ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Криволап А.В.

АКТУАЛЬНІ ПРОБЛЕМИ DATA MINING

Навчальний посібник з виконання лабораторних робіт

Київ - 2023

УДК 004.8

Криволап А.В. Актуальні проблеми Data Mining. Навчальний посібник з виконання лабораторних робіт . – Київ, 2023. – 60 с.

Навчальний посібник призначений для студентів освітньо-наукових програм «Інформатика» та «Штучний інтелект» за спеціальністю 122 «Комп‘ютерні науки», які проходять курс «Актуальні проблеми Data Mining» або споріднений, в якому розглядаються базові задачі Data Mining такі, як класифікація, побудова асоціативних правил, кластеризація.

В посібнику надаються приклади застосування методів 1Rule, баєсівського класифікатора, побудови дерев прийняття рішень, k-найближчих сусідів, Apriori, FP-Growth, Single-link, Complete-link, k-Means, DBSCAN для розв’язання відповідних задач.

Рецензенти О.О.Марченко, доктор фіз.-мат.наук
С.С.Шкільняк, доктор фіз.-мат.наук

Затверджено Вченою радою факультету комп’ютерних наук та кібернетики КНУТШ Протокол № 9 від 21 березня 2023р.

ЗМІСТ

ВСТУП

Дисципліна інтелектуальний аналіз даних або Data Mining почала свій розвиток, як частина дисципліни виявлення знань в базах даних. Саме отримання знань та нової інформації з існуючих даних є основною ціллю тих задач, що розглядаються в межах дисципліни. Важливими ознаками Data Mining є більш прикладний характер та той факт, що для вирішення проблем, що розглядаються в її межах, застосовуються результати інших областей знань, зокрема статистики, штучного інтелекту, методів оптимізації, теорії інформації [1]. Тому можна сказати, що основна увага в межах дисципліни приділяється саме застосуванню відповідних результатів. Цей посібник також зосереджено на розгляді основних базових методів, запропонованих для розв'язку основних задач, які ставляться в рамках Data Mining.

В Data Mining Curriculum [2], що був запропонований як опис тем, що утворюють дисципліну, наводяться такі базові задачі, як класифікація, кластеризація, пошук асоціативних правил, візуалізація. Ці проблеми складають один з аспектів області досліджень. Іншим аспектом є визначення та дослідження поняття даних, яким чином вони повинні зберігатися та оброблятися для отримання найбільшої користі. Важливим є також їх поєднання, тобто аналіз, як тип даних та інші характеристики впливає на застосування того чи іншого методу для розв'язання проблеми.

Наявність оптимізованих програмних реалізацій різних моделей призводить до того, що одна з основних задач, що вирішується інженером даних це вибір найкращої моделі та визначення оптимальних параметрів для неї. Проте, це не є можливим без ґрунтовного розуміння суті відповідних методів та математичних результатів, що лежать в їхній основі. Застосування базових алгоритмів, хоч і на простих наборах даних, дозволяє краще зрозуміти особливості їх використання, основні переваги та недоліки.

Тема, що розглядається в першому розділі посібника, це попередня

обробка даних. Це зумовлено тим, що дані, якими оперує дослідник не завжди є ідеальними і готовими до використання для побудови тієї чи іншої моделі. Саме тому, більшість стандартів Data Mining, зокрема CRISP-DM [3,4], відмічають підготовку даних одним з перших етапів аналізу. Застосування необхідних технік дозволяє значно покращити результати наступних етапів. Правильно підготовлені дані є надійним фундаментом для наступних кроків.

Задачі класифікації та основним методам її розв'язку присвячено другий розділ. Дана проблема є більше прогностичною. Різні результати та ідеї лежать в основі алгоритмів, запропонованих для її вирішення. Деякі з них демонструються на прикладі і дозволяють отримати уявлення про те, що різні методи найкраще працюють з певним типом вхідних даних. Серед таких алгоритмів — наївний баєсівський класифікатор, використання дерев прийняття рішень, k-найближчих сусідів [3-8].

Задача побудови асоціативних правил, або пошуку шаблонів, що часто зустрічаються, вирізняється постановкою серед інших задач, таких як класифікація та кластеризація. Проте, вона також спрямована на виявлення закономірностей, хоч і працює з транзакційними даними. В третьому розділі даються основні визначення і розглядається приклад застосування декількох алгоритмів, як найпростішого Apriori, так і більш складного та ефективного FP-Growth [3-8].

В четвертому розділі посібника досліджується проблема кластеризації та методи її вирішення. Варто відмітити, що часто ця задача є частиною попередніх кроків для вирішення інших задач. В той час, як основною ціллю може бути побудова класифікатора, кластеризація дозволяє оптимізувати дані, що подаються на вхід алгоритму класифікацій. Наприклад, дозволяє зменшити розмірність простору вхідних атрибутів, виявити аномальні приклади в навчальній вибірці. Методи, направлені на вирішення даної проблеми також є різноманітними. Серед них варто відмітити різні варіації методу k-середніх [3-8]. В розділі також наводиться порівняння

розбиття на кластери, отримане різними варіантами алгоритму.

В цілому, хоч розглянуті задачі досить різні, вони мають також багато спільних рис. Явно чи неявно близькість об’єктів в просторі атрибутів є одним з основних критеріїв. Виявлення нетривіальних закономірностей є ключовою метою кожної проблеми. І, найголовніше, велика кількість варіантів використання для вирішення прикладних задач як прогностичного так і описового характеру. Це все дійсно дозволяє перетворити набір значень, що інакше б мали незначну користь в по-справжньому вагомий результат.

1. ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

Окремим кроком вирішення тієї чи іншої задачі Data Mining є підготовка даних для наступного їх використання в певному алгоритмі. Існує багато прикладів, в яких належним чином виконані етапи попередньої обробки даних суттєво вплинули на загальний результат. До таких етапів можна віднести наступні підзадачі [4,5]:

- Видалення аномальних записів та викидів
- Заповнення відсутніх значень
- Визначення несуттєвих атрибутів
- Перетворення даних у форму, очікувану відповідним алгоритмом.
- Зменшення об’єму набору даних

1.1 **Заповнення відсутніх значень**

Методи Data Mining передбачають, що для кожного об’єкта, значення всіх атрибутів є визначеними. Проте, для реальних наборів це не завжди так і ця проблема має бути вирішена, щоб відповідні методи могли бути застосовані.

Найпростішим рішенням є прибрати атрибути, що мають відсутні значення з розгляду. Але заповнені значення даних атрибутів можуть містити інформацію корисну для вирішення проблеми, що розглядається, наприклад, вони можуть підвищити точність класифікації.

У випадку, коли відсутні значення, все ж, визначаються, способом, що не потребує додаткових обчислень є використання окремих констант, які б фактично означали, що значення відсутнє, але могли б бути використані при побудові моделі. Складнішим, проте не завжди таким, що призводить до кращих результатів є використання довільних значень, серед тих, які також приймає відповідний атрибут. Можливим також є і використання середнього значення, або медіани. Окремим випадком є використання середнього всередині класу, якому належить запис. Також, відповідні значення можуть визначатися, використовуючи інші атрибути, таким чи-

ном вирішуючи окрему підзадачу прогнозування значення змінної. Таким, що потребує великих затрат часу є введення відсутніх значень експертами вручну.

Всі варіанти зазначені вище, крім заповнення експертами та відкидання атрибута, призводять до деякого спотворення даних, проте це виправдовується вищою оцінкою якості отриманої моделі.

1.2 **Видалення шумів та викидів**

Наступним недоліком наборів даних є наявність шумів та викидів. Шуми це, зазвичай, похибки отриманні при вимірюванні певних характеристик, пов’язані з недосконалістю приладів, або іншими зовнішніми факторами. Викиди, або аномальні значення часто бувають спричинені схожими факторами, проте полягають не в незначних відхиленнях від очікуваного значення, а в суттєвих, коли значення лежить поза межами можливих значень атрибутів.

Для боротьби з зашумленими даними відбувається, найчастіше, за рахунок різних технік згладжування. Прикладом такої техніки є перетворення відповідного атрибуту з неперервного до дискретного з використання обмеженого набору значень, або “кошиків”, на які замінюються всі близькі значення. Також дана проблема може бути вирішена з використанням регресії, коли значення апроксимуються певною функцією і значення атрибута для записів з набору замінюються значення відповідної гладкої функції.

Боротьба з викидами полягає в їх ідентифікації та відкидання з навчальної вибірки. В цьому зазвичай допомагає кластеризація та візуалізація даних, дослідження середнього значення та медіани. Зазвичай заміна використання середнього на використання медіани призводить до меншого випливу викидів на результат роботи алгоритму.

1.3 Трансформація даних

Кожен метод Data Mining потребує того, щоб атрибути набору даних були відповідного типу, в такому разі, метод показує найкращі результати. Наприклад, для баєсівського класифікатору, дерев прийняття рішень, дані мають бути категорійними, або дискретними числовими, з обмеженою множиною значень. В такому випадку, решта атрибутів, потребують дискретизації, або групування, у випадку неперервних числових атрибутів це відбувається шляхом поділу множини всіх значень на інтервали і заміною всіх значень на певну константу, що відповідає інтервалу, куди входить значення. Також для дискретизації може використовуватися кластеризація, коли значення атрибута розбиваються на кластери і замінюються значенням, що відповідає кластеру.

Деякі методи не можуть працювати з категорійними атрибутами і потребують їх кодування. Якщо значення категорійного атрибуту впорядковані, то їх можна пронумерувати послідовними числами, проте в загальному випадку це може призвести до росту похибки і спотворення даних. Наприклад, якщо закодувати категорійний атрибут, що має значення “зелений”, “жовтий”, “червоний” числами 0, 1, 2, то виявиться, що різниця між “червоним” та “зеленим” вдвічі більша ніж між “червоним” та “жовтим”. Проте ця інформація була відсутня в початковому наборі даних. Вирішити дану проблему дозволяє кодування за допомогою атрибутів прапорців, або hot-one кодування. За цього кодування, кожному значенню категорійного атрибута співставляється новий атрибут, значення якого рівне 1 для тих об’єктів, які мали відповідне значення атрибута і 0 інакше. Це дозволяє більш точно представити категорійний атрибут, проте загальна кількість атрибутів в такому разі значно зросте.

Проблеми, які вирішує трансформація даних не обмежуються перетворенням типів атрибутів. Важливою задачею є, також, нормалізація даних. Нормалізація полягає в гарантуванні того, що всі атрибути будуть мати схожі інтервали можливих значень, таким чином, гарантуючи, що різні

атрибути будуть мати однаковий вплив на результат. Наприклад, якщо набір даних має два атрибути, один з яких приймає значення від 0 до 10, а інший від 0 до 10000 і використовується евклідова відстань для визначення того, які об’єкти вважати близькими. В даному випадку, вплив другого атрибута буде значно переважати і може нівелювати вплив першого взагалі. Варто зазначити, що нормалізація найбільш суттєво впливає на результат методів, заснованих на понятті відстані (наприклад, k-найближчих сусідів порівняно з баєсівським класифікатором). Найбільш відомими підходами до нормалізації є мін-максна нормалізація, та нормалізація з використанням стандартного відхилення.

Ще одним напрямком, який можна віднести до трансформації даних, є техніки зменшення об’єму даних. Це пов’язано з тим, що швидкодія методів напряму залежить від кількості атрибутів та записів, що складають набір даних і важливим є оптимізація їх кількості без суттєвої втрати точності. До означених технік відносять методи зменшення розмірності простору атрибутів, зокрема з використанням кластеризації. Також різноманітні способи стиснення даних. Окремо слід виділити і методи семплінгу, або використання певної підмножини набору даних, обраної довільним чином. Такий підхід працює швидше за стиснення даних, хоча і в більшій мірі впливає на якість.

2. ЗАДАЧА КЛАСИФІКАЦІЇ

Однією з основних задач, що досліджуються в межах Data Mining є задача класифікації. В загальному класифікація полягає в віднесенні нових об'єктів до певного класу (зазвичай задається окремим категорійним атрибутом), базуючись на наявних розмічених даних, та враховуючи схожість об'єктів, що належать одному класу. Вхідними даними є набір даних, в якому кожен об'єкт має мітку, до якого класу він належить. Дана множина називається навчальною множиною. Тому задача класифікації, на відміну від, задачі кластеризації, відноситься до навчання з учителем.

Зазвичай процес класифікації відбувається в два етапи — побудова моделі (класифікатора), що описує дані на основі заданої навчальної множини: використання моделі для визначуння класу нових записів. Залежно від підходу, співвідношення обчислювальних ресурсів, яких потребують етапи може варіюватися. Процес побудови класифікатора може відбуватися ітеративно, після кожної ітерації модель оцінюється за різними показниками і в разі потреби вносяться зміни. Критерії за якими оцінюють класифікатор, включають в себе точність, інтерпретовність, здатність до узагальнення. Точність зазвичай визначається за допомого спеціальної тестової множини, яка виділяється з загальної навчальної множини. Інтерпретовність означає простоту розуміння отриманих параметрів моделі та як вони відносяться з задачею, що розв'язується . Здатність до узагальнення — це характеристика того, що модель не була перенавчена. У випадку перенавчання, класифікатор демонструє високу точність на навчальній вибірці, але точність на записах, що не зустрічалися при навчанні є низькою.

Варто зазначити, що для вирішення задачі класифікації було запропоновано багато різноманітних підходів, серед них: нейронні мережі, дерева прийняття рішень, оптимізаційні методи, теорія ймовірності та інші. Також цікаво, що пошук методів, що мають вищу точність та якість класифікації призвів до появи методів, що засновані на ансамблі класифікаторів. В цьому випадку, використовується набір класифікаторів, кожен з

яких є слабким класифікатором і має точність меншу за найкращих представників, тобто вважався “застарілим”.

В наступних розділах будуть розглянуті приклади деяких базових методів класифікації, таких як: 1-Rule, Баєсівський класифікатор (Naive Bayes), використання дерев прийняття рішень (Decision Tree), k-Найближчих-Сусідів (k-NN).

2.1 Алгоритм 1-Rule

Одним з найпростіших алгоритмів класифікації є алгоритм 1-Rule, або алгоритм побудови елементарних правил. Даний алгоритм також можна розглядати, як частковий випадок дерев прийняття рішень, в якому дерево обмежене кореневим вузлом та відповідними листками.

1-Rule метод має гіршу точність, ніж решта розглянутих алгоритмів, проте його основними перевагами є швидкість, простота та інтерпретовність. Він дозволяє швидко виділити найбільш загальні закономірності. Метод полягає в формуванні правил класифікації, що використовують лише один атрибут (вигляду “атрибут має певне значення, отже належить до певного класу”) і виборі з таких правил того, що має найменшу похибку. Найкраще це зробити використовуючи таблиці частоти для кожного атрибуту відносно класу.

Побудуємо правила для класифікації згідно методу 1-Rule для наступного набору даних: {(0,0,1,0,0), (1,0,0,1,1), (2,0,1,0,1), (0,1,0,0,1), (0,1,1,0,1), (0,1,1,1,0), (1,0,0,1,0), (2,0,0,0,1), (2,1,1,0,1), (0,1,1,1,0)}. Та використаємо їх для визначення класу об’єкту (2,1,1,1,?). В даному прикладі останній атрибут використовується для позначення класу, до якого відноситься об’єкт. Позначимо атрибути Q_1, Q_2, Q_3, Q_4,S, та представимо у табличному вигляді для наочності. Результатом є таблиця 2.1.1.

Для кожного атрибуту побудуємо таблиці частоти відносно значення класу. Результати представлені у вигляді таблиць 2.1.2-2.1.5. Тут кожен рядок відповідає певному значенню атрибута, а колонка — класу.

Таблиця 2.1.1

Q_1	Q_2	Q_3	Q_4	S
0	0	1	0	0
1	0	0	1	1
2	0	1	0	1
0	1	0	0	1
0	1	1	0	1
0	1	1	1	0
1	0	0	1	0
2	0	0	0	1
2	1	1	0	1
0	1	1	1	0
2	1	1	1	?

Таблиця 2.1.2

Q_1\S	0	1
0	3	2
1	1	1
2	0	3

Таблиця 2.1.3

Q_2\S	0	1
0	2	3
1	2	3

Таблиця 2.1.4

Q_3\S	0	1
0	1	3
1	3	3

Таблиця 2.1.5

Q_4\S	0	1
0	1	5
1	3	1

Для кожного атрибута можна сформувати правила з найменшою похибкою серед правила, що використовують лише цей атрибути. Згідно з таблицями частот — це пара значення атрибуту і клас для комірок, що мають найбільшу частоту в кожному рядку. Це наступні правила:

- Якщо $Q_1 = 0$, то $S = 0$. Якщо $Q_1 = 1$, то $S = 0$. Якщо $Q_1 = 2$, то $S = 1$.
- Якщо $Q_2 = 0$, то $S = 1$. Якщо $Q_2 = 1$, то $S = 1$.
- Якщо $Q_3 = 0$, то $S = 1$. Якщо $Q_3 = 1$, то $S = 0$.
- Якщо $Q_4 = 0$, то $S = 1$. Якщо $Q_4 = 1$, то $S = 0$.

Правило “Якщо $Q_1 = 1$, то $S = 0$” може бути замінено правилом “Якщо $Q_1 = 1$, то $S = 1$”, а правило “Якщо $Q_3 = 1$, то $S = 0$” може бути замінено правилом “Якщо $Q_3 = 1$, то $S = 1$” без впливу на похибку.

Похибкою груп правил для кожного атрибуту є частина невірно класифікованих прикладів згідно цього правила. Кількість невірно класифікованих прикладів можна отримати з відповідних таблиць — це сума значень в комірках, що не відповідають правилам. В результаті маємо для Q_1 похибку 0.3, для Q_2 — 0.4, для Q_3 — 0.4, для Q_4 — 0.2. Отже правила для останнього атрибуту мають найменшу похибку і будуть використані для класифікації. Це правила “Якщо $Q_4 = 0$, то $S = 1$” та “Якщо $Q_4 = 1$, то $S = 0$”. Згідно них, новий об’єкт буде віднесено до класу 0.

2.2 Наївний баєсівський класифікатор

Баєсівський класифікатор ґрунтується на формулі умовної ймовірності, запропонованою Баєсом і розгляді задачі класифікації як задачі теорії ймовірності. В межах даного підходу окремими подіями є події, що об’єкт має певне значення атрибута і об’єкт належить певному класу. Формула умовної ймовірності використовується для обчислення ймовірності настання події належності об’єкта певному класу за умови того що настали події наявності у об’єкта певних значень атрибутів.

Нехай розмірність простору атрибутів *n*, і кожен атрибут має скінчену множину можливих значень. Позначимо $x_i^j \in A_i$ – *j*-те значення *i*-го атрибуту, а f_i^j відповідну подію. В свою чергу c_k будемо позначати подію, що об’єкт належить класу *k*. Згідно формули Баєса для умовної ймовірності, ймовірність того, що даний об’єкт відноситься до класу *k* визначається наступним чином $P(c_k|X)=\frac{P(X|c_k)P(c_k)}{P(X)}$, де X — це подія, що об’єкт має певні значення атрибутів, задані вектором $X=\left(x_1^{j_1},x_2^{j_2}\ldots,x_n^{j_n}\right)$.

Для визначення, до якого класу віднести об’єкт X — потрібно знайти *k* для якого $P(c_k|X)$ буде максимальним. Тобто максимальною буде

ймовірність того, що об’єкт належить даному класу за умови настання події, що об’єкт має відповідні значення атрибутів. Зважаючи на те, що дільник $P(X)$ однаковий для всіх класів, його можна прибрати, отже залишається знайти $k=argmax P(X|c_k)P(c_k)$.

Основним припущення методу Баєса є те, що події для окремих атрибутів є незалежними і тому ми можемо розписати $P(X|c_k)=\prod P(f_i^{j_i}|c_k)$. Необхідні ймовірності легко обчислюються згідно тренувального набору даних. Де $P(f_i^j|c_k)$ — фактично відношення кількості прикладів, що відносяться до класу *k* і мають *j*-те значення *i*-го атрибута, до всіх прикладів, що відносяться до класу *k*.

Розв’яжемо приклад з розділу 2.1, використовуючи метод Баєса. Так як всього 4 приклади для класу 0 і 6 прикладів для класу 1, і додатково не задано апріорні ймовірності належності певному класу, то ймовірності обчислюються з набору даних і $P(S=0)=0.4$ та $P(S=1)=0.6$. В свою чергу з таблиць 2.1.2-2.1.5 можемо обчислити умовні ймовірності того, що атрибути набувають певних значень для об’єктів певного класу. Загальна кількість прикладів кожного класу — 4 і 6 відповідно, залишається поділити значення комірок на ці числа. Таким чином отримуємо наступні значення.

$P(Q_1=0|S=0)=3/4$ $P(Q_1=1|S=0)=1/4$ $P(Q_1=2|S=0)=0$

$P(Q_1=0|S=1)=2/6$ $P(Q_1=1|S=1)=1/6$ $P(Q_1=2|S=1)=3/6$

$P(Q_2=0|S=0)=1/2$ $P(Q_2=1|S=0)=1/2$

$P(Q_2=0|S=1)=1/2$ $P(Q_2=1|S=1)=1/2$

$P(Q_3=0|S=0)=1/4$ $P(Q_3=1|S=0)=3/4$

$P(Q_3=0|S=1)=1/2$ $P(Q_3=1|S=1)=1/2$

$P(Q_4=0|S=0)=1/4$ $P(Q_4=1|S=0)=3/4$

$P(Q_4=0|S=1)=5/6$ $P(Q_4=1|S=1)=1/6$

Так як ми отримали $P(Q_1=2|S=0)=0$, то для того, щоб уникнути множників нульових, застосуємо розмиття за Лапласом. Для цього перерахуємо ймовірності для даного класу і атрибута, додавши до кожного значення

певний коефіцієнт розмиття, він має бути меншим ніж існуючі значення, використаємо для нашого прикладу 0.1, так отримаємо оновлені ймовірності.

$$P(Q_1=0|S=0)=31/43 \quad P(Q_1=1|S=0)=11/43 \quad P(Q_1=2|S=0)=1/43$$

Після того, як всі потрібні ймовірності отримано, визначимо клас для об'єкту (2,1,1,1). Для цього визначимо, для якого класу відповідний добуток ймовірностей буде більшим.

$$P(S=0)P(Q_1=2|S=0)P(Q_2=1|S=0)P(Q_3=1|S=0)P(Q_4=1|S=0)=\frac{4*1*1*3*3}{10*43*2*4*4}$$

$$P(S=1)P(Q_1=2|S=1)P(Q_2=1|S=1)P(Q_3=1|S=1)P(Q_4=1|S=1)=\frac{6*3*1*1*1}{10*6*2*2*6}$$

Отже для S=0, маємо 9/3440, а для S=1, маємо 43/3440, а значить відносимо новий об'єкт до класу 1.

2.3 Використання дерев прийняття рішень

Дерева прийняття рішень можуть бути застосовані не лише для зручного опису інформації, а також у якості ефективних моделей класифікації. Вони використовуються як окремо так і в межах таких методів, як Random Forest.

Деревом прийняття рішень в рамках задачі класифікації називається дерево, листки якого розмічені мітками класів, до яких може належати запис, вузли розмічені атрибутами набору даних, а дуги розмічені значеннями атрибуту, яким позначено вузол, з якого виходить дуга. Таким чином для класифікації запису необхідно починаючи з кореневого вузла пройти деревом відповідно до значень атрибутів даного запису. Досягнутий таким чином листок, буде містити клас, до якого й буде віднесено запис.

Одним з алгоритмів побудови дерев прийняття рішень є алгоритм ID3, що визначається рекурсивно. Визначимо його відносно множини атрибутів *Q*, множини записів *A* та атрибуту *S*, що позначає клас. Алгоритм також може бути застосований використовуючи різні критерій вибору атрибуту, що найкраще характеризує дані, це також один з параметрів алгоритму, йо-

го будемо позначати *FeatureCriteria*(*A*,*q*). Алгоритм ID3 має наступні етапи:

1. Якщо всі записи *A* мають однакове значення атрибуту *S*, то деревом для даної множини буде листок з цим значенням.
2. Інакше, якщо $Q=\varnothing$, то визначити значення атрибута *S*, що зустрічається найчастіше серед записів з *A* і результатом буде дерево, що складається з листка, що містить відповідне значення.
3. Інакше визначити атрибут $q\in Q$ такий, що *FeatureCriteria*(*A*,*q*) максимальне. Позначимо $A_q^i=\{a|a\in A\wedge q(a)=q_i\}$, де $\{q_1,q_2,\dots,q_n\}$ – всі можливі значення атрибута *q*. Результатом буде дерево з коренем, що має мітку *q*, для кожного q_i будується дуга з міткою q_i, що виходить з кореня до дерева, що будується наступним чином:
   - Якщо $A_q^i=\varnothing$, то деревом буде листок з міткою, рівною значенню атрибуту *S*, що зустрічається найчастіше серед записів з *A*.
   - Інакше дерево буде результатом роботи ID3($Q\setminus\{q_i\}, A_q^i, S$).

Серед різноманітних підходів до визначення критерію вибору атрибута, найчастіше застосовуються інформаційний критерії та індекс Джині

Згідно з інформаційним критерієм максимізується приріст інформації, що визначається за формулою $Gain(A,q)=H(A,S)-\sum_{i=1}^{n}\frac{|A_q^i|}{|A|}H(A_q^i,S)$, де $\{q_1,q_2,\dots,q_n\}$ – всі можливі значення атрибута *q, a* A_q^i — всі записи з *A*, що мають значення q_i атрибута *q*. Під $H(A,S)$ позначається ентропія множини *A* відносно атрибута *S*. І визначається як $H(A,S)=-\sum_{i=1}^{s}\frac{m_i}{m}\log_2(\frac{m_i}{m})$, де *s* – кількість різних значень атрибута *S,* а $m_i=|\{a|a\in A\wedge S(a)=s_i\}|$ — кількість записів з *A*, що мають значення s_i атрибута *S.* А *m* – загальна кількість записів в *A*.

Критерій, що базується на індексі Джині в свою чергу визначається наступним чином: $GiniCriteria(A,q)=Gini(A,S)-Gini_q(A,S)$, де

$Gini(A,S)=1-\sum_{i=1}^{s}(\frac{m_i}{n})^2$. Змінні s, m_i та m використані аналогічно попередньому критерію. Тоді індекс згідно з новим розподілом, можна визначити, як $Gini_q(A,S)=\sum_{i=1}^{n}\frac{|A_q^i|}{|A|}Gini(A_q^i,S)$.

Використаємо алгоритм ID3 для задачі з розділу 2.1. Як критерій обирається інформаційний критерій, випадок застосування критерію, що базується на індексі Джині є аналогічним. Набір даних відповідно {(0,0,1,0,0), (1,0,0,1,1), (2,0,1,0,1), (0,1,0,0,1), (0,1,1,0,1), (0,1,1,1,0), (1,0,0,1,0), (2,0,0,0,1), (2,1,1,0,1), (0,1,1,1,0)}.

Можемо бачити, що пункти 1 та 2 алгоритму не виконуються, отже необхідно визначити атрибут, за яким буде розбити множину. Загальна ентропія рівна $H(A,S)=-\frac{4}{10}\log_2(\frac{4}{10})-\frac{6}{10}\log_2(\frac{6}{10})\approx 0.97$. Позначимо A_i^j підмножини A, для яких значення атрибута Q_i рівне j. Тоді маємо наступні значення критеріїв:

$$Gain(A,Q_1)=H(A,S)-\frac{|A_1^0|}{|A|}H(A_1^0,S)-\frac{|A_1^1|}{|A|}H(A_1^1,S)-\frac{|A_1^2|}{|A|}H(A_1^2,S)$$

$$Gain(A,Q_2)=H(A,S)-\frac{|A_2^0|}{|A|}H(A_2^0,S)-\frac{|A_2^1|}{|A|}H(A_2^1,S)$$

$$Gain(A,Q_3)=H(A,S)-\frac{|A_3^0|}{|A|}H(A_1^0,S)-\frac{|A_3^1|}{|A|}H(A_3^1,S)$$

$$Gain(A,Q_4)=H(A,S)-\frac{|A_4^0|}{|A|}H(A_4^0,S)-\frac{|A_4^1|}{|A|}H(A_4^1,S)$$

Обчислимо ентропію підмножини A_1^0 , до неї відноситься 5 записів, для яких Q_1=0, з них 3 записи мають S=0, а 2 записи мають S=1, отже:

$$H(A_1^0,S)=-\frac{3}{5}\log_2(\frac{3}{5})-\frac{2}{5}\log_2(\frac{2}{5})\approx 0.97 \quad .$$

Аналогічно:

$$H(A_1^1,S)=-\frac{1}{2}\log_2(\frac{1}{2})-\frac{1}{2}\log_2(\frac{1}{2})=1 \qquad H(A_1^2,S)=-0*\log_2(0)-1*\log_2(1)=0$$

$$H(A_2^0,S)=-\frac{3}{5}\log_2(\frac{3}{5})-\frac{2}{5}\log_2(\frac{2}{5})\approx 0.97 \qquad H(A_2^1,S)=-\frac{3}{5}\log_2(\frac{3}{5})-\frac{2}{5}\log_2(\frac{2}{5})\approx 0.97$$

$$H(A_3^0,S)=-\frac{3}{4}\log_2(\frac{3}{4})-\frac{1}{4}\log_2(\frac{1}{4})\approx 0{,}81 \qquad H(A_3^1,S)=-\frac{1}{2}\log_2(\frac{1}{2})-\frac{1}{2}\log_2(\frac{1}{2})=1$$

$$H(A_4^0,S)=-\frac{1}{6}\log_2(\frac{1}{6})-\frac{5}{6}\log_2(\frac{5}{6})\approx 0.65 \qquad H(A_4^1,S)=-\frac{3}{4}\log_2(\frac{3}{4})-\frac{1}{4}\log_2(\frac{1}{4})\approx 0.81$$

Підставимо в формули приросту, маємо

$$Gain(A,Q_1)\approx 0.97-0.5*0.97-0.2*1-0.3*0\approx 0.285$$

$$Gain(A,Q_2)\approx 0.97-0.5*0.97-0.5*0.97\approx 0$$

$$Gain(A,Q_3)\approx 0.97-0.4*0.81-0.6*1\approx 0.046$$

$$Gain(A,Q_4)\approx 0.97-0.6*0.65-0.4*0.81-0.3*0\approx 0.256$$

Найбільше значення критерію маємо для Q_1. Обираємо даний атрибут для поділу, результати для Q_1=0, Q_1=1, Q_1=2 представлено в таблицях 2.3.1 - 2.3.3 відповідно.

Таблиця 2.3.1

Q_2	Q_3	Q_4	S
0	1	0	0
1	0	0	1
1	1	0	1
1	1	1	0
1	1	1	0

Таблиця 2.3.2

Q_2	Q_3	Q_4	S
0	0	1	1
0	0	1	0

Таблиця 2.3.3

Q_2	Q_3	Q_4	S
0	1	0	1
0	0	0	1
1	1	0	1

Можна бачити, що у випадку Q_1=2 — всі записи мають однакове значення для *S*. Отже це піддерево буде листком з відповідною міткою. У випадку Q_1=1 — всі записи мають однакові значення атрибутів, крім атрибута, що задає мітку класу. Тому подальше розбиття не є доцільним і тому замінимо дане піддерево листком з міткою, яку мають більшість записів — 1.

Продовжимо розгляд алгоритму для даних з таблиці 2.3.1, з попередніх розрахунків $H(A_1^0,S)=-\frac{3}{5}\log_2(\frac{3}{5})-\frac{2}{5}\log_2(\frac{2}{5})\approx 0.97$.

$$Gain(A_1^0,Q_2)=H(A_1^0,S)-\frac{|B_2^0|}{|A_1^0|}H(B_2^0,S)-\frac{|B_2^1|}{|A_1^0|}H(B_2^1,S)$$

$$Gain(A_1^0,Q_3)=H(A_1^0,S)-\frac{|B_3^0|}{|A_1^0|}H(B_3^0,S)-\frac{|B_3^1|}{|A_1^0|}H(B_3^1,S)$$

$$Gain(A_1^0,Q_4)=H(A_1^0,S)-\frac{|B_4^0|}{|A_1^0|}H(B_4^0,S)-\frac{|B_4^1|}{|A_1^0|}H(B_4^1,S)$$

Тут B_i^j позначимо підмножини A_1^0 , для яких значення атрибута Q_i рівне *j*. Обчислимо значення потрібних ентропій.

$$H(B_2^0,S)=-0*\log_2(0)-1*\log_2(1)=0 \qquad H(B_2^1,S)=-\frac{1}{2}\log_2(\frac{1}{2})-\frac{1}{2}\log_2(\frac{1}{2})=1$$

$$H(B_3^0,S)=-0*\log_2(0)-1*\log_2(1)=0 \qquad H(B_3^1,S)=-\frac{1}{4}\log_2(\frac{1}{4})-\frac{3}{4}\log_2(\frac{3}{4})\approx 0.81$$

$$H(B_4^0,S)=-\frac{2}{3}\log_2(\frac{2}{3})-\frac{1}{3}\log_2(\frac{1}{3})\approx 0.92 \qquad H(B_4^1,S)=-0*\log_2(\frac{3}{4})-1*\log_2(1)=0$$

Підставимо в формули, та отримуємо значення критеріїв.

$$Gain(A_1^0,Q_2)\approx 0.97-0.2*0-0.8*1\approx 0.17$$

$$Gain(A_1^0,Q_3)\approx 0.97-0.2*0-0.8*0.81\approx 0.322$$

$$Gain(A_1^0,Q_4)\approx 0.97-0.6*0.92-0.4*0\approx 0.418$$

Отже на даному етапі обираємо атрибут Q_4 . В такому випадку при Q_4=1. Маємо листок з міткою 0. А для Q_4=1 отримаємо набір даних, в якому значення атрибута Q_2 співпадають з міткою класу, тому очевидно, що в цій підзадачі наступним буде обрано даний атрибут, бо ентропія роз-

биття за ним буде рівна 0, а отже приріст інформації буде найбільшим. Отримане дерево прийняття рішень зображено на рисунку 2.3.1.

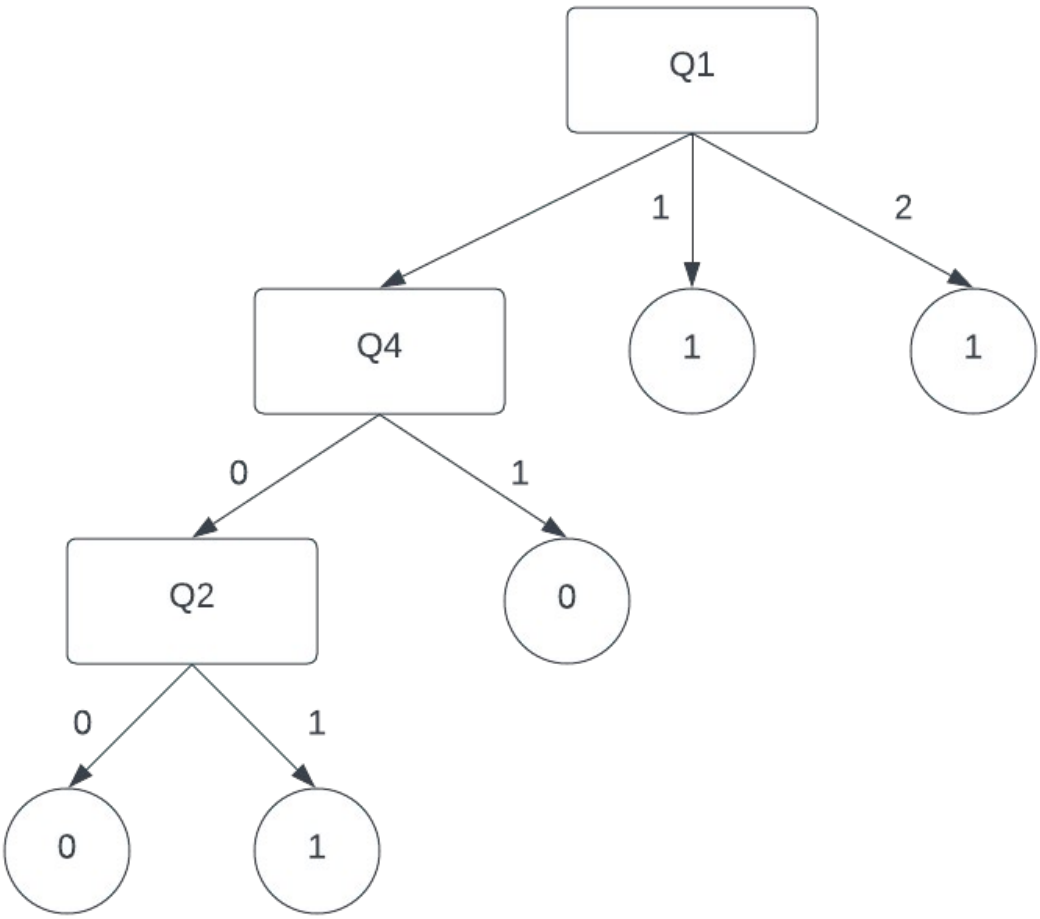

Рисунок 2.3.1

Для класифікації запису (2,1,1,1,?) потрібно рухатися деревом згідно значень атрибутів, в такому разі отримуємо результат на першому кроці і відносимо даний об’єкт до класу 1.

2.4 Алгоритм k-найближчих сусідів

Ідея використання існуючих прикладів, найбільш подібних до об’єкту, що розглядається для визначення його класу є природньою. Одним з алгоритмів, що ґрунтується на ній є алгоритм k-найближчих сусідів. Згідно алгоритму, для того, щоб визначити клас об’єкта, потрібно знайти k при-

кладів з навчальної вибірки, що є найближчими до нього, та використати інформацію про те, до яких класів вони належать. Визначення класу може відбуватися як за допомогою простого голосування, так і за допомогою зваженого. За простого голосування об'єкту призначається клас, до якого належить більшість зі знайдених сусідів. За зваженого голосування кількість голосів за кожен клас враховується як сума голосів кожного сусіда, що належить класу, а кількість голосів окремого сусіда визначається як обернений квадрат відстані до поточного об'єкту.

Кількість найближчих прикладів, що використовуються, k – є параметром алгоритму та визначається таким чином, щоб гарантувати як узагальнюючу здібність алгоритму, так і стійкість до викидів. Рекомендації стосовно даного параметру є об'єктом дослідження. Наприклад, для двовимірного простору атрибутів, показано, що вибір k більше ніж 4 є недоцільним і не призведе до суттєвого збільшення точності.

Особливістю даного алгоритму є те, що саму модель не можна відділити від даних. Результатом цього є те, що при розв'язку задачі відсутня фаза навчання моделі, проте для визначення класу об'єктів потрібно використати всі наявні навчальні приклади.

Розглянемо задачу з розділу 2.1 при значенні параметру k рівному 3 та евклідовій відстані, як відстані між записами. Отже маємо тренувальну множину {(0,0,1,0,0), (1,0,0,1,1), (2,0,1,0,1), (0,1,0,0,1), (0,1,1,0,1), (0,1,1,1,0), (1,0,0,1,0), (2,0,0,0,1), (2,1,1,0,1), (0,1,1,1,0)} та запис (2,1,1,1,?), для якого потрібно визначити клас. Обчислимо відстань до кожного з записів, та результат представимо у вигляді таблиці 2.4.1. В стовпці d, буде позначено відстань, а в стовпці votes – кількість голосів відповідного об'єкта при зваженому голосуванні. Ми можемо точно визначити два найближчих сусіди, це точки (2,1,1,0,1) та (2,0,1,0,1), точки (1,0,0,1,1), (1,0,0,1,0) та (2,0,0,0,1). Яку б з даних точок ми не обрали, на результат класифікації це не впливатиме. Дві найближчі точки належать до класу 1, тому навіть обравши приклад (1,0,0,1,0) за будь-якого типу голосування отримаємо

перевагу класу 1. У випадку з незваженим голосуванням це 2 проти 1, а якщо голосування зважене, то 1.5 проти 0.33.

Таблиця 2.4.1

Q_1	Q_2	Q_3	Q_4	S	d	votes
0	0	1	0	0	2.45	1/6
1	0	0	1	1	1.73	1/3
2	0	1	0	1	1.44	1/2
0	1	0	0	1	2.45	1/6
0	1	1	0	1	2.24	1/3
0	1	1	1	0	2	1/4
1	0	0	1	0	1.73	1/3
2	0	0	0	1	1.73	1/3
2	1	1	0	1	1	1
0	1	1	1	0	2	1/4

2.5 Вправи до розділу

1. Для представлених нижче наборів даних використати алгоритм 1Rule для побудови класифікатора та використати його для визначення класу об’єкта.

2. Для представлених нижче наборів даних побудувати наївний баєсівський класифікатор та використати його для визначення класу об’єкта.

3. Для представлених нижче наборів даних побудувати дерево прийняття рішень за алгоритмом ID3 (використати інформаційний критерій або заснований на індексі Джині) та використати його для визначення класу об’єкта.

4. Для прикладу з розділу 2.3 використати критерій, що базується на індексі Джині та порівняти з результатом представленим на рис. 2.3.1.

5. Використати алгоритм k-найближчих сусідів для k=3 та визначити клас об’єкта для представлених наборів даних. Порівняти результати отримані з використанням різних підходів до голосування.

Набори даних:

1. {(0,1,2,1), (1,0,1,1), (0,1,1,0), (0,0,1,1), (0,0,2,1), (1,1,2,0), (1,0,2,1), (1,0,0,0), (0,0,0,0), (0,0,1,1)}.Визначити клас для (1,1,1).
2. {(0,1,2,1), (1,0,1,0), (0,1,1,1), (0,0,1,1), (0,0,2,1), (1,1,2,1), (1,0,2,0), (1,0,0,1), (0,0,0,0), (0,0,1,0)}.Визначити клас для (1,1,1).
3. {(0,1,2,0), (1,0,1,0), (0,1,1,0), (0,0,1,0), (0,0,2,1), (1,1,2,1), (1,0,2,1), (1,0,0,1), (0,0,0,0), (0,0,1,1)}.Визначити клас для (1,1,1).
4. {(0,1,2,0), (1,0,1,1), (0,1,1,0), (0,0,1,1), (0,0,2,0), (1,1,2,1), (1,0,2,1), (1,0,0,1), (0,0,0,0), (0,0,1,1)}.Визначити клас для (1,1,1).

3. ЗАДАЧА ПОШУКУ АСОЦІАТИВНИХ ПРАВИЛ

Задача пошуку асоціативних правил пов'язана зі знаходженням взаємозв'язків між повторюваними подіями. В початковому вигляді вона була сформульована як проблема аналізу користувацького кошика (market basket analysis). Базуючись на транзакціях (покупках, зроблених клієнтом за один візит) потрібно сформулювати правила вигляду "з того, що було придбано певний набір продуктів слідує, що буде також придбано інший набір". Приклади, що підтверджують ці правила повинні мати місце достатньо часто. В той час, як кількість випадків, коли правило не підтвердилося, тобто був придбаний набір з засновку правила, але не було придбано набір з висновку, повинна бути мінімальною.

Дамо формальне визначення асоціативних правил та сформулюємо задачу їх пошуку. Якщо ми маємо $I = \{i_1, i_2, i_3, ...i_n\}$ – множину всіх товарів, або елементів. Тоді транзакцією T будемо називати підмножину множини I всіх товарів, $T \subset I$. Питання кількості елементів не розглядається, лише їх належність транзакції. Якщо $X \subset T$, то будемо говорити, що набір елементів X міститься в транзакції T. Сукупність всіх транзакцій, що відбулися, згідно з якими, потрібно побудувати асоціативні правила будемо позначати D. D може містити транзакції, що складаються з однакових елементів, проте представляти різні реальні транзакції.

Асоціативним правилом називається правило вигляду $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \cap Y = \varnothing$. Важливими характеристиками кожного асоціативного правила відносно сукупності всіх транзакцій D є підтримка та достовірність. Підтримка (support) правила $X \Rightarrow Y$ визначається як частина транзакцій з D, що містять $X \cup Y$, або $supp(X \Rightarrow Y) = supp(X \cup Y)$. Достовірність (confidence) правила визначається, як частина транзакцій, що містять Y, серед тих транзакцій з D, що містять X. Знаючи підтримку всіх множин, достовірність може бути визначена, як $conf(X \Rightarrow Y) = supp(X \cup Y) / supp(X)$.

Задача пошуку асоціативних правил в такому разі формулюється наступним чином: для заданого рівня мінімальної підтримки (minsupport) та мінімальної достовірності (minconfidence) та заданої сукупності транзакцій D знайти всі асоціативні правила, що мали підтримку та достовірність не нижчі за задані рівні. Таким чином, наявність мінімальної підтримки гарантує, що правило є шаблоном, що повторюється достатньо часто. А відповідність рівню мінімальної достовірності вказує на те, що правило має низьку похибку.

В загальному випадку задача пошуку асоціативних правил розбивається на дві підзадачі:

- Задача пошуку наборів елементів, що мають достатній рівень підтримки. Ці набори називаються наборами, що часто зустрічаються (frequent patterns)
- Задача генерування правил з отриманих наборів, які задовольняли б вимогу мінімальної достовірності.

З означених підзадач, найбільш критичною з точки зору швидкодії є перша задача. Якщо розглядати найпростіший підхід і перебирати всі можливі набори, рахуючи їх підтримку — складність зростає експоненціально в залежності від загальної кількості елементів. Маючи вже знайдені набори, що часто зустрічаються, можна легко для них побудувати всі можливі правила та обчислити їх достовірність. Таким чином, алгоритми, що застосовуються для розв’язку проблеми, зосереджені на тому, щоб зменшити кількість наборів, що потенційно можуть мати достатній рівень підтримки і потребують перевірки.

Були запропоновані різні підходи до вирішення задачі побудови наборів, що часто зустрічаються, серед яких алгоритм Apriori, що базується на властивості антимонотонності, алгоритм FP Growth в якому додатково використовуються спеціальні структури даних.

Якщо набір елементів F має достатню підтримку, правила $X \Rightarrow Y$ такі, що $X \cup Y = F$ можуть бути отримані шляхом наступним чином. Для кожної

власної підмножин $S \subset F$ набору F ми можемо розглянути правило $S \Rightarrow F / S$. Таким чином для вирішення підзадачі генерації правил, що мають необхідний рівень достовірності, маючи знайдені всі набори елементів, що часто зустрічаються, необхідно лише згенерувати всі можливі правила згідно з описаним вище прикладом та відсіяти ті, достовірність яких менша за minconfidence.

3.1 Алгоритм Apriori

Одним з найперших запропонованих алгоритмів для знаходження наборів елементів, що мають достатню підтримку, був алгоритм Apriori. Даний алгоритм заснований на властивості антимонотонності, що в тій чи іншій мірі використовується у всіх алгоритмах, направлених на вирішення даної задачі. Властивість антимонотонності полягає в тому, що підтримка набору не може бути більшою за підтримку будь-якого набору, що є його підмножиною, тобто $\forall X,Y \subset I \quad X \subset Y \Rightarrow supp(X) \geq supp(Y)$. Дана властивість ґрунтується на тому, що, якщо набір міститься в транзакції, то довільна підмножина також міститься в даній транзакції. Використання властивості антимонотонності дозволяє одразу прибрати з розгляду набір елементів, що є надмножиною набору елементів, для якого раніше було показано, що він не має достатньої підтримки.

Алгоритм Apriori полягає в послідовній побудові наборів, що мають достатню підтримку, починаючи з одноелементних наборів, з яких формуються двоелементні набори і т.д. До того моменту, поки нові набори не можуть бути побудовані. Окрема ітерація, що відповідає за побудову *n*-елементних наборів, використовуючи *n*-1-елементні, має наступні кроки:

- Генерація нових потенційних кандидатів використовуючи набори, що мають достатню підтримку отримані на попередній ітерації.
- Фільтрування згенерованих наборів згідно з властивістю антимонотонності.

- Обчислення підтримки отриманих наборів та відкидання тих, що не мають достатньої підтримки.

Для одноелементних наборів перші два кроки алгоритму полягають в тому, що утворюються всі можливі одноелементні набори.

Генерація n-елементних наборів відбувається шляхом злиття n-1-елементних наборів з достатньою підтримкою, що відрізняються лише одним елементом. Позначимо F_n множину наборів з n елементів, що мають потрібну підтримку. Таким чином для генерації кандидатів використовуються пари наборів, такі, що $A,B \in F_{n-1}, (A \cap B) = n-2$ і додається множина $C = A \cup B$.

Після того, як всі можливі кандидати з n елементів було сформовано, вони фільтруються і відкидаються набори елементів, що включають себе хоча б один набір з n-1 елементу, що не входить в F_{n-1}. Згідно з властивістю антимонотонності такі набори не можуть мати необхідної підтримки і тому немає потреби для них обчислювати підтримку додатково.

Для всіх відфільтрованих наборів, обчислюється підтримка. Це відбувається шляхом обходу всієї множини транзакцій, що потребує найбільших затрат обчислювальних ресурсів. Саме тому більшість модифікацій алгоритму спрямовані на оптимізацію даного кроку. Це відбувається за рахунок використання додаткових структур даних, чи зміні підходу до визначення підтримки набору як в алгоритмі Eclat, що буде розглянуто наступним.

Послідовність кроків повторюється до того моменту, доки може бути утворено набори більшого розміру. Після чого для всіх отриманих наборів з достатньою підтримкою генеруються правила та перевіряється їх достовірність.

Розглянемо множину транзакції представлену в таблиці 3.1.1 та для неї знайдемо всі набори елементів, що мають підтримку більше 40% використовуючи алгоритм Apriori. Для даної задачі множина всіх елементів $I = \{a,b,c,d,e,f,g,h\}$.

Таблиця 3.1.1

Id транзакції	Елементи
1	a, b, c, d, e
2	a, c, d, f
3	a, b, c, d, e, g
4	c, d, e, f
5	c, e, f, h
6	d, e, f
7	a, f, g
8	d, e, g, h
9	a, b, c, f
10	c, d, e, h

Обчислимо підтримку одноелементних наборів, результати представлено в таблиці 3.1.2

Таблиця 3.1.2

Набір	Id транзакцій в яких містяться	Підтримка
a	1, 2, 3, 7, 9	50%
b	1, 3, 9	30%
c	1, 2, 3, 4, 5, 9, 10	70%
d	1, 2, 3, 4, 6, 8, 10	70%
e	1, 3, 4, 5, 6, 8, 10	70%
f	2, 4, 5, 6, 7, 9	60%
g	3, 7, 8	30%
h	5, 8, 10	30%

Серед одноелеметних наборів, лише набори для елементів *b*, *g*, *h* не мають достатньої підтримки. Решту елементів використаємо для генерації двоелементних наборів. На даному кроці можна поєднувати довільні одноелементні набори, бо вони завжди будуть відрізнятися лише одним

елементом. Так як інших власних підмножини, крім тих, з яких утворений двоелементний набір не існує, немає потреби в кроці фільтрації.

Для утворених двоелементних наборів підрахуємо підтримку, результати разом з множиною транзакцій, в яких містяться ці набори зображені в таблиці 3.1.3

Таблиця 3.1.3

Набір	Id транзакцій в яких містяться	Підтримка
a, c	1, 2, 3, 9	40%
a, d	1, 2, 3	30%
a, e	1, 3	20%
a, f	2, 7, 9	30%
c, d	1, 2, 3, 4, 10	50%
c, e	1, 3, 4, 5, 10	50%
c, f	2, 4, 5, 9	40%
d, e	1, 3, 4, 6, 8, 10	60%
d, f	2, 4, 6	30%
e, f	4, 5, 6	30%

Можна побачити, що лише набори $\{a,c\},\{c,d\},\{c,e\},\{d,e\},\{c,f\}$ мають достатній рівень підтримки. З них ми можемо обрати відповідно пари, що відрізняються лише одним елементом $\{a,c\}ma\{c,d\}$, $\{a,c\}ma\{c,e\}$ $\{a,c\}ma\{c,f\}$, $\{c,d\}ma\{c,e\}$, $\{c,d\}ma\{d,e\}$, $\{c,d\}ma\{c,f\}$, $\{c,e\}ma\{d,e\}$, $\{c,e\}ma\{c,f\}$. Об'єднавши відповідні пари, отримаємо множину $\{a,c,d\},\{a,c,e\},\{a,c,f\},\{c,d,e\},\{c,d,f\},\{c,e,f\}$ триелементних наборів. На кроці фільтрації ми можемо відкинути наступні набори, бо вони мають підмножини, підтримка яких недостатня. Набір $\{a,c,d\}$ містить $\{a,d\}$, набір $\{a,c,e\}$ містить $\{a,e\}$, набір $\{a,c,f\}$ містить $\{a,f\}$, набір $\{c,d,f\}$ містить $\{d,f\}$, набір $\{c,e,f\}$ містить $\{e,f\}$. Отже серед згенерованих наборів залишається лише набір $\{c,d,e\}$ для якого підтримка буде рівна 40%, бо він міститься в транзакціях 1, 3, 4, 10.

Результати можна представити у вигляді таблиці 3.1.4.

Таблиця 3.1.4

n	F_{n}
1	$\{a\},\{c\},\{d\},\{e\},\{f\}$
2	$\{a,c\},\{c,d\},\{c,e\},\{d,e\},\{c,f\}$
3	$\{c,d,e\}$

Для набору $\{c,d,e\}$ побудуємо всі можливі асоціативні правила, для цього розглянемо всі власні підмножини $\{c\},\{d\},\{e\},\{c,d\},\{c,e\},\{d,e\}$. З них можна утворити наступні правила, значення достовірності для яких зазначені в таблиці 3.1.5.

Таблиця 3.1.5

Правило	Достовірність
$\{c\}\Rightarrow\{d,e\}$	$conf(\{c\}\Rightarrow\{d,e\})=supp(\{c,d,e\})/supp(\{c\})=4/7$
$\{d\}\Rightarrow\{c,e\}$	$conf(\{d\}\Rightarrow\{c,e\})=supp(\{c,d,e\})/supp(\{d\})=4/7$
$\{e\}\Rightarrow\{c,d\}$	$conf(\{e\}\Rightarrow\{c,d\})=supp(\{c,d,e\})/supp(\{e\})=4/7$
$\{c,d\}\Rightarrow\{e\}$	$conf(\{c,d\}\Rightarrow\{e\})=supp(\{c,d,e\})/supp(\{c,d\})=4/5$
$\{c,e\}\Rightarrow\{d\}$	$conf(\{c,e\}\Rightarrow\{d\})=supp(\{c,d,e\})/supp(\{c,e\})=4/5$
$\{d,e\}\Rightarrow\{c\}$	$conf(\{d,e\}\Rightarrow\{c\})=supp(\{c,d,e\})/supp(\{d,e\})=4/6$

3.2 Алгоритм Eclat

Одним з основних недоліків алгоритму Apriori є те, що для обчислення підтримки набору елементів потрібен повний прохід бо базі транзакцій. Одне з можливих рішень було запропоновано в рамках алгоритму Eclat та полягало у використанні так званого вертикального формату даних. Даний підхід полягає в тому, що для кожного набору елементів зберігається множина транзакцій, в яких міститься цей набір. Всі кроки алгоритму аналогічні крокам алгоритму Apriori, лише на етапі генерації наборів також обчислюється множина транзакцій, кожна з яких містить новий набір. Це відбувається згідно наступної формули: $T_{A\cup B}=T_A\cap T_B$, де T_A позначатимемо множину транзакцій, що містять набір A.

В розв’язку задачі для алгоритму Apriori можна бачити у відповідних таблицях крім підтримки набору також обчислені множини транзакцій, в яких містяться відповідні набори. Наприклад для набору $\{a,c\}$ маємо $T_{\{a,c\}}=T_{\{a\}}\cap T_{\{c\}}=\{1,2,3,7,9\}\cap\{1,2,3,4,5,9,10\}=\{1,2,3,9\}$.

Результат також можна зобразити за допомогою дерева на рисунку 3.2.1. Де сірим позначено набори, що не мають достатньої підтримки, і вузли також містять множину транзакцій для кожного набору.

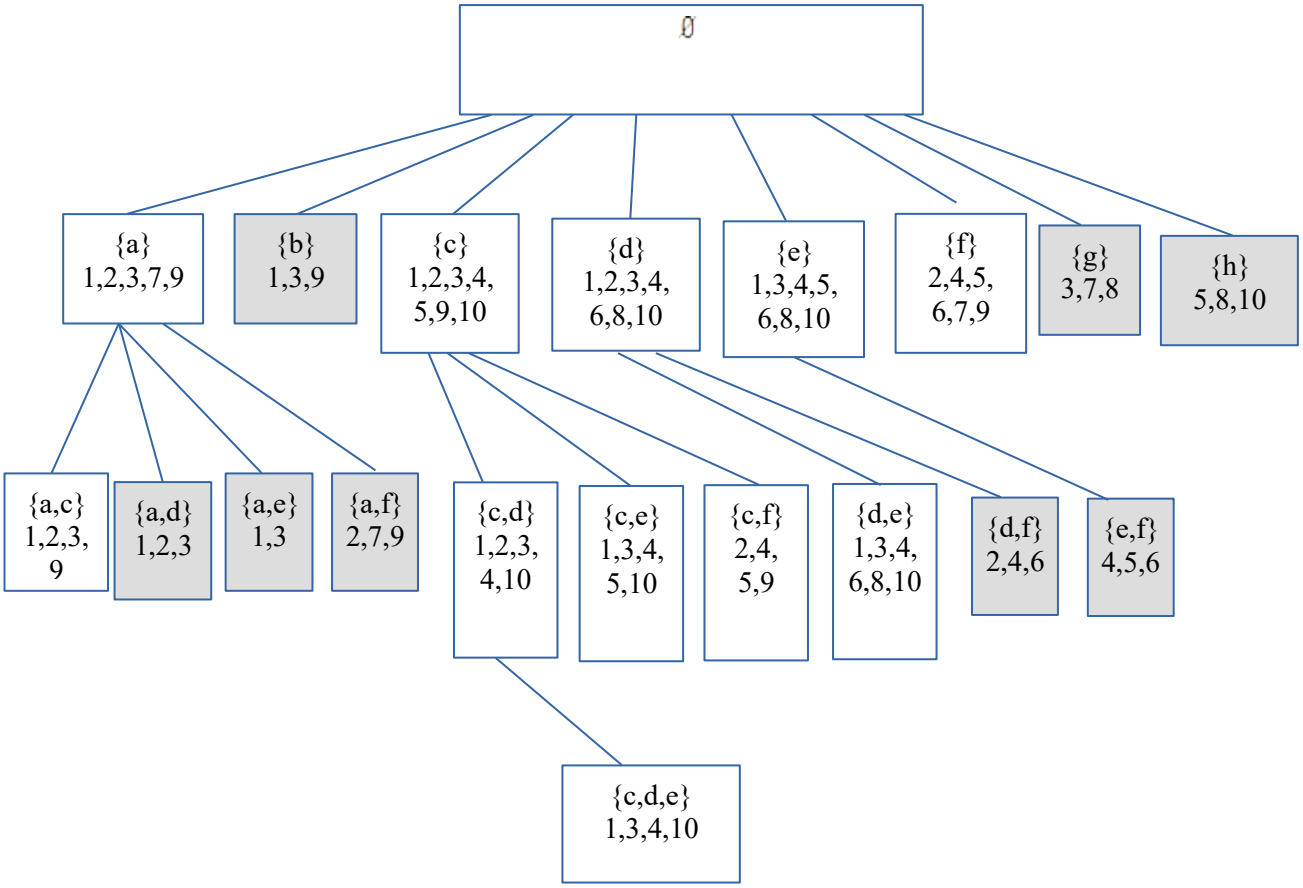

Рисунок 3.2.1

3.3 Алгоритм FP Growth

Ще одним підходом до вирішення проблеми оптимізації підрахунку підтримки наборів елементів може бути використання спеціальних структур даних. Саме на цьому побудований алгоритм FP Growth. В межах даного алгоритму за допомогою одного проходу по базі транзакцій формується дерево спеціального вигляду, що має назву FP-Tree. Воно дозволяє ви-

значити підтримку наборів елементів та верхня оцінка розміру дерева залежить від кількості елементів, а не від кількості транзакцій.

Мітками вузлів в дереві FP-Tree є пара — елемент та кількість транзакцій. Також в процесі побудови дерева, для кожного елементу будується список, що містить всі вершини дерева розмічені даним елементом.

Алгоритм FP Growth складається з наступних етапів:

- Підрахунок підтримки одноелементних наборів, видалення елементів, підтримка яких недостатня. Впорядкування елементів в порядку спадання підтримки в межах транзакції.
- Прохід списком транзакцій та побудова дерева FP-Tree.
- Використання побудованого на попередньому кроці дерева для знаходження наборів елементів, що мають достатню підтримку. Відбувається в порядку зростання підтримки окремих елементів.

Етап побудови дерева відбувається шляхом обходу множини всіх транзакцій. На початку етапу дерево складається з єдиного кореневого вузла, який містить мітку *Null*, або порожню множину. Окремий крок полягає в розгляді однієї транзакції. Етап побудови продовжується до того моменту, поки всі транзакції не будуть враховані. Таким чином на побудову дерева потрібен лише один обхід множини транзакцій. Для окремої транзакції, що розглядається, знаходиться найдовший префікс (враховуючи впорядкованість елементів згідно спаданням їх загальної підтримки в межах транзакції) такий, що в дереві вже існує шлях з відповідним префіксом. В цьому разі в мітці кожного вузла зі знайденого префіксу та шляху, кількість транзакцій збільшується на 1. Частина транзакції що залишилася, додається до дерева у вигляді шляху, що починається з останнього вузла знайденого в дереві префіксу. Мітками вузлів цього шляху є елементи транзакції, а кількість транзакцій рівна 1. Отримане після обробки всіх транзакцій дерево є FP-Tree для зазначеної множини транзакцій.

Етап побудови наборів, що мають достатню підтримку полягає в послідовному розгляді елементів в порядку зростання їх підтримки. Для кожного з них за FP-Tree деревом будуються всі шляхи, що ведуть з кореня до всіх вузлів, мітка яких містить даний елемент. Згідно отриманих шляхів визначаються набори, що мають достатню підтримку, так як підтримка набору рівна тому, яку кількість разів він зустрічається серед знайдених шляхів.

Таким чином, алгоритм дозволяє отримати шукані набори елементів в результаті декількох обходів множини транзакцій та кількості обходів дерева FP-Tree, обмеженою числом елементів. Загалом FP Growth алгоритм має значно кращу оцінку швидкості, ніж класичний алгоритм Apriori, проте потребує місце для збереження спеціального дерева.

Розглянемо приклад з розділу 3.1 та для нього використаємо алгоритм FP Growth. Для цього, спочатку, обчислимо підтримку всіх одноелементних наборів та відсортуємо їх за її спаданням. Результат можна побачити в таблиці 3.3.1.

Таблиця 3.3.1

Набір	Id транзакцій в яких містяться	Підтримка
c	1, 2, 3, 4, 5, 9, 10	70%
d	1, 2, 3, 4, 6, 8, 10	70%
e	1, 3, 4, 5, 6, 8, 10	70%
f	2, 4, 5, 6, 7, 9	60%
a	1, 2, 3, 7, 9	50%
b	1, 3, 9	30%
g	3, 7, 8	30%
h	5, 8, 10	30%

Як і раніше, маємо елементи *b*, *g*, *h*, що не мають потрібної підтримки, видалимо їх з транзакцій, а решту елементів відсортуємо згідно з отриманим порядком — *c*, *d*, *e*, *f*, *a*. Результуючі транзакції представлені в таблиці 3.3.2.

Таблиця 3.3.2

Id транзакції	Елементи
1	c, d, e, a
2	c, d, f, a
3	c, d, e, a
4	c, d, e, f
5	c, e, f
6	d, e, f
7	f, a
8	d, e
9	c, f, a
10	c, d, e

Перша транзакцій містить елементи *c*, *d*, *e*, *a*. Так як на початку дерево містить лише кореневий елемент, ми просто додаємо відповідний шлях до дерева. Результат представлено на рисунку 3.3.1.

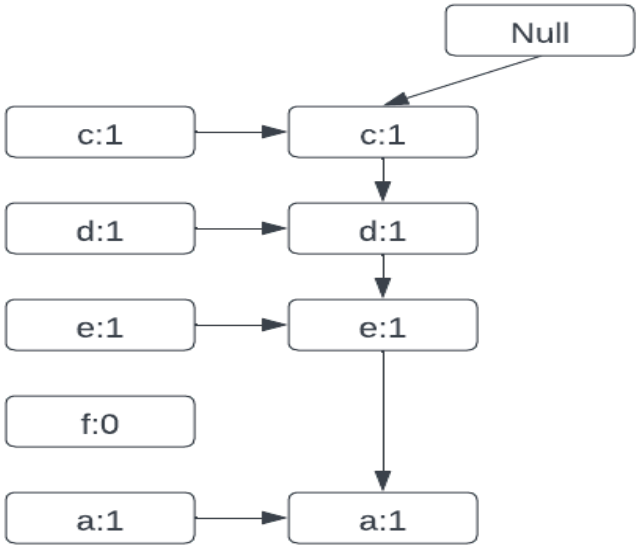

Рисунок 3.3.1

Наступна транзакція складається з *c*, *d*, *f*, *a.* В дереві вже існує шлях, що співпадає з транзакцією за префіксом і цей префікс *c*, *d*. Ми збільшимо лі-

чильники міток відповідних елементів та додамо шлях *f*, *a*, починаючи з останнього вузла знайденого префіксу. Результат представлено на рисунку 3.3.2.

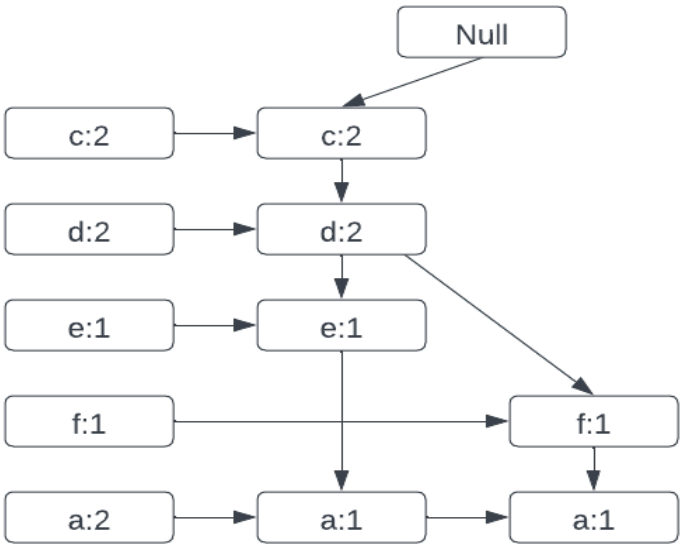

Рисунок 3.3.2

Наступна транзакція співпадає з першою, отже нам потрібно лише збільшити лічильники відповідних міток, що зображено на рисунку 3.3.3.

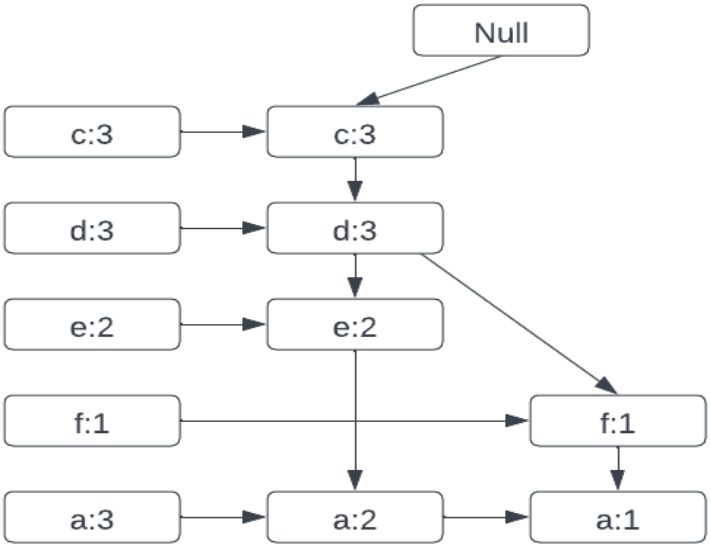

Рисунок 3.3.3

Наступні транзакції — *c*, *d*, *e*, *f* та *c*, *e*, *f* мають префікси *c*, *d*, *e* та *c* відповідно. Результати послідовного додавання шляхів зображено на рисунку 3.3.4 та рисунку 3.3.5

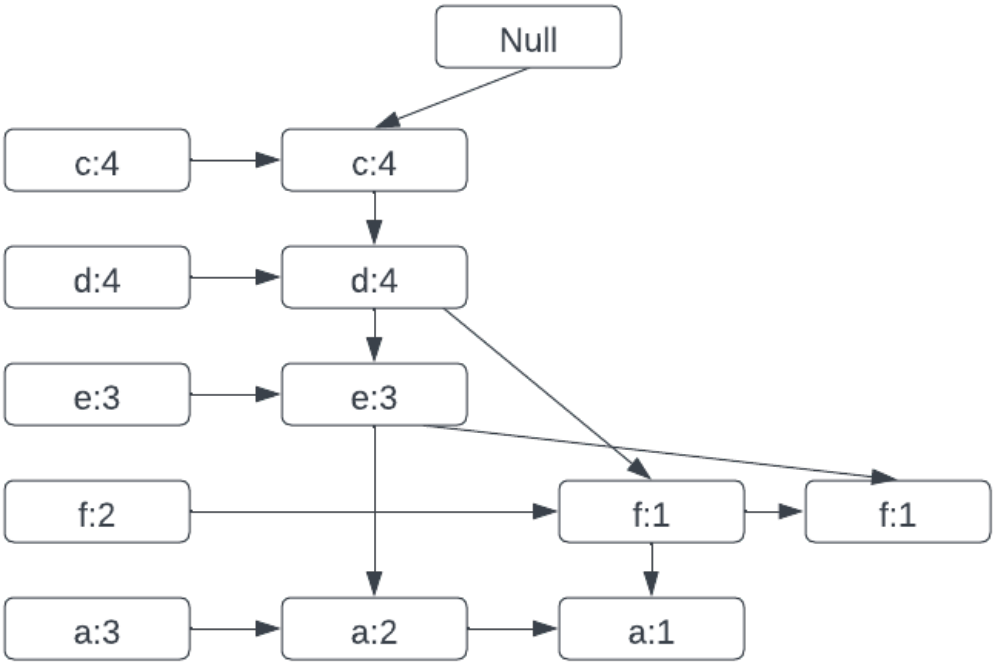

Рисунок 3.3.4

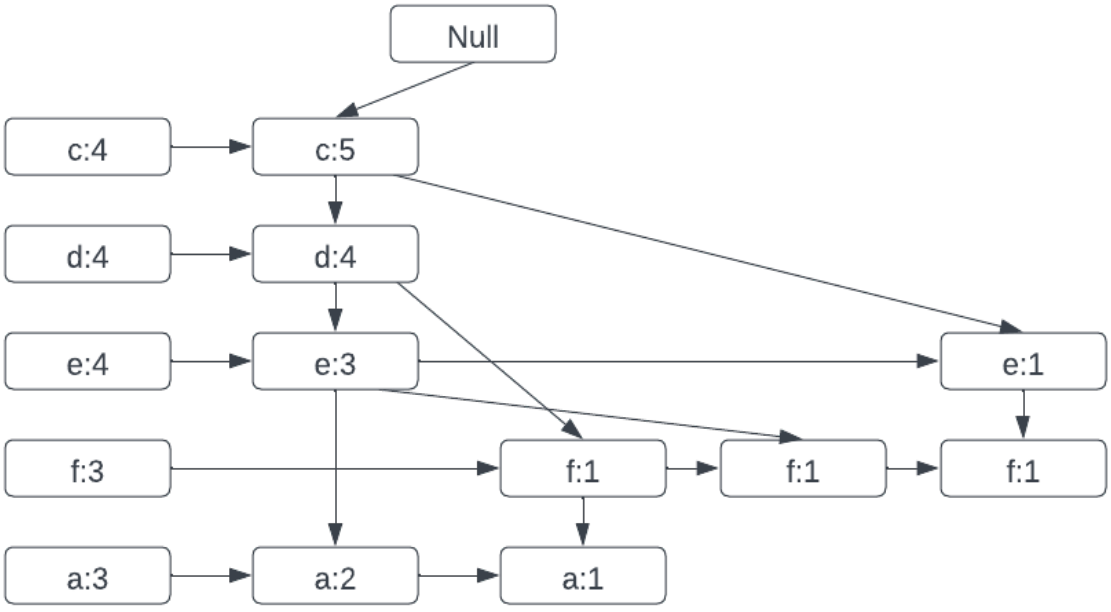

Рисунок 3.3.5

Наступні транзакції — *d*, *e*, *f* та *f*, *a* не мають префікси в дереві, тому нові шляхи будуть додані починаючи з кореня. Результати послідовного додавання шляхів зображено на рисунку 3.3.6.

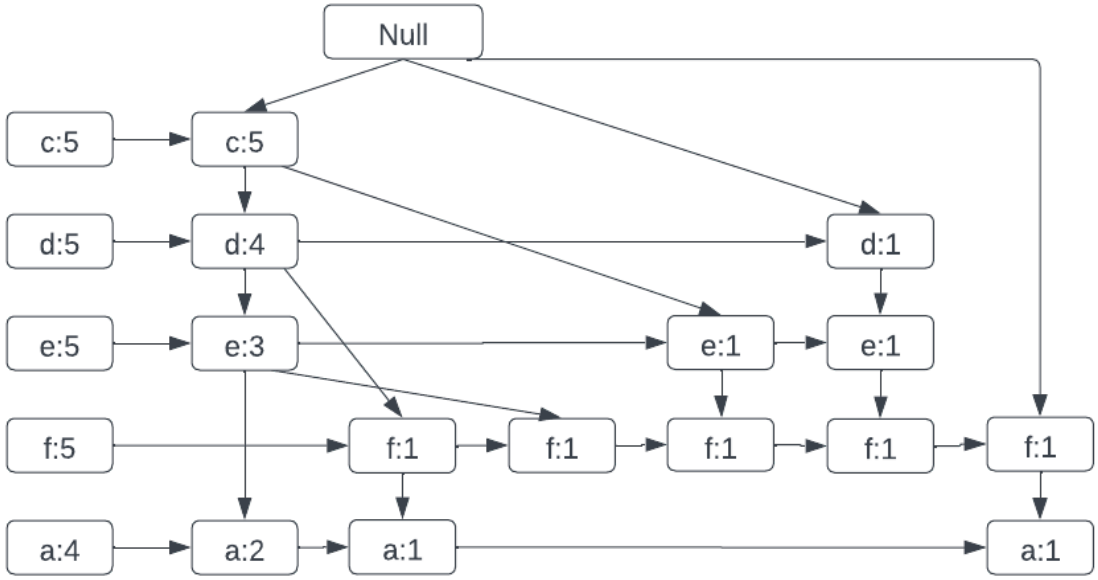

Рисунок 3.3.6

Наступні транзакції — *d*, *e* та *c*, *d*, *e* та вже наявні в дереві як префікси існуючих шляхів, тому потрібно лише змінити мітки вузлів відповідно. А для транзакції *c*, *f*, *a* потрібно додати новий шлях, враховуючи префікс *c*. Результуюче дерево зображено на 3.3.7. Варто відмітити також, що в процесі побудови дерева також кожен новий вузол додавався до списку вузлів для відповідного елемента. Це дозволить спростити процес пошуку всіх шляхів в дереві, що ведуть до певного елементу, який є ключовим елементом етапу генерування наборів елементів, що часто зустрічаються в транзакціях.

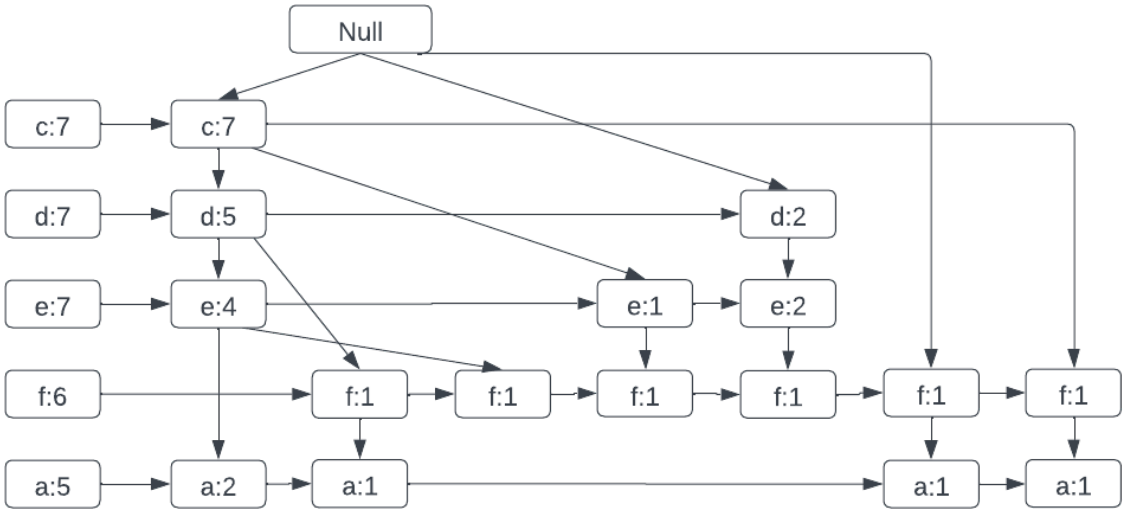

Рисунок 3.3.7

Починаючи з елемента *a* випишемо всі шляхи, що ведуть до нього в дереві, враховуючи мітку при елементі *a*, як кількість $\{\{cdea:2\},\{cdfa:1\},\{fa:1\},\{cfa:1\}\}$. Рахуючи за отриманими шляхами кількість входжень двоелементних наборів, що містять *a*, маємо $\{\{ca:4\},\{da:3\},\{ea:2\},\{fa:3\}\}$. Достатню підтримку має лише $\{ca:4\}$.

Проведемо аналогічні дії відносно елемента *f*. Це дозволить проаналізувати всі набори, що містять *f*, але не містять *a*. В результаті маємо $\{\{cdf:1\},\{cdef:1\},\{cef:1\},\{def:1\}\{cf:1\}\{f:1\}\}$. Розглядаючи двоелементні набори: $\{\{cf:4\},\{df:3\},\{ef:2\}\}$. Достатню підтримку має лише $\{cf:4\}$.

Для елемента *e* маємо наступні шляхи: $\{\{cde:4\},\{ce:1\},\{de:2\}\}$. Що означає, що двоелементні та триелементні набори з достатньою підтримкою наступні: $\{cde:4\},\{ce:5\},\{de:6\}$.

Для елемента *d* маємо наступні шляхи: $\{\{cd:5\},\{d:2\}\}$. Що означає, що двоелементний набір $\{cd:5\}$ має достатню підтримку.

Об’єднавши всі знайдені набори отримаємо результат, що співпадає з наведеним в таблиці 3.1.4 для алгоритму Apriori. Процес побудови правил для отриманих наборів не відрізняється від приведеного в розділі 3.1.

3.4 Вправи до розділу

Використовуючи алгоритм Apriori або FP Growth побудувати асоціативні правила для наступних множин транзакцій, що мають minsupport=40% та minconfidence=75%.

1. {(a,b,c,d), (b,c,d), (a,e,f,g,h), (b,c,d,e,g,j), (b,c,d,e,f), (a,f,g), (a,i,j), (a,b,e,h), (f,g,h,i,j), (e,f,h)}.

2. {(a,b,c,f), (b,c,f), (b,d,e,g,h), (b,c,e,f,g,j), (b,c,d,e,f), (a,d,g), (a,i,j),(a,b,e,h), (d,g,h,i,j), (d,e,h)}

3. {(a,b,c,d,e,f), (b,c,d), (b,e,g,h), (b,c,g,j), (b,c,d,e,f), (a,e,f), (a,i,j), (a,b,c,e,h), (e,6,h,i,j), (b,f,h)}

4. ЗАДАЧА КЛАСТЕРИЗАЦІЇ

Задача кластеризації є однією з основних проблем Data Mining. Вона полягає у виділенні сукупностей подібних записів з набору даних, що аналізується. Ці сукупності, або множини прийнято називати кластерами, хоча історично виникали різні терміни для їх позначення в межах різних дисциплін. Порівнюючи задачі кластеризації та класифікації, можна знайти як багато подібного, так і багато відмінностей. Обидві задачі базуються на понятті подібності. Правильний вибір відстані, що використовується, нормалізація значень, поєднання категорійних та числових ознак — всі ці аспекти мають значний вплив на результат роботи алгоритмів покликаних на розв'язання обох проблем. В той же час, основною відмінністю між задачами є те, що в межах класифікації, для тренувальної множини задано цільову ознаку, яка вказує на те, до якого класу відноситься запис, при кластеризації такої ознаки не існує. В такому випадку, про точність кластеризації не доводиться говорити, але можливо обчислити показники якості. Такі показники дозволяють формально виразити інтуїтивне розуміння, що найкраще розбиття на кластери має максимізувати відмінності між елементами різних кластерів і мінімізувати відмінності всередині кластеру.

Найчастіше кластеризація є одним з попередніх кроків процесу Data Mining. Результати його є часто вхідними даними інших процесів. Це дозволяє структурувати набір даних, що аналізується. Однією з важливих задач, які покликана вирішити кластеризація, є зменшення розмірності простору ознак. Зважаючи на те, наскільки зросла кількість ознак в сучасних базах, дана задача набуває все більшого значення. Це дозволяє, зокрема, покращити точність результуючого класифікатора за рахунок розгляду більшої кількості атрибутів на результат, без значного впливу на швидкодію.

В межах алгоритмів та прикладів, що розглядаються в даному розділі, будемо використовувати Евклідову відстань між записами (без врахування нормалізації), або функцію відмінності для категорійних ознак.

Одним з підходів до вирішення проблеми кластеризації є застосування ієрархічних алгоритмів, найбільш розповсюдженими з яких є агломеративний тип. Вони орієнтовані на побудову кластерів способом “знизу-вгору”, тобто починаючи з найпростіших можливих кластерів, послідовно формувати складніші, шляхом злиття. Важливо, що в такому разі попередня оцінка кількості кластерів не є обов’язковою.

Іншим підходом є використання одного з алгоритмів, заснованих на алгоритмі k-Means. В цьому випадку головна увага приділяється процесу розбиття на підмножини, подібність елементів всередині яких, значно переважає подібність між елементами різних підмножин шляхом ітеративного покращення розбиття на кластери. Більшість алгоритмів, що відносяться до цього типу, передбачає наявність інформації про цільову кількість кластерів.

Окремо виділяються алгоритми кластеризації, що базуються на понятті щільності. За такого підходу кластери трактуються, як множини записів, що розташовані щільно один відносно одного. Таким чином поєднується критерій близькості точок всередині кластеру та віддаленості точок з різних кластерів.

Загалом, варто відмітити, що, як і для задачі класифікації, спектр ідей та підходів, що успішно застосовується є широким.

4.1 Ієрархічні алгоритми

Ієрархічні алгоритми направлені на побудову дендрограми, що задає ієрархію кластерів. Ієрархічні алгоритми поділяються на агломеративні та дивізимні. Агломератині базуються на тому, що на першому кроці формуються найменші можливі кластери і послідовно відбувається їх об’єднання в більші кластери, поки не буде утворено один кластер, що буде складатися з усіх точок. Дівізимні алгоритми ж, навпаки, засновані на поділі кластерів на менші і роботу алгоритми починають зі стану, коли всі записи

містяться в єдиному кластері. Найчастіше застосовується агломеративний підхід.

Загальна схема складається з наступних кроків:

- Виділити кожен запис з набору даних, що розглядається в окремий кластер.
- Визначити два кластери, відстань між якими найменша і об’єднати їх. Повторити цей крок, поки не залишиться лише один кластер

Саме за тим, як визначається відстань між двома кластерами відрізняються окремі методи. Серед способів, які найчастіше використовуються виділимо наступні:

- Single-link. Відстань визначається як найменша з усіх можливих відстаней між записам, один з яких належить першому кластеру, а інший — другому.
- Complete-link. Відстань визначається як найбільша з усіх можливих відстаней між записам, один з яких належить першому кластеру, а інший — другому.
- Average-link. Відстань визначається як середнє всіх можливих відстаней між записам, один з яких належить першому кластеру, а інший — другому.
- Centroid-link. Відстань визначається як відстань між центрами мас кластерів.
- Median-link. Відстань визначається як відстань між медіанами кластерів.
- Ward. Відстань визначається як приріст середньоквадратичної похибки, отриманий внаслідок об’єднання цих кластерів.

Аналогічно до інших алгоритмів кластеризації середньоквадратична похибка визначається як середнє квадратів відстаней між записами, що належать кластеру і центром ваг кластеру. З наведених підходів, саме метод Уорда показує найкращі результати.

Розглянемо приклад в двовимірному просторі ознак з Евклідовою відстанню для демонстрації Single-link, Complete-link, та Average-link. {(1,5),(2,4),(2,7),(2,9),(3,1),(5,5),(5,7),(6,6),(7,6),(10,6)} — використаємо даний набір записів рисунок 4.1.1. Позначимо точки A, B, C, D, E, F, G, H, I, J відповідно.

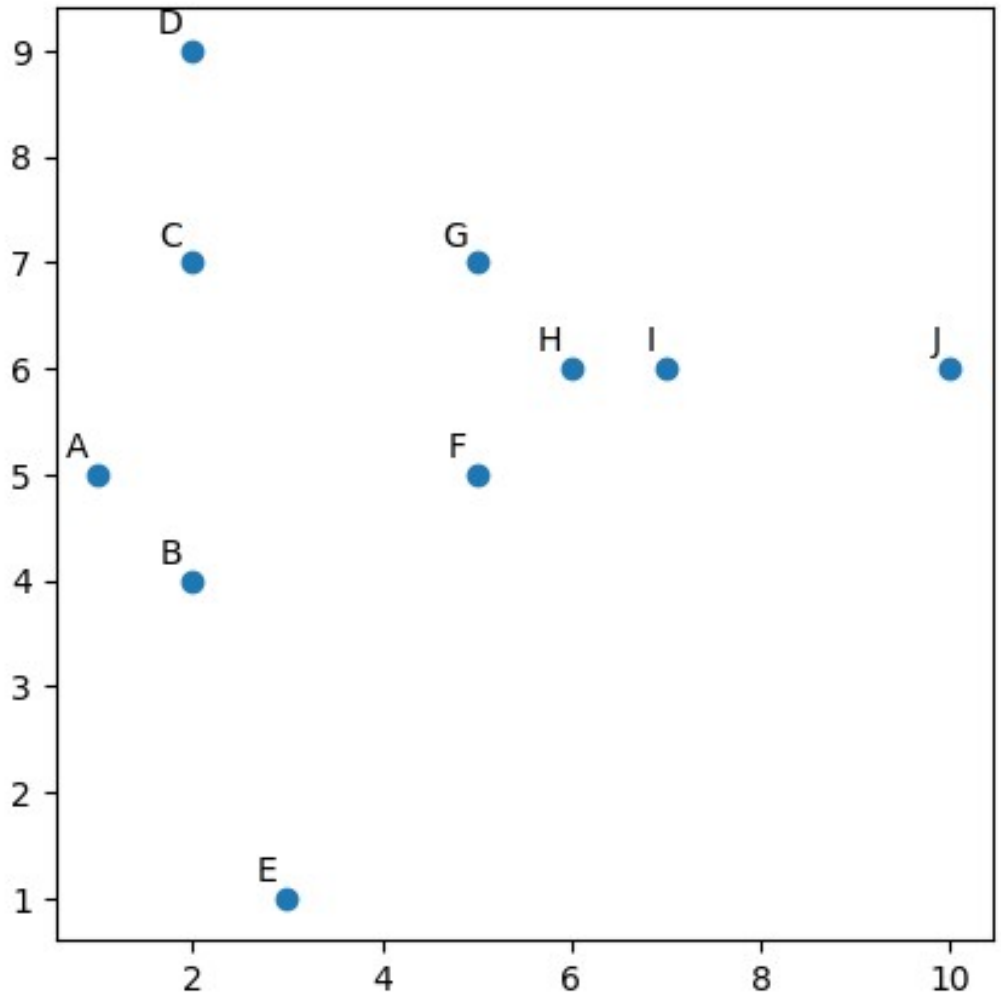

Рисунок 4.1.1

Обчислимо матрицю відстаней, результат представлено в таблиці 4.1.1.

Розглянемо метод Single-link. На початку маємо кластери {A}, {B}, {C}, {D}, {E}, {F}, {G}, {H}, {I}, {J}, кожен з яких складається з єдиної точки.

Таблиця 4.1.1

	A	B	C	D	E	F	G	H	I	J
A	0	1.4	2.2	4.1	4.5	4	4.5	5.1	6.1	9.1
B	1.4	0	3	5	3.2	3.2	4.2	4.5	5.4	8.2
C	2.2	3	0	2	6.1	3.6	3	4.1	5.1	8.1
D	4.1	5	2	0	8.1	5	3.6	5	5.8	8.5
E	4.5	3.2	6.1	8.1	0	4.5	6.3	5.8	6.4	8.6
F	4	3.2	3.6	5	4.5	0	2	1.4	2.2	5.1
G	4.5	4.2	3	3.6	6.3	2	0	1.4	2.2	5.1
H	5.1	4.5	4.1	5	5.8	1.4	1.4	0	1	4
I	6.1	5.4	5.1	5.8	6.4	2.2	2.2	1	0	3
J	9.1	8.2	8.1	8.5	8.6	5.1	5.1	4	3	0

Найменша відстань згідно матриці між кластерами {H} та {I} – 1. Об’єднуємо відповідні кластери. Маємо {A}, {B}, {C}, {D}, {E}, {F}, {G}, {H, I}, {J}. Таблицю відстаней можна оновити для поточного кроку, об’єднавши записи, що відповідають кластерам, що зливаються і обираючи найменше з двох значень, як відстань до інших кластерів — таблиця 4.1.2.

Таблиця 4.1.2

	A	B	C	D	E	F	G	H,I	J
A	0	1.4	2.2	4.1	4.5	4	4.5	5.1	9.1
B	1.4	0	3	5	3.2	3.2	4.2	4.5	8.2
C	2.2	3	0	2	6.1	3.6	3	4.1	8.1
D	4.1	5	2	0	8.1	5	3.6	5	8.5
E	4.5	3.2	6.1	8.1	0	4.5	6.3	5.8	8.6
F	4	3.2	3.6	5	4.5	0	2	1.4	5.1
G	4.5	4.2	3.	3.6	6.3	2	0	1.4	5.1
H,I	5.1	4.5	4.1	5	5.8	1.4	1.4	0	3
J	9.1	8.2	8.1	8.5	8.6	5.1	5.1	3	0

Згідно з оновленою таблицею, найближчими є декілька пар кластерів, відстань між якими однакова(1.4) — це {A} і {B}, {F} і {H, I}, {G} і {H, I}. Оберемо першу пару, керуючись порядком, в якому задані точки, отримаємо набір кластерів {A, B}, {C}, {D}, {E}, {F}, {G}, {H, I}, {J} та матрицю відстаней, представлену в таблиці 4.1.3

Таблиця 4.1.3

	A,B	C	D	E	F	G	H,I	J
A,B	0	2.2	4.1	3.2	3.2	4.2	4.5	8.2
C	2.2	0	2	6.1	3.6	3	4.1	8.1
D	4.1	2	0	8.1	5	3.6	5	8.5
E	3.2	6.1	8.1	0	4.5	6.3	5.8	8.6
F	3.2	3.6	5	4.5	0	2	1.4	5.1
G	4.2	3.	3.6	6.3	2	0	1.4	5.1
H,I	4.5	4.1	5	5.8	1.4	1.4	0	3
J	8.2	8.1	8.5	8.6	5.1	5.1	3	0

Наступним кроком об’єднаємо {F} і {H, I} – результат в таблиці 4.1.4.

Таблиця 4.1.4

	A,B	C	D	E	F,H,I	G	J
A,B	0	2.2	4.1	3.2	3.2	4.2	8.2
C	2.2	0	2	6.1	3.6	3	8.1
D	4.1	2	0	8.1	5	3.6	8.5
E	3.2	6.1	8.1	0	4.5	6.3	8.6
F,H,I	3.2	3.6	5	4.5	0	1.4	3
G	4.2	3.	3.6	6.3	1.4	0	5.1
J	8.2	8.1	8.5	8.6	3	5.1	0

Наступним кроком об’єднаємо {G} і {F, H, I} – результат в таблиці 4.1.5.

Таблиця 4.1.5

	A,B	C	D	E	F,G,H,I	J
A,B	0	2.2	4.1	3.2	3.2	8.2
C	2.2	0	2	6.1	3	8.1
D	4.1	2	0	8.1	3.6	8.5
E	3.2	6.1	8.1	0	4.5	8.6
F,G,H,I	3.2	3	3.6	4.5	0	3
J	8.2	8.1	8.5	8.6	3	0

Згідно матриці, найменша відстань — 2, між кластерами {C} та {D}. Об'єднаємо їх, наступним кроком буде об'єднання кластерів {A, B} та {C, D}, що зображено в таблицях 4.1.6 і 4.1.7 відповідно.

Таблиця 4.1.6

	A,B	C,D	E	F,G,H,I	J
A,B	0	2.2	3.2	3.2	8.2
C,D	2.2	0	6.1	3	8.1
E	3.2	6.1	0	4.5	8.6
F,G,H,I	3.2	3	4.5	0	3
J	8.2	8.1	8.6	3	0

Таблиця 4.1.7

	A,B,C,D	E	F,G,H,I	J
A,B.C.D	0	3.2	3	8.1
E	3.2	0	4.5	8.6
F,G,H,I	3	4.5	0	3
J	81	8.6	3	0

Згідно з таблицею, наступна найменша відстань — це 3. Керуючись порядком точок, об'єднаємо кластери {A, B, C, D} та {F, G, H, I}

Таблиця 4.1.8

	A,B,C,D,F,G,H,I	E	J
A,B.C.D,F,G,H,I	0	3.2	3
E	3.2	0	8.6
J	81	8.6	0

Після цього до великого кластеру {A, B, C, D, F, G, H, I}, додамо послідовно {J} та {E}. В результаті отримаємо дендрограму зображену на рисунку 4.1.2. Варто відмітити, що по вісі ординат відмічена та мінімальна відстань при якій відповідні кластери було об'єднано і так як в деяких випад-

ках одночасно декілька пар кластерів вважалися найближчим, це і представлено на дендрограмі.

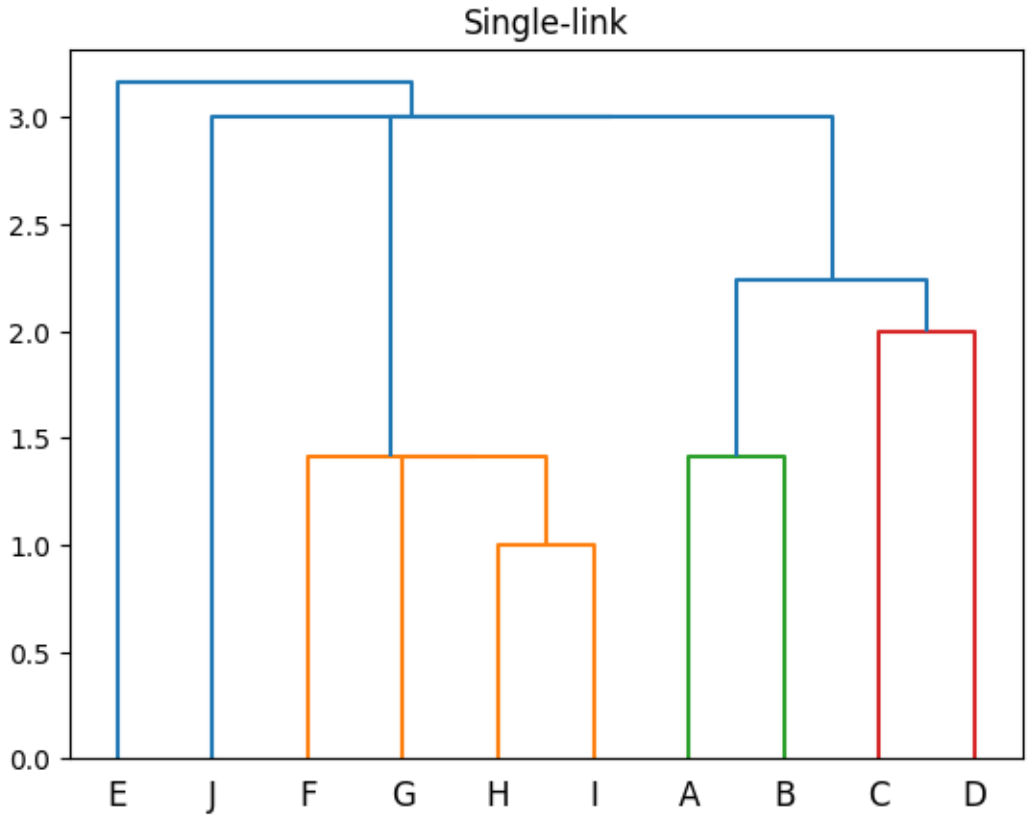

Рисунок 4.1.2

Для Complete-link на кожному кроці відстань між кластерами, що не змінюються та новим кластером, в матриці рахується, як максимум відстаней тих кластерів, що об'єднуються. У випадку Average-link – рахується середнє з врахуванням кількостей елементів в кожному кластері, що зливаються. На початкових кроках означені алгоритми об'єднують схожі точки, зокрема для одноелементних кластерів відстані обчислені за трьома розглянутими методами співпадають. Проте наступні кроки можуть суттєво відрізнятися. Single-link метод схильний до формування кластерів, що є більш гомогенними, відстань між точками всередині яких мінімальна. Complete-link метод, в свою чергу, приводить до формування більш компактних кластерів, які можна назвати сфероподібними у відповідному просторі. Average-link метод покликаний зменшити вплив екстремальних значень відстані між точками різних кластерів. Дендрограми для набору

даних, що розглядається, згідно з методами Complete-link та Average-link представлені на рисунках 4.1.3 та 4.1.4.

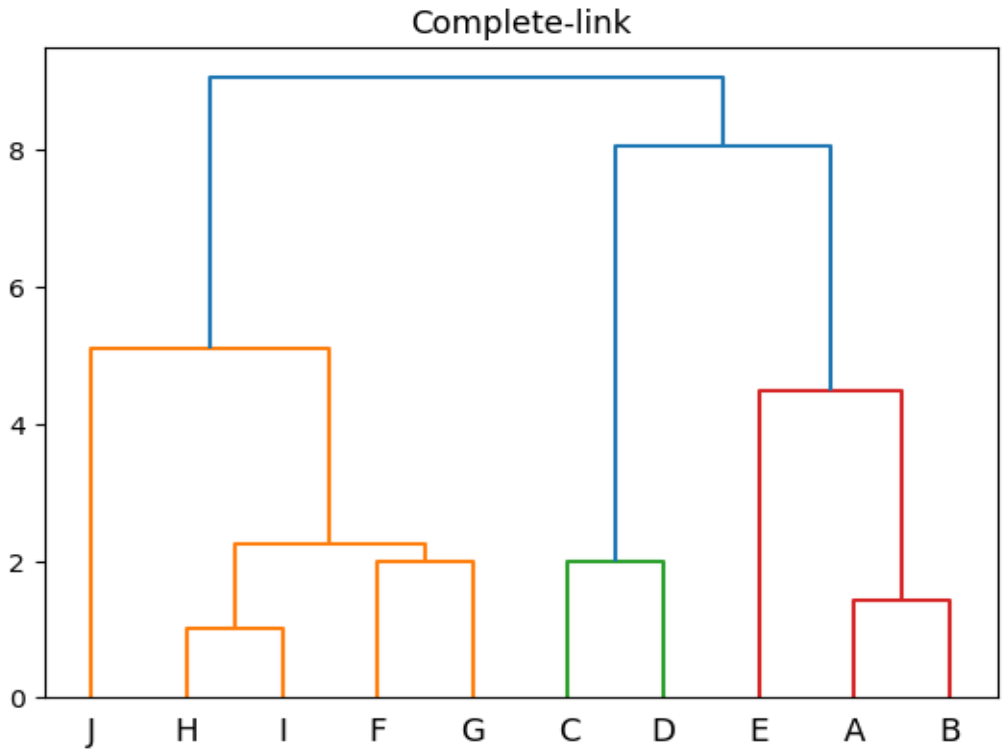

Рисунок 4.1.3

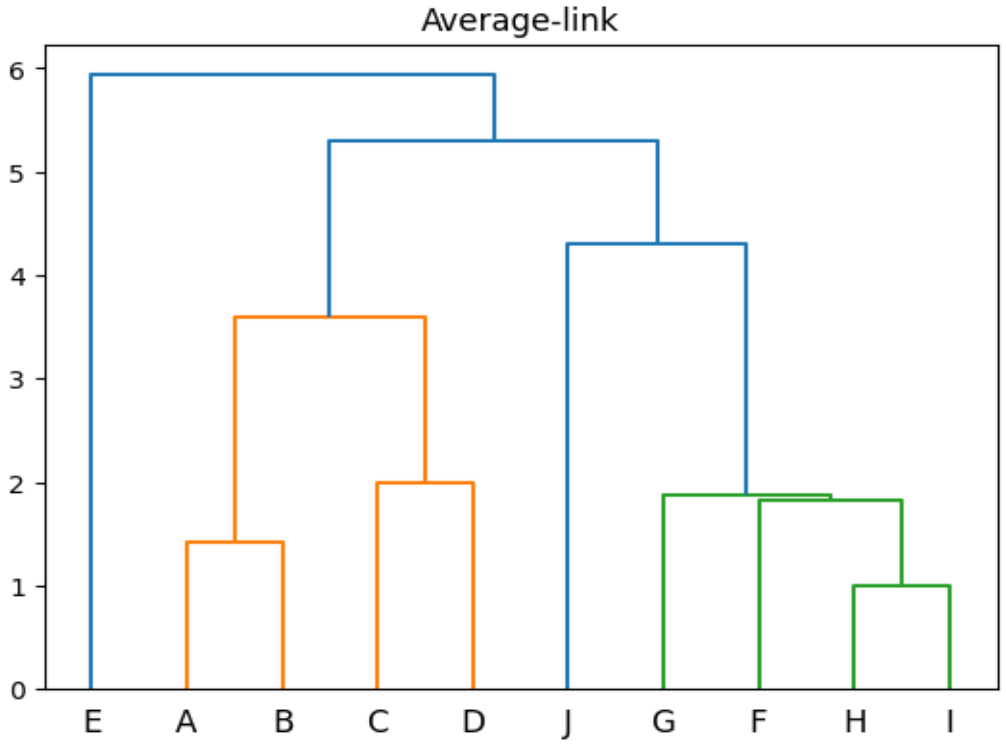

Рисунок 4.1.4

4.2 Алгоритм k-Means

Одним з найбільш популярних алгоритмів для кластеризації є алгоритм k-Means. Це призвело, зокрема, до появи різних варіацій алгоритму, що будуть розглянуті окремо.

Суть алгоритму полягає в ітеративному розбитті набору даних на задану кількість кластерів, де розбиття на кожній ітерації базується на розбитті з попередньої. Ключовими елементами алгоритму є те, що бажана кількість кластерів є вхідним параметром, та значний вплив початкового розбиття, що найчастіше відбувається довільним чином, або з використанням певних евристик. Загалом це призводить до ризику того, що розв'язок зійдеться до локального мінімуму, і серед всіх можливих розбиттів на кластери, отримане не буде найкращим.

Після вибору вхідних параметрів, зокрема, бажаної кількості кластерів, метод відбувається згідно наступної послідовності кроків:

- Вибір початкових центрів кластерів. Зазвичай обираються довільним чином серед вхідних даних.
- Визначення кластеру для кожної точки згідно з тим, який центр є найближчим.
- Виходячи з поточного розподілу, обчислення нового центру кожного кластеру, як центру мас всіх записів, віднесених до відповідного кластеру.
- Перевірка виконання умови зупинки, якщо вона не виконується, то повернення до другого кроку, кроку перерозподілу записів згідно нових центрів кластерів.

Умовою зупинки алгоритму є те, що розподіл на кластери не змінився з попередньої ітерації, тобто всі записи відносяться до кластеру, до якого вони були віднесені на попередній ітерації. Також часто використовують додаткову умову зупинки — умову того, що зміна середньоквадратичної помилки повинна перевищувати певний поріг. Тобто, у випадку, якщо кластеризація не покращується суттєво, робота алгоритму припиняється.

Розглянемо приклад з попереднього розділу і застосуємо до нього алгоритм k-Means при k=3. Оберемо початковими центрами кластерів точки A, F, J. Згідно з відстанями між точками, обчисленими в таблиці 4.1.1. — до кластеру з центром в A віднесемо точки {A, B, C, D, E}, де точка E рівновіддалена від точок A і F, проте виберемо перший по порядку кластер. Кластер з центром в F буде, відповідно, {F, G, H, I}. А останній кластер, складатиметься лише з J. Обчислимо координати центрів мас кожного кластеру, виходячи з координат їх елементів, отримаємо $C_1=\left(\frac{1+2+2+2+3}{5};\frac{5+4+7+9+1}{5}\right)=(2;5.2)$ $C_2=\left(\frac{5+5+6+7}{4};\frac{5+7+6+6}{4}\right)=(5.75;6)$ $C_3=(10;6)$. Зобразимо це на рисунку 4.2.1. і обчислимо відстані від точок до центрів мас кластерів в таблиці 4.2.1. Легко побачити, що згідно таблиці, найближчими центрами мас для точок будуть центри мас кластерів, до яких вони відносяться і розподіл не зміниться, таким чином виконується умова зупинки і результатом буде даний розподіл.

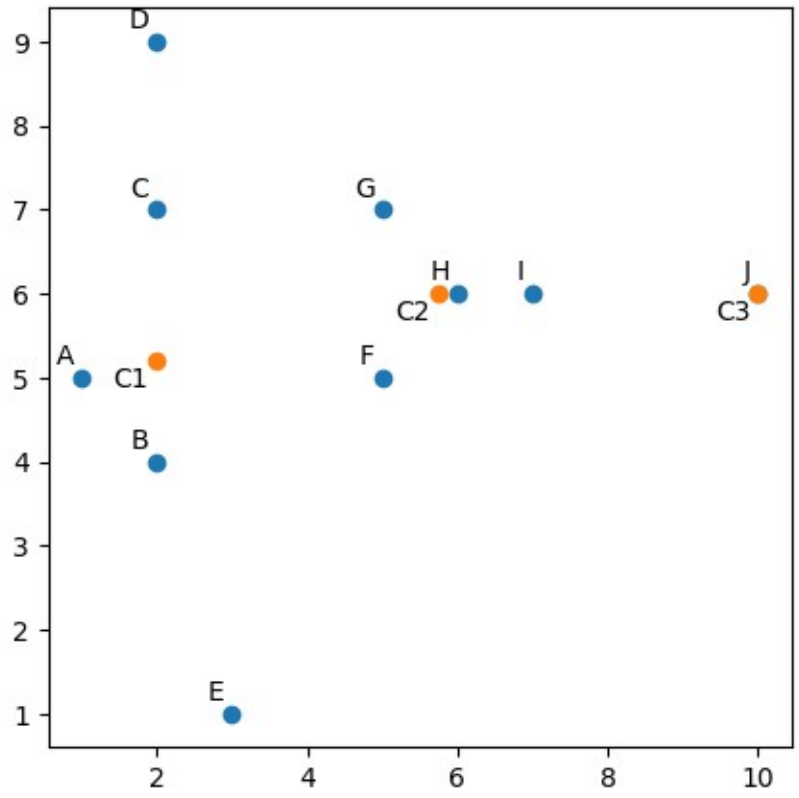

Рисунок 4.2.1

	A	B	C	D	E	F	G	H	I	J
C_1	1	1.2	2.8	3.8	4.3	3	3.5	4.1	5.1	8
C_2	4.9	4.25	3.9	4.8	5.7	1.25	1.25	0.25	1.25	4.25
C_3	9	8.2	8.1	8.5	8.6	5.1	5.1	4	3	0

Таблиця 4.2.1

Виберемо інші початкові центри кластерів і дослідимо, як зміниться результат. Цього разу початковими центрами оберемо точки F, G та H. В такому разі кластери після першої ітерації будуть наступні: {F, A, B, E}, {G, C, D}, {H, I, J}. В такому випадку центри мас можуть бути обчислені наступним чином: $C_1=\left(\frac{5+1+2+3}{4};\frac{5+5+4+1}{4}\right)=(2.75;3.75)$ $C_2=\left(\frac{5+2+2}{3};\frac{7+7+9}{3}\right)=(3;7.66)$ $C_3=\left(\frac{6+7+10}{3};\frac{6+6+6}{3}\right)=(7.66;6)$. Результат зображено на рисунку 4.2.2

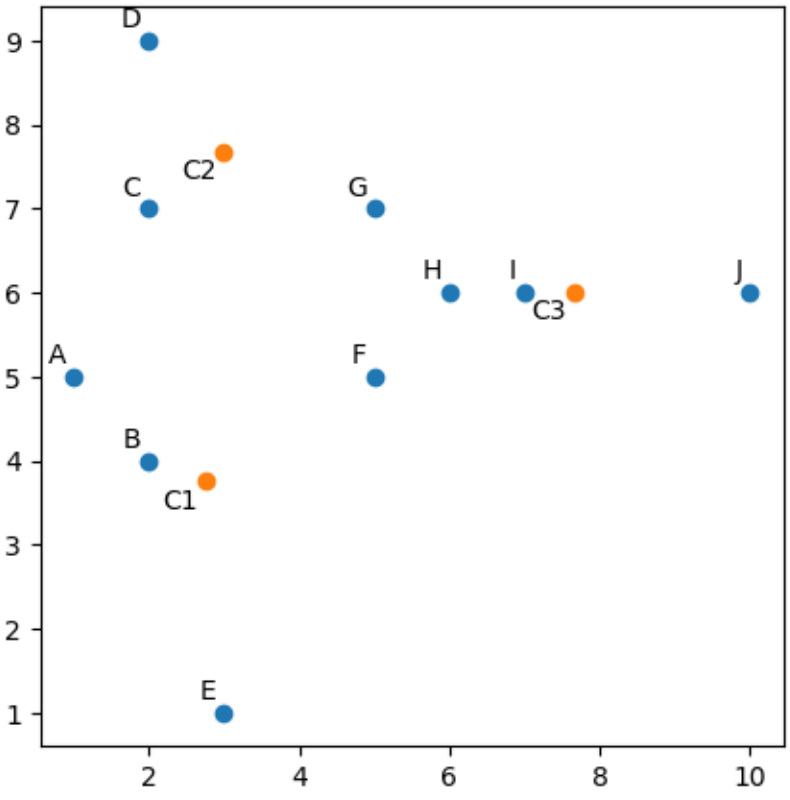

Рисунок 4.2.2

Обчисливши матрицю відстаней від точок до центрів кластерів (таблиця 4.2.2), бачимо, що розподіл не зміниться. Отже, отримали результат. Також, неважко бачити, що отриманий розподіл відрізняється від того, що був знайдений раніше. Цей факт можна пояснити тим, що набір даних для якого розв'язується задача кластеризації, не має чітко виражених кластерів і тому, кожного разу алгоритм зупинився, досягнувши локального мінімуму похибки.

Таблиця 4.2.2

	A	B	C	D	E	F	G	H	I	J
C_1	2.2	0.8	3.3	5.3	2.9	2.6	4.0	4.0	4.8	7.6
C_2	3.3	3.8	1.2	1.7	6.66	3.3	2.1	3.4	4.3	7.2
C_3	6.7	6.0	5.7	6.4	6.8	2.8	2.8	1.66	0.66	2.34

4.3 Алгоритм k-Medoids

Одним з численних варіантів алгоритму k-Means є алгоритм k-Medoid, або k-Медіан. Основною відмінністю є те, що на кожній ітерації у якості нових центрів кластерів обираються не центри мас, а медіани, тобто точки, що є найближчими до центрів мас. Це дозволяє не обчислювати заново кожної ітерації відстані від центрів кластерів до всіх точок, бо ці відстані вже відомі. Також використання медіан дозволяє зменшити вплив аномальних викидів на результат кластеризації. Хоча такі точки значною мірою можуть впливати на обчислення центру мас кластера, віддаляючи його від основної множини подібних точок всередині кластеру, проте за рахунок того, що центром є одна з точок, вдається нівелювати цей вплив.

Використаємо метод k-медіан для того, щоб кластеризувати приклад, розглянутий в попередньому розділі і перевіримо, чи дозволить це досягнути кращих результатів. Перший крок буде співпадати, оберемо знову

точки F, G та H, як початкові центри кластерів. Початкове розбиття в такому разі буде ідентичним — {F, A, B, E}, {G, C, D}, {H, I, J}. Також ідентичними будуть і центри мас цих кластерів. $C_1=(2.75;3.75)$ $C_2=(3;7.66)$ $C_3=(7.66;6)$. Згідно з таблицею 4.2.2 медіанами будуть точки B, C, I як найближчі до центрів мас.

Визначимо розподіл відносно нових центрів кластерів, в такому випадку отримаємо наступні множини: { B, A, E}, {C, D}, {I, F, G, H, J}. Центрами мас даних кластерів будуть $C_1=\left(\frac{2+1+3}{3};\frac{4+5+1}{3}\right)=(2;3.33)$, $C_2=\left(\frac{2+2}{2};\frac{7+9}{2}\right)=(2;8)$, $C_3=\left(\frac{7+5+5+6+10}{5};\frac{6+5+7+6+6}{5}\right)=(6.6;6)$, а медіанами — точки B, C (або D), I. Так як другий кластер маж лише дві точки, кожну з них можна обрати медіаною, проте на розподіл це не вплине і склад кластерів не зміниться. Це і буде результатом роботи алгоритму.

Для порівняння якості результатів k-Means та k-Medoid обчислимо суму квадратів відстаней між точками кластерів і центрами мас відповідного кластеру. Так як кількість точок однакова, не будемо ділити на неї, щоб знайти саме середнє значення.

Для методу k-Means

$$SSE_{1,1}=(5-2.75)^2+(5-3.75)^2+(1-2.75)^2+(5-3.75)^2+(2-2.75)^2+(4-3.75)^2$$
$$+(3-2.75)^2+(1-3.75)^2\approx 31$$
$$SSE_{1,2}=(5-3)^2+(7-7.66)^2+(2-3)^2+(7-7.66)^2+(2-3)^2+(9-7.66)^2\approx 8.7$$
$$SSE_{1,3}=(6-7.66)^2+(6-6)^2+(7-7.66)^2+(6-6)^2+(10-7.66)^2+(6-6)^2\approx 11$$
$$SSE_1\approx 31+8.7+11=50.7$$

Для методу k-Medoid

$$SSE_{2,1}=(2-2)^2+(4-3.33)^2+(1-2)^2+(5-3.33)^2+(3-2)^2+(1-3.33)^2\approx 10.6$$
$$SSE_{2,2}=(2-2)^2+(7-8)^2+(2-2)^2+(9-8)^2=2$$
$$SSE_{2,3}=(7-6.6)^2+(6-6)^2+(5-6.6)^2+(5-6)^2+(5-6.6)^2+(7-6)^2+(6-6.6)^2+(6-6)^2$$
$$+(10-6.6)^2+(6-6)^2=16.6$$
$$SSE_2\approx 10.6+2+16.6=29.2$$

Можемо бачити, що значення середньоквадратичної похибки майже вдвічі менше, що підтверджує актуальність методу k-Medoid і якість отриманої кластеризації. Також варто відмітити, що кількість необхідних обчислень вдалося також скоротити, хоча в порівнянні з k-Means даний алгоритм потребує знаходження медіан на кожному кроці.

4.4 Алгоритм Fuzzy k-Means

Іншою важливою модифікацією алгоритму k-Means, окрім k-Medoid є алгоритм fuzzy k-Means. Він полягає в застосуванні нечіткої логіки до питання належності елемента кластеру, іншими словами, для кожної точки визначається міра належності кожному кластеру. Виконується це таким чином, щоб сума мір належності для однієї точки була рівна 1 і міра належності точки кожному кластеру була обернено пропорційна відстані до центру кластера і рівна 1, у випадку, якщо точка співпадає з центром кластеру. Також в формулі обчислення центру мас кластера, тобто центру кластера для наступної ітерації, міра належності виступає у ролі ваги кожної окремої точки.

Використаємо приклад з попередніх розділів використаний для порівняння алгоритмів k-Means та k-Medoid. Початковими центрами кластерів також оберемо точки F, G та H. Округлені значення належності точок відповідним кластерам, можна побачити в таблиці 4.4.1. Показовим є те, що зважаючи на те, що точки обрані початковими центрами розташовані достатньо близько, для багатьох точок міри належності різним кластерам є досить близькими.

Таблиця 4.4.1

	A	B	C	D	E	F	G	H	I	J
U_1	0.37	0.4	0.33	0.29	0.4	1	0	0	0.24	0.31
U_2	0.33	0.31	0.39	0.42	0.29	0	1	0	0.24	0.31
U_3	0.3	0.29	0.28	0.29	0.31	0	0	1	0.52	0.38

Використавши отримані міри належності як ваги та обчисливши центри мас згідно цього отримаємо $C_1=(4;5.1)$ $C_2=(4;6.1)$ $C_3=(4.9;5.6)$. Це відображено на рисунку 4.4.1. Можна бачити, що в даному випадку, використання нечіткої модифікації алгоритму не допомагає виправити вплив вибору початкових центрів кластерів, проте як і у випадку k-Medoid дозволяє зменшити вплив аномальних вимірів.

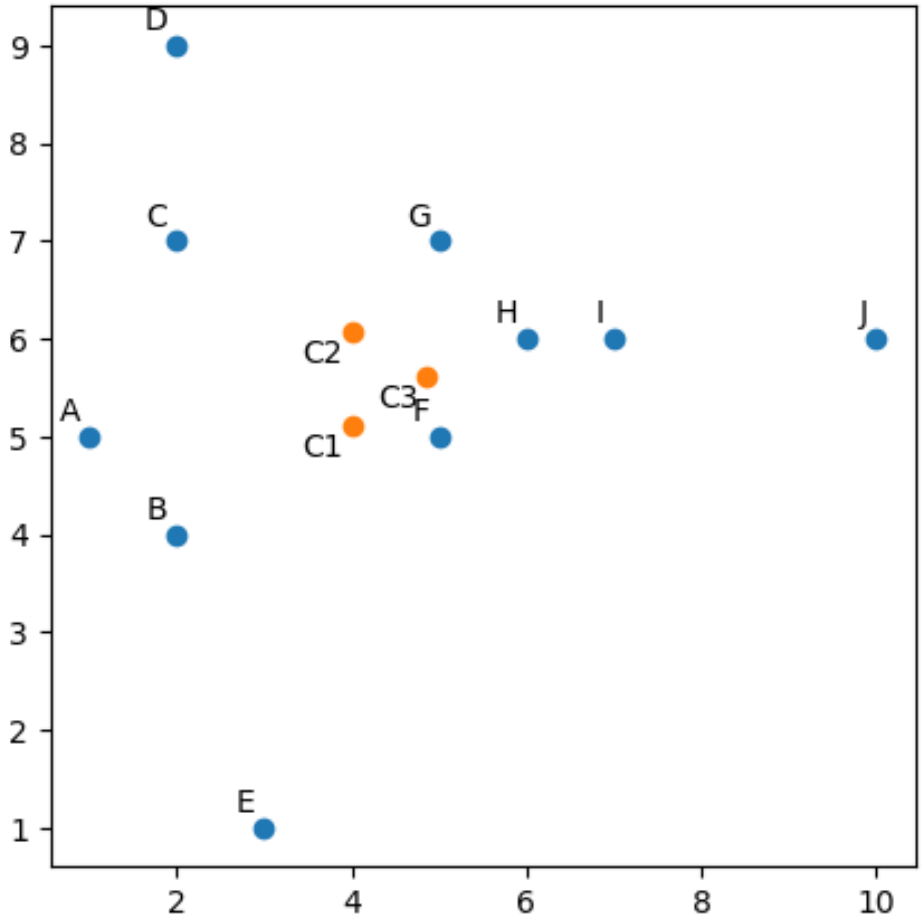

Рисунок 4.4.1

4.5 Алгоритм DBSCAN

Окрім поняття близькості, або подібності при кластеризації також використовується поняття щільності. Саме щільність дозволяє формалізувати властивість кластерів містити точки, що подібні між собою, але відрізняються від інших. Це визначається за рахунок того, що щільність в центрі

кластера є вищою ніж за його межами, якщо побудувати опуклу оболонку для всіх точок, що належать кластеру.

Найбільш відомим алгоритмом, що базується на понятті щільності є алгоритм DBSCAN (Density-Based Spatial Clusteing of Applications with Noise). Він дозволяє знаходити кластери довільної форми, враховуючи щільність записів. Вхідними параметрами алгоритму є радіус *eps* в якому враховується сусідство точок та мінімальну необхідну кількість сусідів *MinPts*.

Базовим поняттям DBSCAN є поняття основної точки, або точки ядра (core point). Основними точками називаються точки, в *eps* радіусі яких знаходиться більше ніж *MinPts* точок. Точка p_2 є безпосередньо досяжною згідно зі щільністю, відносно точки p_1, якщо точка p_1 є основною, а точка p_2 знаходяться в межах *eps* відносно неї. Точка p_2 є досяжною згідно зі щільністю, відносно точки p_1, якщо існує послідовність безпосередньо досяжних точок, що їх зв'язують. Точка, не досяжна згідно зі щільністю від жодної основної точки, називається викидом.

Алгоритм DBSCAN полягає в визначенні для кожної точки, до якої категорії вона належить, чи це основна точка, чи одна з точок, досяжних відносно них за щільністю, чи викид. Основні точки досяжні одна відносно іншої, разом з досяжними від них точками, формують окремі кластери. Решта точок вважається викидом і тому жодному кластеру не належить. Алгоритм послідовно аналізує точки, що ще не були віднесені до певної категорії. Якщо точка має достатню кількість сусідів, вона позначається, як основна, та аналізуються її сусіди, всі вони відносяться до того ж кластеру, що й основна точка, якщо сусідня точка також є основною, процедура повторюється для неї рекурсивно. Якщо точна не має сусідів, вона одразу ж позначається як викид. Головною перевагою даного методу є можливість ефективного застосування до наборів даних значного об'єму, адже щільність є локальною ознакою.

Для набору даних {(1,5),(2,4),(2,7),(2,9),(3,1),(5,5),(5,7),(6,6),(7,6),(10,6)}, що використовувався в попередніх розділах і зображено на рисунку 4.1.1, знайдемо основні точки для параметрів *eps*=2 *MinPts*=2. Для цього в матриці відстаней відмітимо всі комірки, що не перевищують *eps*, таким чином відмічаючи всіх сусідів для кожної точки. Результат представлено в таблиці 4.5.1.

Таблиця 4.5.1

	A	B	C	D	E	F	G	H	I	J
A	0	1.4	2.2	4.1	4.5	4	4.5	5.1	6.1	9.1
B	1.4	0	3	5	3.2	3.2	4.2	4.5	5.4	8.2
C	2.2	3	0	2	6.1	3.6	3	4.1	5.1	8.1
D	4.1	5	2	0	8.1	5	3.6	5	5.8	8.5
E	4.5	3.2	6.1	8.1	0	4.5	6.3	5.8	6.4	8.6
F	4	3.2	3.6	5	4.5	0	2	1.4	2.2	5.1
G	4.5	4.2	3	3.6	6.3	2	0	1.4	2.2	5.1
H	5.1	4.5	4.1	5	5.8	1.4	1.4	0	1	4
I	6.1	5.4	5.1	5.8	6.4	2.2	2.2	1	0	3
J	9.1	8.2	8.1	8.5	8.6	5.1	5.1	4	3	0

Не враховуючи комірки, що складають діагональ, знайдемо точки для яких відповідний їм рядок, або стовпчик містить не менше ніж *MinPts* відмічених комірок — це точки F, G, H. Дані точки будуть основними для заданих параметрів. Так як вони всі досяжні одна відносно іншої, то вони відносяться до одного кластеру, разом з усіма точками, досяжними від них згідно щільності — це точка I. Це означає, що згідно алгоритму DBSCAN для заданих параметрів, можна виділити лише один кластер — {F, G, H, I}, решта точок вважаються викидами. Якщо збільшити значення параметру *eps* до 2.5, кількість отриманих кластерів зросте, отримаємо додатково кластер {A, B, C, D}, проте точки E та J так і залишаться викидами.

4.6 Вправи до розділу

1. Для представлених нижче наборів даних побудувати дендрограму, застосувавши ієрархічні алгоритми Single-link, Complete-link та Average-link.
2. Для розбиття на 3 кластери, порівняти якість результатів отриманих різними алгоритмами в попередній задачі.
3. Для представлених нижче наборів даних провести кластеризацію, застосувавши методи k-Means та k-Medoid.
4. Використовуючи параметри *eps*=2.5, *MinPts*=2, за допомогою алгоритму DBSCAN, серед представлених нижче наборів даних, знайти основні точки, та виділити кластери.

Набори даних:

1. {(1,1),(1,8),(2,2),(2,5),(3,1),(4,3),(5,2),(6,1),(6,8),(8,6)}
2. {(1,1),(1,2),(1,5),(2,8),(3,7),(4,2),(7,5),(8,3),(8,7),(9,3)}
3. {(2,1),(2,4),(3,5),(3,6),(4,1),(4,9),(5,4),(5,6),(7,2),(9,8)}
4. {(1,4),(2,5),(2,8),(3,4),(3,5),(4,1),(4,7),(5,6),(7,6),(8,1)}

СПИСОК ЛІТЕРАТУРИ

1. Марченко О.О., Россада Т.В. Актуальні проблеми Data Mining: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. – Київ. – 2017. – 150 с.
2. Data Mining Curriculum. ACM SIGKDD: electronic resorce. URL: https://kdd.org/exploration_files/CURMay06.pdf, 2006. 10 p
3. Bowles M. Machine learning in Python: essential techniqes for predictive analysis: book. Indiana. John Wiley & Sons, 2015. 326 p.
4. Larose D.T., Larose C.D. Discovering knowledge in data: an introduction to Data Mining. Second edition: book. New Jersey. John Wiley & Sons, 2014. 316 p.
5. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. Third edition: book. Waltham. Morgan Kaufmann Publishers, 2012. 703p.
6. Kantardzic M. Data Mining. Concepts, Models, Methods, and Algorithms. Third edition: book. New Jersey. John Wiley & Sons, 2020. 639 p.
7. Witten I.H., Frank E. Data Mining. Practical Machine Learning Tools and Techniques. Second edition: book. San Francisco. Morgan Kaufmann Publishers, 2005. 525p.
8. Harrington P. Machine Learning in action: book. Shelter Island. Manning Publications, 2012. 354p.