванні та розумінні даних. Ось кілька ключових аспектів Python у контексті аналітики даних. Бібліотеки для аналізу даних. Python має такі популярні бібліотеки, як NumPy і Pandas, які надають потужні структури даних (такі як масиви та DataFrame) і функції для обробки та аналізу даних. Вони дозволяють ефективно виконувати операції фільтрації, сортування, групування, агрегування і об’єднання даних.

Python пропонує різноманітні бібліотеки для візуалізації даних, такі як Matplotlib, Seaborn і Plotly. Вони дозволяють будувати графіки, діаграми, графи та інші візуалізації, що допомагають зрозуміти дані та виявити корисні інсайти.

Python має потужні бібліотеки для машинного навчання, такі як Scikit-learn, TensorFlow і PyTorch. Вони надають інструменти для побудови моделей класифікації, регресії, кластеризації, рекомендаційних систем та багатьох інших. Ці бібліотеки дозволяють використовувати різні алгоритми та проводити оптимізацію моделей.

Для реалізації даних алгоритмів, використовувався IDE від JetBrains PyCharm (рис. 3.1). PyCharm є інтегрованою середовищем розробки (IDE) для мови програмування Python, розробленим компанією JetBrains. Воно надає розширені можливості для розробки програм на Python та підтримує різні функціональні можливості, що полегшують процес програмування.

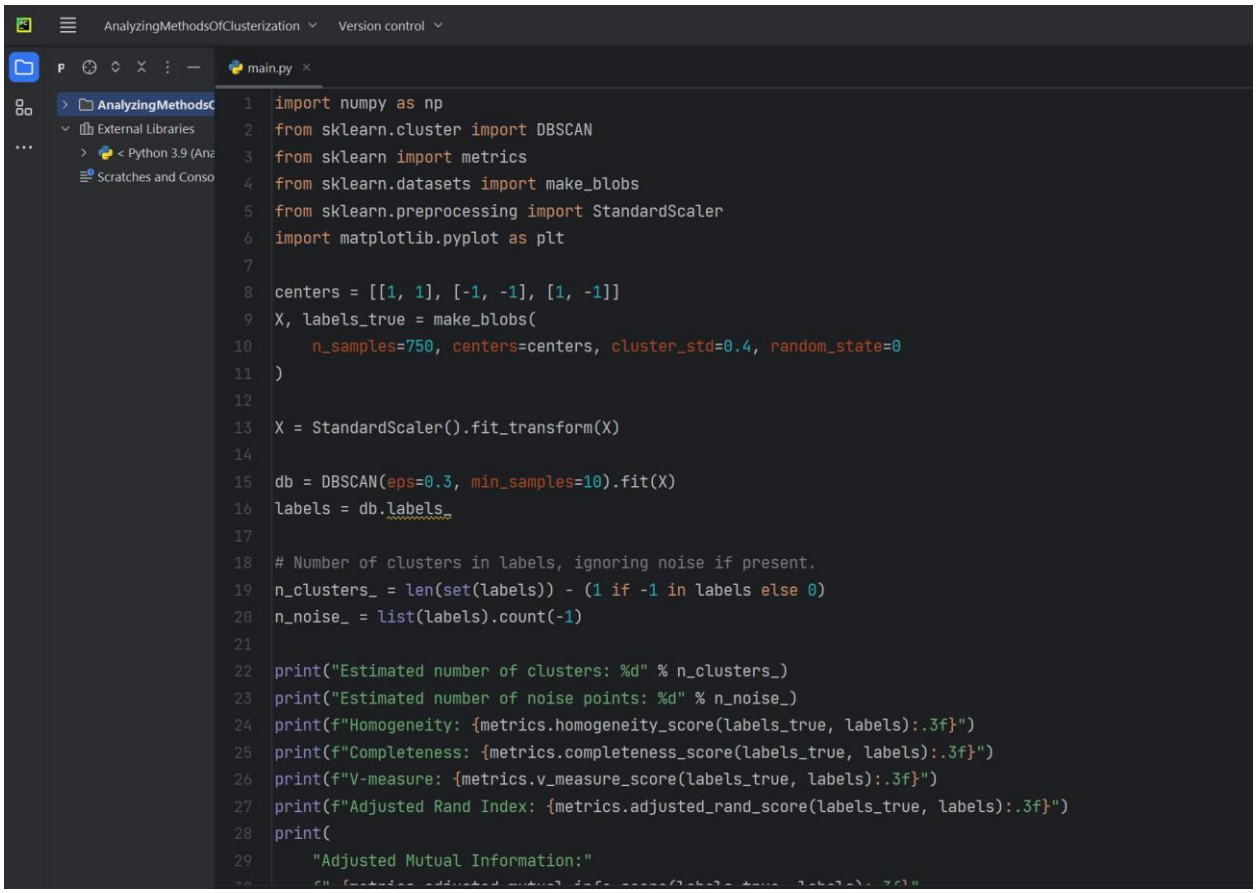



Рисунок 3.1 – Приклад інтерфейсу PyCharm

Ось декілька ключових особливостей PyCharm:

– редагування коду PyCharm надає потужний редактор коду з розумним автодоповненням, підсвічуванням синтаксису, переходом до визначень функцій та багато інших корисних функцій. Він також підтримує форматування коду за стандартами PEP 8, що дозволяє писати чистий і зрозумілий код;

– відладка PyCharm має вбудований відладчик, який дозволяє крокувати по коду, встановлювати точки зупину та аналізувати значення змінних під час виконання програми. Це допомагає знайти та виправити помилки в коді швидше та ефективніше;

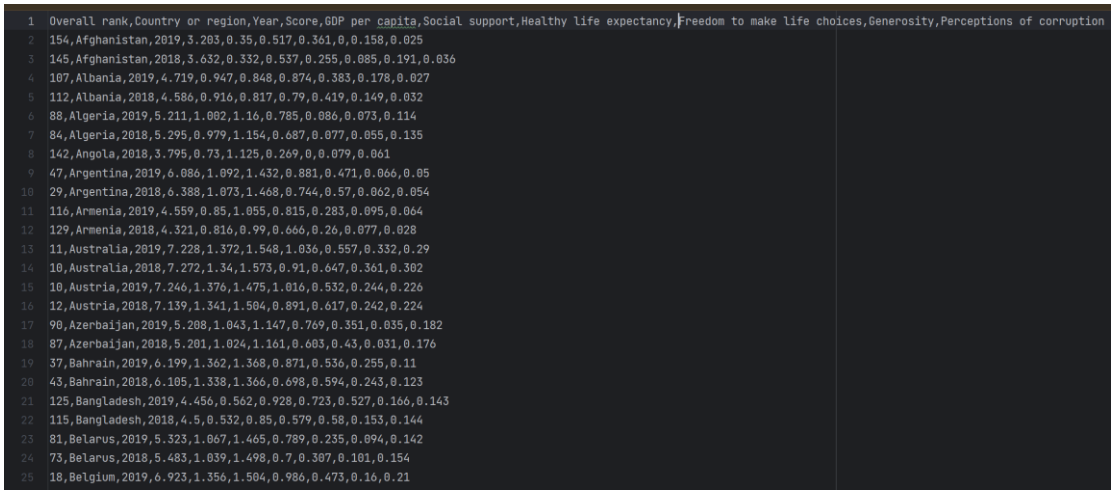
– керування проектами PyCharm дозволяє легко створювати та керувати проектами Python. Воно автоматично визначає структуру проекту, надає можливість встановлювати та керувати залежностями через інструменти, такі як pip та virtualenv;

– підтримка систем контролю версій PyCharm інтегрується з популярними системами контролю версій, такими як Git, SVN та Mercurial. Це дозволяє командам розробників легко працювати з кодом, здійснювати коміти, злиття та інші операції безпосередньо з інтерфейсу PyCharm;

– інструменти аналізу коду PyCharm надає різноманітні інструменти для аналізу якості коду, виявлення потенційних.

## 3.2 Опис вибірки даних

Для даної роботи було обрано вибірку даних рівень щастя країн, яка має такий вигляд (рис. 3.2).



```
Overall rank,Country or region,Year,Score,GDP per capita,Social support,Healthy life expectancy,Freedom to make life choices,Generosity,Perceptions of corruption
154,Afghanistan,2019,3.203,0.35,0.517,0.361,0,0.158,0.025
145,Afghanistan,2018,3.632,0.332,0.537,0.255,0.085,0.191,0.036
107,Albania,2019,4.719,0.947,0.848,0.874,0.383,0.178,0.027
112,Albania,2018,4.586,0.916,0.817,0.79,0.419,0.149,0.032
88,Algeria,2019,5.211,1.002,1.16,0.785,0.086,0.073,0.114
84,Algeria,2018,5.295,0.979,1.154,0.687,0.077,0.055,0.135
142,Angola,2018,3.795,0.73,1.125,0.269,0,0.079,0.061
47,Argentina,2019,6.086,1.092,1.432,0.881,0.471,0.066,0.05
29,Argentina,2018,6.388,1.073,1.468,0.744,0.57,0.062,0.054
116,Armenia,2019,4.559,0.85,1.055,0.815,0.283,0.095,0.064
129,Armenia,2018,4.321,0.816,0.99,0.666,0.26,0.077,0.028
11,Australia,2019,7.228,1.372,1.548,1.036,0.557,0.332,0.29
10,Australia,2018,7.272,1.34,1.573,0.91,0.647,0.361,0.302
10,Austria,2019,7.246,1.376,1.475,1.016,0.532,0.244,0.226
12,Austria,2018,7.139,1.341,1.504,0.891,0.617,0.242,0.224
90,Azerbaijan,2019,5.208,1.043,1.147,0.769,0.351,0.035,0.182
87,Azerbaijan,2018,5.201,1.024,1.161,0.603,0.43,0.031,0.176
37,Bahrain,2019,6.199,1.362,1.368,0.871,0.536,0.255,0.11
43,Bahrain,2018,6.105,1.338,1.366,0.698,0.594,0.243,0.123
125,Bangladesh,2019,4.456,0.562,0.928,0.723,0.527,0.166,0.143
115,Bangladesh,2018,4.5,0.532,0.85,0.579,0.58,0.153,0.144
81,Belarus,2019,5.323,1.067,1.465,0.789,0.235,0.094,0.142
73,Belarus,2018,5.483,1.039,1.498,0.7,0.307,0.101,0.154
18,Belgium,2019,6.923,1.356,1.504,0.986,0.473,0.16,0.21
```

Рисунок 3.2 – Вибірка даних рівень щастя країн

Про дані: це дослідження аналізує зв’язок між показником індексу щастя в 2018 і 2019 роках і набором незалежних змінних, таких як «Загальний рейтинг», «ВВП на душу населення», «Соціальна підтримка», «Очікувана тривалість здорового життя», «Свобода». зробити життєвий вибір», «Щедрість» і «Сприйняття корупції». Метою цього дослідження є дослідження впливу цих незалежних змінних на рівень щастя людей протягом цих двох років. Крім того, було проведено аналіз по країнах, щоб вивчити варіації змінних між найщасливішою країною, яка займає перше місце, та Індією.

Залежні та незалежні змінні це дві основні змінні будь-якого експерименту чи дослідження. Незалежний – це той, який змінюється або контролюється для вивчення його впливу на залежну змінну. Залежним є змінна, яка досліджується та вимірюється.

Тоді їх можна розглядати як причину (незалежна змінна) та наслідок (залежну змінну). Незалежний контролюється експериментатором, тоді як залежний змінюється у відповідь на незалежного.

Приклади даних зміних:

– вплив споживання фруктів на фізичну стійкість. Вживання фруктів, фізична витривалість;

– вплив споживання цукру на вагу. Споживання цукру, вага (DV).

Залежні та незалежні змінні. Першою незалежною змінною є ВВП на душу населення, який відображає економічний добробут жителів країни. Друга змінна, свобода робити життєвий вибір, представляє рівень автономії та контролю, який люди мають над своїм життям. Третя змінна, очікувана тривалість здорового життя, вказує на тривалість часу, протягом якого люди можуть розраховувати прожити в доброму здоров’ї. Четверта змінна, сприйняття корупції, відображає ступінь, до якого корупція сприймається як поширена в суспільстві. Нарешті, п’ята змінна, соціальна підтримка, вимірює ступінь, до якої люди мають доступ до підтримки з боку родини, друзів та інших соціальних мереж.

Вивчаючи зв’язок між цими незалежними змінними та показником індексу щастя, дослідження має на меті сприяти розумінню факторів, які впливають на рівень щастя людини. Результати цього дослідження можуть бути корисними для політиків та інших зацікавлених сторін у визначенні областей для втручання та покращення для підвищення загального добробуту населення.

Перелік типів даних в датасеті:

– overall rank (загальний рейтинг);

– country or region (країна або регіон);

– score (оцінка: список оцінок щастя різних країн);

– DP per capital (ВВП на душу населення);

– social support (соціальна підтримка: соціальна підтримка різних країн);

– healthy life expectancy (очікувана тривалість здорового життя);

– freedom to make life choices (свобода вибору життя: оцінка сприйняття свободи різними країнами);

– generosity (щедрість: щедрість (якість бути добрим і щедрим);

– perceptions of corruption (сприйняття корупції).

Наступним кроком імпортуємо наш датасет (рис. 3.3).

Лістинг 3.1 Імпорт датасету:

```
import pandas as pd
df = pd.read_csv(‘happy.csv’)
print(df.head())
df.info()
data = df.drop("Rate", axis = 1)
print(df["Rate"].values)
```

```
K:\pyCharm\AnalyzingMethodsOfClusterization\venv\Scripts\python.exe K:\pyCharm\AnalyzingMethodsOfClusterization\exampleDataSet.py
   Rate   Overall rank  ... Generosity  Perceptions of corruption
0     0            154  ...      0.158                      0.025
1     0            145  ...      0.191                      0.036
2     0            107  ...      0.178                      0.027
3     0            112  ...      0.149                      0.032
4     0             88  ...      0.073                      0.114

[5 rows x 11 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 11 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Rate                          312 non-null    int64
 1    Overall rank                 312 non-null    int64
 2   Country or region             312 non-null    object
 3   Year                          312 non-null    int64
 4   Score                         312 non-null    float64
 5   GDP per capital               312 non-null    float64
 6   Social support                312 non-null    float64
 7   Healthy life expectancy       312 non-null    float64
 8   Freedom to make life choices  312 non-null    float64
 9   Generosity                    312 non-null    float64
 10  Perceptions of corruption     312 non-null    float64
dtypes: float64(7), int64(3), object(1)
memory usage: 26.9+ KB
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]

Process finished with exit code 0
```

Рисунок 3.3 – Імпорт датасета

Лістинг 3.2 Нормалізація датасету:

```
# scaling and normalizing the dataset
std_scaler = StandardScaler()
data_cluster = data.copy()
data_cluster[data_cluster.columns] = std_scaler.fit_transform(data_cluster)
print(data_cluster.describe())
```

Після внесених змін датасет має такий вигляд (рис. 3.4).

| | Rate | Overall rank | ... | Generosity | Perceptions of corruption |
|---|---|---|---|---|---|
| count | 312.000000 | 312.000000 | ... | 3.120000e+02 | 3.120000e+02 |
| mean | 0.000000 | 0.000000 | ... | -6.262797e-17 | 2.960595e-16 |
| std | 1.001606 | 1.001606 | ... | 1.001606e+00 | 1.001606e+00 |
| min | -1.080522 | -1.720983 | ... | -1.893962e+00 | -1.172192e+00 |
| 25% | -1.080522 | -0.860492 | ... | -7.679980e-01 | -6.466660e-01 |
| 50% | 0.163474 | 0.000000 | ... | -7.688940e-02 | -3.103296e-01 |
| 75% | 1.407470 | 0.860492 | ... | 6.426919e-01 | 3.124183e-01 |
| max | 1.407470 | 1.720983 | ... | 4.297543e+00 | 3.631113e+00 |

Рисунок 3.4 – Нормалізація датасета

Нормалізація датасету – це процес приведення значень признаків в датасеті до спільної шкали або діапазону. Це робиться з метою забезпечення однорідності та стабільності при обробці та аналізі даних. Основні причини для нормалізації датасету включають наступне:

– уникнення перекосу: якщо признаки в датасеті мають різні масштаби або одиниці виміру, може виникнути перекос у впливі цих признаків на модель або алгоритм машинного навчання. Нормалізація допомагає уникнути цього перекосу, приводячи значення признаків до спільної шкали, такої як діапазон від 0 до 1;

– покращення інтерпретованості результатів: нормалізація датасету може сприяти кращому розумінню та інтерпретації результатів. Значення признаків будуть знаходитись в одному діапазоні, що спрощує порівняння та визначення їх вагомості в контексті задачі аналізу даних;

– покращення швидкості збіжності алгоритмів: деякі алгоритми машинного навчання, зокрема ті, що використовують оптимізаційні методи, можуть швидше збігатися, якщо дані нормалізовані. Це може покращити швидкість тренування моделі та зменшити кількість ітерацій, необхідних для досягнення оптимального розв’язку.

## 3.3 Реалізація методів

### 3.3.1 DBSCAN

Лістинг 3.3 Реалізація алгоритму DBSCAN:

```
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
data = pd.read_csv('happy.csv')
X = data[['Score','GDP per capital','Social support','Healthy life expectancy','Freedom to make life choices','Generosity','Perceptions of corruption']].values
dbscan = DBSCAN(eps=0.3, min_samples=5)
labels = dbscan.fit_predict(X)
print("Мітки кластерів та шуму:")
print(labels)
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title('DBSCAN')
plt.show()
```

Даний код виконує кластеризацію даних з використанням алгоритму DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Дана бібліотека sklearn.cluster.DBSCAN є класом, який надає реалізацію алгоритму кластеризації DBSCAN (Density-Based Spatial Clustering of Applications with Noise) в бібліотеці Scikit-learn (sklearn). DBSCAN є алгоритмом неконтрольованого навчання, який використовує щільність точок у просторі для кластеризації даних.

Результат виконання коду показано на рисунку 3.5.

```
K:\pyCharm\AnalyzingMethodsOfClusterization\venv\Scripts\python.exe K:\pyCharm
Мітки кластерів та шуму:
[-1 -1  0  0  1  1 -1  1  1  0  0  2  2  2  2  1  1  1  1 -1 -1  1  1  2
  2  1 -1 -1  1  1  1  1  1  1 -1 -1  1  1 -1 -1  3  3 -1 -1  4  3 -1 -1
  2  2 -1 -1  3  3  1  1  1  1  1  1 -1 -1 -1  3 -1  2 -1  1  1  1  1  2
  1  2  2  1  1  1  1  0  0  1  1  1  1  3  3  2  2  1  1  5  5  3  0 -1
  2  2 -1  4  1  1  1  1  3 -1 -1 -1  1  1 -1 -1  1  1  2  2 -1 -1  1  1
  0  0  0  0  2  2  2  2  1  1 -1 -1  1  1  1  1 -1  1  1  1  3  3  1  1
  1  1  1  1  4  4  1  1  1  1 -1 -1 -1 -1  1  1  1  1  2  2  1 -1 -1 -1
 -1  1  1  3  3  2  2 -1 -1  1  1  1  1  1  1  1  1  1  1  1 -1 -1  3  3 -1
 -1  5  5 -1 -1  2  2  2  2  1  1  3  3 -1 -1  1  1  1  2  2 -1 -1  3  3
  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1  1  1  1  1 -1 -1  1  1  3  3
  1  1  3 -1 -1 -1  1  1  1  1 -1 -1  5  5  1  1 -1 -1  1  1 -1 -1 -1 -1
  2  2  2  2 -1 -1  1  1  1 -1 -1 -1  1  1 -1 -1  1  1  0  0  1  1  1  1
  3  3 -1 -1  2 -1  2  2  2  2  1  1 -1 -1 -1 -1  1  1 -1 -1  3  3 -1 -1]

Process finished with exit code 0
```

Рисунок 3.5 – Мітки кластерів

Мітки кластерів і шуму, що повертаються алгоритмом DBSCAN, є числові значення, які присвоюються кожній точці даних у вибірці. Ось їх значення.

Кластерні позначки – кожна точка даних може бути віднесена до певного кластера. Кластерні позначки позначають належність точок даних до певних кластерів. Зазвичай кластерні мітки представлені цілими числами, де кожне число відповідає певному кластеру. Наприклад, якщо у вас є 3 кластери, мітки кластерів можуть бути 0, 1 та 2.

Мітка шуму – деякі точки даних можуть бути позначені як шум або викиди, тобто вони не належать жодному кластеру. Зазвичай мітка шуму позначається як -1 або окреме значення, яке відрізняється від міток кластерів.

Після застосування алгоритму DBSCAN та отримання міток кластерів та шуму, можна використовувати ці мітки для подальшого аналізу та інтерпретації результатів. Наприклад, можливо досліджувати різні кластери,

аналізувати їх характеристики та порівнювати їх між собою. Мітка шуму може бути корисною для виявлення аномалій або викидів у даних.

Дана програма буде візуалізувати діаграму розсіювання (рис. 3.6).

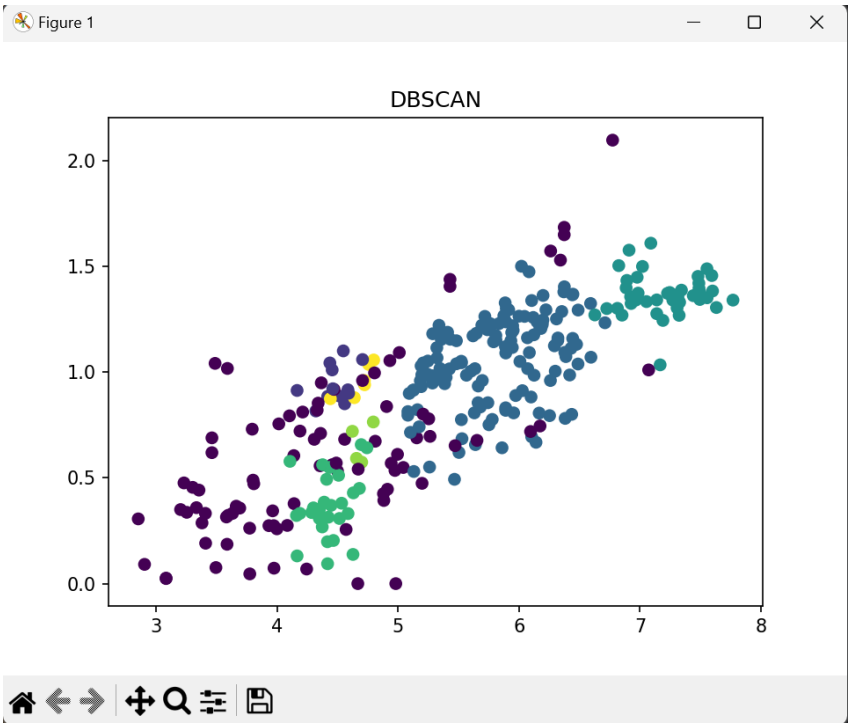



Рисунок 3.6 – Діаграма розсіювання

Діаграма розсіювання – це тип графіка, який використовується для візуалізації числових змінних або ознак на двовимірній площині. Вона складається з точок, розташованих на графіку відповідно до їх координат на осі *x* та *y*.

Діаграма розсіювання дозволяє досліджувати взаємозв'язок або розподіл між двома змінними. Кожна точка даних представляє одне спостереження або приклад, де значення на осі x відповідає значенню першої ознаки, а значення на осі y відповідає значенню другої ознаки.

Аналізуючи діаграму розсіювання, можна помітити такі характеристики:

– кореляція розташування точок на графіку може свідчити про наявність залежності між змінними. Якщо точки згруповані навколо певної лінії або

мають спільну тенденцію, це може вказувати на позитивну або від’ємну кореляцію між ознаками;

– кластеризація діаграма розсіювання також може використовуватись для візуалізації кластерної структури в даних. Якщо точки формують групи або скопища, це може вказувати на наявність кластерів в даних;

– аномалії окремі точки, відхилені від основної структури даних, можуть вказувати на аномалії або викиди.

Діаграма розсіювання є потужним інструментом для візуального аналізу даних і допомагає нам отримати уявлення про розподіл і взаємозв’язок змінних. У контексті алгоритму DBSCAN, діаграма розсіювання може бути корисною для візуалізації результатів кластеризації та розуміння структури даних.

Оцінка роботи алгоритму.

Лістинг 3.4 Код для оцінювання роботи алгоритму:

```
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn import metrics
import matplotlib.pyplot as plt


# Завантаження даних
data = pd.read_csv(‘happy.csv’)
X = data[[‘Score’, ‘GDP per capital’, ‘Social support’, ‘Healthy life expectancy’, ‘Freedom to make life choices’, ‘Generosity’, ‘Perceptions of corruption’]].values
df = pd.read_csv("happy.csv")
labels_true = df["Rate"].values
```

```
# Застосування алгоритму DBSCAN
dbscan = DBSCAN(eps=0.3, min_samples=5)
labels = dbscan.fit_predict(X)

# Виведення міток кластерів та шуму
print("Мітки кластерів та шуму:")
print(labels)

# Візуалізація результатів
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title('DBSCAN')
plt.show()

# Оцінка алгоритма

# Оцінка приблизної кількості кластерів
estimated_num_clusters = len(set(labels)) - (1 if -1 in labels else 0)
print("Оцінена кількість кластерів:", estimated_num_clusters)

# Оцінка приблизної кількості шумових точок
estimated_num_noise = list(labels).count(-1)
print("Оцінена кількість шумових точок:", estimated_num_noise)

# Обчислення метрик
homogeneity = metrics.homogeneity_score(labels_true, labels)
completeness = metrics.completeness_score(labels_true, labels)
v_measure = metrics.v_measure_score(labels_true, labels)
adjusted_rand_index = metrics.adjusted_rand_score(labels_true, labels)
adjusted_mutual_info = metrics.adjusted_mutual_info_score(labels_true,
labels)
```

```
silhouette_coefficient = metrics.silhouette_score(X, labels)

# Виведення результатів
print("Homogeneity:", homogeneity)
print("Completeness:", completeness)
print("V-measure:", v_measure)
print("Adjusted Rand Index:", adjusted_rand_index)
print("Adjusted Mutual Information:", adjusted_mutual_info)
print("Silhouette Coefficient:", silhouette_coefficient)
```

Результат роботи програми показано на рисунку 3.7.

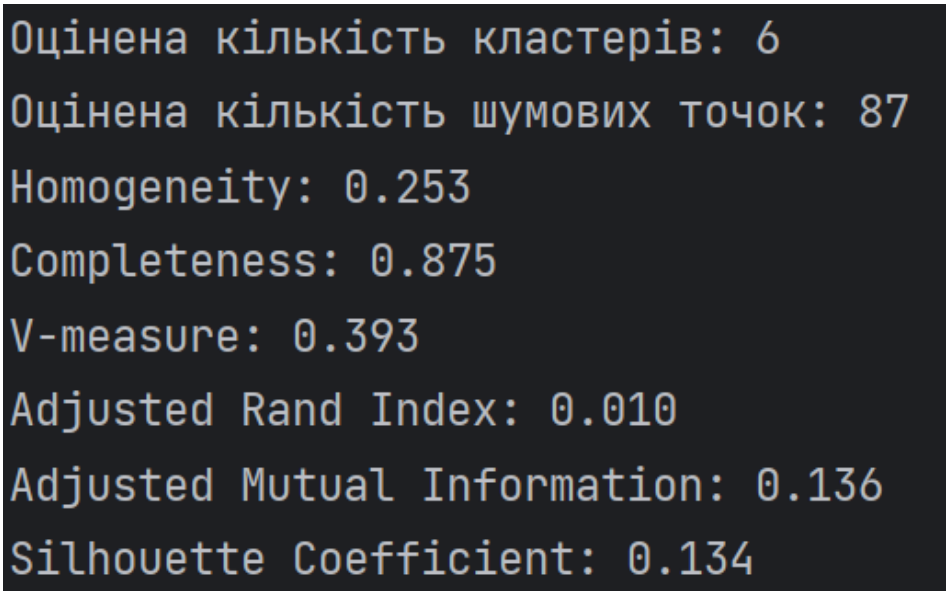



Рисунок 3.7 – Результат DBSCAN

### 3.3.2 Optics

Лістинг 3.5 Програмна реалізація OPTICS:

```
import numpy as np
```

```
import pandas as pd
from sklearn.cluster import OPTICS
from sklearn import metrics
import matplotlib.pyplot as plt

# Завантаження даних
data = pd.read_csv('happy.csv')
X = data[['Score', 'GDP per capital', 'Social support', 'Healthy life expectancy', 'Freedom to make life choices', 'Generosity', 'Perceptions of corruption']].values
labels_true = data["Overall rank"].values
# Застосування алгоритму OPTICS
optics = OPTICS(min_samples=5)
optics.fit(X)
# Отримання міток кластерів та шуму
labels = optics.labels_

# Виведення міток кластерів та шуму
print("Мітки кластерів та шуму:")
print(labels)

# Візуалізація результатів
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title('OPTICS')
plt.show()

# Оцінка алгоритма

# Оцінка приблизної кількості кластерів
estimated_num_clusters = len(set(labels)) - (1 if -1 in labels else 0)
```

```
print("Оцінена кількість кластерів:", estimated_num_clusters)

# Оцінка приблизної кількості шумових точок
estimated_num_noise = list(labels).count(-1)
print("Оцінена кількість шумових точок:", estimated_num_noise)

# Обчислення метрик
homogeneity = metrics.homogeneity_score(labels_true, labels)
completeness = metrics.completeness_score(labels_true, labels)
v_measure = metrics.v_measure_score(labels_true, labels)
adjusted_rand_index = metrics.adjusted_rand_score(labels_true, labels)
adjusted_mutual_info = metrics.adjusted_mutual_info_score(labels_true, labels)
completeness = metrics.completeness_score(labels_true, labels)
v_measure = metrics.v_measure_score(labels_true, labels)
adjusted_rand_index = metrics.adjusted_rand_score(labels_true, labels)
adjusted_mutual_info = metrics.adjusted_mutual_info_score(labels_true, labels)

silhouette_coefficient = metrics.silhouette_score(X, labels)

# Виведення результатів
print("Homogeneity: %0.3f" % homogeneity)
print("Completeness: %0.3f" % completeness)
print("V-measure: %0.3f" % v_measure)
print("Adjusted Rand Index: %0.3f" % adjusted_rand_index)
print("Adjusted Mutual Information: %0.3f" % adjusted_mutual_info)
print("Silhouette Coefficient: %0.3f" % silhouette_coefficient)
```

Результат роботи програми показано на рисунку 3.8.

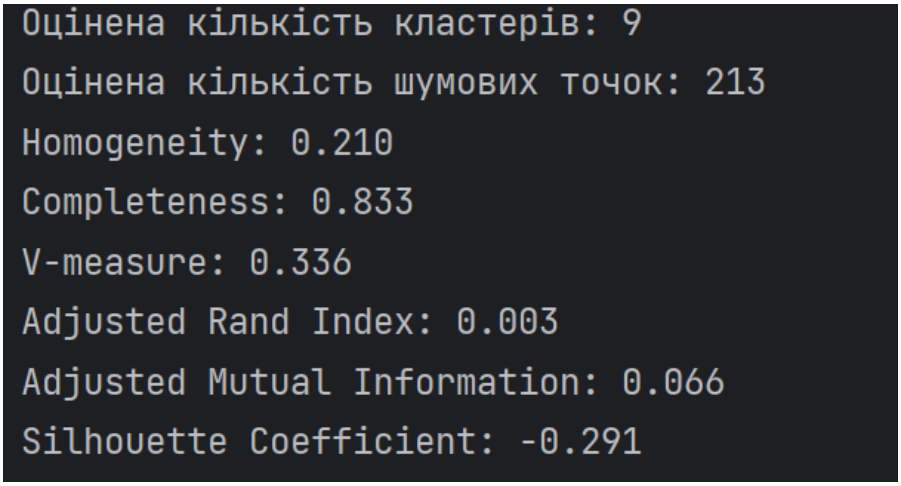



Рисунок 3.8 – результат роботи OPTICS

### 3.3.3 Denclue

Лістинг 3.6 Програмна реалізація DENCLUE:

```
import numpy as np
import pandas as pd
from DenclueAlg import Denclue
from sklearn import metrics
import matplotlib.pyplot as plt

# Завантаження даних
data = pd.read_csv('happy.csv')
X = data[['Score', 'GDP per capital', 'Social support', 'Healthy life expectancy', 'Freedom to make life choices', 'Generosity', 'Perceptions of corruption']].values
labels_true = data["Overall rank"].values
```

```
# Застосування алгоритму Denclue
denclue = Birch()
denclue.fit(X)
# Отримання міток кластерів
labels = denclue.labels_

# Виведення міток кластерів
print("Мітки кластерів:")
print(labels)

# Візуалізація результатів
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title('Denclue')
plt.show()

# Оцінка алгоритма

# Оцінка приблизної кількості кластерів
estimated_num_clusters = len(set(labels))
print("Оцінена кількість кластерів:", estimated_num_clusters)

# Обчислення метрик
homogeneity = metrics.homogeneity_score(labels_true, labels)
completeness = metrics.completeness_score(labels_true, labels)
v_measure = metrics.v_measure_score(labels_true, labels)
adjusted_rand_index = metrics.adjusted_rand_score(labels_true, labels)
adjusted_mutual_info = metrics.adjusted_mutual_info_score(labels_true,
labels)
silhouette_coefficient = metrics.silhouette_score(X, labels)
```

```
# Виведення результатів
print("Homogeneity: %0.3f" % homogeneity)
print("Completeness: %0.3f" % completeness)
print("V-measure: %0.3f" % v_measure)
print("Adjusted Rand Index: %0.3f" % adjusted_rand_index)
print("Adjusted Mutual Information: %0.3f" % adjusted_mutual_info)
print("Silhouette Coefficient: %0.3f" % silhouette_coefficient)
```

Результат роботи програми показано на рисунку 3.9.

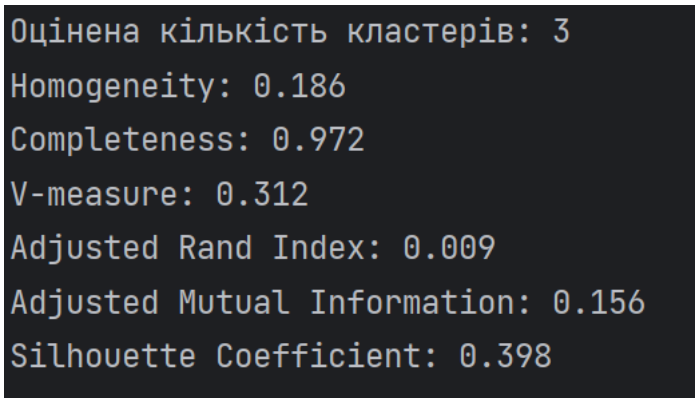



Рисунок 3.9 – Результат DENCLUE

Оцінка якості алгоритмів представлена у вигляді таблиці 3.1.

Таблиця 3.1 – Оцінка якостей алгоритмів

| **Назва методу** | **Homogeneity** | **Completeness** | **V-measure** |
|---|---|---|---|
| DBSCAN | 0,253 | 0,875 | 0,393 |
| OPTICS | 0,210 | 0,833 | 0,336 |
| DENCLUE | 0,186 | 0,972 | 0,312 |

Продовження таблиці 3.1

| **Назва методу** | **Adjusted Rand Index** | **Adjusted Mutual Information** | **Silhouette Coefficient** |
|---|---|---|---|
| DBSCAN | 0,01 | 0,136 | 0,134 |
| OPTICS | 0,003 | 0,066 | -0,291 |
| DENCLUE | 0,009 | 0,156 | 0,398 |

Проаналізувавши отримані, найкращі результати має алгоритм Denclue.

Також в бібліотеці sklearn, є порівняльна характеристика алгоритмів (рис. 3.10).

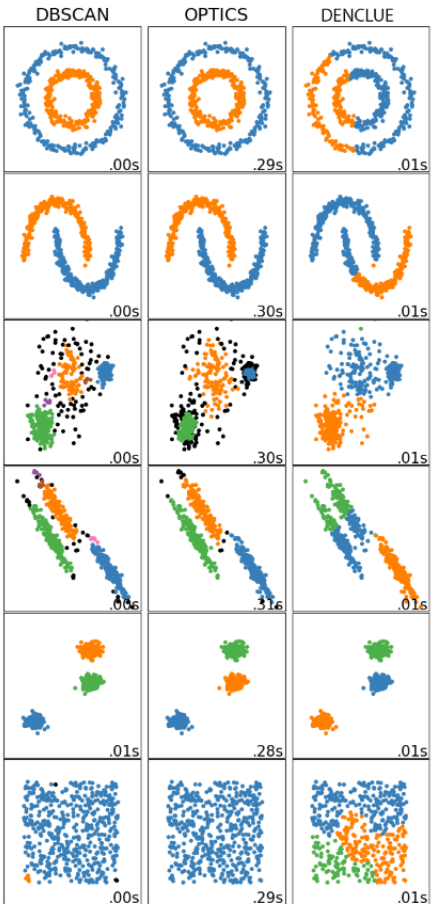



Рисунок 3.10 – Порівняльна таблиця

# ВИСНОВКИ

У рамках кваліфікаційної роботи були проаналізовані методи кластеризації даних на основі щільності розподілу даних. Було використано датасет, який має дані про рівень щастя людей у різних країнах.

Аналіз методів кластеризації на основі щільності розподілу даних, зокрема Denclue, може допомогти виявити цікаві закономірності та групування в наборі даних. Основна ідея за такими методами полягає в тому, щоб приділити увагу густині точок в просторі ознак, визначаючи регіони з високою щільністю як кластери.

Застосування методів кластеризації на основі щільності розподілу даних має свої переваги і обмеження. Однією з головних переваг є можливість виявлення нелінійних та нерегулярних структур даних. Вони можуть ефективно розпізнавати складні форми кластерів, які не завжди можуть бути виявлені за допомогою інших методів кластеризації, таких як k-середніх або ієрархічна кластеризація.

Однак методи кластеризації на основі щільності розподілу даних також мають свої обмеження. Одним з основних обмежень є необхідність налаштування параметрів, таких як ширина ядра та порогові значення щільності, що можуть впливати на результати кластеризації. Відповідний вибір цих параметрів може бути нетривким завданням, особливо в разі складних даних або великих наборів даних.

Загалом, методи кластеризації на основі щільності розподілу даних є потужними інструментами для виявлення складних структур та групування даних.

Результати роботи апробовано у вигляді тези доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [31].

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, IEEE Access, 10, pp. 124738-124746.

2. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

3. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, Сучасні інформаційні системи, 6(3), С. 5-12.

4. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, Сучасні інформаційні системи, 7(1), С. 5-13.

5. Abourezq, M., & Idrissi, A. (2016). Database-as-a-service for big data: An overview. *International Journal of Advanced Computer Science and Applications*, *7*(1).

6. Kalibjian, J. (2013). " Big Data" Management and Security Application to Telemetry Data Products. In International Telemetering Conference Proceedings (Vol. 49).

7. Sarbaini, S., Saputri, W., & Muttakin, F. (2022). Cluster Analysis Menggunakan Algoritma Fuzzy K-Means Untuk Tingkat Pengangguran Di Provinsi Riau. Jurnal Teknologi Dan Manajemen Industri Terapan, 1(2), 78-84.

8. Fernandez-Yague, M. A., Hymel, L. A., Olingy, C. E., McClain, C., Ogle, M. E., García, J. R., ... & Botchwey, E. A. (2022). Analyzing immune response to

engineered hydrogels by hierarchical clustering of inflammatory cell subsets. Science Advances, 8(8), eabd8056.

9. Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., & Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Engineering Applications of Artificial Intelligence, 110, 104743.

10. Clustering scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/clustering.html (дата звернення 06.05.2023).

11. Metrics and scoring: quantifying the quality of predictions documentation. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата звернення 12.05.2023).

12. Бодянський, Є. В., Шафроненко, А. Ю., & Климова, І. М. (2021). Метод адаптивної достовірної нечіткої кластеризації даних на основі еволюційного алгоритму. Збірник наукових праць Харківського національного університету Повітряних Сил, (2 (68)), 80-83.

13. Бодянський, Є. В., Шафроненко, А. Ю., & Климова, І. М. (2020). Рекурентна достовірна нечітка кластеризація великих даних з використанням функції належності спеціального типу.

14. Шафроненко, А. Ю., & Москаленко, В. В. (2021, December). ПРАВДОПОДІБНА НЕЧІТКА КЛАСТЕРИЗАЦІЯ ДАНИХ НА ОСНОВІ ЕВОЛЮЦІЙНИХ ПРОЦЕДУР. In The 5th International scientific and practical conference “Science, innovations and education: problems and prospects”(December 8-10, 2021) CPN Publishing Group, Tokyo, Japan. 2021. 1068 p. (p. 383).

15. Bodyanskiy, Y., Shafronenko, A., & Mashtalir, S. (2020). Online robust fuzzy clustering of data with omissions using similarity measure of special type. In Lecture Notes in Computational Intelligence and Decision Making: Proceedings of the XV International Scientific Conference “Intellectual Systems of Decision

Making and Problems of Computational Intelligence"(ISDMCI'2019), Ukraine, May 21–25, 2019 15 (pp. 637-646). Springer International Publishing.

16. Shafronenko, A., Bodyanskiy, Y. V., Klymova, I., & Holovin, O. (2020, May). Online credibilistic fuzzy clustering of data using membership functions of special type. In CMIS (pp. 744-753).

17. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Irina, K. (2021, September). Online Credibilistic Fuzzy Clustering Method Based on Cauchy Density Distribution Function. In 2021 11th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 704-707). IEEE.

18. Hu, Z., Bodyanskiy, Y. V., Tyshchenko, O. K., & Shafronenko, A. (2019, July). Fuzzy clustering of incomplete data by means of similarity measures. In 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON) (pp. 957-960). IEEE.

19. Bodyanskiy, Y. V., Shafronenko, A., & Klymova, I. (2021, April). Adaptive Recovery of Distorted Data Based on Credibilistic Fuzzy Clustering Approach. In COLINS (pp. 6-15).

20. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Popov, S. (2020, September). Evolving neo-fuzzy system for distorted data online processing. In 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 352-355). IEEE.

21. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Popov, S. (2020, September). Evolving neo-fuzzy system for distorted data online processing. In 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 352-355). IEEE.

22. Bodyanskiy, Y., Shafronenko, A., & Volkova, V. (2012). Adaptive clustering of incomplete data using neuro-fuzzy Kohonen network. Artificial Intelligence Methods and Techniques for Business and Engineering Applications"–Rzeszow-Sofia: ITHEA, 287-296.

23. Shafronenko, A., & Bodyanskiy, Y. V. (2020). Adaptive fuzzy clustering approach based on evolutionary cat swarm optimization. In CMIS (pp. 832-842).

24. Bodyanskiy, Y. V., Shafronenko, A., & Rudenko, D. (2019). Online Neuro Fuzzy Clustering of Data with Omissions and Outliers based on Completion Strategy. In CMIS (pp. 18-27).

25. Bodyanskiy, Y., Shafronenko, A., Klymova, I., & Polyvoda, V. (2022). Robust Recurrent Credibilistic Modification of the Gustafson-Kessel Algorithm. In Lecture Notes in Computational Intelligence and Decision Making: 2021 International Scientific Conference" Intellectual Systems of Decision-making and Problems of Computational Intelligence", Proceedings (pp. 613-623). Springer International Publishing.

26. Shafronenko, A., Bodyanskiy, Y., & Rudenko, D. (2020). Neuro-fuzzy clustering of Distorted Data Using Cat Swarm Optimization. LAP LAMBERT Academic Publishing.

27. Bodyanskiy, Y., Shafronenko, A., & Pliss, I. (2022). Clusterization of vector and matrix data arrays using the combined evolutionary method of fish schools. Системні дослідження та інформаційні технології: міжнародний науково-технічний журнал,№ 4.

28. Ішков, В. В., Козій, Є. С., Чернобук, О. І., Васильченко, Н. В., & Кузнецова, С. С. (2022). Аналіз методів кластеризації ділянок різної потужності вугільного пласта для створення їх природної типізації за вмістом германію (на прикладі пласта с6 шахти «Дніпровська»).

29. Ali, N., Chen, J., Fu, X., Hussain, W., Ali, M., Iqbal, S. M., ... & Thanh, H. V. (2023). Classification of reservoir quality using unsupervised machine learning and cluster analysis: Example from Kadanwari gas field, SE Pakistan. Geosystems and Geoenvironment, 2(1), 100123.

30. Biasetton, N., Disegna, M., Barzizza, E., & Salmaso, L. (2023). A new adaptive membership function with CUB uncertainty with application to cluster analysis of Likert-type data. Expert Systems with Applications, 213, 118893.

31. Авлякулов Т.Е. (2023). НЕЧІТКА КЛАСТЕРИЗАЦІЯ ДАНИХ З РІЗНОЮ ЩІЛЬНІСТЮ РОЗПОДІЛУ: 27-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI стол ітті». Зб. матеріалів форуму. Т. 7. Харків: ХНУРЕ. С. 78-79.