math_2023_056

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА**

**Факультет математики та інформатики**
**кафедра математичного моделювання**

# ПОРІВНЯННЯ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДАНИХ

**Кваліфікаційна робота**
**Рівень вищої освіти – перший (бакалавр)**

***Виконав:***
студент 4 курсу,
групи 407
**Микита Крештанович**
***Керівник:***
кандидат фізико-математичних наук,
доцент
**Дорошенко Ірина Вікторівна**

***До захисту допущено***
***на засіданні кафедри***
***протокол № _ від «_»_____ 2023 р.***
***Зав. кафедрою _____ проф. Черевко І.М.***

**Чернівці – 2023**

## Анотація

В даній роботі розглянуто порівняння алгоритмів кластеризації даних: K-Means, Hierarchical Agglomerative Clustering (HAC), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization clustering using Gaussian Mixture Models (GMM). Порівняння здійснюється завдяки наперед згенерованим наборам даних, які мають різний характер поведінки: концентричні кола (2 кластери), смужки (3), хмари (3), нероздільна множина (1), серпи (2). Для кожного з наборів даних застосовано перелічені методи і визначено найкращий алгоритм кластеризації для певного типу даних. Алгоритми кластеризації даних застосовано до трьох наборів реальних даних. Створено інтерактивний веб-застосунок для інтерактивної кластеризації даних згаданими алгоритмами, який розгорнуто на хмарному сервері shinyapps.io.



***Ключові слова:*** кластеризація даних, кластерний аналіз, K-Means, Hierarchical Agglomerative Clustering (HAC), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization clustering using Gaussian Mixture Models (GMM).



## Annotation

This work deals with the comparison of data clustering algorithms: K-Means, Hierarchical Agglomerative Clustering (HAC), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization clustering using Gaussian Mixture Models (GMM). The comparison is made thanks to pre-generated data sets that have different behavior: concentric circles (2 clusters), stripes (3), clouds (3), inseparable set (1), crescents (2). For each of the data sets, the listed methods are applied and the best clustering algorithm for a certain type of data is determined. Data clustering algorithms were applied to three sets of real data. An interactive web application for interactive data clustering using the mentioned algorithms has been created, which is deployed on the shinyapps.io cloud server.



***Keywords:*** data clustering, cluster analysis, K-Means, Hierarchical Agglomerative Clustering (HAC), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization clustering using Gaussian Mixture Models (GMM).


Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ **Микита Крештанович**
(підпис)

# ЗМІСТ

# ВСТУП

Кластерний аналіз – це неконтрольований процес, який поділяє набір об’єктів на однорідні групи. Сьогодні існує безліч алгоритмів кластеризації, описаних у наукових публікаціях, що стосуються різних сфер життя: розпізнавання образів, штучному інтелекті, інформаційних технологіях, обробці зображень, біології, психології, маркетингу тощо. Виходячи з цього, користувачам часто важко визначити відповідний алгоритм для власних досліджень або порівняти нові ідеї з наявними результатами.

Проблема кластеризації даних широко вивчається в літературі для аналізу даних та машинного навчання у різних сферах життєдіяльності людини. За відсутності конкретної позначеної інформації, кластеризацію можна вважати короткою моделлю даних, яку можна інтерпретувати в сенсі підсумкової або генеративної моделі. Основну задачу кластеризації можна сформулювати так: маючи набір точок даних, розділити їх на набір груп, які максимально відрізняються.

Це дуже приблизне визначення і варіації у визначенні проблеми можуть бути значними залежно від конкретної моделі, що використовується. Наприклад, генеративна модель може визначати подібність на основі ймовірнісного генеративного механізму, тоді як підхід, заснований на відстані, використовуватиме для кількісного визначення традиційну функцію відстані. Крім того, конкретний тип даних також має значний вплив на визначення проблеми.

**Метою роботи** є дослідження, практичне застосування та порівняння існуючих методів кластеризації даних K-Means, Hierarchical Agglomerative Clustering (HAC), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization clustering using Gaussian Mixture Models (GMM).

**Об’єктом дослідження** є використання різних за характером скупчення та поведінки даних у процесі кластеризації.

**Предмет дослідження** – застосування алгоритмів кластеризації даних:

K-Means, HAC, DBSCAN, GMM.

**Основні завдання роботи:**

- ознайомитися з загальними прийомами, що використовуються у кластерному аналізі;
- охарактеризувати типи даних, що застосовуються у кластеризації;
- проілюструвати переваги та недоліки різних алгоритмів кластеризації;
- здійснити кластеризацію даних за допомогою різних алгоритмів кластеризації;
- описати, у яких випадках краще використовувати певний алгоритм;
- створити Shiny веб-застосунок для інтерактивної кластеризації даних;
- застосувати розглянуті методи кластеризації до реальних даних.

## РОЗДІЛ І. Загальні підходи, що використовуються у кластерному аналізі

Кластеризація даних (або просто кластеризація), яку також називають кластерним аналізом, аналізом сегментації, таксономічним аналізом або неконтрольованою класифікацією, є методом створення груп об’єктів або кластерів таким чином, що об’єкти в одному кластері є дуже схожими, а об’єкти в різних кластерах – досить різні.

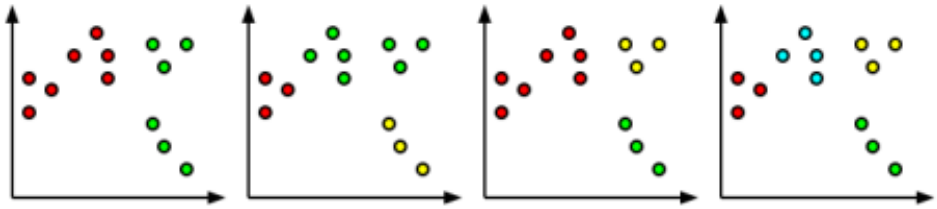


Рисунок 1. Приклади наборів даних, які можна кластеризувати

Проблема кластеризації виникає серед таких доменів додатків:

• Проміжний крок для інших фундаментальних проблем інтелектуального аналізу даних: оскільки кластеризацію можна вважати формою узагальнення даних, він часто служить ключовим проміжним кроком для багатьох фундаментальних проблем інтелектуального аналізу даних, таких як класифікація або аналіз викидів. Компактний підсумок даних часто буває корисним для різних типів аналізу конкретних програм.

• Спільна фільтрація: в методах спільної фільтрації кластеризація забезпечує узагальнення користувачів-однодумців. Оцінки, надані різними користувачами один одному, використовуються для виконання спільної фільтрації. Це можна використовувати для надання рекомендацій у різноманітних програмах.

• Сегментація клієнтів: ця програма дуже схожа на спільне фільтрування, оскільки створює групи подібних клієнтів у даних. Основна відмінність від спільної фільтрації полягає в тому, що замість використання рейтингової інформації для цілей кластеризації можна використовувати довільні атрибути об’єктів.

• Узагальнення даних: багато методів кластеризації тісно пов’язані з методами зменшення розмірності. Такі методи можна вважати формою узагальнення даних. Узагальнення даних може бути корисним для створення компактних представлень даних, які легше обробляти та інтерпретувати в широкому спектрі програм.

• Динамічне виявлення трендів: багато форм динамічних і потокових алгоритмів можна використовувати для виконання виявлення трендів у різноманітних програмах соціальних мереж. У таких програмах дані динамічно кластеризуються потоковим способом і можуть використовуватися для визначення важливих моделей змін. Прикладами таких потокових даних можуть бути багатовимірні дані, текстові потоки, потокові дані часових рядів і дані траєкторії. Ключові тенденції та події в даних можна виявити за допомогою методів кластеризації.

• Аналіз мультимедійних даних: низка різних видів документів, таких як зображення, аудіо чи відео, підпадають під загальну категорію мультимедійних даних. Визначення подібних сегментів має численні застосування, наприклад, визначення подібних фрагментів музики чи подібних фотографій. У багатьох випадках дані можуть бути мультимодальними та містити різні типи. У таких випадках проблема стає ще більш складною.

• Аналіз біологічних даних: біологічні дані стали всеохоплюючими за останні кілька років завдяки успіху досліджень геному людини та збільшенню можливостей збору різних видів даних про експресію генів. Біологічні дані зазвичай структуровані або як послідовності, або як мережі. Алгоритми кластеризації дають гарне уявлення про ключові тенденції в даних, а також про незвичні послідовності.

• Аналіз соціальної мережі: у цих програмах структура соціальної мережі використовується для визначення важливих спільнот у базовій мережі. Виявлення спільноти має важливе застосування в аналізі соціальних мереж, оскільки воно забезпечує важливе розуміння структури спільноти в

мережі. Кластеризація також має застосування для узагальнення соціальних мереж, що корисно в ряді програм.

Вищезазначений список програм не є вичерпним, але в той же час він демонструє широке розмаїття проблем, які можна вирішити за допомогою алгоритмів кластеризації. Робота в сфері кластеризації даних зазвичай поділяється на кілька широких категорій:

• орієнтація на техніку: оскільки кластеризація є досить популярною проблемою, не дивно, що численні методи, такі як імовірнісні методи, методи на основі відстані, спектральні методи, методи на основі щільності та методи на основі зменшення розмірності, використовуються для процесу кластеризації. Кожен із цих методів має свої переваги та недоліки та може добре працювати в різних сценаріях і проблемних областях. Певні типи даних, такі як дані великої розмірності, великі дані або потокові дані, мають власний набір проблем і часто вимагають спеціальних методів.

• центрований тип даних: різні програми створюють різні типи даних з різними властивостями. Наприклад, апарат ЕКГ створюватиме точки часових рядів, які сильно корелюють один з одним, тоді як соціальна мережа створюватиме суміш документів і структурних даних. Деякі з найпоширеніших прикладів – це категоричні дані, дані часових рядів, дискретні послідовності, мережеві дані та ймовірнісні дані. Очевидно, що характер даних значною мірою впливає на вибір методології, яка використовується для процесу кластеризації. Крім того, деякі типи даних є складнішими, ніж інші, через поділ різних типів атрибутів, таких як поведінка або контекстуальні атрибути.

• додаткові відомості про варіації кластеризації: низка ідей також розроблена для різних типів варіацій кластеризації. Наприклад, візуальний аналіз, контрольований аналіз, ансамблевий аналіз або багаторакурсний аналіз можна використовувати для отримання додаткової інформації. Крім того, проблема перевірки кластерів також важлива з точки зору отримання конкретної інформації про продуктивність кластеризації.

Проблеми кластеризації можна вирішувати за допомогою різноманітних методів. Крім того, для етапу попередньої обробки даних потрібні спеціальні методи.

## 1.1. Методи вибору функцій

Фаза вибору функції є важливим етапом попередньої обробки, який необхідний для підвищення якості основної кластеризації. Не всі функції однаково актуальні для пошуку кластерів, оскільки деякі можуть бути більш шумними, ніж інші. Тому часто корисно використовувати фазу попередньої обробки, на якій галасливі та нерелевантні функції відсікаються від суперечок. Вибір ознак і зменшення розмірності тісно пов'язані. Під час вибору об'єктів вибираються оригінальні підмножини об'єктів. У зменшенні розмірності лінійні комбінації ознак можуть бути використані в таких методах, як аналіз головних компонентів [32], щоб ще більше посилити ефект вибору ознак. Перевагою першого є більша інтерпретабельність, тоді як перевагою останнього є те, що для процесу представлення потрібна менша кількість перетворених напрямків.

Слід зазначити, що вибір функцій також можна інтегрувати безпосередньо в алгоритм кластеризації, щоб отримати кращу інформацію про місцевість. Це особливо корисно, коли різні функції мають відношення до різних місцевостей даних. Мотивуючим фактором для алгоритмів кластеризації підпростору великої розмірності є невдача алгоритмів глобального вибору ознак. Як зазначено в [7]: *..багато реальних прикладів даних, деякі точки корельовані щодо заданого набору вимірів, а інші – щодо різних вимірів. Таким чином, не завжди можливо відсікти занадто багато вимірів без водночас значної втрати інформації» стор.61.* Тому використання локального вибору ознак шляхом інтеграції процесу вибору ознак в алгоритм є найкращим способом досягнення цієї мети. Такі локальні виділення ознак також можуть бути розширені до проблеми зменшення розмірності [6, 11] і іноді називаються локальним зменшенням розмірності.

## 1.2. Імовірнісні та генеративні моделі

У ймовірнісних моделях основна ідея полягає в моделюванні даних із *генеративного процесу*. Спочатку передбачається конкретна форма генеративної моделі (наприклад, суміш гаусів), а потім параметри цієї моделі оцінюються за допомогою алгоритму Максимізації Очікування (МО) [14]. Доступний набір даних використовується для оцінки параметрів таким чином, щоб вони мали *максимальну вірогідність підходу* до генеративної моделі. Враховуючи цю модель, ми потім оцінюємо генеративні ймовірності (або ймовірності відповідності) базових точок даних. Точки даних, які добре відповідають розподілу, матимуть високу ймовірність відповідності, тоді як аномалії матимуть дуже низьку ймовірність відповідності.

Загальний принцип генеративної моделі на основі суміші полягає в *припущенні*, що дані були згенеровані із суміші $k$ розподілів із розподілами ймовірностей $G_1 \dots G_k$ за допомогою наступного процесу:

Виберемо розподіл даних з апріорною ймовірністю $a_i$, де $i \in \{1 \dots k\}$, щоб вибрати один із $k$ розподілів. Припустимо, що вибрано $r$. Згенеруємо точку даних $G_r$. Розподіл ймовірностей $G_r$ обирається з безлічі різних можливостей. Цей генеративний процес вимагає визначення кількох параметрів, таких як попередні ймовірності та параметри моделі для кожного розподілу $G_r$. Моделі з різними рівнями гнучкості можуть бути розроблені в залежності від того, чи визначені попередні ймовірності як частина постановки проблеми, чи передбачається кореляція між атрибутами в межах компонента суміші. Параметри моделі та ймовірність призначення точок даних кластерам циклічно залежать один від одного. Отож, ітераційні методи потрібні для вирішення даної циклічності. Генеративні моделі зазвичай розв'язуються з використанням підходу МО, який починається з випадкової або евристичної ініціалізації, а потім ітеративно використовує два кроки для усунення циклічного обчислення:

- (E-Step) Визначити очікувану ймовірність віднесення точок даних до

кластерів з використанням поточних параметрів моделі.

- (M-Step) Визначити оптимальні параметри моделі кожної суміші, використовуючи ймовірності призначення як вагові коефіцієнти.

Основна властивість МО-моделей полягає в тому, що їх можна відносно легко узагальнити для різних типів даних, якщо генеративна модель для кожного компонента правильно вибрана для окремого компонента суміші $G_r$. До прикладу:

- Для числових даних можна використовувати модель суміші Гауса для моделювання кожного компонента $G_r$.

- Для категоріальних даних модель Бернуллі може бути використана для $G_r$, щоб моделювати генерацію дискретних значень.

- Для даних послідовності можна використовувати приховану модель Маркова (Hidden Markov Model) для $G_r$, щоб змоделювати генерацію послідовності. Цікаво, що HMM сама по собі є особливим видом моделі суміші, в якій різні компоненти суміші залежать один від одного через переходи. Таким чином, кластеризацію даних послідовності з сумішшю HMM можна вважати дворівневою моделлю суміші.

Генеративні моделі є одними з найбільш фундаментальних методів кластеризації, оскільки вони намагаються зрозуміти основний *процес*, за допомогою якого створюється кластер. Існує ряд цікавих зв'язків між іншими методами кластеризації та генеративними моделями, розглядаючи особливі випадки з точки зору попередніх ймовірностей або параметрів суміші. До прикладу, окремий випадок, у якому кожна попередня ймовірність фіксується на тому самому значенні, а всі компоненти суміші, як припускається, мають однаковий радіус уздовж усіх вимірів, зводиться до м'якої версії алгоритму $k$-середніх.

## 1.3. Алгоритми на основі відстані

Можна стверджувати, що багато спеціальних форм генеративних алгоритмів зводяться до алгоритмів на основі відстані. Це пояснюється тим, що компоненти суміші в генеративних моделях часто використовують функцію відстані в межах розподілу ймовірностей. До прикладу, розподіл Гауса представляє ймовірності генерації даних у термінах евклідової відстані від середнього значення суміші. У результаті можна показати, що генеративна модель із розподілом Гауса має дуже тісний зв'язок із алгоритмом $k$-середніх. Насправді можна стверджувати, що багато алгоритмів, заснованих на відстані, є скороченнями або спрощеннями різних типів генеративних моделей.

Методи, засновані на дистанції, часто є бажаними через їхню простоту та легкість реалізації в різноманітних сценаріях. Алгоритми на основі відстані можна загалом розділити на два типи:

• *Плоский:* у цьому випадку дані поділяються на кілька кластерів за один кадр, як правило, з використанням представників розділення. Вибір репрезентативної функції поділу та функції відстані є вирішальним і регулює поведінку основного алгоритму. У кожній ітерації точки даних призначаються їхнім найближчим представникам поділу, а потім представник коригується відповідно до точок даних, призначених кластеру. Доцільно порівняти це з ітераційною природою алгоритму МО, в якому м'які призначення виконуються на М-кроці, а параметри моделі (аналогічні представникам кластера) коригуються на О-кроці. Деякі поширені методи створення розділів:

- *k-Середні (k-Means:):* у цих методах представники поділу відповідають середньому значенню кожного кластера. Представник розділення не витягується з вихідного набору даних, а створюється як функція базових даних. Евклідова відстань використовується для обчислення відстаней. Метод k-Середні вважається одним із найпростіших і класичних методів кластеризації даних [29], а також, мабуть, одним із найбільш широко

використовуваних методів у практичних реалізаціях через його простоту.

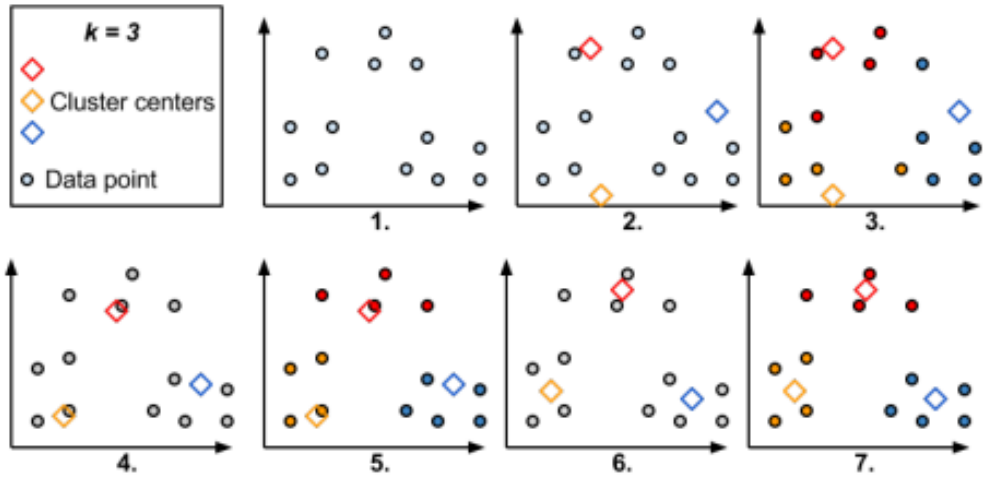



Рисунок 2. Класифікація нової точки даних залежатиме від вибраного розміру k.

- *k-Медіани ( k-Medians):* у цих методах медіана вздовж кожного виміру замість середнього використовується для створення представника поділу. Як і у випадку підходу k-Середні, представники поділу не беруться з вихідного набору даних. Підхід k-Медіан є більш стійким до шуму та викидів, оскільки медіана набору значень зазвичай менш чутлива до екстремальних значень у даних. Слід також зазначити, що термін «k-Медіани» іноді перевантажений у літературі, оскільки він також використовується для позначення підходу k-Медоїд, у якому представники поділу беруться з оригінальних даних. Методи k-Медіани і k-Медоїд слід розглядати як різні методи.

- *k-Медоїд (k-Medoid):* у цих методах представник поділу вибирається з вихідних даних. Такі методи особливо корисні, коли точки даних для кластеризації є довільними об’єктами, і часто немає сенсу говорити про функції цих об’єктів. До прикладу, для набору мережевих або дискретних об’єктів послідовності може бути недоцільним акцент на їхньому середньому значенні або медіані. У таких випадках представники поділу витягуються з даних, а для покращення якості цих представників використовуються

ітераційні методи. У кожній ітерації один із представників замінюється на представника з поточних даних, щоб перевірити, чи покращується якість кластеризації. Цей метод зазвичай вимагає набагато більше ітерацій, ніж попередні. Однак, на відміну від попередніх двох методів, останній можна використовувати в сценаріях, де немає сенсу говорити про середні значення або медіани об'єктів даних (до прикладу, об'єктів структурних даних).

• *Ієрархічний:* у цих методах кластери представлені ієрархічно за допомогою дендограми з різними рівнями деталізації. Залежно від того, чи створено це ієрархічне представлення зверху вниз чи знизу вгору, ці представлення можна вважати агломеративними або роздільними.

*Агломеративний:* даний метод використовує висхідний підхід, у якому все починається з окремих точок даних і здійснюється послідовне об'єднання кластерів, щоб створити деревоподібну структуру. Можливі різноманітні варіанти щодо того, як ці кластери можуть бути об'єднані, що забезпечує різні компроміси між якістю та ефективністю. Деякими прикладами таких варіантів є однозв'язування, зв'язування всіх пар, зв'язування центроїда та кластеризація з вибірковим зв'язком. У кластеризації з одним зв'язком використовується найкоротша відстань між будь-якою парою точок у двох кластерах. У зв'язку всіх пар використовується середнє значення за всіма парами, тоді як у зв'язку вибірки для обчислення середньої відстані використовується вибірка точок даних у двох кластерах. У центроїдному зчепленні використовується відстань між центроїдами. Деякі варіації цих методів мають недолік у з'єднанні ланцюжків, у якому більші кластери природним чином схильні мати ближчу відстань до інших точок і, отже, залучатимуть все більшу кількість точок. Одинарна тяга кластеризації особливо чутлива до цього явища. Деякі домени даних, такі як мережеві кластери, також більш сприйнятливі до такої поведінки.

- *Розділовий:* у даному методі використовується низхідний підхід для послідовного розділення точок даних у деревоподібну структуру. Для виконання розділення на кожному кроці можна використовувати будь-який

плоский алгоритм кластеризації. Поділ, що розділяє, забезпечує більшу гнучкість як щодо ієрархічної структури дерева, так і рівня балансу в різних кластерах. Не обов'язково мати ідеально збалансоване дерево з точки зору глибини різних вузлів або дерево, у якому ступінь кожної гілки дорівнює рівно двом. Це дозволяє побудувати структуру дерева, яка допускає різні компроміси в балансуванні глибини вузла та ваги вузла (кількості точок даних у вузлі). До прикладу, у низхідному методі, якщо різні гілки дерева незбалансовані з точки зору ваги вузлів, тоді для поділу на кожному рівні можна вибрати найбільший кластер. Такий підхід [35] використовується в METIS для створення добре збалансованих кластерів у великих соціальних мережах, в яких проблема дисбалансу кластерів є особливо гострою. Хоча METIS не є алгоритмом на основі відстані, ці загальні принципи також застосовуються до алгоритмів на основі відстані.

Методи, засновані на відстані, дуже популярні, оскільки їх можна використовувати практично з будь-яким типом даних, якщо для цього типу даних створено відповідну функцію відстані. Проблема кластеризації може бути зведена до проблеми пошуку функції відстані для цього типу даних. Таким чином, розробка функції відстані сама по собі стала важливою сферою дослідження для інтелектуального аналізу даних [3, 52]. Спеціальні методи також часто розроблялися для конкретних областей даних, таких як дані категорій або часових рядів [18, 25]. Звичайно, у багатьох сфераxx, таких як дані великої розмірності, якість функцій відстані знижується через багато нерелевантних розмірів [26] і може демонструвати як помилки, так і ефекти концентрації, що знижує статистичну значущість результатів інтелектуального аналізу даних. У таких випадках можна використовувати або надмірність у більших частинах матриці попарної відстані, щоб абстрагувати шум у обчисленнях відстані за допомогою спектральних методів [11], або проекції, щоб безпосередньо знайти кластери у відповідних підмножинах атрибутів [7].

## 1.4. Методи на основі щільності та сітки

Методи на основі щільності та сітки є двома тісно пов’язаними класами, які намагаються дослідити простір даних на високому рівні деталізації. Щільність у будь-якій конкретній точці простору даних визначається або в термінах кількості точок даних у заздалегідь заданому обсязі його локалізації, або в термінах більш гладкої оцінки щільності ядра [48]. Як правило, простір даних досліджується на досить високому рівні деталізації, і використовується фаза постобробки, щоб «з’єднати» щільні області простору даних у довільну форму. Методи на основі сітки – це специфічний клас методів на основі щільності, у яких окремі регіони простору даних, що досліджуються, формуються у структуру, подібну до сітки. Структури, подібні до сітки, часто особливо зручні через більшу легкість у складанні різних щільних блоків на етапі пост-обробки. Такі сітоподібні методи також можна використовувати в контексті методів високої розмірності, оскільки сітки меншої розмірності визначають кластери на підмножинах розмірностей [4].

Основною перевагою цих методів є те, що оскільки вони досліджують простір даних на високому рівні деталізації, їх можна використовувати для реконструкції повної форми розподілу даних. Двома класичними методами для методів на основі щільності та методів на основі сітки є DBSCAN [20] та STING [53] відповідно. Основна проблема методів на основі щільності полягає в тому, що вони природним чином визначені на точках даних у безперервному просторі. Тому вони часто не можуть бути осмислено використані в дискретному або неевклідовому просторі, якщо не використовується підхід вбудовування. Таким чином, багато довільних типів даних, таких як дані часових рядів, не так легко використовувати з методами на основі щільності без спеціальних перетворень. Інша проблема полягає в тому, що обчислення щільності стає все важче визначити з більшою розмірністю через більшу кількість клітинок у базовій структурі сітки та

розрідженість даних у базовій сітці.

## 1.5. Використання методів зменшення розмірності

Методи зменшення розмірності тісно пов’язані як з вибором ознак, так і з групуванням, оскільки вони намагаються використовувати близькість і кореляцію між розмірами для зменшення розмірності представлення. Таким чином, методи зменшення розмірності часто можна вважати вертикальною формою кластеризації, у якій стовпці даних кластеризуються за допомогою аналізу кореляції або близькості, на відміну від рядків. Тому виникає природне запитання, чи можливо виконати ці кроки одночасно, шляхом кластеризації рядків і стовпців даних разом. Ідея полягає в тому, що одночасна кластеризація рядків і стовпців, швидше за все, буде ефективнішою, ніж виконання кожного з цих кроків окремо. Цей ширший принцип привів до численних алгоритмів, таких як матрична факторизація, спектральна кластеризація, ймовірнісне приховане семантичне індексування та співкластеризація. Деякі з цих методів, наприклад спектральна кластеризація, дещо відрізняються, але, тим не менш, базуються на тій самій загальній концепції. Ці методи також тісно пов’язані з методами прогнозованої кластеризації, які зазвичай використовуються для даних великої розмірності в літературі з баз даних.

### 1.5.1. Генеративні моделі для зменшення розмірності

У цих моделях генеративний розподіл ймовірностей використовується для інтегрованого моделювання зв’язків між точками даних і вимірами. До прикладу, узагальнений розподіл Гаусса можна вважати сумішшю довільно корельованих (орієнтованих) кластерів, параметри яких можна дізнатися за допомогою МО-алгоритму. Звичайно, це часто непросто зробити надійно зі збільшенням розмірності через більшу кількість параметрів, залучених до процесу навчання. Добре відомо, що такі методи, як МО, дуже чутливі до

переобладнання, в якому кількість параметрів значно збільшується. Це пояснюється тим, що МО-методи намагаються зберегти всю інформацію в термінах м’яких ймовірностей, замість того, щоб робити важкий вибір щодо вибору точок і розмірів непараметричними методами. Тим не менш, багато особливих випадків для різних типів даних були успішно вивчені за допомогою генеративних моделей.

Особливо поширеним методом зменшення розмірності є тематичне моделювання в текстових даних [28]. Цей метод також іноді називають імовірнісним латентним семантичним індексуванням (PLSI). У тематичному моделюванні кластер асоціюється з набором слів і набором документів одночасно. Основними параметрами, які потрібно вивчити, є ймовірності присвоєння слів (розмірів) темам (кластерам) і документів (точок даних) темам (кластерам). Таким чином, це природним чином створює м’яку кластеризацію даних як з точки зору рядка, так і з точки зору стовпця. Потім вони вивчаються комплексно. Ці методи знайшли великий успіх у літературі з інтелектуального аналізу тексту, і було запропоновано багато методів, таких як Latent Dirichlet Allocation (LDA), які змінюються за цим принципом із використанням більш узагальнених пріоритетів [12].

### 1.5.2. Матрична факторизація та кокластеризація

Методи матричної факторизації та спільної кластеризації також є широко використовуваними класами методів зменшення розмірності. Ці методи зазвичай застосовуються даних, що представлені як розріджені невід’ємні матриці, хоча в принципі можна узагальнити ці методи також для інших типів матриць. Однак справжньою привабливістю цього підходу є додаткова можливість інтерпретації, властива методам невід’ємної матриці факторизації, у яких точка даних може бути виражена як невід’ємна лінійна комбінація понять у базових даних. Методи факторизації невід’ємної матриці тісно пов’язані з кокластеризацією, яка кластеризує рядки та стовпці матриці одночасно.

Нехай $A$ – невід'ємна матриця $n \times d$, яка містить $n$ записів даних, кожна з яких має розмірність $d$. У більшості типових програм, таких як текстові дані, матриця $A$ представляє малі невід'ємні величини, такі як частоти слів, і є не тільки невід'ємною, але й розрідженою. Матрицю $A$ можна приблизно розкласти на дві невід'ємні матриці низького рангу $U$ і $V$ розмірів $n \times k$ і $k \times d$ відповідно. Забігаючи на перед, ці матриці є представниками для кластерів у рядках і стовпцях відповідно, коли рівно $k$ кластерів використовуються для представлення як рядків, так і стовпців. Тому маємо:

$$A \approx U \times V \tag{1.1}$$

Залишкова матриця $R$ уособлює шум у базових даних:

$$R = A - U \times V \tag{1.2}$$

Зрозуміло, що бажано визначити факторизовані матриці $U$ та $V$ так, щоб сума квадратів залишків у $R$ була мінімізованою. Це еквівалентно визначенню невід'ємних матриць $U$ та $V$, так що норма Фробініуса $A - U \times V$ мінімізується. Це задача оптимізації з обмеженнями, в якій обмеження відповідають невід'ємності записів у $U$ та $V$. Тому метод Лагранжа можна використовувати для вивчення параметрів цієї задачі оптимізації [36].

Невід'ємна $n \times k$ матриця $U$ представляє координати кожної з $n$ точок даних у кожному з $k$ новостворених вимірів. Високе позитивне значення запису $(i, j)$ означає, що точка даних $i$ тісно пов'язана з новоствореним виміром $j$. Таким чином, тривіальним способом виконання кластеризації було б призначити кожну точку даних новоствореному виміру, для якого вона має найбільший компонент в $U$. Крім того, якщо точкам даних дозволено належати до кількох кластерів, то для кожного з $k$ у стовпцях $V$, записи зі значенням вище певного порогу відповідають кластерам документів. Таким чином, новостворений набір параметрів можна зробити синонімом кластеризації набору даних. На практиці можна зробити набагато краще, використовуючи $k$-середні для нового представлення. Позитивна

характеристика методу факторизації невід'ємної матриці полягає в тому, що розмір запису в матриці $U$ дає розуміння, наскільки конкретна точка даних пов'язана з певною концепцією (або новостворениим виміром). Ціна цієї кращої інтерпретації полягає в тому, що новостворені набори розмірів зазвичай не ортогональні один одному. Це підводить нас до обговорення фізичної інтерпретації матриці $V$.

В матриці $V$ $k \times d$ надає фактичне представлення кожного з $k$ новостворених вимірів у термінах вихідних $d$ вимірів. Таким чином, кожен рядок у цій матриці є одним із компонентів цієї новоствореної системи осей, і рядки не обов'язково є ортогональними один одному (на відміну від більшості інших методів зменшення розмірності). Великий позитивний запис означає, що цей новостворений вимір тісно пов'язаний із певним виміром у базових даних. Наприклад, у програмі кластеризації документів записи з великими позитивними значеннями в кожному рядку представляють найпоширеніші слова в кожному кластері. Для ідентифікації цих слів можна використати порогову техніку. Зверніть увагу на схожість цього підходу з технікою на основі проекції або більш м'якою технікою PLSI. У контексті програми кластеризації документів ці великі позитивні записи забезпечують кластери слів у базових даних. Тому в контексті текстових даних кожен документ може бути приблизно виражений (через процес факторизації) як невід'ємна лінійна комбінація щонайбільше $k$ векторів кластерів слів. Питома вага цього компонента представляє важливість цього компонента, що робить декомпозицію легкою інтерпретацією.

На даному етапі очевидно, що матриці $U$ та $V$ одночасно забезпечують кластери в рядках (документи) і стовпцях (слова). Цей загальний принцип у застосуванні до розріджених невід'ємних матриць також називають спільною кластеризацією. Звичайно, факторізація невід'ємної матриці є лише одним із можливих способів виконання спільної кластеризації.

### 1.5.3. Спектральні методи

Спектральні методи є цікавою технікою для зменшення розмірності, яка працює з матрицею подібності (або відстані) базових даних замість роботи з вихідними точками та розмірами. Це, звичайно, має свій набір переваг і недоліків. Головною перевагою є те, що тепер можна працювати з довільними об'єктами для зменшення розмірності, а не просто з точками даних, які представлені в багатовимірному просторі. Насправді спектральні методи також виконують подвійне завдання вбудовування цих об'єктів у евклідів простір, одночасно виконуючи зменшення розмірності. Тому спектральні методи надзвичайно популярні для виконання кластеризації довільних об'єктів, таких як набори вузлів у графі. Недоліком спектральних методів є те, що, оскільки вони працюють із матрицею подібності $n \times n$, складність часу навіть для створення матриці подібності масштабується відповідно до квадрата кількості точок даних. Крім того, процес визначення власних векторів цієї матриці може бути надзвичайно дорогим, якщо не потрібна дуже невелика кількість цих власних векторів. Іншим недоліком спектральних методів є те, що набагато складніше створити представлення нижчих розмірів для точок даних, якщо вони не є частиною вихідної вибірки, з якої була створена матриця подібності. Для багатовимірних даних використання такої великої матриці подібності є досить зайвим, якщо тільки дані не є надзвичайно шумними та мають високу розмірність.

Нехай $D$ — база даних, що містить $n$ точок. Першим кроком є створення $n \times n$ матриці $W$ вагових коефіцієнтів, яка представляє попарну подібність між різними точками даних. Це робиться за допомогою теплового ядра. Для будь-якої пари точок даних $\overline{X_i}$ і $\overline{X_j}$ теплове ядро визначає матрицю подібності $W$ вагових коефіцієнтів:

$$W_{ij} = exp\left(-\left\|\overline{X_i} - \overline{X_j}\right\|^2 \quad ^t\right), (1)$$

де t – визначений користувачем параметр. Крім того, значення $W_{ij}$встановлюється рівним 0, якщо відстань між $\overline{X_i}$ і $\overline{X_j}$ перевищує заданий

поріг. Матрицю подібності також можна розглядати як матрицю суміжності графа, в якому кожен вузол відповідає елементу даних, а вага ребра відповідає подібності між цими елементами даних. Тому спектральні методи зводять проблему до пошуку оптимальних розрізів у цьому графі, які відповідають розділам, які слабо з'єднані ребрами, що представляють подібність. Отже, спектральні методи можна вважати технікою на основі графів для кластеризації будь-яких типів даних шляхом перетворення матриці подібності в структуру мережі.

Слід зазначити, що навіть коли відстані дуже шумні, матриця подібності кодує значну кількість інформації через її вичерпний характер. Саме тут корисний спектральний аналіз, оскільки шум у представленні подібності можна абстрагувати за допомогою аналізу власних векторів цієї матриці. Таким чином, ці методи здатні відновлювати та покращувати приховану інформацію в матриці подібності, хоча й за досить високу вартість, яка масштабується з квадратом кількості точок даних.

Відображення точок на одновимірний простір: узагальнення для 1-вимірного випадку є відносно простим. Відображення точки даних $D$ у набір точок $y_1 \dots y_n$ на прямій, у якій подібність відображається на евклідовій відстані на цій прямій. Тому небажано, щоб точки даних, які дуже схожі, відображалися на віддалених точках на цій лінії. Визначимо значення $y_i$,якt мінімізує наступну цільову функцію 0:

$$0 = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \times$$

Цільова функція 0 може бути переписана в термінах матриці Лапласа $L$ від $W$. Матриця Лапласа визначається як $D - W$, де $D$ є діагональною матрицею, що задовольняє $D_{ii} = \sum_{j-1}^{n} W_{ji}$. Нехай $\overline{y} = y_1 \dots y_n$. Цільову функцію 0 можна переписати так:

$$0 = \overline{y}^T \times L \times \overline{y}. (3)$$

Необхідно включити обмеження масштабування, щоб переконатися, що тривіальне значення $y_i = 0$для всіх і не є проблемою. Можливе обмеження

масштабування таке:

$$\overline{y}^T \times D \times \overline{y} = 1. (4)$$

Використання матриці $D$ забезпечує різні ваги для елементів даних, оскільки передбачається, що вузли з більшою схожістю з різними елементами даних більше залучені до процесу кластеризації. Це формулювання оптимізації є у форматі узагальненого власного вектора, і тому значення $\overline{y}$ оптимізується шляхом вибору найменшого власного значення, для якого задовольняється узагальнене співвідношення власного вектора $L \times \overline{y} = \lambda \times D \times \overline{y}$. На практиці, однак, найменше узагальнене власне значення відповідає тривіальному розв'язку, де $\overline{y}$ є (нормализованим) одиничним вектором. Цей тривіальний власний вектор неінформативний. Тоді друге найменше власне значення забезпечує оптимальне рішення, яке є більш інформативним.

Ця модель може бути узагальнена для визначення всіх власних векторів у порядку зростання власного значення. Такі напрямки відповідають послідовним ортогональним напрямкам у даних, що призводить до найкращого відображення. Це призводить до набору з $n$ власних векторів $\overline{e_1}, \overline{e_2} \dots \overline{e_n}$ (з яких перший є тривіальним), з відповідними власними значеннями $0 = \lambda_1 \leq \lambda_2 \dots \leq \lambda_n$. Нехай відповідне векторне представлення даних точки вздовж кожного власного вектора позначаються через $\overline{y}^1 \dots \overline{y}^n$.

Що інтуїтивно представляють малі та великі власні вектори у новому трансформованому просторі? Використовуючи впорядкування елементів даних уздовж власного вектора невеликої величини для створення розрізу, вага ребер через розріз, ймовірно, буде невеликою. Таким чином, це представляє кластер у просторі елементів даних. У той же час, якщо вибрано верхні $k$ найдовших власних векторів, тоді векторні представлення $\overline{y}^n \dots \overline{y}^{n-k+1}$ забезпечують матрицю $n \times k$. Це забезпечує $k$-вимірне вбудоване представлення всього набору даних з $n$ точок, який зберігає максимальну кількість інформації. Таким чином, спектральні методи можуть бути використані для одночасного виконання кластеризації та зменшення

розмірності.

## 1.6. Багатовимірний сценарій

Багатовимірний сценарій є особливо складним для кластерного аналізу через значні варіації в поведінці атрибутів даних у різних частинах даних. Це призводить до численних проблем у інтелектуальному аналізі даних, таких як кластеризація, пошук найближчого сусіда та аналіз викидів. Слід зазначити, що багато алгоритмів для цих задач залежать від використання відстаней як важливої підпрограми. Однак зі збільшенням розмірності відстані все більше втрачають свою ефективність і статистичну значущість через нерелевантні атрибути. Передумова полягає в тому, що послідовно менша частка атрибутів часто залишається релевантною зі збільшенням розмірності, що призводить до розмивання відстаней і посилення ефектів концентрації через поведінку усереднення нерелевантних атрибутів. Ефекти концентрації стосуються того факту, що коли багато функцій є шумними та некорельованими, їхні адитивні ефекти призведуть до того, що всі попарні відстані між точками даних стануть однаковими. Ефекти шуму та концентрації є проблемними способами для алгоритмів кластеризації на основі відстані (та багатьох інших):

• Збільшення шуму від нерелевантних атрибутів може спричинити помилки у представленні відстані, так що воно більше не представлятиме належним чином власну відстань між об’єктами даних.

• Вплив концентрації від нерелевантних розмірів призводить до зниження статистичної значущості результатів алгоритмів на основі відстані, якщо вони використовуються безпосередньо з відстанями, які не були належним чином усунені.

## 1.7. Масштабовані методи кластерного аналізу

Завдяки прогресу програмного та апаратного забезпечення збирати дані

стало дедалі легше в різноманітних сценаріях. Наприклад, у програмах соціального зондування користувачі можуть носити мобільні датчики, що призводить до постійного накопичення даних з часом. Це називається потоковим сценарієм, у якому передбачається, що один прохід дозволений для потоку даних, оскільки дані часто занадто великі, щоб їх можна було зібрати в межах обмежених ресурсів. Навіть коли дані збираються в автономному режимі, це призводить до численних проблем масштабованості з точки зору інтеграції з традиційними системами баз даних або з точки зору використання великих обсягів даних у розподіленому середовищі з інфраструктурою великих даних. Таким чином, можливі різні рівні викликів, залежно від характеру базових даних.

### 1.7.1. Проблеми введення/виведення в управлінні базами даних

Найбільш фундаментальні проблеми масштабованості виникають, коли алгоритм інтелектуального аналізу даних поєднується з традиційною системою баз даних. У таких випадках можна показати, що головне вузьке місце виникає через час вводу-виведення, необхідний для доступу до об’єктів у базі даних. Тому часто корисними є алгоритми, які використовують послідовне сканування даних, а не методи, які випадково отримують доступ до записів даних. У літературі з баз даних було запропоновано ряд класичних методів для вирішення цих проблем масштабованості.

Найпростішими алгоритмами, які можна поширити на цей випадок, є методи плоского розділення, які використовують послідовне сканування бази даних, щоб призначити точки даних представникам. Одним із перших методів [66] у цьому напрямку був CLARANS, у якому ці представники визначаються за допомогою $k$-медоїдного підходу. Метод CLARANS використовує вибірку, виконуючи ітераційний підйом на гору над меншою вибіркою, щоб підвищити ефективність підходу. Іншим методом у цьому напрямку є BIRCH [56], який узагальнює підхід $k$-Середні до процесу кластеризації. Метод CURE [24], знаходить кластери несферичної форми за допомогою більш ніж однієї репрезентативної точки на кластер. Він поєднує

розділення та вибірку для забезпечення більш ефективного процесу кластеризації.

### 1.7.2. Алгоритми потокової передачі

Сценарій потокової передачі є особливо складним для алгоритмів кластеризації через вимоги аналізу в реальному часі, а також еволюцію та зміну концепції базових даних. У той час як алгоритми, орієнтовані на базу даних, вимагають обмеженої кількості проходів над даними, потокові алгоритми вимагають рівно одного проходу, оскільки дані не можуть зберігатися взагалі. На додаток до цієї проблеми, аналіз зазвичай потрібно виконувати в режимі реального часу, і в аналізі потрібно належним чином враховувати зміну моделей у даних.

Для досягнення цих цілей практично всі потокові методи використовують техніку підсумовування для створення проміжних представлень, які можна використовувати для кластеризації. Один із перших методів у цьому напрямку використовує підхід мікрокластеризації [5] для створення та підтримки кластерів із основного потоку даних. Зведена статистика підтримується для мікрокластерів, щоб забезпечити ефективний процес кластеризації. Це поєднується з пірамідальними часовими рамками, щоб охопити аспекти, що розвиваються, основного потоку даних. Потокова кластеризація також може бути розширена до інших типів даних, таких як дискретні дані, дані масивного домену, текстові дані та дані графіків.

### 1.7.3. Структура великих даних

У той час як алгоритми потокового передавання працюють за припущенням, що дані занадто великі для явного зберігання, фреймворк великих даних використовує прогрес у технології зберігання, щоб фактично зберігати дані та обробляти їх. Однак, навіть якщо дані можна зберігати в явному вигляді, часто буває нелегко обробити та витягти з них ідеї. Це пояснюється тим, що збільшення розміру даних передбачає використання розподіленої файлової системи для зберігання інформації, а для забезпечення

достатньої масштабованості потрібні методи розподіленої обробки. Проблема полягає в тому, що якщо великі сегменти даних доступні на різних машинах, часто занадто дорого перемішувати дані між різними машинами, щоб отримати з них інтегровану інформацію. Таким чином, як і в усіх розподілених інфраструктурах, бажано обмінюватися проміжною інформацією, щоб мінімізувати витрати на зв'язок. Для прикладного програміста це іноді може створити труднощі з точки зору відстеження того, де зберігаються різні частини даних, і точного впорядкування зв'язків, щоб мінімізувати витрати.

У цьому контексті структура Google MapReduce [15] забезпечує ефективний метод для аналізу великих обсягів даних, особливо коли природа обчислень включає лінійно обчислювані статистичні функції над елементами потоків даних. Одним із бажаних аспектів цієї структури є те, що вона абстрагує точні деталі того, де зберігаються різні частини даних для прикладного програміста. Багато алгоритмів кластеризації, таких як $k$-середні, природно лінійні з точки зору їх масштабованості з розміром даних.

## РОЗДІЛ ІІ. Типи даних, що вивчаються у кластерному аналізі

Конкретний тип даних має величезний вплив на вибір алгоритму кластеризації. Більшість найперших алгоритмів кластеризації були розроблені з неявним припущенням, що атрибути даних були числовими. Однак це не так у більшості реальних сценаріїв, коли дані можуть бути отримані з будь-якої кількості можливих типів, таких як дискретні (категоричні), часові або структурні. У цьому розділі обговорюється вплив типів даних на процес кластеризації, а також наведено короткий огляд різних типів даних.

### 2.1. Категоріальні дані

Категоріальні дані досить поширені в реальних наборах даних. Це пов’язано з тим, що багато атрибутів у реальних даних, таких як стать, раса чи поштовий індекс, за своєю суттю є дискретними й не мають природного порядку. У багатьох випадках набори даних можуть бути змішаними, в яких деякі атрибути, такі як зарплата, є числовими, тоді як інші атрибути, такі як стать або поштовий індекс, є категоричними. Особливою формою категоріальних даних є дані ринкового кошика, у яких усі атрибути двійкові.

Категоріальні набори даних створюють численні проблеми для алгоритмів кластеризації:

- Коли алгоритми залежать від використання функції подібності або відстані, стандартні показники $L_k$ більше не можна використовувати. Необхідно визначити нові міри подібності для категоріальних даних[32].
- Багато алгоритмів кластеризації, такі як методи $k$-середні або $k$-медіан, створюють представників кластеризації як середні або медіани точок даних у кластерах. У багатьох випадках такі статистичні дані, як середнє або медіана, природно визначені для числових даних, але їх потрібно відповідним чином змінити для дискретних даних.

Коли дані змішані, проблема стає складнішою, оскільки різні атрибути тепер потрібно обробляти гетерогенним чином, а функції подібності мають чітко враховувати неоднорідність, що лежить в основі.

Слід зазначити, що деякі моделі кластеризації більш піддатливі для різних типів даних, ніж інші. Наприклад, деякі моделі залежать лише від функції відстані (або подібності) між записами. Таким чином, поки між записами можна визначити відповідну функцію подібності, то ефективно використовуються методи кластерного аналізу. Спектральна кластеризація — це один клас методів, який можна використовувати фактично з будь-яким типом даних, якщо визначено відповідні функції подібності. Недоліком є те, що методи масштабуються з квадратом розміру матриці подібності. Генеративні моделі також легко узагальнюється для різних типів даних, якщо відповідну генеративну модель можна визначити для кожного компонента суміші. Деякі поширені алгоритми для кластеризації категоріальних даних включають CACTUS [21], ROCK [23], STIRR [22] і LIMBO [9].

## 2.2. Текстові дані

Текстові дані є особливо поширеною формою даних із зростанням популярності Інтернету та соціальних мереж. Текстові дані зазвичай представлені у форматі векторного простору, у якому конкретне впорядкування абстрагується, і тому дані розглядаються як пакет слів. Слід зазначити, що методи для кластеризації текстових даних також можна використовувати для кластеризації атрибутів на основі набору. Текстові дані мають ряд властивостей, які слід брати до уваги:

- Дані надзвичайно багатовимірні та розріджені. Це відповідає тому факту, що лексикон тексту досить великий, але кожен документ містить лише невелику кількість слів. Таким чином, більшість атрибутів приймають нульові значення.
- Значення атрибутів відповідають частоті слів і, отже, зазвичай є

невід’ємними. Це важливо з точки зору багатьох методів спільної кластеризації та матричної факторизації, які використовують цю невід’ємність.

Найбільш ранні методи кластеризації тексту, такі як метод розсіювання [13, 49], використовують методи на основі відстані. Зокрема, для процесу кластеризації використовується комбінація $k$-середніх і агломеративних методів. Згодом проблему кластеризації тексту часто досліджували в контексті тематичного моделювання, де створюється м’яка матриця приналежності слів і документів до кластерів. Методи, що використовуються для генеративного тематичного моделювання, це PLSI та LDA [12, 28]. Ці методи можна вважати м’якими версіями таких методів, як кокластеризація [16, 17, 45] і матрична факторизація [36], які кластеризують рядки та стовпці разом одночасно. Спектральні методи [47] часто використовуються для виконання цього розбиття шляхом створення дводольного графа подібності, який представляє відношення належності рядків (документів) і стовпців (слів). Це не особливо дивно, оскільки відомо, що методи матричної факторизації та спектральна кластеризація тісно пов’язані між собою [38].

## 2.3. Мультимедійні дані

Із зростанням популярності соціальних медіа багато форм мультимедійних даних можуть використовуватися разом із методами кластеризації. До них належать зображення, аудіо та відео. Прикладами соціальних мереж, які містять велику кількість таких даних, є Flickr і Youtube. Навіть Інтернет і звичайні соціальні мережі зазвичай містять значну кількість мультимедійних даних різних типів. У багатьох випадках такі дані можуть зустрічатися в поєднанні з іншими більш звичайними формами текстових даних.

Кластеризація мультимедійних даних часто є складним завданням через різний і неоднорідний характер основного вмісту. У багатьох випадках дані

можуть бути мультимодальними або контекстними, що містять як поведінкові, так і контекстні атрибути. Наприклад, дані зображення, як правило, є контекстними, у яких позиція пікселя представляє його контекст, а значення на пікселі представляє його поведінку. Відео та музичні дані також є контекстними, оскільки часове впорядкування записів даних надає необхідну інформацію для розуміння. Неоднорідність і контекстний характер даних можна вирішити лише за допомогою належного представлення та аналізу даних. Насправді представлення даних є ключовим питанням у всіх формах мультимедійного аналізу, що суттєво впливає на кінцеву якість результатів.

### 2.4. Кластеризація даних часових рядів

Дані часових рядів є досить поширеними у всіх формах даних датчиків, фондових ринках або будь-яких інших видах додатків часового відстеження чи прогнозування. Основним аспектом часових рядів є те, що значення даних не є незалежними одне від одного, вони залежать одне від одного в часі. Зокрема, дані містять контекстний атрибут (час) і поведінковий атрибут (значення даних). Існує значна різноманітність у визначеннях проблем у сценарії часових рядів. Дані часових рядів можна кластеризувати різними способами, залежно від того, чи потрібен онлайн-аналіз на основі кореляції, чи офлайн-аналіз на основі форми.

- У онлайн-аналізі на основі кореляції між різними потоками даних часових рядів відстежуються з часом, щоб створити онлайн-кластери. Такі методи часто корисні для вибору датчиків і прогнозування, особливо коли потоки виявляють кореляцію затримки в рамках кластерних моделей. Ці методи часто використовуються в додатках фондового ринку, де бажано підтримувати групи кластеризованих біржових тикерів в режимі онлайн на основі їхніх моделей кореляції. Таким чином, функції відстані між різними серіями необхідно обчислювати безперервно на основі їх взаємних коефіцієнтів регресії. Деякі приклади цих методів включають підхід

MUSCLES [55] і широкомасштабний метод кореляційного моніторингу часових рядів [57].

• У автономному аналізі на основі форм об’єкти часових рядів аналізуються в автономному режимі, і конкретні деталі про те, коли було створено певний часовий ряд, не важливі. Наприклад, для набору часових рядів ЕКГ, зібраних від пацієнтів, точний час, коли було зібрано інформацію, не є важливим, але загальна форма ряду важлива для цілей кластеризації. У таких випадках важлива функція відстані між двома часовими рядами. Це важливо, оскільки різні часові ряди можуть бути не намальовані на тому самому діапазоні значень даних, а також можуть виявляти ефекти викривлення часу, коли форми можна зіставити лише подовжуючи або звужуючи частини часового ряду в часі. Розробка функції відстані [25] містить ключ до ефективного використання підходу.

Особливо цікавим є випадок багатофакторних часових рядів, у яких багато рядів створюються одночасно протягом певного часу. Класичним прикладом цього є дані траєкторії, в яких різні напрямки координат утворюють різні компоненти багатовимірного ряду. Тому траєкторний аналіз можна розглядати як особливий вид кластеризації часових рядів. Як і у випадку однофакторних часових рядів, ці кроки можна виконати за допомогою онлайн-аналізу (траєкторії рухаються разом у реальному часі) або офлайн-аналізу (подібна форма).

## 2.5. Дискретні послідовності

Багато форм даних створюють дискретні послідовності замість категоріальних. Наприклад, веб-журнали, послідовності команд у комп’ютерних системах і біологічні дані є дискретними послідовностями. Контекстний атрибут у цьому випадку часто відповідає розміщенню (наприклад, біологічні дані), а не часу. Біологічні дані також є одним із найпоширеніших застосувань кластеризації послідовностей.

Як і у випадку безперервних послідовностей, ключовою проблемою є створення функцій подібності між різними об’єктами даних. У цьому контексті зазвичай використовуються численні функції подібності, такі як відстань Хеммінга, відстань редагування та найдовша спільна підпослідовність [39]. Іншою ключовою проблемою, яка виникає в контексті кластеризації дискретних послідовностей, є те, що проміжне та сумарне представлення набору послідовностей може бути обчислювально інтенсивним. На відміну від числових даних, де можна використовувати методи усереднення, набагато складніше знайти такі представлення для дискретних послідовностей. Загальним представленням, яке забезпечує відносно обмежений рівень узагальнення, є дерево суфіксів. Методи використання дерев суфіксів для методів кластеризації послідовностей були запропоновані в CLUSEQ [54].

Генеративні моделі можуть бути використані як для моделювання відстаней між послідовностями, так і для створення ймовірнісних моделей генерації кластерів [50]. Особливо поширеним підходом є використання сумішей HMM [44]. Приховані моделі Маркова можна вважати особливим видом моделі суміші, в якій різні компоненти суміші залежать один від одного. Другий рівень моделювання суміші можна використовувати для створення кластерів із цих різних HMM. Велика частина роботи з кластеризації послідовностей виконується в контексті біологічних даних [19, 31, 41].

## 2.6. Мережеві дані

Графіки та мережі є одними з найбільш фундаментальних (і загальних) представлень усіх даних. Це пояснюється тим, що практично кожен тип даних можна представити як графік подібності зі значеннями подібності на краях. Насправді метод спектральної кластеризації можна розглядати як найзагальніший зв'язок між усіма іншими видами кластеризації та кластеризацією графів. Таким чином, якщо функція подібності може бути

визначена між довільними об’єктами даних, для виконання аналізу можна використовувати спектральну кластеризацію. Кластеризація графів широко вивчалася в класичній літературі, особливо в контексті проблеми двостороннього та багатостороннього розбиття графів. Найбільш класичним з усіх багатосторонніх алгоритмів поділу є метод Кернігана-Ліна [37]. Такі методи можна використовувати в поєднанні з методами укрупнення графів, щоб забезпечити ефективні рішення. Ці методи відомі як методи багаторівневого розбиття графів. Особливо популярним алгоритмом у цій категорії є METIS [35].

У літературі зазвичай використовується ряд методів для створення секцій із графів:

• *Генеративні моделі:* можна визначити генеративну модель практично для будь-якої проблеми кластеризації, доки існує відповідний метод для визначення кожного компонента суміші як розподілу ймовірностей [51].

• *Класичні комбінаторні алгоритми:* ці методи використовують мережевий потік [8] або інші ітераційні комбінаторні методи, щоб створити розділи з базового графа. Слід зазначити, що часто відомо, що вибірка рівних країв створює якісні розділи, коли вона повторюється багато разів [34]. Часто бажано визначити розрізи, які добре збалансовані між різними розділами, оскільки розріз з найменшим абсолютним значенням часто містить переважну більшість вузлів в одному розділі та дуже малу кількість вузлів в інших розділах.

• *Спектральні методи:* їх можна розглядати як послаблення лінійного програмування для цілочисельних програм, що представляють оптимізацію розрізів графіка. Різні цільові функції можуть бути побудовані для різних типів розрізів, таких як ненормалізований розріз, відношення та нормалізований розріз. Таким чином, неперервні розв’язки цих лінійних програм можна використовувати для створення багатовимірного вбудовування для вузлів, до яких можна застосувати звичайні алгоритми

Umeans.

• *Факторізація невід'ємної матриці:* оскільки граф може бути представлений у вигляді матриці суміжності, факторизація невід'ємної матриці може бути використана для того, щоб розкласти його на дві матриці низького рангу. Можна застосувати методи факторизації матриці або до матриці суміжності між вузлами, або до матриці суміжності між вузлами, щоб отримати різні види інформації. Також відносно легко доповнити матрицю вмістом для створення аналітичних методів, які можуть кластеризуватися за допомогою комбінації вмісту та структури [43].

## 2.7. Невизначені дані

Багато форм даних або мають низьку точність, або якість даних була навмисно знижена з метою розробки різних типів алгоритмів мережевого видобутку. Це призвело до появи імовірнісних баз даних. Імовірнісні дані можуть бути представлені або у формі атрибутивної невизначеності, або у формі можливої моделі світу, в якій лише окремі підмножини атрибутів можуть бути присутніми в даних у певний момент часу [2]. Ключова ідея тут полягає в тому, що включення імовірнісної інформації може покращити якість алгоритмів інтелектуального аналізу даних. Наприклад, якщо два атрибути однаково бажано використовувати в алгоритмі в детерміністичному сценарії, але один з них має більшу невизначеність, ніж інший, тоді атрибут з меншою невизначеністю явно більш бажаний для використання. Невизначені алгоритми кластеризації нещодавно також були поширені на область потоків і графів. У контексті графів часто бажано визначити найбільш надійні підграфи в основній мережі. Це ті підграфи, які найважче роз'єднати в умовах крайньої невизначеності [40].

# РОЗДІЛ ІІІ. Алгоритми кластеризації даних

## 3.1. K-Means

Алгоритм $K$-Means є ітераційним алгоритмом, який намагається розділити набір даних на попередньо визначені $K$ окремі підгрупи (кластери), що не перекриваються, де кожна точка даних належить лише одному кластеру. $K$-Means намагається зробити внутрішньокластерні точки даних якомога подібнішими, водночас зберігаючи кластери максимально різними. $K$-Means призначає точки даних кластеру таким чином, щоб сума квадратів відстані між точками даних і центроїдом кластера (середнє арифметичне всіх точок даних, які належать цьому кластеру) була мінімальною. Чим менше варіацій у кластерах, тим більш однорідними (схожими) є точки даних у межах одного кластера.

Алгоритм $K$-Means працює наступним чином:

1) Вкажіть кількість кластерів $K$.
2) Ініціалізуйте центроїди, спочатку перетасувавши набір даних, а потім випадково вибравши $K$ точок даних для центроїдів без заміни.
3) Продовжуйте повторювати, доки центроїди не зміняться, тобто призначення точок даних для кластерів не змінюється.
   - ✓ Обчисліть суму квадратів відстані між точками даних і всіма центроїдами.
   - ✓ Призначте кожну точку даних найближчому кластеру (центроїду).
   - ✓ Обчисліть центроїди для кластерів, взявши середнє значення всіх точок даних, які належать кожному кластеру.

Підхід $K$-Means для вирішення проблеми кластеризації даних називається очікуванням-максимізацією (Expectation-Maximization, EM). Е-крок призначає точки даних найближчому кластеру. М-крок обчислює центроїд кожного кластера.

Цільова функція алгоритму $K$-Means має наступний вигляд:

$$J = \sum_{i=1}^{m}\sum_{k=1}^{K} \omega_{ik} \vee x^i - \mu \vee \quad |^2$$

де $\omega_{ik} = 1$ для точки даних $x_i$, якщо вона належить кластеру $k$; інакше $\omega_{ik} = 0$. Крім того, $\mu$ є центроїдом кластера $x_i$.

Така проблема мінімізації складається з двох частин. Спочатку ми мінімізуємо $J$ відносно $\omega_{ik}$ та фіксуємо $\mu_k$. Потім ми мінімізуємо $J$ відносно $\mu_k$ та фіксуємо $\omega_{ik}$. Технічно кажучи, ми диференціюємо $J$ відносно $\omega_{ik}$ та оновлюємо призначення кластерів (Е-крок). Далі ми диференціюємо $J$ відносно $\mu_k$ та повторно обчислюємо центроїди після присвоєння кластера з попереднього кроку (М-крок). Отже, Е-крок це:

$$\frac{\partial J}{\partial \omega_{ik}} = \sum_{i=1}^{m}\sum_{k=1}^{K} \vee x^{\text{i}} - \mu \vee^2 = \omega_{ik} = \begin{cases} 1, \text{якщо} k = argmin_j \left\| x^i - \mu_j \right\|^2, \\ 0, \text{у всіх інших випадках.} \end{cases}$$

Іншими словами кажучи, призначаємо точку даних $x_i$ найближчому кластеру за сумою квадратів відстані від центроїда кластера.

А м-крок це:

$$\frac{\partial J}{\partial \mu_k} = 2\sum_{i=1}^{m} \omega_{ik}\left(x^i - \mu_k\right) = 0 = \mu_k = \frac{\sum_{i=1}^{m} \omega_{ik} x^{\text{i}}}{\sum_{i=1}^{m} \omega_{ik}}.$$

Останнє рівняння означає переобчислення центроїда кожного кластера для відображення нових призначень.

Варто зауважити кілька речей:

- Оскільки алгоритми кластеризації, включаючи $K$-Means, використовують вимірювання на основі відстані для визначення подібності між точками даних, рекомендується стандартизувати дані, щоб отримати нульове середнє значення і стандартне відхилення – одиницю, оскільки майже завжди атрибути в будь-якому наборі даних матимуть різні одиниці вимірювання, наприклад, вік та дохід людини.
- Враховуючи ітераційну природу $K$-Means і випадкову ініціалізацію центроїдів на початку алгоритму, різні ініціалізації можуть

призвести до утворення різних кластерів, оскільки алгоритм $k$-Means може застрягти в локальному оптимумі та не збігатися з глобальним оптимумом. Тому рекомендується запускати алгоритм, використовуючи різні ініціалізації центроїдів і вибирати результати виконання, які давали б нижчу суму квадратів відстані.

- Якщо приналежність точок даних до кластера не змінюється, то це вказує на відсутність змін у варіації всередині кластера:

$$\frac{1}{m}\sum_{i=1}^{m_k} \vee x^{i} - \mu_{c^k} \vee^{2}.$$

### 3.2. Hierarchical Agglomerative Clustering (HAC)

У агломеративному ієрархічному підході HAC ми визначаємо кожну точку даних як кластер і об'єднуємо існуючі кластери на кожному кроці. Ось чотири різні методи цього підходу:

- ***Одиночне зв'язування:*** у єдиному зв'язуванні ми визначаємо відстань між двома кластерами як мінімальну відстань між будь-якою окремою точкою даних у першому кластері та будь-якою окремою точкою даних у другому кластері. На основі цього визначення відстані між кластерами на кожному етапі процесу ми об'єднуємо два кластери з найменшою відстанню одиничного зв'язку.
- ***Повний зв'язок:*** у повному зв'язку ми визначаємо відстань між двома кластерами як максимальну відстань між будь-якою окремою точкою даних у першому кластері та будь-якою окремою точкою даних у другому кластері. На основі цього визначення відстані між кластерами на кожному етапі процесу ми об'єднуємо два кластери, які мають найменшу повну відстань зв'язку.
- ***Середній зв'язок:*** у середньому зв'язку ми визначаємо відстань між двома кластерами як середню відстань між точками даних у першому кластері та точками даних у другому кластері. На основі цього визначення відстані між кластерами на кожному етапі процесу ми

об’єднуємо два кластери, які мають найменшу середню відстань зв’язку.

- ***Метод центроїда:*** у методі центроїда відстань між двома кластерами є відстанню між двома середніми векторами кластерів. На кожному етапі процесу ми об’єднуємо два кластери, які мають найменшу відстань до центроїда.
- ***Метод Уорда:*** цей метод безпосередньо не визначає міру відстані між двома точками або кластерами. Це підхід, заснований на ANOVA. Односторонній однофакторний дисперсійний аналіз виконується для кожної змінної з групами, визначеними кластерами на цьому етапі процесу. На кожному етапі два кластери зливаються, що забезпечує найменше збільшення комбінованої суми квадратів помилок.

Жоден із цих чотирьох методів не є найкращим. На практиці доцільно випробувати кілька методів, а потім порівняти результати, щоб сформувати загальне судження про остаточне формування кластерів.

Нехай

- $X_1, X_2, \dots, X_k$ – $c$постереження з кластера 1,
- $Y_1, Y_2, \dots, Y_l$ –Спостереження з кластера 2,
- $d(x, y)$ – відстань між вектором спостереження $x$ і вектором спостереження $y$.

Відстань $d_{12}$ позначає відстань між кластерами 1 і 2.

- Одинарне з'єднання: $d_{12} = \min_{i,j} d(X_i, Y_j)$ – відстань між найближчими членами двох кластерів.
- Повне підключення: $d_{12} = \max_{i,j} d(X_i, Y_j)$ – відстань між найдальшими (найбільш несхожими) членами.
- Середнє зв'язування: $d_{12} = \frac{1}{kl} \sum_{i=1}^{k} \sum_{j=1}^{l} d(X_i, Y_j)$ – метод передбачає перегляд відстаней між усіма парами та усереднення всіх цих відстаней. Іншими словами цей метод називається UPGMA (Unweighted Pair Group Mean Averaging) – усереднення незваженого парного групового значення.

- Метод центроїда:$d_{12} = d(\overline{x}, \overline{y})$ передбачає знаходження середнього векторного розташування для кожного з кластерів і визначення відстані між двома центроїдами.

### 3.3. Expectation–Maximization clustering using Gaussian Mixture Models (GMM)

Перейдемо до визначення ймовірнісної моделі. Імовірність спостереження будь-якого спостереження, тобто щільність ймовірності, є зваженою сумою $K$ розподілів Гауса:

$$P(x|\Theta) = \sum_{j=1}^{K} \alpha_j N(x \vee \mu_j, \Sigma_j).$$

Кожна точка є сумішшю $K$ зважених гаусів, які параметризовані середнім значенням і коваріаційною матрицею. Отже, загалом ми можемо описати ймовірність спостереження конкретного спостереження як суміші.

Зверніть увагу, що у випадку 3 кластерів набір параметрів буде таким:

$$\Theta = \{\alpha_1, \alpha_2, \alpha_3, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3\}.$$

Потім ми хочемо знайти значення параметрів, які максимізують вірогідність набору даних. Ми хочемо знайти оцінки максимальної правдоподібності параметрів. Тобто ми хочемо знайти параметри, які максимізують ймовірність спостереження всіх точок даних разом (тобто сукупна ймовірність), розв'язуючи таку задачу оптимізації:

$$\max_{\Theta} P(X|\Theta) = \max_{\Theta} \prod_i P(x_i|\Theta) = \max_{\alpha_j, \mu_j, \Sigma_j} \prod_i$$

Зауважте, що повна сукупна ймовірність набору даних може бути векторизована (тобто розкладена як добуток окремих ймовірностей за допомогою оператора множення) лише в припущенні, що спостереження є ідентично незалежно розподілені. Коли вони є, події є незалежними, і ймовірність спостереження однієї точки даних не залежить від інших ймовірностей.

Замість правдоподібності ми зазвичай максимізуємо логарифм правдоподібності, частково тому, що він перетворює добуток ймовірностей на суму (з якою простіше працювати). Це тому, що натуральний логарифм є монотонно зростаючою увігнутою функцією і не змінює розташування максимуму (місце, де похідна дорівнює нулю, залишиться незмінним).

$$\max_{\Theta} logP(X|\Theta) = \max_{\Theta} log\left(\prod_i P(x_i|\Theta)\right) = \max_{\Theta} \sum_i log\big(P(x_i|\Theta)\big) =$$

$$\max_{\alpha_j,\mu_j,\Sigma_j} \sum_i log\left(\sum_{j=1}^{K} \alpha_j N\big(x_i\big|\mu_j,\Sigma_j\big)\right).$$

Тепер оцінку максимальної правдоподібності можна зробити великою кількістю різних способів. Це можна зробити за допомогою прямої оптимізації (знаходження точок, де часткові похідні є нульовими) або чисельної оптимізації, як градієнтний спуск.

### 3.4. Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN – це непараметричний алгоритм кластеризації на основі щільності: заданий набір точок у певному просторі, він групує разом точки, які щільно розташовані (точки з багатьма сусідами), позначаючи як викиди точки, які лежать окремо в регіонах з низькою щільністю (чиї найближчі сусіди надто далеко). DBSCAN є одним із найпоширеніших алгоритмів кластеризації, а також найбільш цитованим у науковій літературі.

Розглянемо набір точок у деякому просторі, який потрібно кластеризувати. Нехай $\varepsilon$ – параметр, що визначає радіус околиці відносно деякої точки. Для цілей кластеризації DBSCAN точки класифікуються як основні точки, (безпосередньо) доступні точки та викиди наступним чином:

- Точка $p$ є основною, якщо принаймні $minPts$ точок знаходяться на відстані $\varepsilon$ від неї (включаючи $p$).

- Точка $q$ доступна безпосередньо з $p$, якщо точка $q$ знаходиться на відстані $\varepsilon$ від основної точки $p$. Кажуть, що точки доступні безпосередньо з основних точок.
- Точка $q$ досяжна з $p$, якщо існує шлях $p_1,\ldots,p_n$ з $p_1 = p$ і $p_n = q$, де кожен $p_{i+1}$ безпосередньо досяжний з $p_i$. Зауважте, що це означає, що початкова точка та всі точки на шляху мають бути основними точками, за винятком, можливо, $q$.
- Усі точки, недоступні з будь-якої іншої точки, є викидами або шумовими точками.

Тепер, якщо $p$ є основною точкою, тоді вона утворює кластер разом із усіма точками (основними чи неосновними), які доступні з неї. Кожен кластер містить принаймні одну основну точку; неосновні точки можуть бути частиною кластера, але вони формують його «край», оскільки їх не можна використовувати для досягнення більшої кількості точок.

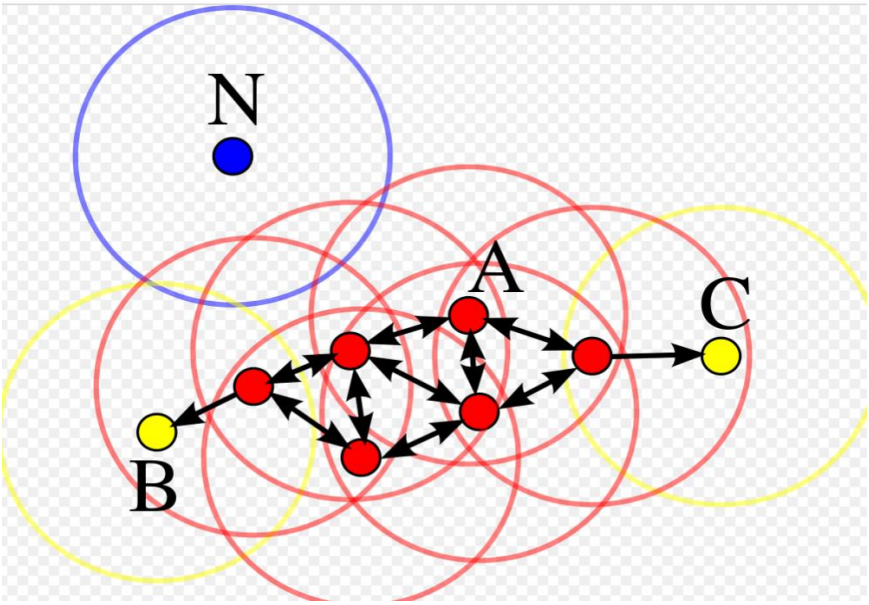



Рисунок 3. Діаграма алгоритму DBSCAN

На діаграмі з Рис. 3 $minPts = 4$. Точка $A$ та інші червоні точки є основними, оскільки область, що оточує ці точки в радіусі $\varepsilon$, містить принаймні 4 точки (включаючи саму точку). Оскільки всі вони доступні один від одного, вони утворюють єдиний кластер. Точки $B$ і $C$ не є основними точками, але доступні з $A$ (через інші основні точки) і, таким чином, також

належать до кластеру. Точка $N$ – це точка шуму, яка не є ані основною точкою, ані доступною безпосередньо.

Досяжність не є симетричним відношенням: за визначенням, лише основні точки можуть досягати неосновних точок. Зворотне твердження – не вірне, тому неосновна точка може бути доступною, але з неї нічого неможливо досягти. Тому для формального визначення розміру кластерів, знайдених DBSCAN, необхідне додаткове поняття зв’язності. Дві точки $p$ і $q$ зв’язані по щільності, якщо існує така точка $o$, що обидві $p$ і $q$ досяжні з $o$. Щільність зв'язності є симетричною. Тоді кластер задовольняє дві властивості:

- Усі точки всередині кластера взаємно пов’язані по щільності.
- Якщо точка досяжна за щільністю з деякої точки кластера, вона також є частиною кластера.

# РОЗДІЛ IV. Порівняння алгоритмів кластеризації даних

## 4.1. Основна ідея дослідження

У даному розділі проведено порівняння кластеризації чотирьом алгоритмами без учителя (без контролю):

- K-Means;
- Hierarchical Agglomerative Clustering (HAC);
- Expectation–Maximization clustering using Gaussian Mixture Models (GMM);
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

Словосполучення «без учителя» означає, що нам невідомо, як правильно поділити на кластери дані. Тобто маємо вхідний набір даних і цей набір даних потрібно розділити на групи (кластери). Для чого це потрібно? Де взагалі використовується кластеризація? – У наш кількість даних зростає експоненціально. Це відбувається завдяки збільшенню накопичувальних можливостей носіїв інформації, тобто завдяки науково-технічному прогресу. Опрацювання даних, тобто їх правильний статистичний аналіз допомагає знайти відповіді на багато питань у економіці, машинобудуванні, бізнесі, медицині. Наприклад, групу хворих на певне захворювання можна кластеризувати, тобто розбити на групи. До однієї з таких груп хворих потраплять пацієнти, які мають ще проблеми з нирками, наприклад. До іншої – пацієнти з цукровим діабетом тощо. Тобто кожна група або кластер пацієнтів матиме власні методи для лікування та певні препарати. Ще одним прикладом є поведінка інвестора на фінансовому ринку. Якщо при виборі портфоліо інвестор має інформацію про групу (кластер) цінних паперів, які він хоче придбати, то цей інвестор і розумітиме ризики, які пов’язані з купівлею даних цінних паперів.

Поділити на кластери означає розділити дані таким чином, щоб дані одного кластера були схожими між собою, а дані з різних кластерів суттєво відрізнялися. Реклама, яку надсилає кожному користувачу Google також

попередньо розділена на кластери. Ми отримуємо рекламу того, що шукали, або схожих товарів. Тобто нам показують лише те, що ми хочемо бачити, але ця тема варта магістерської роботи☺

Отже, кількість алгоритмів кластеризації даних та їх комбінацій стрімко зростає. Наша мета – зрозуміти, як вони працюють та визначити найкращий з них. Варто відзначити, що якщо певний алгоритм виявився найкращим для одних даних (наприклад, медичних), то це не означає, що він так само добре працюватиме з фінансовими даними. Тобто кожен алгоритм має свої переваги та недоліки. Спробуємо просто описати алгоритм роботи кожного алгоритму. Нагадаю, що математичне представлення роботи кожного алгоритму наведено у Розділі 3.

- Алгоритм K-Means є популярним методом кластеризації даних. Його часто використовують у задачах кластеризації. Проте він має великий недолік – необхідно вказувати кількість кластерів на початку роботи алгоритму. Алгоритм методу полягає у виборі кількості кластерів, далі ця кількість випадковим чином розкидається в даних і є центрами кластерів, наступним кроком є обчислення відстаней від кожної точки до центрів кластерів. Кожну точку даних відносять до того кластера, де відстань мінімальна, далі відбувається пере обчислення центру кластера.
- Hierarchical Agglomerative Clustering (HAC) є алгоритмом кластеризації «від листя до стовбура». Тобто на початку відбуваються обчислення матриці відстаней між даними. Далі точки даних з найменшими відстанями об’єднуються у кластер, знову обчислюється матриця відстаней. На основі матриці відстаней, яка постійно перераховується відбувається злиття або не злиття кластерів у один.
- Gaussian Mixture Models (GMM) дають більше гнучкості, ніж K-Means. З GMM припускаємо, що точки даних є розподіленими за Гаусом; це припущення є слабшим за припущення, всі точки

даних розміщено навколо середнього значення (K-Means). Таким чином, маємо аж два параметри для опису форми кластерів: середнє значення та стандартне відхилення. Якщо розглянути приклад у двох вимірах, це означає, що кластери можуть мати будь-яку еліптичну форму (оскільки ми маємо стандартне відхилення в обох напрямках (*xiy*). Таким чином, кожен розподіл Гауса призначається одному кластеру. Щоб знайти параметри розподілу Гауса для кожного кластера (наприклад, середнє значення та стандартне відхилення), використовуватимемо алгоритм оптимізації під назвою «Очікування–максимізація» (EM). Отже, маємо 2 ключові переваги використання GMM. По-перше, GMM набагато гнучкіші з точки зору кластерної коваріації, ніж K-Means; завдяки параметру стандартного відхилення кластери можуть набувати будь-якої форми еліпса, а не бути обмеженими колами. K-Means насправді є окремим випадком GMM, у якому коваріація кожного кластера вздовж усіх вимірів наближається до 0. По-друге, оскільки GMM використовують ймовірності, вони можуть мати кілька кластерів на точку даних. Отже, якщо точка даних знаходиться в середині двох кластерів, що перекриваються, ми можемо просто визначити її клас, сказавши, що вона належить у відсотках *X* до класу 1 і у відсотках *Y* до класу 2. Тобто GMM підтримують змішане членство.

- Density-Based Spatial Clustering of Applications with Noise (DBSCAN) базується на інтуїтивно зрозумілому понятті «кластерів» і «шуму» (кластери – це щільні області в просторі даних, розділені областями меншої щільності точок – шум). Ключова ідея полягає в тому, що для кожної точки кластера околиці даного радіуса повинні містити принаймні мінімальну кількість точок. Методи поділу K-Means та ієрархічна кластеризація працюють добре для пошуку кластерів сферичної

форми або опуклих кластерів. Іншими словами, вони підходять тільки для компактних і добре розділених скупчень. Крім того, на них також сильно впливає наявність шуму та викидів у даних. Реальні дані можуть містити відхилення, наприклад: кластери можуть бути довільної форми, дані можуть містити шум. Тут якраз на допомого приходить алгоритм кластеризації DBSCAN. Алгоритм DBSCAN вимагає на вхід два параметри:

- eps: визначає околиці навколо точки даних, тобто якщо відстань між двома точками менша або дорівнює «eps», то вони вважаються сусідами. Якщо значення eps вибрано замале, велика частина даних буде вважатися викидом. Якщо він вибраний дуже великим, кластери об’єднаються, і більшість точок даних буде в тих самих кластерах. Один із способів знайти значення eps – це графік k-відстаней.
- MinPts: мінімальна кількість сусідів (точок даних) у радіусі eps. Чим більший набір даних, тим більше значення MinPts потрібно вибрати. Як правило, мінімальний MinPts можна отримати з кількості вимірів D у наборі даних як MinPts >= D+1. Мінімальне значення MinPts має бути не менше 3.

### 4.2. Набір даних для кластеризації

У роботі використано сім наборів даних, п'ять з яких є згенерованими з нормального розподілу, два – з рівномірного розподілу. За допомогою лінійної комбінації згенерованих даних отримано наступні множини або хмари даних, які потрібно розділити на кластери чотирьома методами кластеризації. На малюнку нижче показано п'ять типів вхідних даних. Наша мета – здійснити кластеризацію цих наборів за допомогою кожного алгоритму кластеризації та встановити, який алгоритм працює найкраще.

Кожен із п’яти наборів даних на Рис. 4 названий відповідним чином до

розсіювання точок даних: Галактика (Galaxy), Серпи (Sickle), Смужки (Slash), Око (Eye), Рій (Swarm).

Для того, щоб побачити, як на практиці працює кластеризація даних розглянемо ще три набори даних, отримані з платформи Kaggle. Перший набір даних доступний за наступним посиланням https://www.kaggle.com/datasets/arjunbhasin2013/ccdata?datasetId=14701&language=R. Ці дані стосуються сегментації клієнтів для визначення маркетингової стратегії. Даний набір даних узагальнює поведінку використання приблизно 9000 активних власників кредитних карток протягом останніх 6 місяців. Для кожного клієнта файл містить 18 поведінкових змінних. Для здійснення кластеризації використано два стовпці – BALANCE: сума балансу, що залишилася на рахунку клієнта для здійснення покупок та PURCHASES: загальна кількість покупок, здійснених з рахунку.

Другий набір даних https://www.kaggle.com/datasets/harrywang/wine-dataset-for-clustering є результатами хімічного аналізу вин, вирощених в одному регіоні Італії, але отриманих з трьох різних сортів винограду. Всього знайдено 13 типів складових у трьох типах вина. Для здійснення кластеризації даних використано такі складові: Alcohol та Malic_Acid.

Третій набір даних https://www.kaggle.com/datasets/uciml/electric-power-consumption-data-set містить вимірювання споживання електроенергії в одному домогосподарстві з однохвилинною частотою записів протягом майже 4 років. Доступні різні електричні величини та деякі субметричні значення. У даній кваліфікаційній роботі використано такі змінні: Global_active_power та Global_reactive_power. Кожен із згаданих наборів даних названий відповідним чином – Card, Wine, Elect.

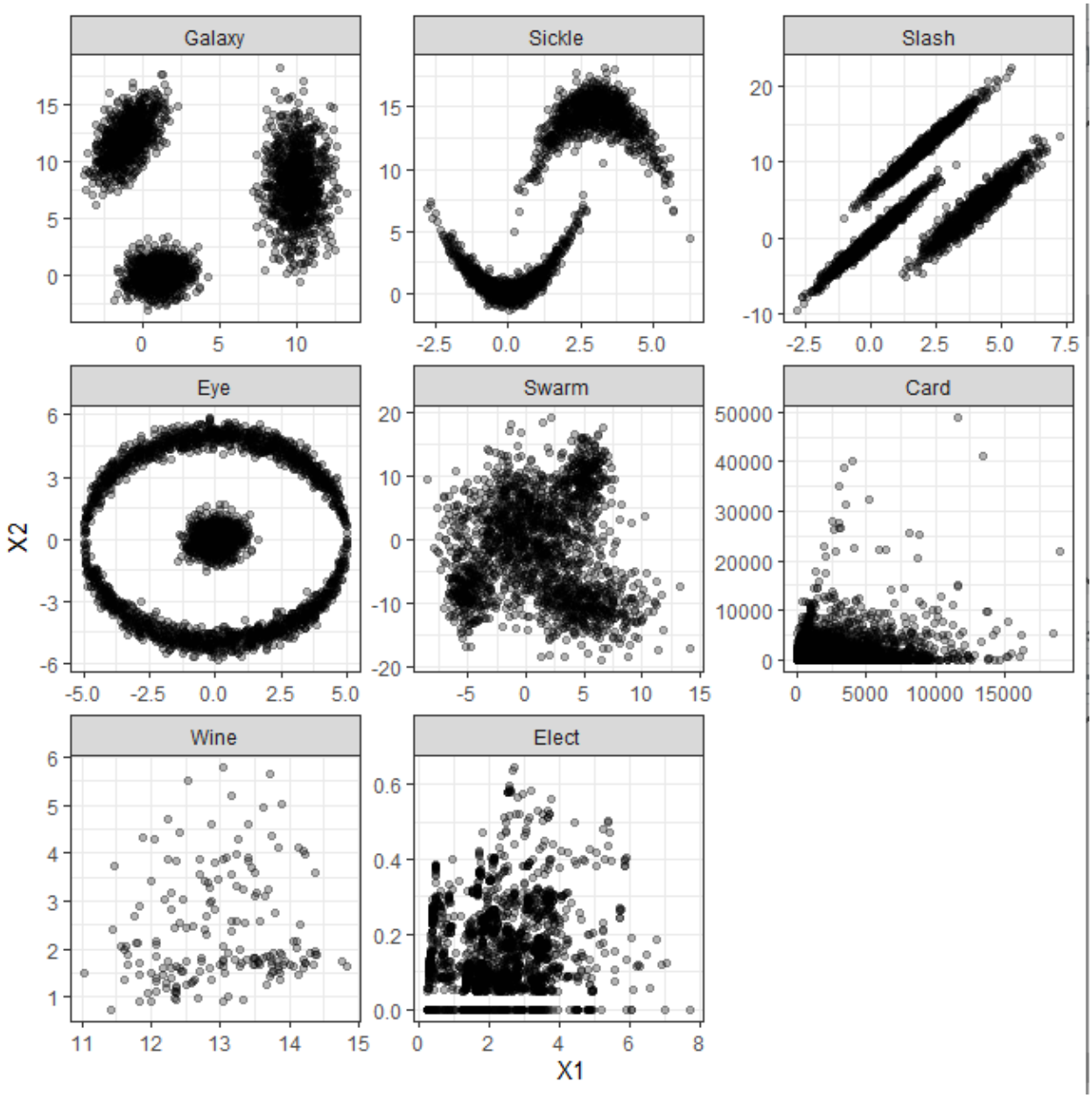



Рисунок 4. Вхідні дані, які необхідно розділити на кластери

### 4.3. Опис програмного середовища R

❖ ***Переваги мови R***

- ✓ R – це найповніший пакет статистичного аналізу, оскільки нові технології та ідеї часто з'являються спочатку в R.
- ✓ R є відкритим вихідним кодом, тому є можливість запускати R будь-де в будь-який час і навіть продавати його на умовах ліцензії.

- ✓ Це кросплатформна програма, яка працює на багатьох операційних системах, найкраще для GNU/Linux і Microsoft Windows.
- ✓ У R кожен може надати виправлення помилок, вдосконалення коду та створювати нові бібліотеки.

❖ ***Недоліки мови R***

- ✓ Якість деяких пакетів у R – не ідеальна.
- ✓ Немає служби підтримки клієнтів мови R, на яку можна поскаржитися, якщо щось не працює.
- ✓ Команди R майже не стосуються керування пам'яттю, тому R може споживати всю доступну пам'ять.

Незважаючи на недоліки, наука про дані є найпопулярнішою технологією сьогодні у світі. Оскільки ця наука в основному складається зі статистики, R є невід'ємним інструментом для роботи в цій галузі.

### 4.4. Результати кластеризації

Отже, у даній роботі згенеровано дані із нормального та рівномірного розподілів. Тобто дані є симуляцією розподілів. Такі симуляції або симуляції за допомогою методу Монте-Карло [43] часто застосовують для перевірки роботи та порівняння алгоритмів. Після того ми використаємо реальні дані для здійснення кластеризації чотирьом методами.

Кожен із алгоритмів вимагає налаштування параметрів кластеризації для його якісної роботи. ***Кластеризація сильно залежить від конкретного набору даних і мети аналізу***. Отже, розглянемо, як кожен алгоритм працює в кожному випадку, щоб дати уявлення про те, коли використовувати певний алгоритм. Кожен набір даних має дві функції $X1$ і $X2$, які генерують точки кластерів із наведених вище розподілів, і мітку набору даних (для цілей візуалізації даних). Назви були просто навіяні формою даних і аж ніяк не є технічними назвами.

Функція calc_cluster приймає набір даних і параметри для кожного

алгоритму як аргументи, обчислює кластери та додає відповідні мітки до кожного кластера даних. Функція plot_cluster приймає назву набору даних як аргумент і будує графіки кластерів, обчислених кожним алгоритмом.

Для кожного набору даних параметри алгоритму змінюються, щоб уникнути помилкового враження про недостатню продуктивність. Важливо відзначити, що DBSCAN іноді виводить «нульовий» кластер даних, який вказує на аутлаєри, виявлені алгоритмом.

Отже, розглянемо результати кластеризації найпростішого випадку – Галактика (Galaxy).

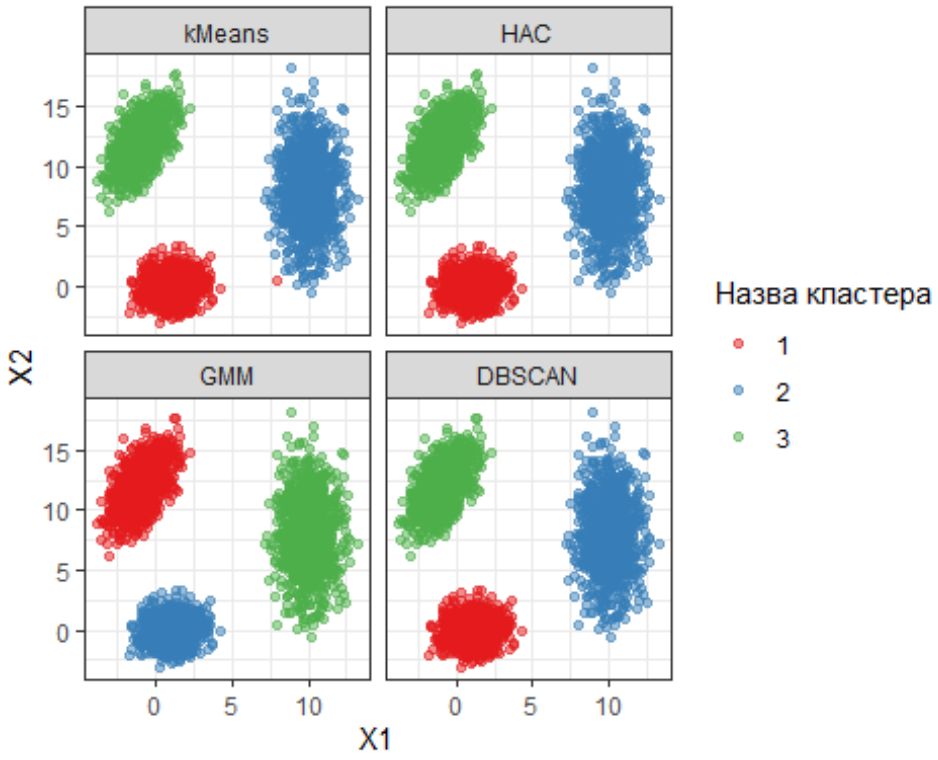



Рисунок 5. Результати кластеризації набору даних Galaxy

Як видно із Рис. 5 у цьому найпростішому випадку немає проблем (за винятком точки «зловмисника» у синьому кластері, заданих k-Means). Наступним набором даних є Sickle. Тобто розподіл даних є уже не таким очевидним для деяких алгоритмів кластеризації.

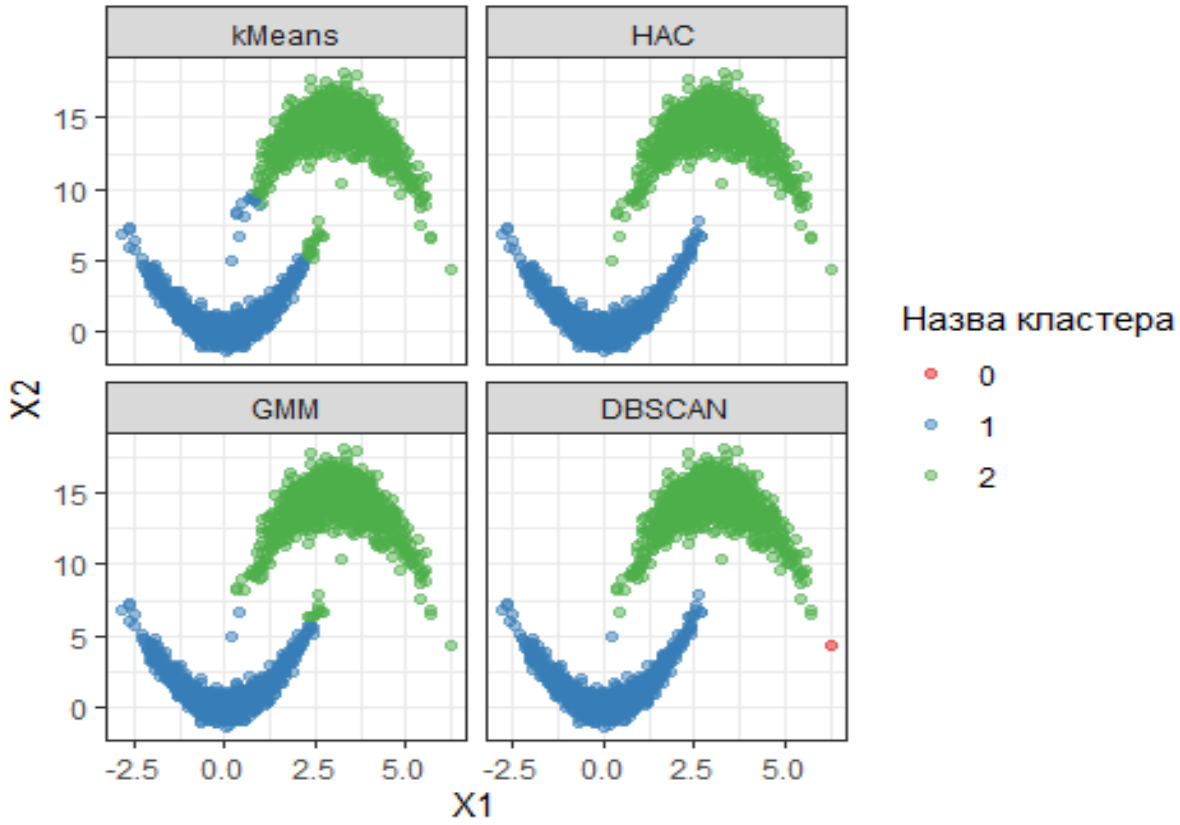



Рисунок 6. Результати кластеризації набору даних Sickle

Як видно із Рис. 6 найкраще з кластеризацією Sickle впоралися алгоритми DBSCAN та HAC. K-Means та GMM містять зелені точки даних у нижньому серпі. Це пов’язано з тим, що центр зеленого кластера знаходиться ближче до крайніх точок з нижнього кластера. Оце і є величезним недоліком вказаних двох підходів. Червона крапка у методі DBSCAN позначає нульовий кластер, тобто аутлаєр (викид). Алгоритм DBSCAN вважає червоне вимірювання аутлаєром в даних. Отже, алгоритм DBSCAN краще пристосовується до конкретних форм кластерів.

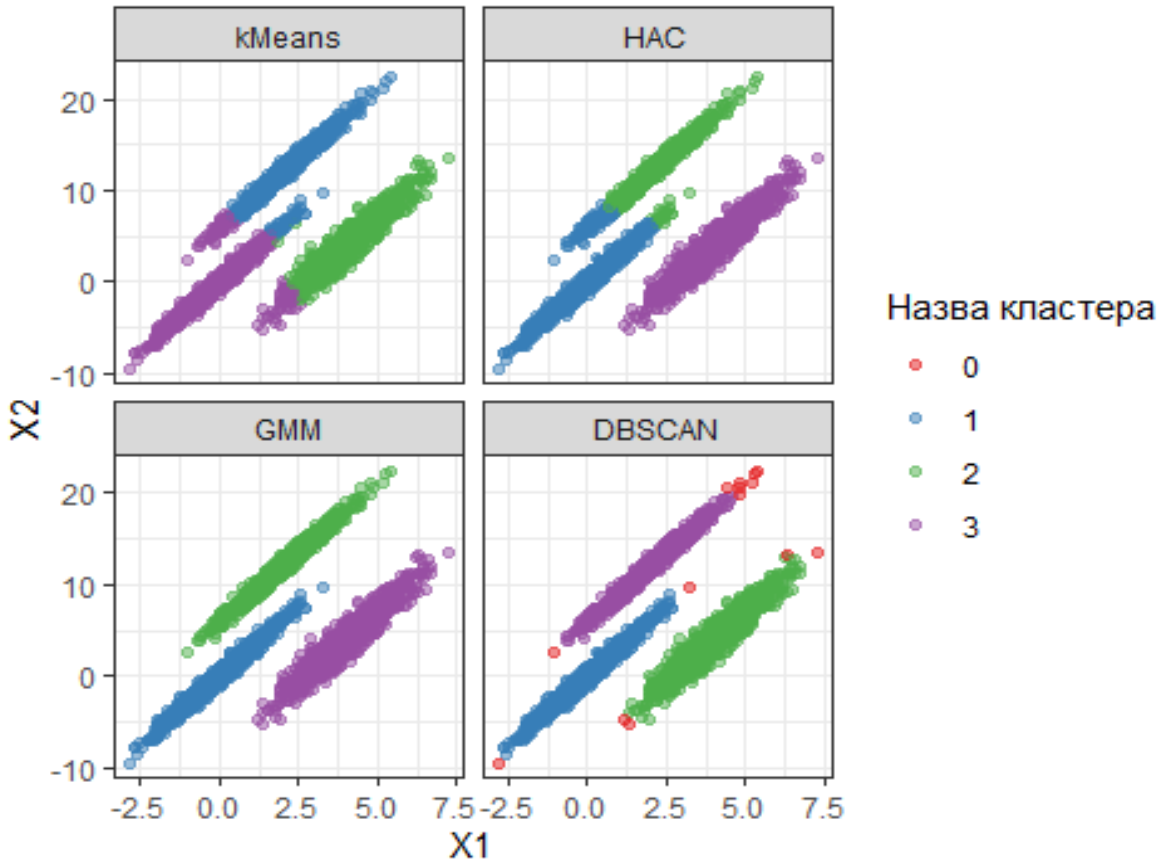



Рисунок 7. Результати кластеризації набору даних Slash

З Рис. 7 зрозуміло, що лідерами у даному випадку є методи DBSCAN та GMM. У цьому випадку пальму першості віддаємо методу GMM, так як DBSCAN вказав на наявність аутлаєрів. K-Means та HAC, як видно на рисунку, не впоралися із кластеризацією у даному випадку.

На Рис. 8 представлено результати кластеризації набору даних Eye чотирьома алгоритмами. Серед лідерів знову алгоритм DBSCAN та алгоритм HAC. K-Means та GMM здійснили неправильну кластеризацію.

Здається, що лідером є алгоритм DBSCAN. Не будемо поспішати з висновками, подивимося на результати кластеризації наступного набору даних – Swarm. У даному випадку бачимо, що K-Means, GMM, HAC чудово впоралися із задачею класифікації, DBSCAN – не впорався.

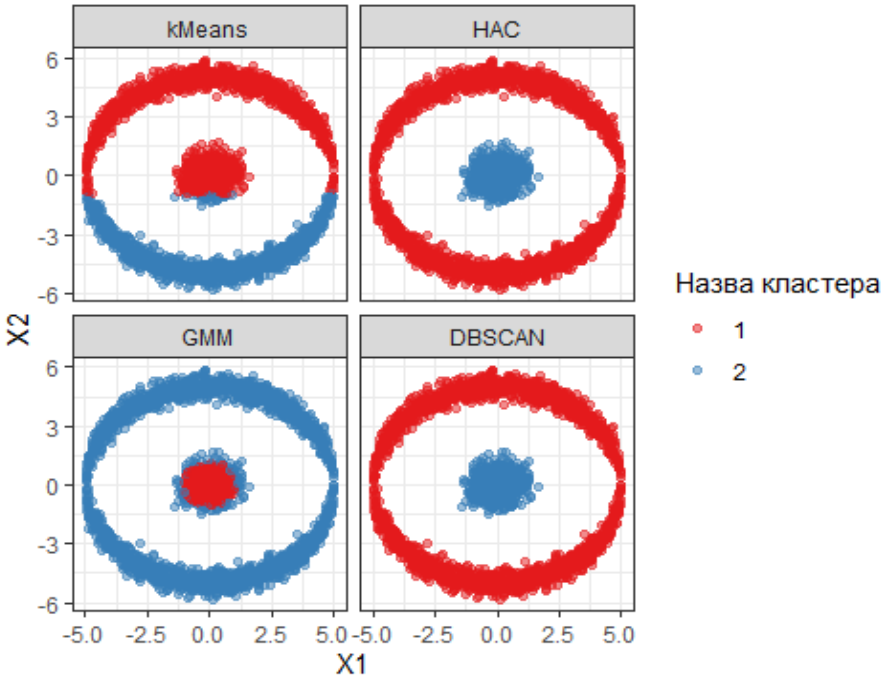



Рисунок 8. Результати кластеризації набору даних Eye

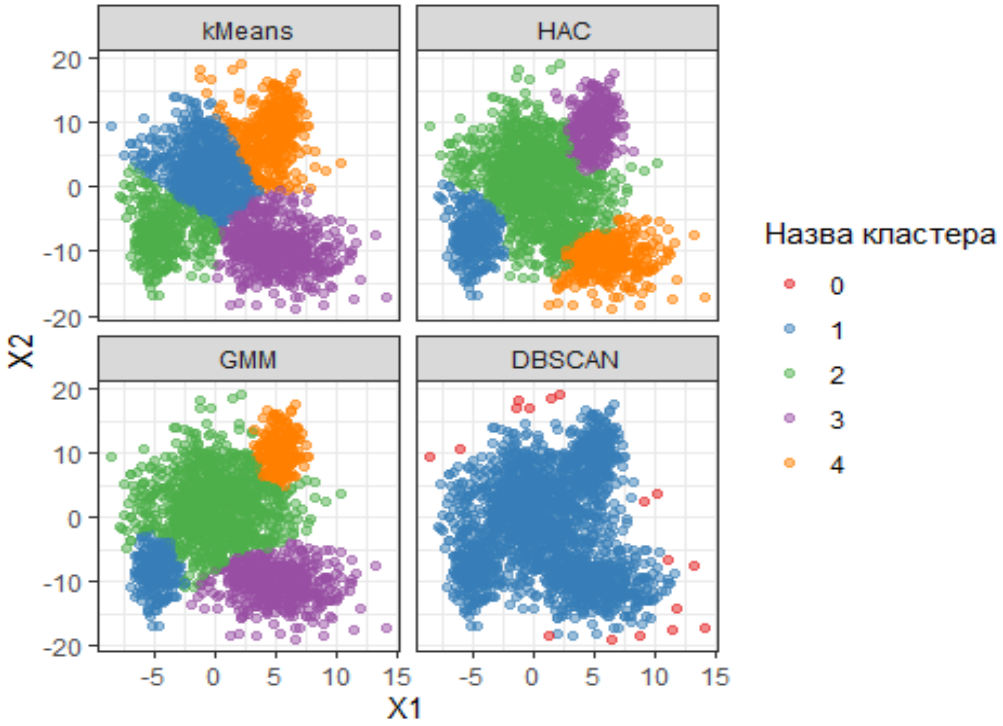



Рисунок 9. Результати кластеризації набору даних Swarm

Для набору даних Swarm (Рис. 9) немає чітких меж між кластерами. Саме у цьому випадку важко використовувати DBSCAN, тому що просто неможливо вибрати параметри для розділення даних на певну кількість кластерів (у нашому випадку від 3 до 6). Наприклад, якщо значення $\varepsilon$ зменшити, щоб визначити меншу околицю, отримаємо більше 15 кластерів.

Отже, розглянемо результати кластеризації. Маємо двох лідерів – DBSCAN (не впорався з набором Swarm) та HAC (не впорався з набором Slash), які правильно кластеризували 4 із 5 наборів даних. На другому місці – алгоритм GMM (не впорався з наборами Eye та Sickle), який правильно кластеризував 3 із 5 наборів даних. Алгоритм K-Means правильно кластеризував лише один набір даних із 5 – Galaxy. K-Means – це інтуїтивно зрозумілий швидкий алгоритм, але він не в змозі обробляти випадки, коли кластери погано розділені або перекриваються, оскільки центр кластера визначається середнім значенням його точок.

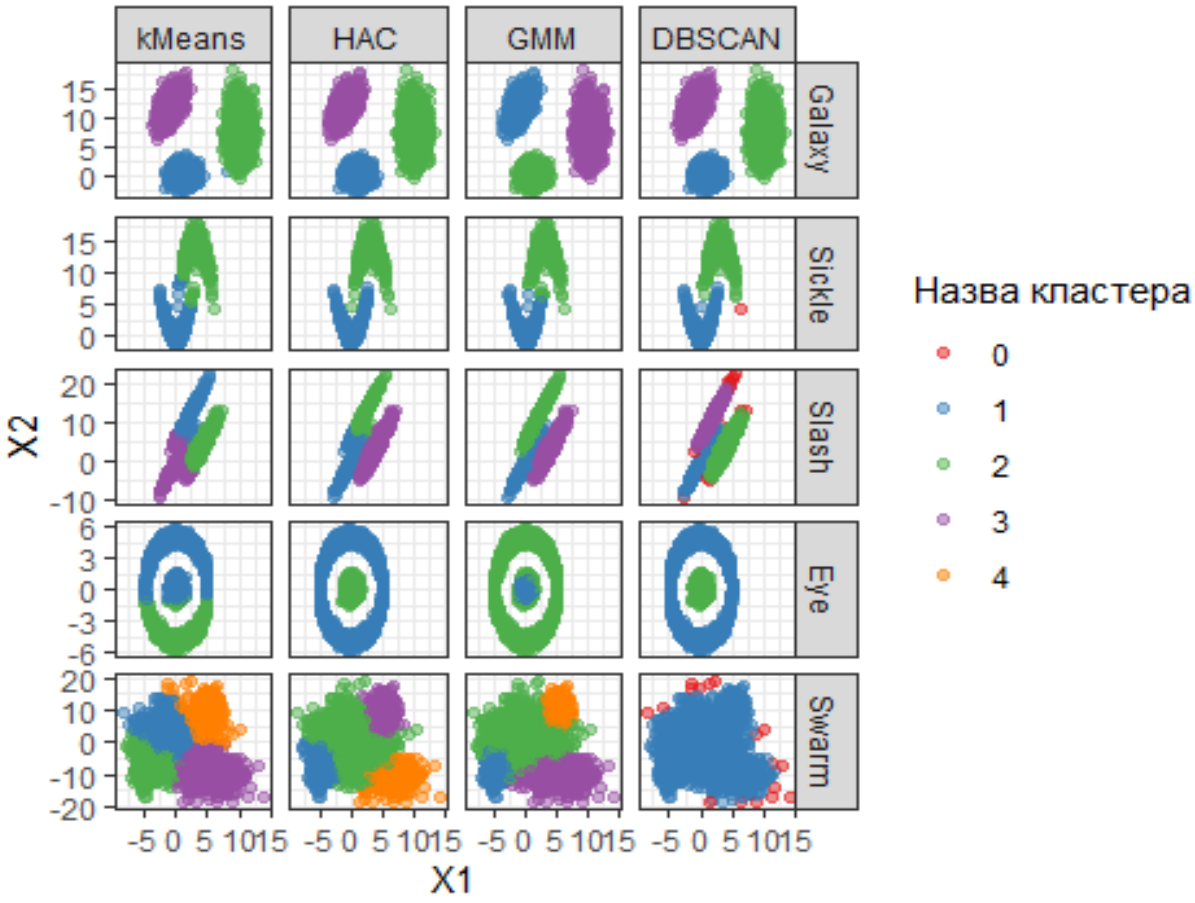



Рисунок 10. Результати кластеризації усіх згенерованих наборів даних

Розглянемо тепер результати кластеризації наборів даних Card, Wine, Elect. На Рис. 11 – 13 показано кластеризовані набори даних, відповідно. На Рис. 11 здійснено кластеризацію даних Card. Як бачимо, метод DBSCAN кластеризує дані на основі їх щільності. Дані є щільними біля точки початку

координат. Тому цей алгоритм відніс зобразив це скупчення одним кластером. Червоні крапочки вважаються аутлайєрами даних. Зелений кластер містить невелику кількість елементів. Інші три алгоритми кластеризації вважають, що ці дані варто розділити на 3 кластери. Проте метод поділу абсолютно відрізняються.

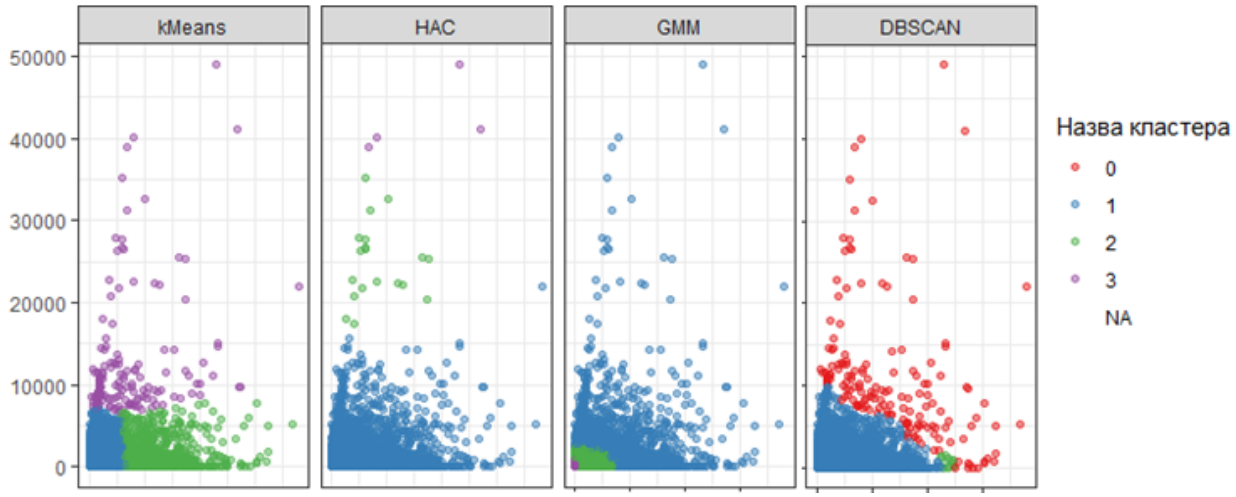



Рис. 11. Кластеризація набору даних Card

На Рис. 12 показано кластеризацію даних Wine. Як бачимо, алгоритми кластеризації kMeans, HAC, GMM здійснили схоже розбиття на 3 групи наших даних. Метод DBSCAN кластеризував дані на два кластери. Знову причиною є щільність точок даних зі зростанням $Y$ координати.

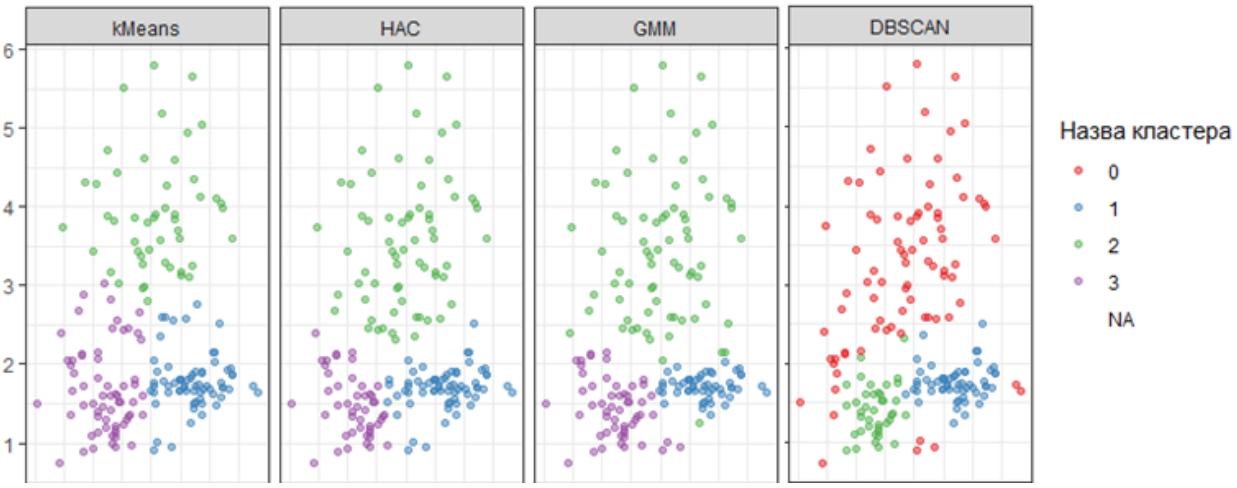



Рис. 12. Кластеризація набору даних Wine

На Рис. 13 показано кластеризацію даних для набору Elect. Тут знову результати кластеризації даних алгоритмами kMeans, HAC, GMM є схожими, а результати кластеризації за допомогою алгоритму DBSCAN відрізняються

від інших.

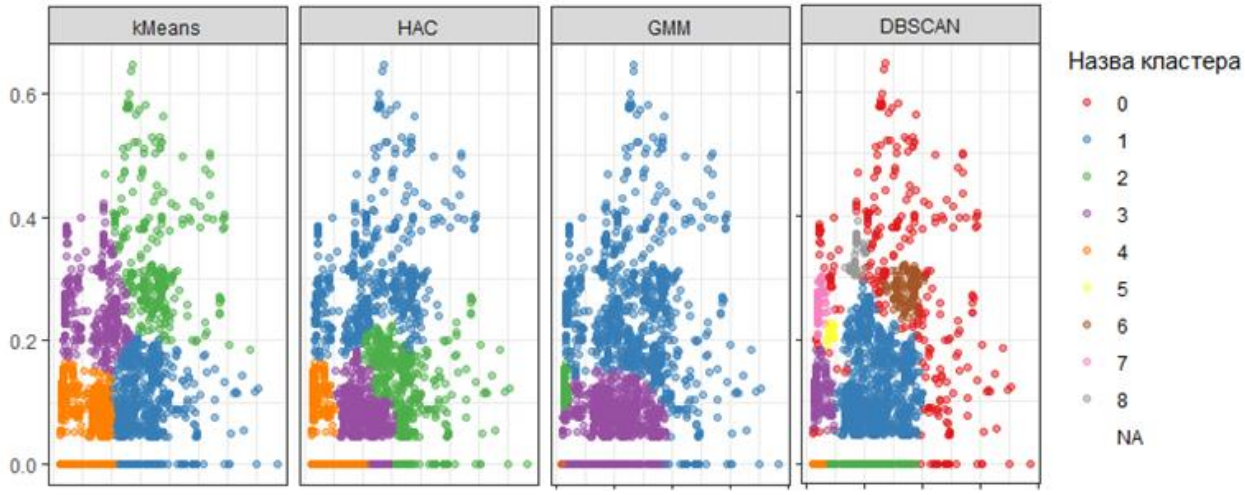



Рис. 13. Кластеризація набору даних Elect

### 4.5. Створення Shiny App

У середовищі R Programming за допомогою пакету RStudio створено Shiny App (інтерактивна веб-сторінка). Цю сторінку розміщено на безкоштовному хмарному сервері shinyapps.io. Меню веб-додатку складається із вигляду вхідних даних, знаходження оптимальної кількості кластерів за допомогою чотирьох підходів та демонстрації або інтерактивної кластеризації даних на вибрану кількість кластерів.

Веб-сторінка доступна за наступним посиланням: https://wc7rar-brainshturm-math0statistics0science.shinyapps.io/Nick/. Веб-додатки Shiny популярні через наступні моменти:

1. Прості у використанні: Shinny app легко створювати та використовувати. Вони дозволяють користувачам взаємодіяти з даними та моделями більш інтуїтивно зрозумілим способом, ніж традиційні інтерфейси командного рядка.

2. Інтерактивність: програми Shiny є інтерактивними, що дозволяє користувачам досліджувати дані та моделі в режимі реального часу. Це полегшує розуміння складних наборів даних і моделей.

3. Можливість налаштування: Shinny app можна налаштувати відповідно до потреб різних користувачів. Їх можна пристосувати до конкретних галузей

або випадків використання.

4. Масштабованість: програми можна масштабувати для обробки великих наборів даних і розгортати на різноманітних платформах, включаючи настільні комп'ютери, сервери та Інтернет.

5. З відкритим вихідним кодом: програми створюються за допомогою R, мови програмування з відкритим кодом. Це означає, що код знаходиться у вільному доступі.

Використовуючи веб-додатки Shiny, кластеризація даних стає набагато зрозумілішою задачею. Використовуємо набір даних про кредитні картки клієнтів банку Card, вірніше, перші 200 записів. Самий набір містить 8500 записів після видалення пропущених значень (NA). Так як для обчислення матриць відстаней потрібен час, використаємо для демонстрації лише перші 200 значень.


| CUST_ID | BALANCE | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PURCHASES | CASH_ADVANCE | PURCHASES_FREQUENCY |
|---|---|---|---|---|---|---|
| C10001 | 40.90 | 95.40 | 0.00 | 95.40 | 0.00 | 0.17 |
| C10002 | 3202.47 | 0.00 | 0.00 | 0.00 | 6442.95 | 0.00 |
| C10003 | 2495.15 | 773.17 | 773.17 | 0.00 | 0.00 | 1.00 |
| C10005 | 817.71 | 16.00 | 16.00 | 0.00 | 0.00 | 0.08 |
| C10006 | 1809.83 | 1333.28 | 0.00 | 1333.28 | 0.00 | 0.67 |
| C10007 | 627.26 | 7091.01 | 6402.63 | 688.38 | 0.00 | 1.00 |
| C10008 | 1823.65 | 436.20 | 0.00 | 436.20 | 0.00 | 1.00 |
| C10009 | 1014.93 | 861.49 | 661.49 | 200.00 | 0.00 | 0.33 |
| C10010 | 152.23 | 1281.60 | 1281.60 | 0.00 | 0.00 | 0.17 |
| C10011 | 1293.12 | 920.12 | 0.00 | 920.12 | 0.00 | 1.00 |
| C10012 | 630.79 | 1492.18 | 1492.18 | 0.00 | 0.00 | 0.25 |
| C10013 | 1516.93 | 3217.99 | 2500.23 | 717.76 | 0.00 | 1.00 |
| C10014 | 921.69 | 2137.93 | 419.96 | 1717.97 | 0.00 | 0.75 |
| C10015 | 2772.77 | 0.00 | 0.00 | 0.00 | 346.81 | 0.00 |
| C10016 | 6886.21 | 1611.70 | 0.00 | 1611.70 | 2301.49 | 0.50 |
| C10017 | 2072.07 | 0.00 | 0.00 | 0.00 | 2784.27 | 0.00 |



Рис. 14. Вигляд вхідних даних

Для кластеризації використовуємо стовпці 2 та 3. Наступні Рис. 15-16 показують результат роботи методів визначення оптимальної кількості кластерів – NbClust, Метод ліктя, Метод силуету, Gap Statistic Method. Кожен із цих методів показує різну оптимальну кількість кластерів. Це ще один доказ того, що немає універсальних підходів та методів. Задача оптимальної кластеризації даних залишається відкритою.

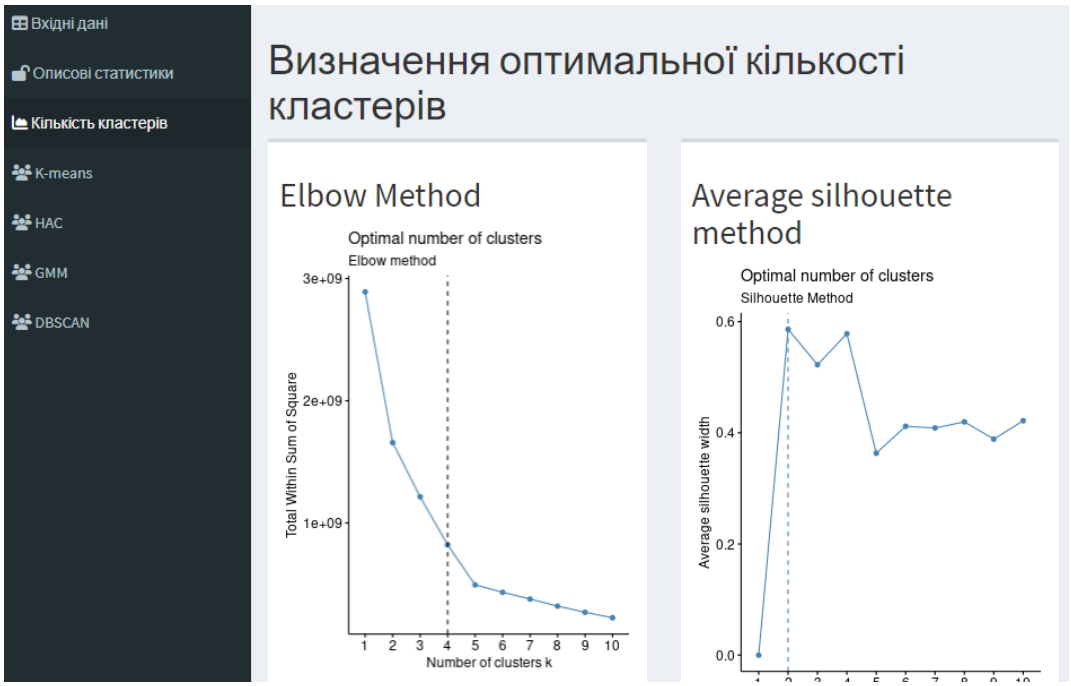



Рис. 15. Визначення оптимальної кількості кластерів

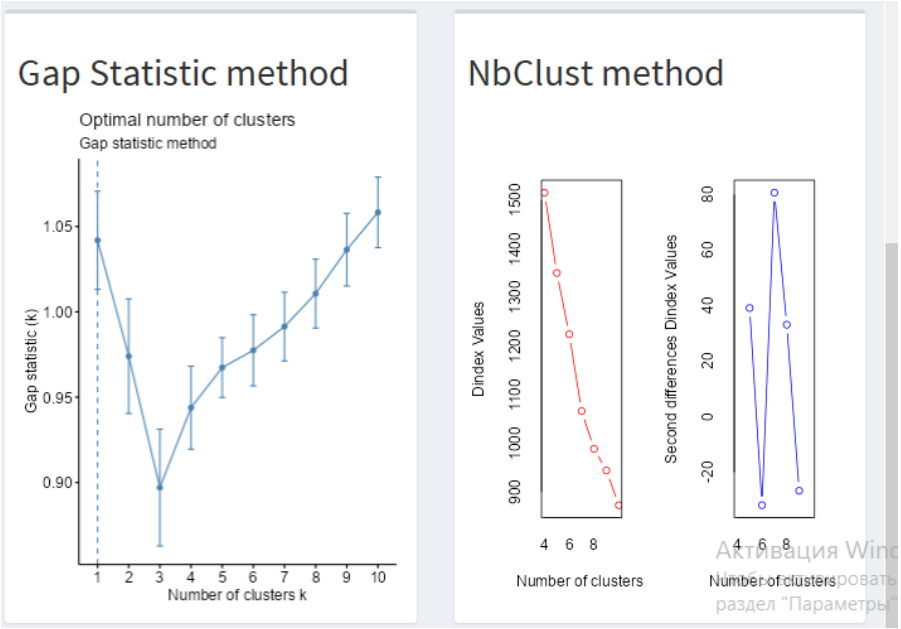



Рис. 16. Визначення оптимальної кількості кластерів

Смуга прокручування дозволяє обрати кількість кластерів для кожного методу кластеризації даних. Статистика кластерів – середнє значення в кластері, кількість елементів, медіана тощо допомагають зробити висновки спеціалістам зі сфери даних, яка розглядається. Таким чином для кожного кластера користувачів кредитних карток може бути застосована своя вигідна пропозиція або обмеження. Наступні Рисунки показують статистику

кластеризації та сам процес кластеризації за допомогою чотирьох підходів.

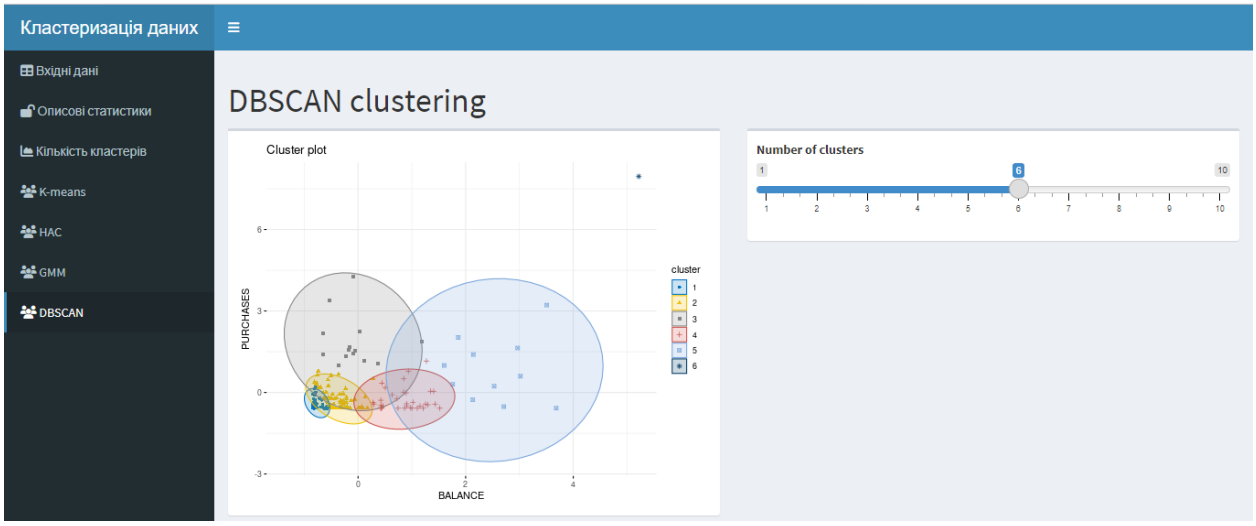



Рис. 17. Кластеризація даних методом DBSCAN на 6 кластерів

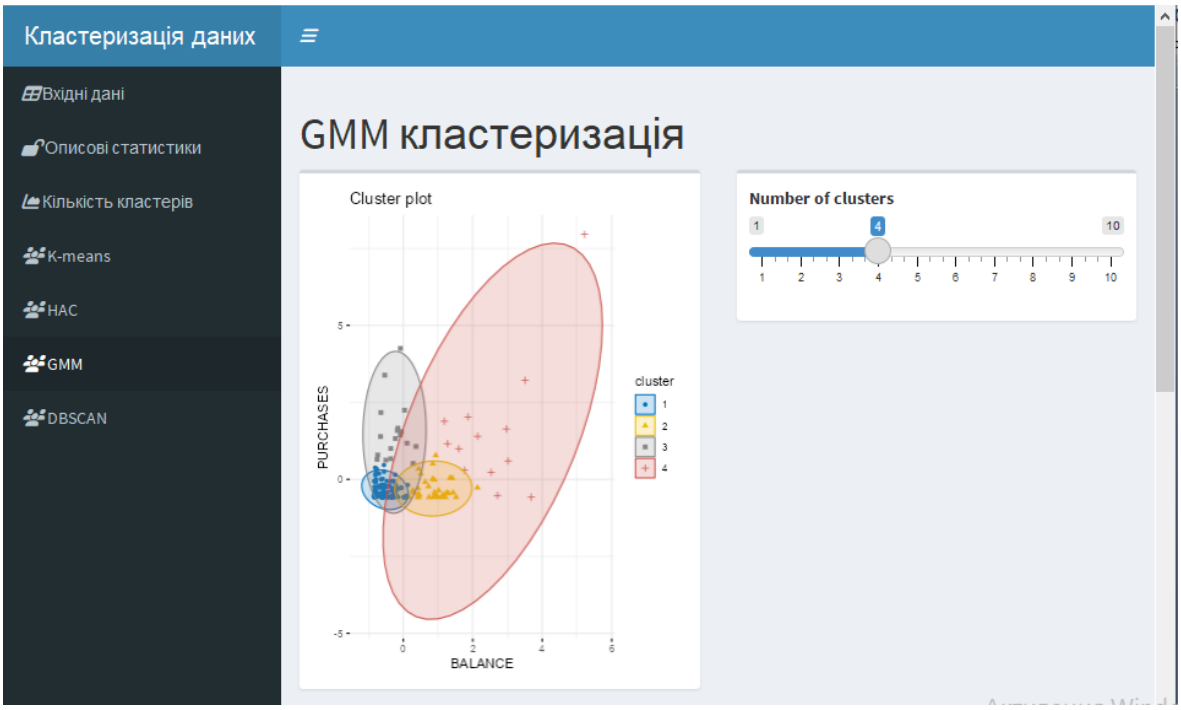



Рис. 17. Кластеризація даних методом GMM на 4 кластери

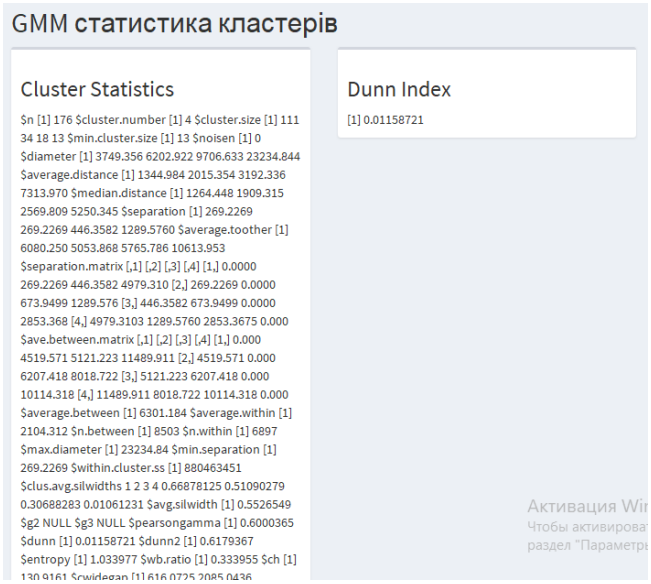



Рис. 18. Статистика GMM кластеризації

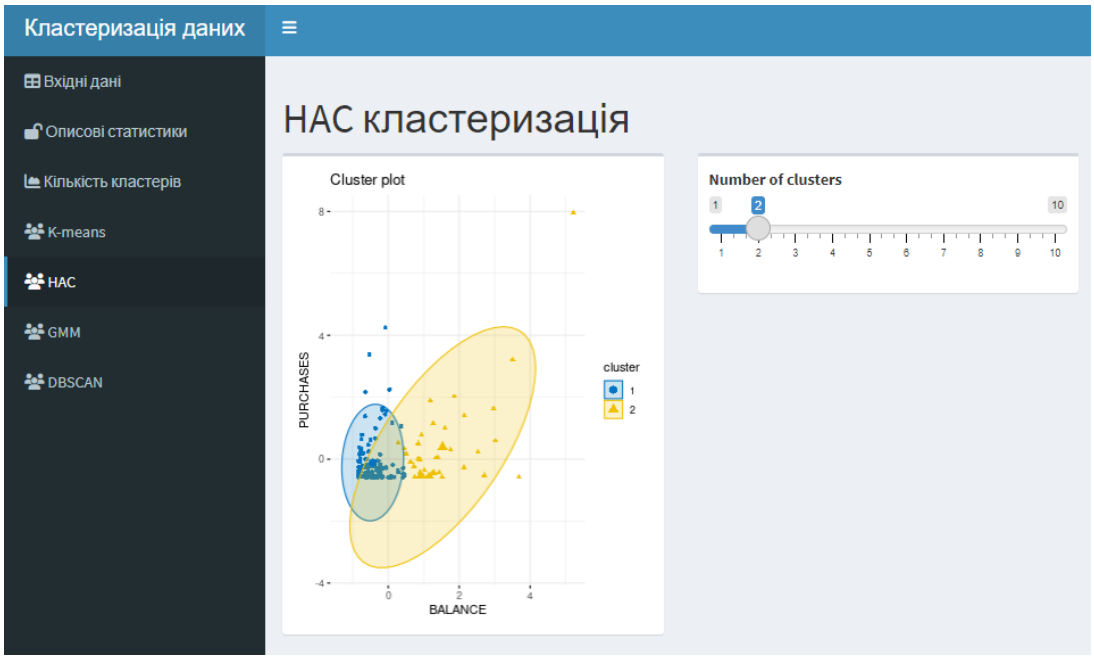



Рис. 19. Кластеризація даних методом HAC на 2 кластери

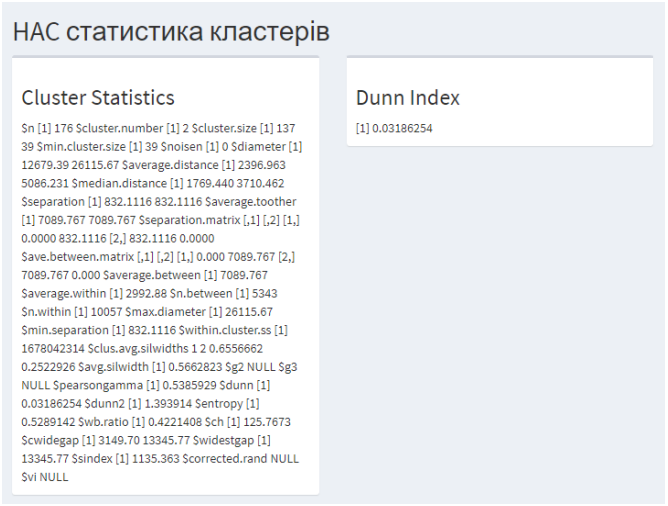


Рис. 20. Статистика HAC кластеризації

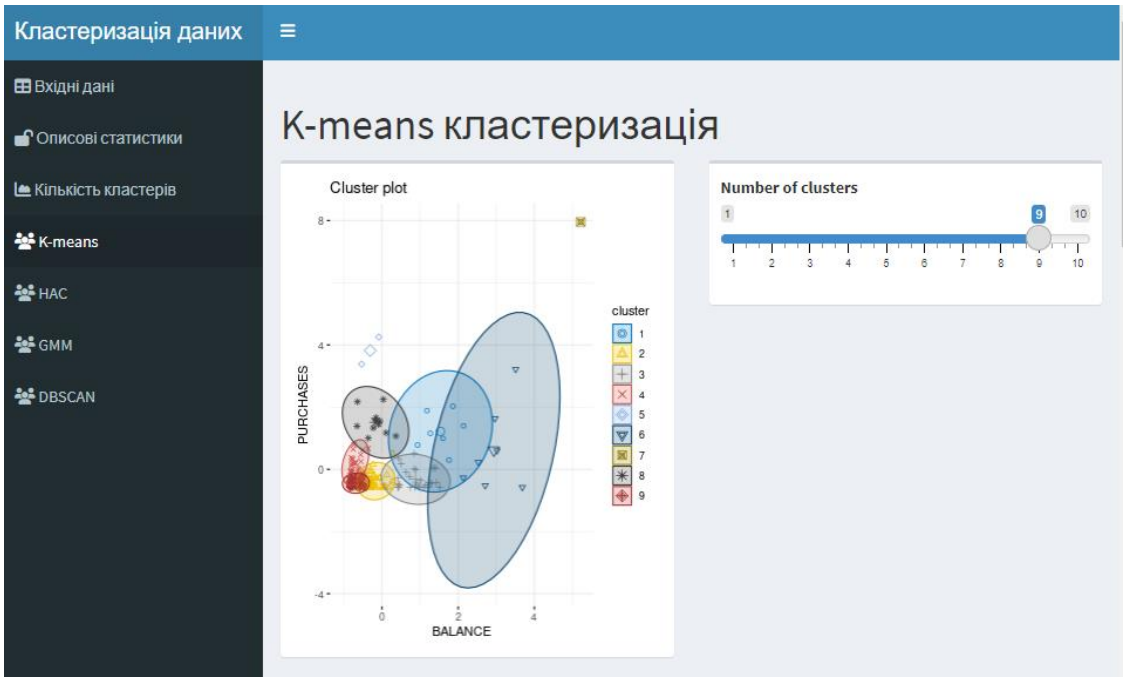



Рис. 21. Кластеризація даних методом k-Means на 9 кластерів

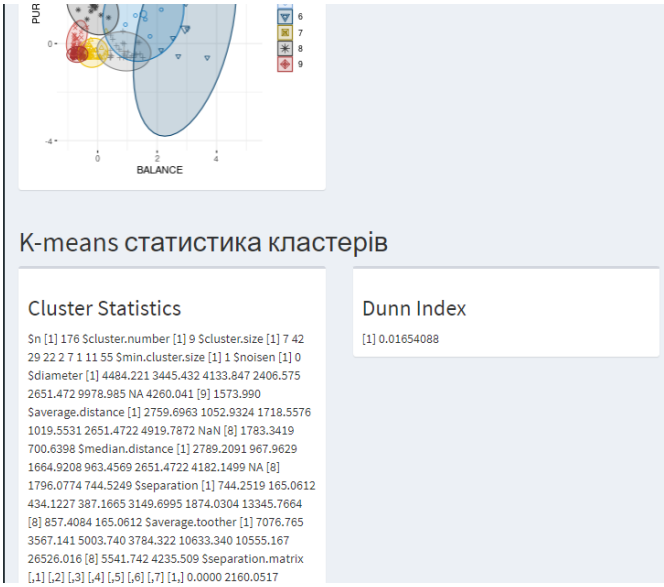



Рис. 22. Статистика k-Means кластеризації

Індекс Dunn є мірою валідності кластерів, яка використовується для оцінки якості рішень кластеризації. Чим меншим є індекс Dunn, тим точніше здійснено кластеризацію даних.

# ВИСНОВКИ

Кластеризація є однією з найбільш фундаментальних проблем інтелектуального аналізу даних через її численні застосування для сегментації клієнтів, цільового маркетингу та узагальнення даних. Численні класи методів були запропоновані в літературі, такі як ймовірнісні методи, методи на основі відстані, методи на основі щільності, методи на основі сітки, методи факторизації та спектральні методи. Проблема кластеризації тісно пов'язана з проблемами зменшення розмірності, особливо через прогнозовані формулювання кластеризації. Багатовимірні дані часто є складними для аналізу через зростаючу розрідженість даних. Методи кластеризації можна розглядати як інтеграцію методів вибору ознак/зменшення розмірності з кластеризацією.

Практична частина дослідження під час написання кваліфікаційної роботи містить порівняння наступних алгоритмів кластеризації даних: K-Means, Hierarchical Agglomerative Clustering (HAC), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization clustering using Gaussian Mixture Models (GMM). Набори даних були попередньо згенерованими за допомогою нормального та рівномірного розподілів. Для кожного з отриманих наборів даних застосовано перелічені алгоритми кластеризації та визначено кращий алгоритм за результатом усіх кластеризацій. Показано, що серед розглянутих алгоритмів найкраще з задачею кластеризації впоралися алгоритми DBSCAN та HAC.

За допомогою трьох наборів даних, отриманих з платформи Kaggle, здійснено кластеризацію реальних даних. Демонстрацію роботи алгоритмів кластеризації даних здійснено з використанням пакету Shiny у створеному веб-додатку. Отриманий веб-додаток розміщено на хмарному сервері shinyapps.io.

## Список використаних джерел


[1]C. C. Aggarwal. A human-computer interactive system for projected clustering, *ACM KDD* Conference, *2001*.

[2]*C. Aggarwal*. Managing and Mining Uncertain Data, *Springer, 2009.*

[3]C. Aggarwal. Towards systematic design of distance functions for data mining applications, KDD Conference, *2003.*

[4]R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD Conference,* 1998.

[5]C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for clustering evolving data streams. *In* VLDB Conference, *2003.*

[6]C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space, *ICDT Conference,* 2001.

[7]C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J.-S. Park. Fast algorithms for projected clustering. In *ACM SIGMOD Conference,* 1999.

[8]R Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms and Applications,* Prentice Hall, 1993.

[9]P. Andritsos et al. LIMBO: Scalable clustering of categorical data. *EDBT Conference,* 2004.

[10]*S. Basu, I. Davidson, and K. Wagstaff*. Constrained clustering: Advances in theory, Algorithms and applications, *Chapman and Hall, 2008.*

[11]M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Computation,* 15(6), 2003.

[12]D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation, *Journal of Machine Learning Research,* 3:993-1022,2003.

[13]D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. *ACMSIGIR Conference,* pages 318-329,1992.

[14]A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society,* B, 39(1): 1-38,1977.

[15]J. Dean and S. Ghemawat. MapReduce: A flexible data processing took, *Communication of the ACM,* 53: 72-77,2010.

[16]I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning, ACM KDD Conference, *2001.*

[17]I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering, *ACM KDD Conference,* 2003.

[18]G. Das and H. Mannila. Context-based similarity measures for categorical databases. *PKDD Conference,* pages 201-210,2000.

[19]M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences,* 95(25): 14863-14868,1998. http://rana.lbl.gov/EisenSoftware.htm

[20]M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *ACM KDD Conference,* pages 226-231,1996.

[21]V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS-clustering categorical data using summaries. *ACMKDD Conference,* 1999.

[22]D. Gibson, J. Kleiberg, and P. Raghavan. Clustering categorical data: An approach based on Dynamical Systems. *VLDB Conference,* 1998.

[23]S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems,* 25(5):345-366,2000.

[24]S. Guha,R. Rastogi, andK. Shim. CURE: An efficient clustering algorithm for large databases. *In* ACM SIGMOD Conference, *1998.*

[25]D. Gunopulos and G. Das. Time-series similarity measures, and time series indexing, *ACM* SIGMOD Conference, *2001.*

[26]A. Hinneburg, C. Aggarwal, and D. Keim. What is the nearest neighbor in high dimensional spaces, *VLDB Conference*, 2000.

[27]A. Hinneburg, D. Keim, and M. Wawryniuk. Hd-eye: Visual mining of high-dimensional data. IEEE Computer Graphics and Applications, *19:22-31,1999.*

[28]T. Hofmann. Probabilistic latent semantic indexing. *ACM SIGIR Conference,* 1999.

[29]A. Jain. Data clustering: 50 years beyond Cmeans. *Pattern Recognition Letters,* 31(8):651- 666, 2010.

[30]A. Jain and R. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[31]D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering,* 16(11): 1370-1386,2004.

[32]I. Jolliffe. *Principal Component Analysis*, Springer, 2002.

[33]D. R. Karger. Random sampling in cut, flow, and network design problems, *STOC,* pp. 648657, 1994.

[34]*L. Kaufman and P. Rousseeuw.* Finding Groups in Data: An Introduction to Cluster Analysis. *Wiley-Interscience, 2005.*

[35]G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing,* 48(1):96-129,1998.

[36]D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization, *Nature,* 401:788-791,1999.

[37]B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs, *Bell System Tech. Journal,* 49:291-307, Feb. 1970.

[38]C. Ding, X. He, and H. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. *SDM Conference,* 2005.

[39]D. Gusfield. *Algorithms for Strings, Trees and Sequences,* Cambridge University Press, 1997.

[40]L. Liu, R. Jin, C. Aggarwal, and Y. Shen. Reliable clustering on uncertain graphs, *ICDM Conference*, 2012.

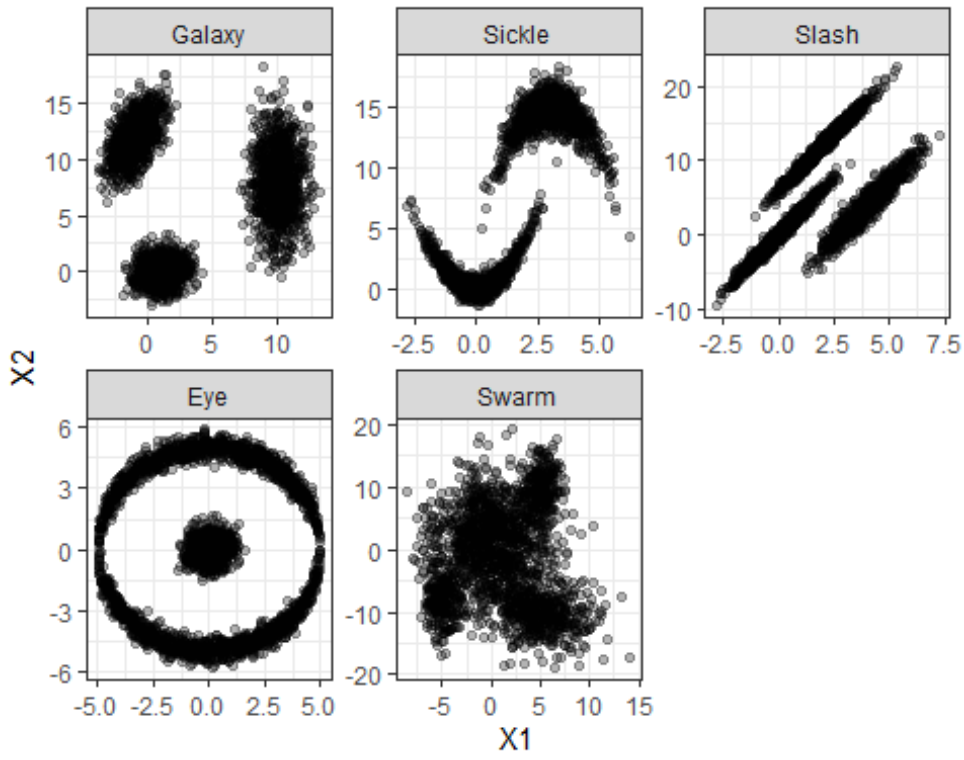
[41]S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: A survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics, *1(1):24-45,2004.*

[42]R. T. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Trans. Knowl. Data Eng.,* 14(5): 1003-1016,2002

[43]Simulations, Of & Zaidi, Habib & Labb, Claire & Morel, Christian. (1999). Improvement of the performance and accuracy of PET Monte Carlo simulations. Proc. SPIE. 3659. 10.1117/12.349537.

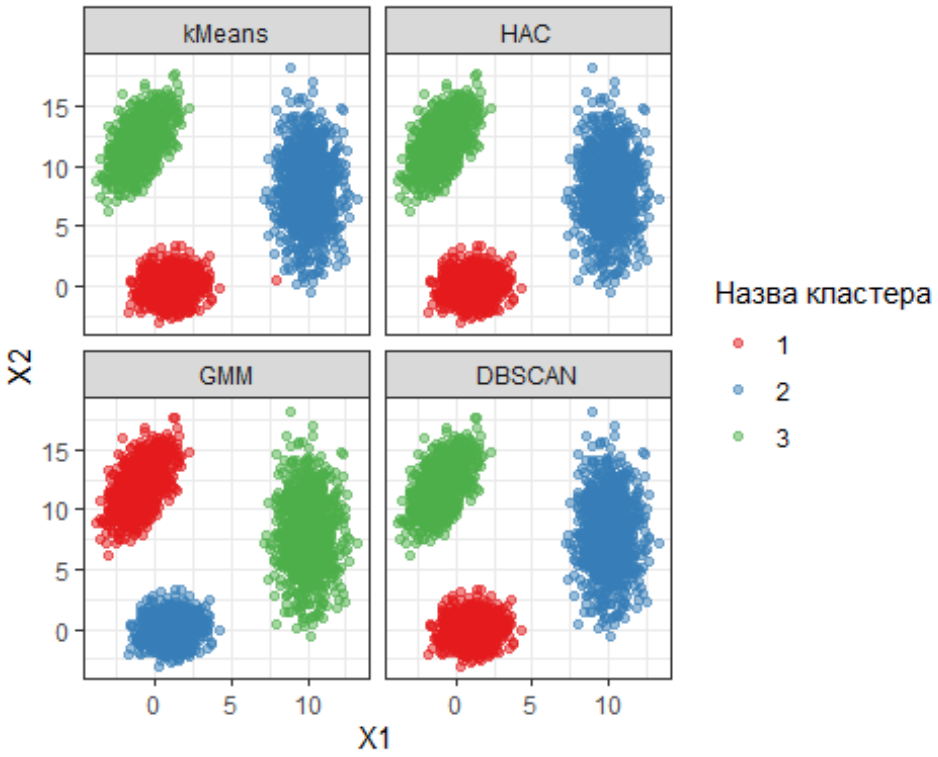
```r
library(dplyr)    # Пакет для обробки даних
library(tidyr)    # Пакет для обробки даних
library(ggplot2)  # Візуалізація даних

library(mclust)   # Пакет для кластеризації методом GMM
library(fpc)      # Пакет для кластеризації методом DBSCAN

set.seed(123)

# Генеруємо дані з нормального розподілу
norm1 <- rnorm(1000, mean = 0, sd = 1)
norm2 <- rnorm(1000, mean = 0, sd = 1)

norm3 <- rnorm(1000, mean = 0, sd = 3)
norm4 <- rnorm(1000, mean = 0, sd = 3)

norm5 <- rnorm(1000, mean = 0, sd = 5)

# Генеруємо дані з рівномірного розподілу
unif1 <- runif(1000, min = -2, max = 2)
unif2 <- runif(1000, min = -5, max = 5)

# Галактика: три добре розділених кластери
galaxy <- data.frame(X1 = c(norm1 + 1, norm1 + 10, norm1 - 1        ),
                     X2 = c(norm2    , norm3 + 8 , norm1 + norm3/2 +
12),
                     dataset = rep("Galaxy", 3000))

# Місяць: два кластери (параболи), які майже перетинаються
sickle <- data.frame(X1 = c(norm1         ,  norm1 + 3        ),
                     X2 = c(norm1^2 + norm2/2, -norm1^2 + norm2 + 15),
                     dataset = rep("Sickle", 2000))

# Смужки: три кластери у вигляді смужок даних
slash <- data.frame(X1 = c(norm1         , norm1 + 4         , norm2 +
2          ),
                    X2 = c(3*norm1 + norm2/2, 3*norm1 + norm3/3 + 4,
3*norm2 + norm1/2 + 12),
                    dataset = rep("Slash", 3000))

# Око: два концентричних кластера
eye <- data.frame(X1 = c(unif2                    , unif2
, norm1/2),
                  X2 = c(sqrt(25 - unif2^2) + norm1/3, -sqrt(25 -
unif2^2) + norm2/3, norm2/2),
                  dataset = rep("Eye", 3000))

# Рій: чотири кластери без чітких меж
swarm <- data.frame(X1 = c(norm1[1:250] - 5, norm3    ,  norm3[1:500]
```

```
+ 5                , norm1[251:500] + 5),
                   X2 = c(norm4[1:250] - 8, norm5 + 2, -
norm3[1:500]/4 + norm4[1:500] - 10, norm4[251:500] + 10),
                   dataset = rep("Swarm", 2000))


# Зв'язуємо усі набори даних в один набір даних
data <- bind_rows(galaxy, sickle, slash, eye, swarm)
data$dataset <- factor(data$dataset, levels = c("Galaxy", "Sickle",
"Slash", "Eye", "Swarm"))

# Створіть нові стовпці для кожної кластеризації
data <- mutate(data, kMeans = NA, HAC = NA, GMM = NA, DBSCAN = NA)
# Візуалізуємо кожен набір даних

ggplot(data, aes(x = X1, y = X2)) +
  geom_point(alpha = 0.3) +
  facet_wrap(. ~ dataset, scales = "free") +
  theme_bw()
```

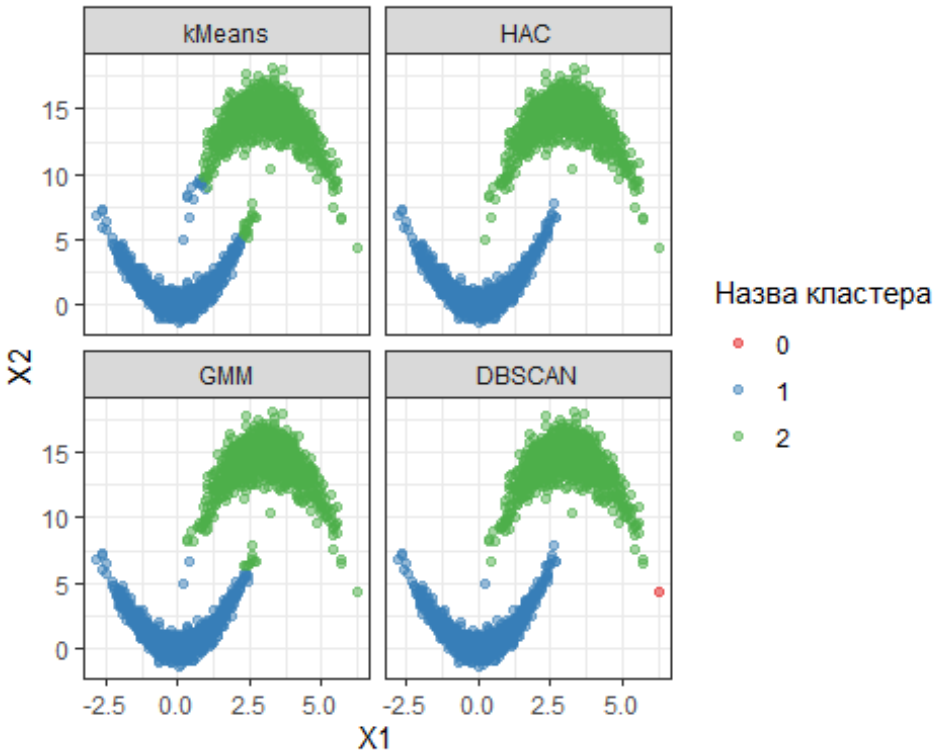



```
# Обчислюємо кластери
calc_cluster <- function(ds, kmeans.centers, hac.method, hac.k, gmm.G,
dbscan.eps) {
  # Нормалізація даних
  ds.scaled <- data.frame(scale(ds[, c("X1", "X2")]))

  # k-Means
  ds$kMeans <- kmeans(ds.scaled[, c("X1", "X2")], centers =
kmeans.centers, nstart = 5)$cluster
```

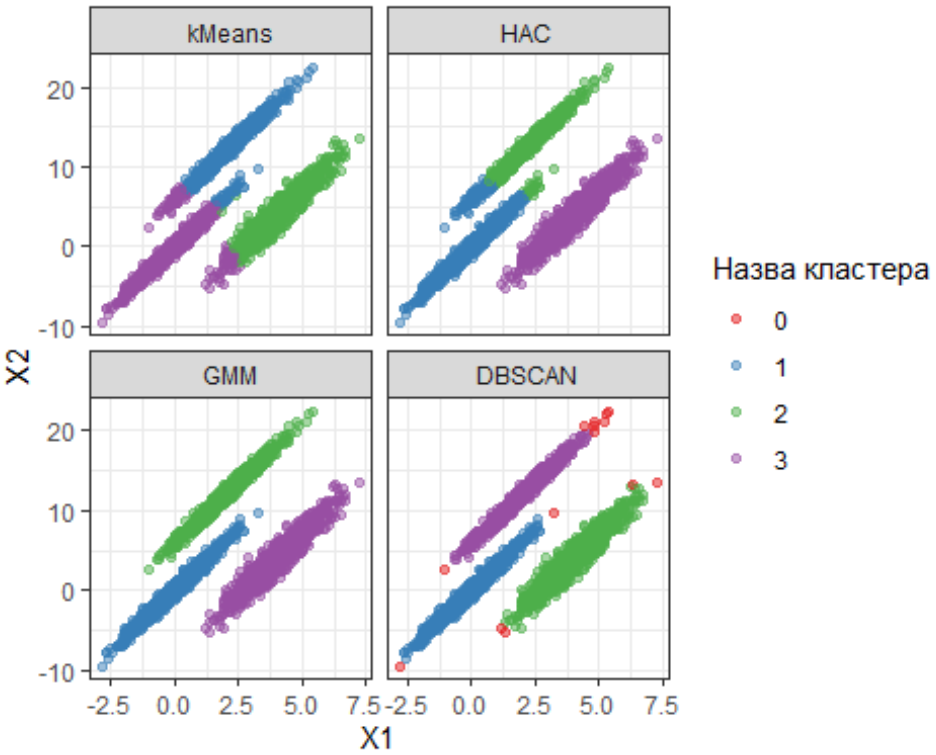
```
  # Hierarchical Agglomerative Clustering (HAC)
  distance <- dist(ds.scaled[, c("X1", "X2")], method = "euclidean")
  ds$HAC <- hclust(distance, method = hac.method) %>% cutree(k =
hac.k)

  # Gaussian Mixture Model (GMM)
  ds$GMM <- Mclust(ds.scaled[, c("X1", "X2")], G =
gmm.G)$classification

  # DBSCAN
  ds$DBSCAN <- dbscan(ds.scaled[, c("X1", "X2")], eps = dbscan.eps,
MinPts = 20)$cluster

  return(ds)
}


# Будуємо кластери
plot_cluster <- function(ds.name) {
  data %>%
    filter(dataset == ds.name) %>%
    gather(-X1, -X2, -dataset, key = "clustering.method", value =
"cluster") %>%
    ggplot(aes(x = X1, y = X2, color = factor(cluster))) +
    geom_point(alpha = .5) +
    facet_wrap(. ~ factor(clustering.method, levels = c("kMeans",
"HAC", "GMM","DBSCAN"))) +
    scale_color_brewer(palette = "Set1") +
    theme_bw() +
    guides(color=guide_legend(title="Назва кластера"))
}


data[data$dataset == "Galaxy", ] <- calc_cluster(data[data$dataset ==
"Galaxy", ],
                                                 kmeans.centers = 3,
                                                 hac.method =
"average", hac.k = 3,
                                                 gmm.G = 3, dbscan.eps
= .5)

## fitting ...
##

|=================================================================
======| 100%

plot_cluster("Galaxy")
```



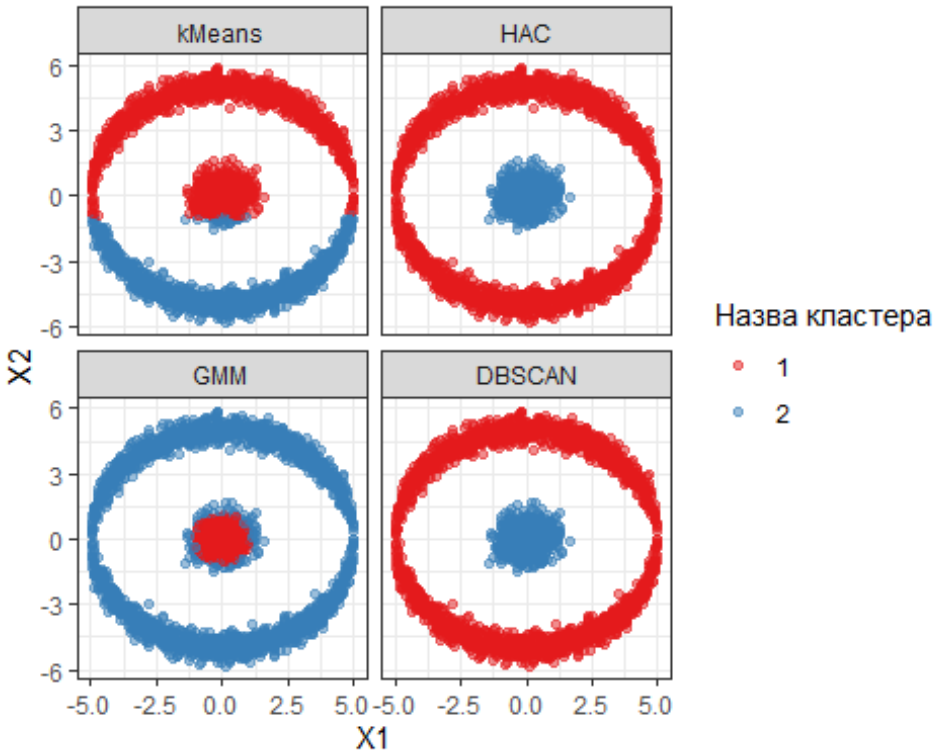

```
data[data$dataset == "Sickle", ] <- calc_cluster(data[data$dataset ==
"Sickle", ],
                                                 kmeans.centers = 2,
                                                 hac.method =
"ward.D", hac.k = 2,
                                                 gmm.G = 2, dbscan.eps
= .5)

## fitting ...
##

|======================================================================
======| 100%

plot_cluster("Sickle")
```



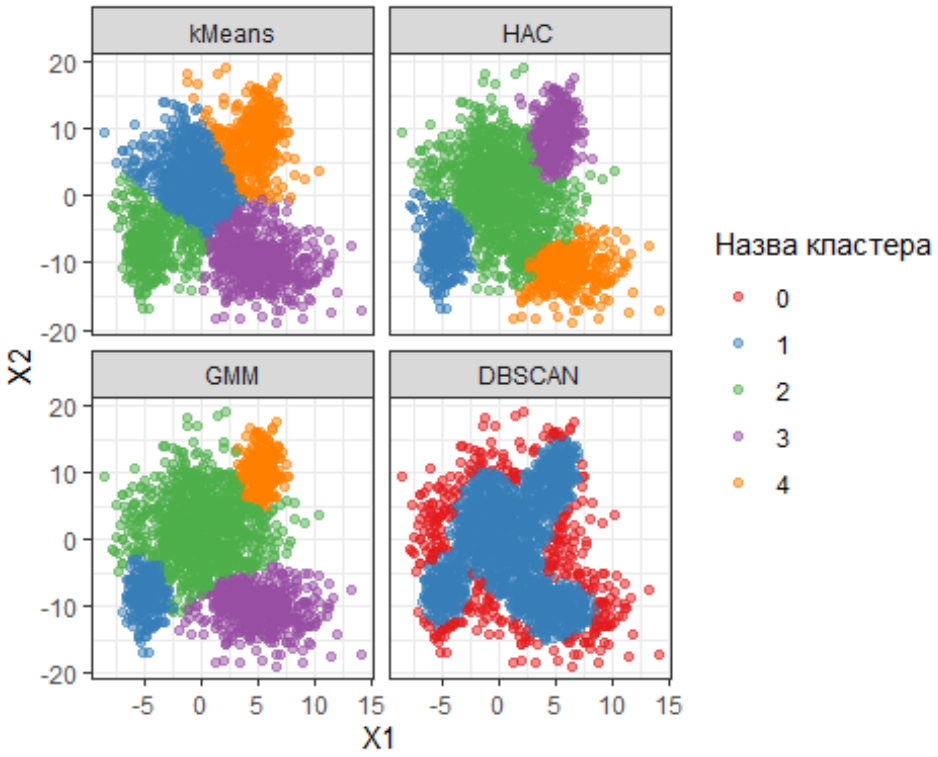

```
data[data$dataset == "Slash", ] <- calc_cluster(data[data$dataset ==
"Slash", ],
                                                kmeans.centers = 3,
                                                hac.method =
"average", hac.k = 3,
                                                gmm.G = 3, dbscan.eps
= .3)

## fitting ...
##

|======================================================================
======| 100%

plot_cluster("Slash")
```



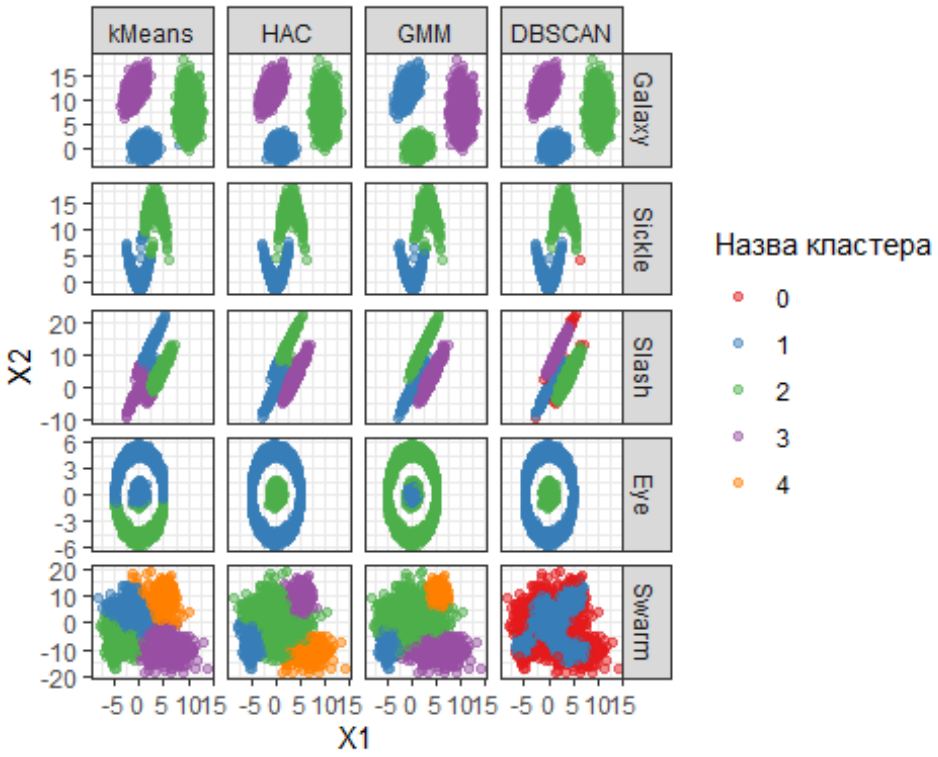

```
data[data$dataset == "Eye", ] <- calc_cluster(data[data$dataset ==
"Eye", ],
                                              kmeans.centers = 2,
                                              hac.method = "single",
hac.k = 2,
                                              gmm.G = 2, dbscan.eps =
.3)

## fitting ...
##

|======================================================================
======| 100%

plot_cluster("Eye")
```




```
data[data$dataset == "Swarm", ] <- calc_cluster(data[data$dataset ==
"Swarm", ],
                                                kmeans.centers = 4,
                                                hac.method = "ward.D",
hac.k = 4,
                                                gmm.G = 4, dbscan.eps
= .2)

## fitting ...
##

|======================================================================
======| 100%

plot_cluster("Swarm")
```




```
data %>%
  gather(-X1, -X2, -dataset, key = "clustering.method", value =
"cluster") %>%
  ggplot(aes(x = X1, y = X2, color = factor(cluster))) +
  geom_point(alpha = .5) +
  facet_grid(dataset ~ factor(clustering.method, levels = c("kMeans",
"HAC", "GMM", "DBSCAN")),
             scales = "free") +
  scale_color_brewer(palette = "Set1") +
  theme_bw() +
  guides(color=guide_legend(title="Назва кластера"))
```




Shiny App

```
# Load libraries
library(shiny)
library(shinydashboard)
library(factoextra)
library(NbClust)
library(fpc)
library (ggplot2)
library(ggthemes)
library(shinythemes)



# Load data
dt<- read.csv("CC GENERAL.csv", sep = ",")
dt <- na.omit(dt[1:200,])
names(dt)

dt$BALANCE_FREQUENCY <- NULL

# This is a Shiny web application. You can run the application by clicking
ui <- shinyUI(
  dashboardPage(
    dashboardHeader(title ="Кластеризація даних"),
    dashboardSidebar(
      sidebarMenu(
        menuItem("Вхідні дані",tabName = "raw",icon=icon("table")),
        menuItem("Описові статистики",tabName = "summary",icon=icon("lock-
open",lib="font-awesome")),
```

```
      menuItem("Кількість кластерів",tabName = "clusters",icon=icon("chart-
area",lib="font-awesome")),
      menuItem("K-means",tabName= "kmeans",icon=icon("users",lib="font-
awesome")),
      menuItem("HAC",tabName= "hac",icon=icon("users",lib="font-awesome")),
      menuItem("GMM",tabName= "pam",icon=icon("users",lib="font-awesome")),
      menuItem("DBSCAN",tabName= "clara",icon=icon("users",lib="font-awesome"))
    )),
  dashboardBody(
    tabItems(
      tabItem(tabName = "plots",h1("Exploratory Analysis - Plots"),fluidRow(
        box(plotOutput("barplot")),
        box(plotOutput("histogram1"))
      ),
      fluidRow(
        box(plotOutput("histogram2"))
      ),

      ),
      tabItem(tabName = "raw",h1("Вхідні
дані"),fluidRow(column(5,tableOutput("rawdata")))),
      tabItem(tabName = "summary",h1("Data Summary"),
              fluidRow(
                box(htmlOutput("summary1")),
                box(htmlOutput("summary2")),
                box(htmlOutput("summary3"))

              )),
      tabItem(tabName = "clusters",h1("Визначення оптимальної кількості
кластерів"),
              fluidRow(
                box(h2("Elbow Method"),plotOutput("method1")),
                box(h2("Average silhouette method"),plotOutput("method2"))),
              fluidRow(
                box(h2("Gap Statistic method"),plotOutput("method3")),
                box(h2("NbClust method"),plotOutput("method4"))
              )),
      tabItem(tabName = "kmeans",h1("K-means кластеризація"),
              fluidRow(
                box(plotOutput("clusterchart")),
                box(sliderInput("clustnum","Number of clusters",1,10,6))
              ),
              h2("K-means статистика кластерів"),
              fluidRow(
                box(h3("Cluster Statistics"),htmlOutput("stat1")),
                box(h3("Dunn Index"),htmlOutput("stat2"))
              )),
      tabItem(tabName = "hac",h1("HAC кластеризація"),
              fluidRow(
```

```
            box(plotOutput("clusterchart1")),
            box(sliderInput("clustnum1","Number of clusters",1,10,6))
          ),
          h2("HAC статистика кластерів"),
          fluidRow(
            box(h3("Cluster Statistics"),htmlOutput("stat3")),
            box(h3("Dunn Index"),htmlOutput("stat4"))
          )),
      tabItem(tabName = "pam",h1("GMM кластеризація"),
          fluidRow(
            box(plotOutput("clusterchart2")),
            box(sliderInput("clustnum2","Number of clusters",1,10,6))
          ),
          h2("GMM статистика кластерів"),
          fluidRow(
            box(h3("Cluster Statistics"),htmlOutput("stat5")),
            box(h3("Dunn Index"),htmlOutput("stat6"))
          )),
      tabItem(tabName = "clara",h1("DBSCAN clustering"),
          fluidRow(
            box(plotOutput("clusterchart3")),
            box(sliderInput("clustnum3","Number of clusters",1,10,6))
          ))

    ))))

# Define server logic required to draw a histogram
# Shiny Server
server <- shinyServer(function(input,output){

  output$barplot <- renderPlot({
    barplot(table(dt$Gender),main="Gender",
            ylab="Count",xlab="Gender",
            col=rainbow(7),
            legend=rownames(table(dt$Gender)))
  })
  output$histogram1 <- renderPlot({
    hist(dt$Rest_blood_pr,main="Histogram",
         xlab="Rest_blood_pr Score",
         ylab="Frequency",
         col="#6600cc",
         labels=TRUE)
  })
  output$histogram2 <- renderPlot({
    hist(dt$Chol,main="Histogram",
         xlab="Chol",
         ylab="Frequency",
         col="#1990cc",
         labels=TRUE)
```

```
  })
  output$rawdata <- renderTable(dt)
  output$summary1 <- renderPrint({names(dt)})
  output$summary2 <- renderPrint({summary(dt)})
  output$summary3 <- renderPrint({str(dt)})
  output$method1 <- renderPlot({
    fviz_nbclust(dt[2:3], kmeans, method = "wss") +
      geom_vline(xintercept = 4, linetype = 2)+
      labs(subtitle = "Elbow method")
  })
  output$method2 <- renderPlot({
    fviz_nbclust(dt[2:3], kmeans, method = "silhouette") +
      labs(subtitle = "Silhouette Method")
  })
  output$method3 <- renderPlot({
    set.seed(123)
    fviz_nbclust(dt[2:3], kmeans, nstart = 25,  method = "gap_stat", nboot = 10,
verbose = FALSE)+
      labs(subtitle = "Gap statistic method")
  })
  output$method4 <- renderPlot({
    NbClust(dt[2:3],distance = "euclidean",
            min.nc = 4, max.nc = 10, method = "kmeans", index = "all")
  })

  output$clusterchart <- renderPlot({
    fviz_cluster((eclust(dt[2:3], "kmeans", k = input$clustnum, nstart = 25, graph =
FALSE)), geom = "point", ellipse.type = "norm",
                 palette = "jco", ggtheme = theme_minimal())

  })
  output$clusterchart1 <- renderPlot({
    fviz_cluster((eclust(dt[2:3], "hclust", k = input$clustnum1, nstart = 25, graph =
FALSE)), geom = "point", ellipse.type = "norm",
                 palette = "jco", ggtheme = theme_minimal())

  })
  output$clusterchart2 <- renderPlot({
    fviz_cluster((eclust(dt[2:3], "pam", k = input$clustnum2, nstart = 25, graph =
FALSE)), geom = "point", ellipse.type = "norm",
                 palette = "jco", ggtheme = theme_minimal())

  })
  output$clusterchart3 <- renderPlot({
    fviz_cluster((eclust(dt[2:3], "clara", k = input$clustnum3, graph = FALSE)), geom =
"point", ellipse.type = "norm",
                 palette = "jco", ggtheme = theme_minimal())

  })
```

```
  output$stat1 <- renderPrint({
    cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "kmeans", k = input$clustnum, nstart =
25, graph = FALSE))$cluster)
  })
  output$stat2 <- renderPrint({
    (cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "kmeans", k = input$clustnum, nstart =
25, graph = FALSE))$cluster))$dunn
  })
  output$stat3 <- renderPrint({
    cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "hclust", k = input$clustnum1, nstart = 25,
graph = FALSE))$cluster)
  })
  output$stat4 <- renderPrint({
    (cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "hclust", k = input$clustnum1, nstart =
25, graph = FALSE))$cluster))$dunn
  })
  output$stat5 <- renderPrint({
    cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "pam", k = input$clustnum2, nstart = 25,
graph = FALSE))$cluster)
  })
  output$stat6 <- renderPrint({
    (cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "pam", k = input$clustnum2, nstart = 25,
graph = FALSE))$cluster))$dunn
  })
  output$stat7 <- renderPrint({
    cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "clara", k = input$clustnum3, nstart = 25,
graph = FALSE))$cluster)
  })
  output$stat8 <- renderPrint({
    (cluster.stats(dist(dt[2:3]),(eclust(dt[2:3], "clara", k = input$clustnum3, nstart = 25,
graph = FALSE))$cluster))$dunn
  })
  output$clustdata <- renderTable(cd)


})

# Run the application
shinyApp(ui = ui, server = server)
```