УДК 004.04
DOI https://doi.org/10.24144/2616-7700.2025.46(1).149-165

Н. Бойко

Національний університет «Львівська політехніка»,
доцент кафедри системи штучного інтелекту,
канд екон. наук
nataliya.i.boyko@lpnu.ua
ORCID: https://orcid.org/0000-0002-6962-9363

ГРАФОВІ АЛГОРИТМИ КЛАСТЕРИЗАЦІЇ: ВИДІЛЕННЯ ЗВ'ЯЗКОВИХ КОМПОНЕНТ ТА ОПТИМІЗАЦІЯ МЕТОДІВ ГРУПУВАННЯ

У даній роботі досліджено методи кластеризації даних, зокрема алгоритм BFS (обхід у ширину) та його оптимізацію для підвищення ефективності. Кластеризація є важливим етапом аналізу великих обсягів інформації, що використовується у різних сферах, зокрема у медичних дослідженнях, аналізі ринкових тенденцій та машинному навчанні. Метою дослідження було виявлення сильних і слабких сторін BFS у задачах групування даних, а також розробка методів його покращення. Для цього було проаналізовано особливості різних підходів до кластеризації, визначено їхні переваги та недоліки, а також проведено порівняння BFS із іншими методами, такими як KNN (метод k найближчих сусідів). Результати дослідження показали, що BFS є ефективним для групування даних, особливо у випадках, коли об'єкти мають природну кластерну структуру. Основними перевагами алгоритму є його здатність працювати з розрідженими даними, обробляти аномальні значення (викиди) та легко адаптуватися до різних порогових значень. Однак основним недоліком залишається його обмежена масштабованість при обробці великих наборів даних. Проаналізовано що BFS у модифікованій формі є потужним інструментом для кластеризації, який може бути застосований у різних сферах аналізу даних. Однак подальші дослідження необхідні для розширення його можливостей, зокрема шляхом впровадження механізмів автоматичного налаштування параметрів, а також адаптації алгоритму для роботи з великими даними за допомогою розподілених обчислень.

Ключові слова: кластеризація, BFS, машинне навчання, оптимізація алгоритму, аналіз даних, викиди, порогове значення.

1. Вступ. У сучасну епоху цифрових технологій автоматизація процесів відіграє ключову роль у різних сферах діяльності. Від найпростіших математичних обчислень у калькуляторах до складних алгоритмів, що імітують людське мислення, кожне завдання вимагає чіткої програмної реалізації. Людина приймає рішення, спираючись на багаторічний досвід та сформовані нейронні зв'язки. Для того щоб навчити машину виконувати подібні дії, необхідно розробляти алгоритми, які дозволяють розбивати складні завдання на підзадачі [1, 4].

Один із поширених підходів до автоматизації — це класифікація та кластеризація даних. Ці методи широко застосовуються у різних галузях, зокрема в медицині, де необхідно групувати препарати, аналізи та дослідження. Особливо актуальним є аналіз даних, пов'язаних із пандемією COVID-19, що охопила весь світ і спричинила серйозні соціальні та медичні наслідки.

У даній роботі розглядається можливість застосування алгоритмів кластеризації для аналізу даних про перебіг хвороби. Дослідження ґрунтується на наборі даних, що містить інформацію про ускладнення, які перенесли пацієнти

з COVID-19. Основне завдання – здійснити кластеризацію пацієнтів за ступенем тяжкості перебігу захворювання з використанням графових алгоритмів. Особливу увагу буде приділено алгоритму пошуку в ширину (BFS) та його ефективності для даного типу задач [1, 14].

Метою роботи є експериментальне визначення ефективності використання алгоритму BFS для кластеризації записів у датасеті, що містить дані про перебіг COVID-19.

Для досягнення поставленої цілі в роботі розглядаються певні завдання дослідження:

1. Ознайомитися з існуючими методами кластеризації та їх застосуванням у медичній сфері.
2. Дослідити графові алгоритми кластеризації та оцінити їх переваги.
3. Проаналізувати можливості використання алгоритму BFS для групування даних.
4. Виконати експериментальне дослідження кластеризації пацієнтів за складністю перебігу COVID-19.
5. Оцінити результати кластеризації та зробити висновки щодо доцільності використання BFS у таких задачах.

2. Аналіз літературних джерел. Основні концепції кластеризації розглянуті Володимиром Естівілл-Кастро у своїх дослідженнях. Автор зазначає, що різноманітність алгоритмів кластеризації пояснюється неоднозначністю самого поняття «кластер». Оскільки очікуваний результат кластеризації залежить від конкретної задачі, існує багато моделей і підходів, які можуть бути ефективними для одних випадків, але не працювати в інших [4, 6].

Еніл Джейн сформулював загальноприйняті вимоги до кластеризації, згідно з якими:

1. Об'єкти всередині одного кластера повинні бути максимально схожими.
2. Об'єкти різних кластерів повинні суттєво відрізнятися.
3. Вимірювання подібності має бути чітким і практично обґрунтованим.

Дослідники Руй Ксу та Дональд Вунш описують стандартний процес кластеризації, який складається з наступних етапів [5]:

1. Витяг і вибір найбільш репрезентативних ознак.
2. Розробка алгоритму кластеризації відповідно до специфіки задачі.
3. Оцінка отриманих результатів і перевірка коректності алгоритму.
4. Інтерпретація кластеризації для отримання практичних висновків.

Окремий напрям у кластеризації становлять графові алгоритми. Вперше ідея випадкових графів була запропонована Гілбертом у 1959 році. Він розглядав процес генерації рівномірного випадкового графа з n вершинами, у якому кожне можливе ребро додається з імовірністю p. У таких графах розподіл ступенів підпорядковується пуассонівському закону, а утворення щільних кластерів малоймовірне через рівномірний розподіл зв'язків [2, 3].

У подальших дослідженнях з'явилися узагальнення моделі Гілберта, такі як l-розподіл. У цій моделі граф формується з $n = l * k$ вершинами, розбитими на l груп по k елементів. Імовірність існування зв'язку між вершинами в межах однієї групи є вищою (p), ніж між вершинами з різних груп (q). Такі моделі використовуються для знаходження природних кластерів у графах [7].

МакШеррі розглядає графову кластеризацію також у контексті розфарбовування графів, що є важливим для вирішення проблеми розподілу ресурсів [8].

За словами Калулапаті, головною перевагою графових алгоритмів кластеризації є можливість ефективного моделювання взаємозв'язків між об'єктами. Графові методи кластеризації належать до методів навчання без учителя, оскільки вони не потребують заздалегідь відомих міток класів. Вони дозволяють виявляти природні структури в даних на основі певних міри подібності [9, 11].

Таким чином, літературний аналіз показує, що графові методи кластеризації є ефективними для групування даних у різних сферах, включаючи медицину. Вони можуть бути застосовані для аналізу перебігу захворювань, зокрема COVID-19, що робить їх перспективним інструментом для медичної аналітики.

3. Методи та засоби дослідження. Основний принцип кластеризації полягає в тому, що об'єкти всередині одного кластера мають бути максимально схожими між собою, тоді як об'єкти з різних кластерів повинні суттєво відрізнятися. Головною відмінністю цього методу від класифікації є те, що кількість груп не задається наперед, а формується в ході виконання алгоритму [12].

Процес кластеризації передбачає кілька ключових етапів:

– Вибір вибірки — визначення набору об'єктів, які підлягають кластеризації.
– Визначення характеристик — встановлення змінних, за якими оцінюються об'єкти, а за потреби — нормалізація їх значень.
– Обчислення схожості — визначення міри подібності між об'єктами.
– Застосування методу кластеризації — поділ об'єктів на групи відповідно до обраного алгоритму.
– Інтерпретація результатів — аналіз отриманих кластерів та їх візуалізація.

Після проведення первинного аналізу результати можуть бути уточнені шляхом зміни метрики або методу кластеризації для досягнення оптимального результату.

Щоб об'єднати об'єкти в групи, необхідно спочатку оцінити ступінь їх подібності. Для цього кожному об'єкту задається вектор характеристик, який може містити як числові, так і категорійні ознаки. У багатьох випадках перед обчисленням відстаней проводиться нормалізація значень, що дозволяє всі характеристики мати однаковий вплив на результати кластеризації [13].

Окремий напрямок у кластеризації базується на теорії графів. У таких методах множина об'єктів представляється у вигляді графа, де вершини відповідають об'єктам, а ребра – зв'язкам між ними, вага яких відповідає мірам подібності. Графові методи дозволяють зручно візуалізувати структуру даних та мають високу гнучкість для роботи з різними типами об'єктів.

Суть таких алгоритмів у тому, що вибірка об'єктів представляється як графа $G = (V, E)$, вершинам якого відповідають об'єкти, а ребра мають вагу, рівний «відстанню» між об'єктами. Перевагою графових алгоритмів кластеризації є наочність, відносна простота реалізації та можливість внесення різних удосконалень, що ґрунтуються на геометричних міркуваннях. Основними алгоритмами є алгоритм виділення зв'язкових компонентів, алгоритм побудови мінімального покриваючого (остовного) дерева та алгоритм пошарової кластеризації [14].

Серед основних графових алгоритмів виділяють метод виділення зв'язкових компонентів — видаляє ребра, які перевищують певний поріг відстані, залишаючи тільки найближчі зв'язки. Внаслідок цього утворюються окремі компоненти, які й визначають кластери. Вибір відповідного порогу здійснюється на основі аналізу розподілу відстаней між об'єктами [15].

В алгоритмі виділення зв'язкових компонент задається вхідний параметр R і в графі видаляються всі ребра, для яких відстань більше R. Сполученими залишаються лише найближчі пари об'єктів. "Сенс алгоритму у тому, щоб підібрати таке значення R, яке буде у діапазоні всіх «відстаней», при яких граф «розвалиться» на кілька зв'язкових компонент". Отримані компоненти є кластери.

Для вибору параметра R зазвичай будується гістограма розподілів попарних відстаней. У завданнях з добре вираженою кластерною структурою даних на гістограмі буде два піки – один відповідає внутрішньо кластерним відстаням, другий — міжкластерним відстані. Параметр R підбирається із зони мінімуму між цими піками. У цьому керувати кількістю кластерів з допомогою порога відстані досить складно [10, 16].

Метод мінімального покриваючого дерева — на початковому етапі створюється мінімальне дерево, що охоплює всі вершини, після чого поступово видаляються найбільш довгі зв'язки, що дозволяє виокремити окремі групи. Наприклад, на графі, що містить дев'ять об'єктів, після побудови мінімального покриваючого дерева можна видалити зв'язок між двома найбільш віддаленими вершинами, що приведе до розбиття графа на два окремих кластери. Надалі цей процес можна продовжувати, поділяючи отримані групи на дрібніші.

Нехай задано множину об'єктів X, що складається з n елементів: $X = \{x_1, x_2, \ldots, x_n\}$, де кожен об'єкт x_i описується множиною характеристик, представлених у вигляді вектора у d-вимірному просторі: $x_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \in R^d$.

Завдання кластеризації полягає у знаходженні такого розбиття множини X на k підмножин (кластерів) C: $C = \{C_1, C_2, \ldots C_k\}$, де $C_i \subset X$ та виконуються наступні умови:

1. Повне покриття: $\bigcup_{i=1}^{k} C_i = X$, тобто кожен об'єкт має бути віднесений до якогось кластера.
2. Взаємна виключність (для чіткої кластеризації): $C_i \cap C_j = \varnothing$, $\forall i \neq j$, тобто кожен об'єкт належить лише одному кластеру (у випадку нечіткої кластеризації ця умова не виконується).
3. Оптимальність кластеризації: необхідно мінімізувати певну функцію вартості (критерій кластеризації), яка залежить від відстаней між об'єктами всередині кластерів.

У випадку використання графового підходу кластеризації, множину об'єктів X можна представити у вигляді неорієнтованого зваженого графа: $G = (V, E, W)$, де: $V = X$ — множина вершин (об'єкти), E — множина ребер між об'єктами, $W : E \to R^+$ — функція ваги, яка визначає міру подібності між об'єктами.

Завдання кластеризації у цьому випадку полягає у розбитті графа на підграфи $G_1, G_2, \ldots, G_k$, такі що:

1. Об'єкти всередині одного кластера мають бути тісно зв'язані (мінімальна

внутрішньокластерна відстань): $\sum_{e \in E(C_i)} W(e)$ — максимальна, де $E(C_i)$ — множина ребер всередині кластера C_i.

2. Об'єкти з різних кластерів мають бути слабко зв'язані (максимальна міжкластерна відстань):$\sum_{e \in E(C_i, C_j)} W(e)$ — мінімальна, для $i \neq j$, де $E(C_i, C_j)$ — множина ребер між кластерами C_i та C_j.

Одним з варіантів алгоритму є кластеризація за допомогою мінімального остовного дерева (MST), яка передбачає:

1. Побудову мінімального покриваючого дерева T для графа G.
2. Видалення найбільш довгих ребер у MST, що призводить до утворення підграфів, які стають кластерами.

Іншим підходом є кластеризація на основі зв'язності компонентів, де:

– вибирається поріг R,
– видаляються всі ребра з вагою $W(e) > R$,
– отримані зв'язні компоненти визначають кластери.

Таким чином, задача кластеризації може бути сформульована як оптимізаційна задача розбиття множини об'єктів на підмножини з урахуванням міри подібності та обраного критерію якості. Використання графових методів дозволяє ефективно враховувати структуру даних, що є особливо корисним для аналізу складних взаємопов'язаних об'єктів.

Реалізація графового алгоритму для вирішення задачі передбачає побудову окремих компонент зв'язності за таким алгоритмом, що наступний об'єкт потраплятиме в інший кластер, якщо деяка відстань до усіх об'єктів поточного кластеру переходить межу порогу. Тож задачу можна сформулювати наступним чином: побудуємо повний зв'язний граф для усіх вершин, а далі видалимо усі ребра, чия вага перевищує порогове значення. Таким чином наш повний граф розколеться на окремі компоненти зв'язності та кожна компонента являтиме собою окремий кластер.

Задача проста у реалізації, доки не постає питання ресурсозатратності. Такі операції потребують значних витрат як часу, так і пам'яті. Тому варто розглянути можливі варіанти оптимізації.

Перший крок — відмова від побудови повного графа, замінивши цей процес динамічним підходом. Замість створення всієї структури наперед, ми будемо працювати безпосередньо з датасетом, оновлюючи лише списки переглянутих і непереглянутих об'єктів. Це дозволить суттєво зменшити витрати ресурсів [9, 13].

Розглянемо, як працює цей алгоритм на прикладі вибірки даних A:

– Обираємо будь-який елемент a з $A[0]$ для перевірки.
– Визначаємо множину його сусідів і перевіряємо, чи відповідають вони заданому пороговому значенню.
– Додаємо знайдених сусідів у список для подальшої перевірки.
– Позначаємо поточний об'єкт як перевірений.
– Оновлюємо список ідентифікаторів, видаляючи всі елементи, які вже були відвідані або поставлені в чергу на перевірку.
– Обираємо наступний елемент для аналізу і повторюємо кроки 1–4.

Такий підхід реалізує алгоритм обходу графа в ширину, але без явного формування графової структури з визначеним набором ребер.

Попри те, що алгоритм уже виглядає оптимізованим, він усе ще має недоліки, які впливають на швидкість роботи. Основні проблеми виникають у двох випадках:

1. Коли спостерігається велике скупчення об'єктів у невеликій області.
2. Коли в датасеті є поодинокі викиди — значення, що значно відрізняються від інших.

Для подальшої оптимізації алгоритму впровадимо два додаткові кроки:

– Нормалізація даних — приведення значень до єдиної шкали для коректного аналізу.
– Фільтрація за процентилем — вибір тільки тих значень, які перевищують певний поріг.

Запускаючи алгоритм на модифікованому наборі даних, відфільтруємо два типи об'єктів: ті, що вже належать до кластеризованих груп та викиди, які значно відрізняються від загальної вибірки. У кожному з цих випадків алгоритм відноситиме об'єкт до найближчого кластера.

Для перевірки коректності роботи алгоритму використаємо візуалізацію результатів на графіку.

Процес тестування складався з таких етапів:

– Візуалізація початкового набору даних (було використано кілька груп елементів, щоб оцінити, чи алгоритм правильно відтворить їхнє групування).
– Об'єднання всіх даних у єдиний датасет.
– Відображення сформованого датасету.
– Запуск основного алгоритму.
– Візуалізація отриманого результату.

Розглянемо три приклади тестування.

У першому з них створимо чотири набори точок у тривимірному просторі, які утворюватимуть сферичні групи.

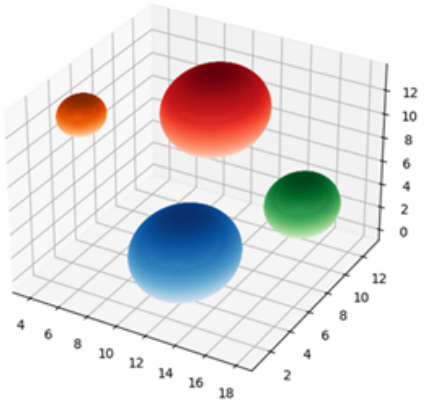
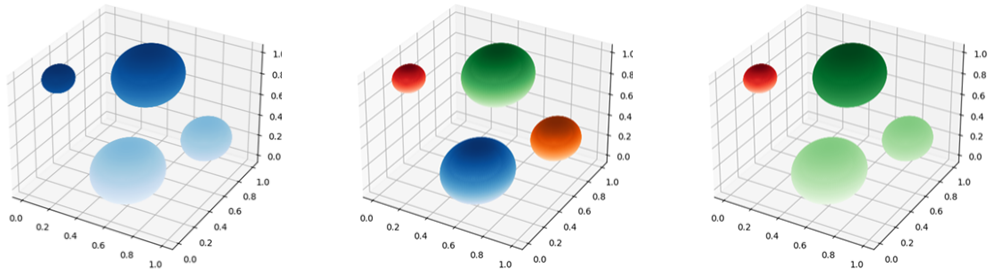


Рис. 1. Візуалізація вхідного датасету.

Кожна сфера налічує близько 10 000 тис точок. Тож величина цілого датасету становить $\sim$ 40 000 тис. елементів, що не є малим числом. У випадку формування повного графу, було отримано $\frac{n(n-1)}{2} =\sim 799\ 980\ 000 =\sim 10^9$ ребер. Тож тепер проробені кроки оптимізації доводять необхідність своєї присутності.

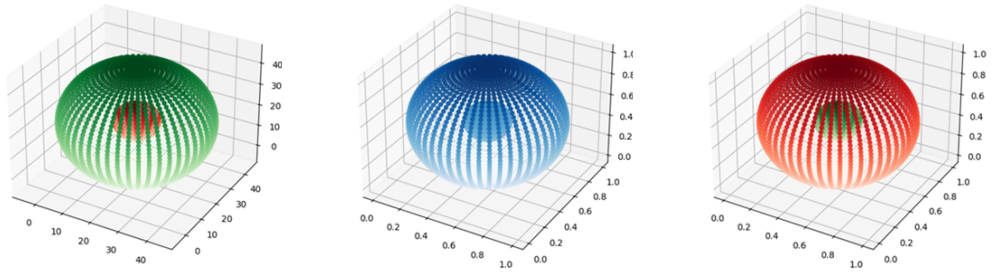


a) b) c)

Рис. 2. a) Візуалізація всього датасету; b) Візуалізація результату кластеризації датасету: з пороговим значенням рівним 1; c) — з пороговим значенням рівним 10.

На Рис. 2 можна спостерігати вплив заданого порогового значення на результат кластеризації. Можна стверджувати наступне, що зі зменшенням порогового значення кількість кластерів зростатиме.

У другому прикладі спробуємо протестувати цікавіший приклад, візьмемо ті ж сфери, проте помістимо одну всередину іншої.

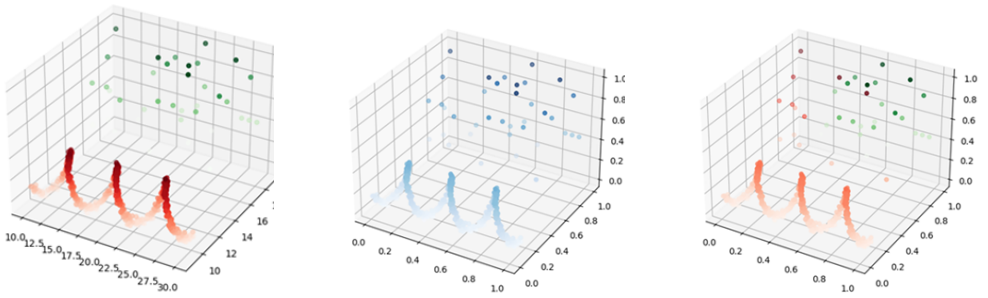


a) b) c)

Рис. 3. a) Візуалізація вхідного датасету; b) Візуалізація цільного датасету; c) Візуалізація результату кластеризації датасету.

Як можна побачити, алгоритм успішно виконує поставлену задачу. Оскільки основним критерієм групування є відстань між точками, вибране порогове значення відіграє ключову роль. У нашому випадку воно достатньо мале, щоб сформувати суцільну поверхню сфери як єдиний кластер, але водночас недостатньо велике, щоб об'єднати внутрішню і зовнішню сфери в один загальний кластер.

Останнім прикладом розглянемо випадок з кількома нетиповими елементами для датасету.



a) b) c)

Рис. 4. a) Візуалізація вхідного датасету; b) Візуалізація цільного датасету; c) Візуалізація результату кластеризації датасету.

Можна стверджувати, що алгоритм виправдав очікування.

4. Експерименти. Приступимо до покрокового виконання завдання, де для виконання задачі, було обрано датасет "Covid_19.xlsx" з онлайн ресурсу kaggle.com. Дослідимо даний набір даних на Рис. 5.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3308 entries, 2020-09-20 11:23:15 to 2021-04-05 03:18:16
Data columns (total 13 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Age                                   3303 non-null   object
 1   Gender                                3297 non-null   object
 2   Region                                3286 non-null   object
 3   Do you smoke?                         3269 non-null   object
 4   Have you had Covid`19 this year?      3305 non-null   object
 5   IgM level                             3232 non-null   object
 6   IgG level                             3248 non-null   object
 7   Blood group                           3252 non-null   float64
 8   Do you vaccinated influenza?          3294 non-null   object
 9   Do you vaccinated tuberculosis?       3303 non-null   object
 10  Have you had influenza this year?     3299 non-null   object
 11  Have you had tuberculosis this year?  3299 non-null   object
 12  Maximum body temperature              1533 non-null   float64
dtypes: float64(2), object(11)
```

Рис. 5. Інформація про датасет.

Як видно на Рис. 5 в наборі даних наявні поля, що містять інформацію про стать людини, її вік, максимальну температуру тіла, результат тесту на ковід антигени тощо. Більшість даних збережені у типі «object».

df

	Age	Gender	Region	Do you smoke?	Have you had Covid`19 this year?	IgM level	IgG level	Blood group	Do you vaccinated influenza?	Do you vaccinated tuberculosis?	Have you had influenza this year?	Have you had tuberculosis this year?	Maximum body temperature
Date time													
2020-09-20 11:23:15	40-65	Female (Жінка)	Ukraine, Lviv (Львів)	No	Maybe (можливо)	<0.9 (negative/ негативний)	0.9-1.1 (indefinable/ невизначений)	2.0	Yes	Yes	No	No	38.1
2020-09-20 11:23:15	23-40	Female (Жінка)	Ukraine, Chernivtsi	No	Yes	>1.1 (positive/ позитивний)	0.9-1.1 (indefinable/ невизначений)	2.0	No	Yes	No	No	37.0
2020-09-20 11:23:15	23-40	Female (Жінка)	Ukraine, Lviv (Львів)	No	Maybe (можливо)	<0.9 (negative/ негативний)	<0.9 (negative/ негативний)	2.0	No	Yes	Yes	No	37.8
2020-09-20 11:23:15	40-65	Male (Чоловік)	Ukraine, Chernivtsi	No	No	<0.9 (negative/ негативний)	<0.9 (negative/ негативний)	2.0	No	Yes	No	No	NaN

Рис. 6. Візуалізація перших рядків датасету.

Для того аби перейти до етапу кластеризації, для кожного пацієнта потрібно обчислити деяку величину, що визначатиме числове значення складності протікання коронавірусу. Реалізуємо алгоритм перетворення значень рядків в числові еквіваленти та виведемо перші декілька екземплярів на Рис. 7:

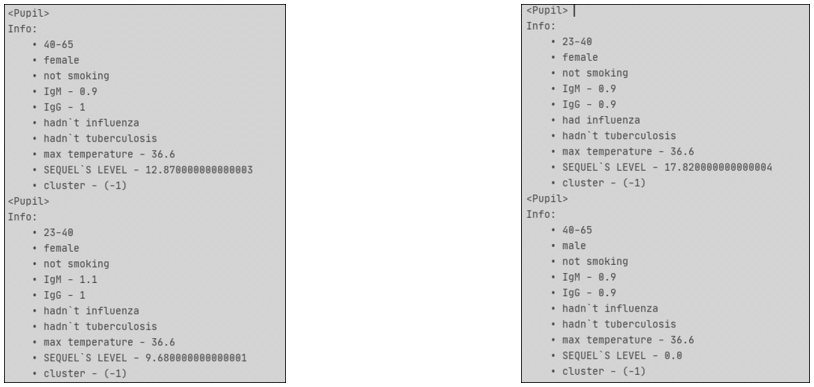


Рис. 7. Візуалізація опрацьованих записів датасету.

Тепер, маючи датасет із визначеними числовими рівнями захворюваності, можемо перейти до процесу кластеризації. На Рис. 7 представлено результати програмного аналізу записів датасету. Усі поля отримують певне числове значення, яке використовується у формулі для розрахунку SEQUEL's LEVEL — змінної, що визначає рівень тяжкості захворювання.

На початковому етапі всім екземплярам присвоюється кластер (-1), а в процесі кластеризації цей показник оновлюється відповідно до актуального кластерного номера. Запустимо алгоритм, представлений вище, на наборі даних для кластеризації записів (Рис. 8).

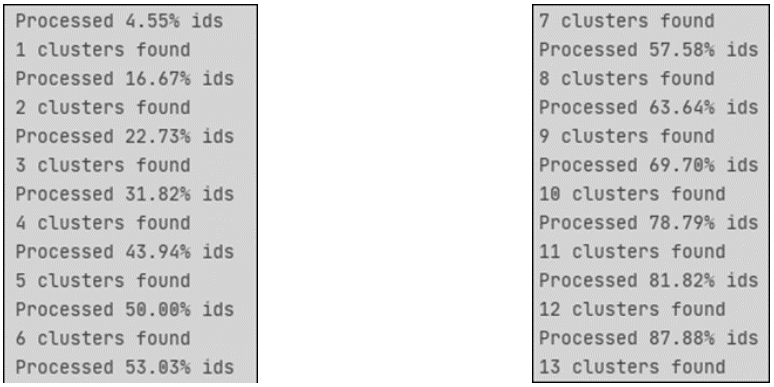


Рис. 8. Візуалізація консольного виводу результату кластеризації.

Як зображено на Рис. 8 з поточними вхідними параметрами порогового значення отримали 13 кластерів. Після виконання кластеризації формується новий датасет з двома додатковими колонками: рівнем складності захворювання та номером кластера приналежності. Виведемо оновлений датасет для огляду результату (Рис. 9).

Для дослідження роботи алгоритму використаємо різні вхідні порогові значення. При вхідних параметрах *D, tolerance, threshold = 1000, 3, 1* в результаті

```
df_processed = pd.read_csv('clustered_covid.csv')

df_processed.head(20)
```

	Date time	Age	Gender	Region	Do you smoke?	Have you had Covid`19 this year?	IgM level	IgG level	Blood group	Do you vaccinated influenza?	Do you vaccinated tuberculosis?	Have you had influenza this year?	Have you had tuberculosis this year?	Maximum body temperature	SEQUEL`S LEVEL	Cluster
0	2020-09-20 11:23:15	40-65	Female (Жінка)	Ukraine, Lviv (Львів)	No	Maybe (можливо)	<0.9 (negative/ негативний)	0.9-1.1 (indefinable/ невизначений)	2.0	Yes	Yes	No	No	38.1	12.8700	4
1	2020-09-20 11:23:15	23-40	Female (Жінка)	Ukraine, Chernivtsi	No	Yes	>1.1 (positive/ позитивний)	0.9-1.1 (indefinable/ невизначений)	2.0	No	Yes	No	No	37.0	9.6800	2
2	2020-09-20 11:23:15	23-40	Female (Жінка)	Ukraine, Lviv (Львів)	No	Maybe (можливо)	<0.9 (negative/ негативний)	<0.9 (negative/ негативний)	2.0	No	Yes	Yes	No	37.8	17.8200	5
3	2020-09-20 11:23:15	40-65	Male (Чоловік)	Ukraine, Chernivtsi	No	No	<0.9 (negative/ негативний)	<0.9 (negative/ негативний)	2.0	No	Yes	No	No	NaN	0.0000	1
4	2020-09-20 11:23:15	16-22	Male (Чоловік)	Ukraine, Lviv (Львів)	No	Yes	>1.1 (positive/ позитивний)	<0.9 (negative/ негативний)	1.0	No	Yes	Yes	No	37.4	14.8500	4
5	2020-09-20 11:23:15	16-22	Male (Чоловік)	Ukraine, Lviv (Львів)	No	No	<0.9 (negative/ негативний)	<0.9 (negative/ негативний)	2.0	Yes	Yes	No	No	NaN	0.0000	1
6	2020-09-20 11:23:15	16-22	Male (Чоловік)	Ukraine, Lviv (Львів)	No	No	<0.9 (negative/ негативний)	<0.9 (negative/ негативний)	1.0	No	No	No	No	NaN	0.0000	1
7	2020-09-20 11:23:15	23-40	Female (Жінка)	Ukraine, Lviv (Львів)	No	Maybe	<0.9 (negative/ негативний)	0.9-1.1 (indefinable/ невизначений)	1.0	No	Yes	Maybe (можливо)	No	38.3	19.8000	6

Рис. 9. Візуалізація перших рядків кластеризованого датасету.

одержали 14 кластерів. Спробуємо змінити їх на *D, tolerance, threshold = 100, 3, 1.*

```
Processed 68.18% ids
1 clusters found
Processed 77.27% ids
2 clusters found
Processed 86.36% ids
3 clusters found
```

Рис. 10. Візуалізація консольного виводу результату кластеризації.

Тепер змінимо наступним чином: *D, tolerance,threshold = 1000, 3, 3.*

```
Processed 6.06% ids
1 clusters found
Processed 50.00% ids
2 clusters found
Processed 53.03% ids
3 clusters found
Processed 57.58% ids
4 clusters found
Processed 65.15% ids
```

```
5 clusters found
Processed 71.21% ids
6 clusters found
Processed 78.79% ids
7 clusters found
Processed 81.82% ids
8 clusters found
Processed 87.88% ids
9 clusters found
```

Рис. 11. Візуалізація консольного виводу результату кластеризації.

При першій зміні отримали 3 кластери, при наступній 9. Як видно на Рис. 11, кластеризація відбувається поступово. Відсоток опрацьованих записів датасету збільшується, охоплюючи все більшу кількість рядків. Також прослідковується

наступна тенденція поділу: при збільшенні порогового значення, кількість кластерів зменшується.

Проаналізуємо результати прискорення, яких вдалося досягти за допомогою оптимізації, описаної в частині програмної реалізації. На етапі графової кластеризації будуть брати участь лише та частина записів, які після нормалізації даних є представниками скупчення максимально подібних екземплярів.

Даний алгоритм графової кластеризації має оцінку таку ж як і алгоритм BFS. Складність стандартного обходу в ширину — $O(n+m)$, де n — кількість вершин (еквівалент кількості записів), m — кількість ребер у графі. З урахуванням того що звичайний датасет на початку формує повний граф, то m можна прирівняти до $\frac{n(n-1)}{2}$. А отже загальну складність графової кластеризації доцільно представити у вигляді — $O(n^2)$.

Тож, давайте виведемо наступну інформацію у термінал: кількість вхідних записів, кількість записів, що братимуть участь в етапі графової кластеризації.

```
Record`s number: 3308
Records for clustering: 66
```

Рис. 12. Скрін-фіксація кількості записів для опрацювання/кластеризації.

Як видно на Рис. 12 на вхід було отримано набір даних розмірністю в 3308 рядків, проте на етап кластеризації потрапило лише 66. Порівняємо обчислювальну складність, передбачалось $O(n^2) = 3308^2 \sim 10\ 942\ 864 \sim 10^7$, а в результаті оптимізації отримали $O(n^2) = 66^2 \sim 4\ 356 \sim 10^3$.

Ефективність прооблених кроків є очевидною і при таких даних, проте для більш об'єктивної оцінки згенеруємо штучний датасет розмірністю 100 000 та зафіксуємо результати (Рис. 13).

```
1 clusters found
Processed 45.45% ids
2 clusters found
Processed 49.20% ids
3 clusters found
Processed 53.48% ids
4 clusters found
Processed 55.61% ids
5 clusters found
Processed 57.22% ids
```

```
6 clusters found
Processed 58.29% ids
7 clusters found
Processed 60.43% ids
8 clusters found
Processed 62.03% ids
9 clusters found
Processed 64.17% ids
10 clusters found
Processed 65.78% ids
```

```
11 clusters found
Processed 68.45% ids
12 clusters found
Processed 70.05% ids
13 clusters found
Processed 72.19% ids
14 clusters found
Processed 74.33% ids
15 clusters found
Processed 75.94% ids
```

```
16 clusters found
Processed 78.07% ids
17 clusters found
Processed 80.75% ids
18 clusters found
Processed 83.42% ids
19 clusters found
Processed 87.70% ids
20 clusters found
```

Рис. 13. Візуалізація консольного виводу результату кластеризації.

Як зображено на Рис. 13 наведену кількість даних кластеризувало на 20 окремих кластерів. Перейдемо до складності:

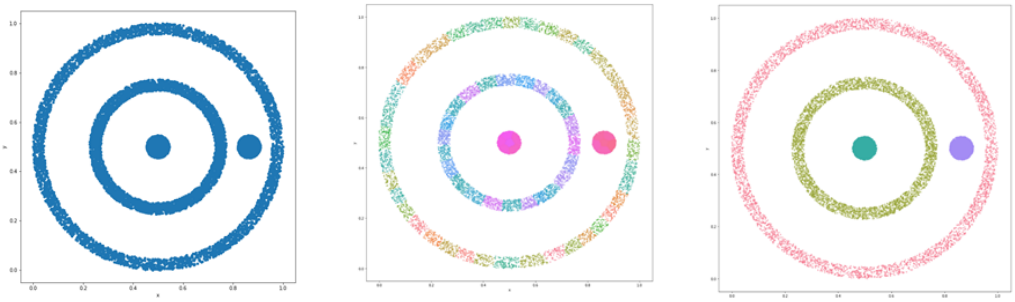
```
Record`s number: 100000
Records for clustering: 187
```

Рис. 14. Скрін-фіксація кількості записів для опрацювання/кластеризації.

Тож як зафіксовано на Рис. 14 з 100000 записів до алгоритму перейшло лиш 187. Оцінимо складність, передбачалось $O(n^2) = 100\ 000^2 \sim 10^{10}$, а в результаті

оптимізації було отримали $O(n^2) = 187^2 \sim 34\ 969 \sim 10^5$. Отримані результати явно свідчать про доцільність застосування реалізованої оптимізації.

5. Результати. Проаналізуємо результати експериментів загальних методів кластеризації наведених вище.



a) b) c)

Рис. 15. a) Візуалізація вхідних даних, конкатенованих в суцільний датасет; b) Результат кластеризації Affinity Propagation; c) Результат кластеризації на BFS.

На Рис. 15 представлено вхідні дані для застосування алгоритму, як приклад один з методів кластеризації та безпосередньо кластеризація на BFS.

Усі застосовані алгоритми дали різні результати, це цілком пояснюється тим, що кожен з них при створенні був спрямований на вирішення якоїсь конкретної моделі. Деякі алгоритми застосовуються для посереднього групування, на вхід потребують початкові дані наповнення кластерів та використовують їх для подальших кроків, деяким потрібно задати вхідну кількість кластерів тощо.

Можна зробити висновок, що в ході постановки конкретної задачі, можна визначити тип очікуваного результату, та, відштовхуючись від нього, підбирати алгоритм.

У даному випадку, коли задачею є групування в окремі кластера екземплярів з мінімальною відстанню між собою та відсутністю будь-якого вхідного розподілу, ідеально вписується використання алгоритму графової кластеризації. Так як саме цей алгоритм надає можливість оптимально розподілити об'єкти на групи по наближенню їх числового значення.

Аналізуючи результати отримані при проведенні експериментів, в першу чергу хочеться зазначити, що алгоритм таки справився з поставленою задачею. Проаналізуємо вплив вхідних параметрів на результат (Табл. 1).

Як можна спостерігати в Табл. 1, в залежності від вхідних параметрів, а саме величини порогового значення, одержували різні результати. Чим менше порогове значення надавалось, тим дрібніший поділ на кластера був отриманний. Можна стверджувати, що параметр порогового значення є прямим об'єктом впливу на подрібнення результату кластеризації.

Якщо говорити про ефективність проведеної оптимізації, щодо даного алгоритму, то було досягнуто високої швидкодії виконання (Табл. 2).

Тож за отриманими результатами в Табл. 2, можна відстежити значне пришвидшення роботи алгоритму, а при опрацюванні наборів даних великих масштабів — це є важливим критерієм.

Варто зазначити й те, що даний алгоритм підлягає удосконаленню. А саме,

Таблиця 1.

Таблиця залежності результату від вхідних параметрів

К-ть записів	Вхідні параметри			Отриманий результат
N	D	Tolerance	threshold	Кількість кластерів
3308	1000	3	1	14
3308	1000	3	4	3
3308	1000	3	3	9
100 000	1000	3	3	20
100 000	1000	3	1	34

Таблиця 2.

Порівняння алгоритмів по складності по часу виконання

К-ть записів	Очікувана складність звичайної BFS кластеризації	Складність оптимізованого алгоритму
3308	10^7	10^3
100 000	10^{10}	10^5
100 000 000	10^{16}	10^8

його можна імплементувати до алгоритму оптимального підбору порогового значення, що зробить цей алгоритм абсолютно самостійним. Тобто на вхід даватиметься лише набір деяких екземплярів для кластеризації, а вже алгоритм групуватиме їх за різними пороговими значеннями та оцінювати результат поточного значення. В залежності від того, порогове значення змінюватиметься, та буде відбуватися його оцінка, і так до моменту отримання оптимального результату.

Обговорення результатів. У процесі дослідження було проведено детальний аналіз алгоритмів кластеризації та їх застосування в задачах групування даних. Основний акцент був зроблений на використанні BFS як базового алгоритму кластеризації, зокрема його модифікації для підвищення ефективності. Однією з ключових знахідок стало застосування додаткової оптимізації за допомогою алгоритму KNN, що дозволило покращити точність і швидкодію.

Отримані результати підтвердили, що BFS добре справляється із завданнями групування, особливо у випадках, коли дані мають чітко виражені кластери. Водночас, алгоритм демонструє певні обмеження при роботі з високорозмірними даними або з нерівномірним розподілом кластерів.

Важливою перевагою представленого підходу є його здатність ефективно працювати з поодинокими точками-викидами, що є суттєвою проблемою для багатьох методів кластеризації. Використання додаткового етапу нормалізації та фільтрації даних сприяло точнішій ідентифікації таких об'єктів та їх коректній класифікації.

Оптимізація BFS дозволила зменшити часові витрати на виконання кластеризації, однак залишаються випадки, коли алгоритм може втрачати ефективність, зокрема при роботі з великими наборами даних. Використання адаптивного порогового значення могло б додатково покращити продуктивність.

Результати дослідження відкривають кілька напрямів для подальшого вдосконалення методу:

1. Розширення алгоритму для роботи з великими обсягами даних. Незважаючи на оптимізацію, BFS все ще має певні обмеження в масштабованості. Дослідження можливостей паралельної обробки або використання розподілених обчислень може значно підвищити продуктивність.
2. Інтеграція з іншими методами машинного навчання. Поєднання алгоритмів кластеризації з нейронними мережами або методами глибокого навчання може покращити якість аналізу даних та їхню адаптивність до складних структур.
3. Автоматичний підбір порогового значення. Наразі вибір порогового значення залишається евристичним. Розробка методу для автоматичного визначення оптимального порогу дозволила б значно покращити точність кластеризації та її універсальність.
4. Аналіз ефективності на реальних даних. Використання алгоритму для кластеризації медичних або соціальних даних може допомогти оцінити його практичну цінність та адаптувати методику до реальних сценаріїв.

Загалом, дослідження підтвердило ефективність модифікованого алгоритму BFS у задачах кластеризації. Оптимізація дозволила покращити швидкість роботи та якість кластеризації, проте залишаються відкриті питання щодо його масштабованості та автоматичного налаштування параметрів. Подальші дослідження у цьому напрямі можуть зробити метод ще більш гнучким та ефективним у широкому спектрі застосувань.

6. Висновки. В ході дослідження обраної теми було проаналізовано різні алгоритми кластеризації. На основі даних досліджень можна зробити наступні висновки.

По-перше, в жодному випадку не можна виділити один найкращий алгоритм. Кожен з існуючих вартий уваги і знаходить своє застосування виходячи з поставлених вимог та обмежень.

Для детального розбору та аналізу був обраний алгоритм кластеризації BFS. Як було описано детально в програмній реалізації самого алгоритму, було частково модифіковано основний алгоритм, в першу чергу для загальної оптимізації швидкодії кластеризації. До того ж друга частина оптимізації звертається до алгоритму кластеризації KNN. Це доводить про тісний взаємозв'язок між алгоритмами.

Якщо ж аналізувати даний спосіб реалізації, то було отримано швидкий та якісний алгоритм, який знає як справлятися з викидами та легко абстрагується до різних порогових значень. Тож якщо коротко, то:

– Графовий алгоритм кластеризації оптимально справляється з групуванням записів в наборі даних.
– З використанням оптимізації методу, можна досягти високої швидкодії алгоритму.
– Можлива оптимізація кластеризації на BFS — імплементація алгоритму для оптимального підбору вхідного порогового значення.

Отже, проведене дослідження підтверджує ефективність запропонованого методу, а також демонструє можливості його адаптації до різних типів задач

кластеризації. Отже, проведене дослідження підтверджує ефективність запропонованого методу, а також демонструє можливості його адаптації до різних типів задач кластеризації.

Подяка. Дослідження створено в рамках проекту, що фінансується Національним фондом досліджень України, зареєстрованим № 2021.01/0103 «Методи та засоби дослідження маркерів старіння та їх впливу на постковідні ефекти для подовження працездатного періоду», який виконується на кафедрі систем штучного інтелекту Інституту комп'ютерних наук та інформаційних технологій Національного університету «Львівська політехніка».

Список використаної літератури

1. Sarker A., Shamim S. M., Shahiduz Zama Dr. Md., Mustafizur Rahman Md. Employee's performance analysis and prediction using K-means clustering & decision tree algorithm. *Global Journal of Computer Science and Technology.* 2018. Vol. 18. pp 1–4.
2. Saxena A., Prasad M., Gupta A., Bharill N., Patel O. P., Tiwari A., and Lin C. T. A review of clustering techniques and developments. *Neurocomputing.* 2017. Vol. 26. pp. 664–681.
3. Hossain M. Z., Akhtar M. N., Ahmad R. B., and Rahman M. A dynamic K-means clustering for data mining. *Indonesian Journal of Electrical Engineering and Computer Science.* 2016. Vol. 13, No. 2. pp. 521–526. DOI: https://doi.org/10.11591/ijeecs.v13.i2.pp521-526
4. Yim O., Ramdeen K. T. Hierarchical Cluster Analysis: Comparison of Three Linkage Measures and Application to Psychological Data. *The Quantitative Methods for Psychology.* 2015. Vol. 11, No. 1. pp. 8–21. DOI: https://doi.org/10.20982/tqmp.11.1.p008
5. Sneath P., Sokal R. Hierarchical Cluster Analysis: Comparison of Three Linkage Measures and Application to Psychological Data. *The Quantitative Methods for Psychology.* 2015. Vol. 11, No. 1. pp. 8–21. DOI: https://doi.org/10.20982/tqmp.11.1.p008
6. Fraley C., Raftery A. E. How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. *Technical Report. Department of Statistics University of Washington.* 1998. Vol. 41, No. 8. pp. 578–588. DOI: https://doi.org/10.1093/comjnl/41.8.578
7. Murtagh F. A survey of recent advances in hierarchical clustering algorithms which use cluster centers. *Computer Journal.* 2020. Vol. 26, No. 4, pp. 354–359.
8. Ptitsyn A., Hulver M., Cefalu W., York D., and Smith S. R. Unsupervised clustering of gene expression data points at hypoxia as possible trigger for metabolic syndrome. *BMC Genomics.* 2016. Vol. 7, No. 1. 318. DOI: https://doi.org/10.1186/1471-2164-7-318
9. Tung A. K., Hou J., Han J. Spatial clustering in the presence of obstacles. *Proceedings 17th International Conference on Data Engineering :* IEEE. Heidelberg : Germany, 02–06 April, 2001. pp. 359–367. DOI: https://doi.org/10.1109/ICDE.2001.914848
10. Boehm C., Kailing K., Kriegel H., Kroeger P. Density connected clustering with local subspace preferences. *Fourth IEEE International Conference on Data Mining (ICDM'04) :* IEEE. Brighton : UK, 01–04 November, 2004. pp. 27–34. DOI: https://doi.org/10.1109/ICDM.2004.10087
11. Boyko N., Hetman S., Kots I. Comparison of Clustering Algorithms for Revenue and Cost Analysis. *Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2021).* Lviv : Ukraine. 2021. Vol. 1. pp. 1866–1877. URL: https://ceur-ws.org/Vol-2870/paper136.pdf (date of access: 09.12.2024).
12. Procopiuc C. M., Jones M., Agarwal P. K., Murali T. M. A Monte Carlo algorithm for fast projective clustering. *ACM SIGMOD Intern. conf. on management of data.* Madison, Wisconsin : USA, 4–6 June, 2002. pp. 418–427.
13. Boyko N. Application of mathematical models for improvement of "cloud" data processes organization. *Mathematical Modeling and Computing.* 2016. Vol. 3, No. 2. pp. 111–119. DOI: https://doi.org/10.23939/mmc2016.02.111
14. Slamet C., Rahman A., Ramdhani M. A., and Darmalaksana W. Clustering the verses of the Holy Qur'an using K-means algorithm. *Asian Journal of Information Technology.* 2016. Vol. 15, No. 24. pp. 5159–5162. DOI: http://doi.org/10.3923/ajit.2016.5159.5162
15. Boyko N., Kmetyk-Podubinska K., Andrusiak I. Application of Ensemble Methods of Strengthening in Search of Legal Information. *Lecture Notes on Data Engi-*

neering and Communications Technologies. 2021. Vol. 77. pp. 188–200. DOI: https://doi.org/10.1007/978-3-030-82014-5_13
16. Bekiros S., Nguyen D. K., Sandoval J. L., and Uddin G. S. Information diffusion, cluster formation and entropy-based network dynamics in equity and commodity markets. *European Journal of Operational Research.* 2017. Vol. 256, No. 3. pp. 945–961. DOI: https://doi.org/10.1016/j.ejor.2016.06.052

Boyko N. Graph-Based Clustering Algorithms: Connected Component Detection and Optimization of Grouping Methods.

This paper explores data clustering methods, focusing on the BFS (Breadth-First Search) algorithm and its optimization to enhance efficiency. Clustering is a crucial step in analyzing large volumes of data and is widely used in various fields, including medical research, market trend analysis, and machine learning. The main objective of the study was to identify the strengths and weaknesses of BFS in data grouping tasks and develop methods for its improvement. To achieve this, different clustering approaches were analyzed, their advantages and disadvantages were identified, and BFS was compared with other methods, such as KNN (k-Nearest Neighbors). The research results demonstrate that BFS is effective for data grouping, especially when objects exhibit a natural cluster structure. The key advantages of the algorithm include its ability to work with sparse data, handle outliers, and easily adapt to different threshold values. However, its primary limitation is its scalability when processing large datasets. The analysis confirms that the modified BFS is a powerful clustering tool applicable in various data analysis domains. However, further research is required to expand its capabilities, particularly by developing mechanisms for automatic parameter tuning and adapting the algorithm for large-scale data processing through distributed computing.

Keywords: clustering, BFS, machine learning, algorithm optimization, data analysis, outliers, threshold value.

References

1. Sarker, A., Shamim, S. M., Shahiduz Zama, Dr. Md., & Mustafizur Rahman Md. (2018). Employee's performance analysis and prediction using K-means clustering & decision tree algorithm. *Global Journal of Computer Science and Technology,* 18, 1–4.
2. Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., & Lin, C. T. (2017). A review of clustering techniques and developments. *Neurocomputing,* 26. 664–681.
3. Hossain, M. Z., Akhtar, M. N., Ahmad, R. B., & Rahman, M. (2016). A dynamic K-means clustering for data mining. *Indonesian Journal of Electrical Engineering and Computer Science,* 13(2), 521–526. https://doi.org/10.11591/ijeecs.v13.i2.pp521-526
4. Yim, O., & Ramdeen, K. T. (2015). Hierarchical Cluster Analysis: Comparison of Three Linkage Measures and Application to Psychological Data. *The Quantitative Methods for Psychology,* 11(1), 8–21. https://doi.org/10.20982/tqmp.11.1.p008
5. Sneath, P., & Sokal, R. (2015). Hierarchical Cluster Analysis: Comparison of Three Linkage Measures and Application to Psychological Data. *The Quantitative Methods for Psychology.* 11(1), 8–21. https://doi.org/10.20982/tqmp.11.1.p008
6. Fraley, C., & Raftery, A. E. (1998). How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. *Technical Report. Department of Statistics University of Washington,* 41(8), 578–588. https://doi.org/10.1093/comjnl/41.8.578
7. Murtagh, F. (2020). A survey of recent advances in hierarchical clustering algorithms which use cluster centers. *Computer Journal,* 26(4), 354–359.
8. Ptitsyn, A., Hulver, M., Cefalu, W., York, D., & Smith, S. R. (2016). Unsupervised clustering of gene expression data points at hypoxia as possible trigger for metabolic syndrome. *BMC Genomics,* 7(1), 318. https://doi.org/10.1186/1471-2164-7-318
9. Tung, A. K., Hou, J., & Han, J. (02–06 April, 2001). Spatial clustering in the presence of obstacles. In *Proceedings 17th International Conference on Data Engineering.* Heidelberg: Germany. https://doi.org/10.1109/ICDE.2001.914848

10. Boehm, C., Kailing, K., Kriegel, H., & Kroeger, P. (01–04 November, 2004). Density connected clustering with local subspace preferences. In *Fourth IEEE International Conference on Data Mining (ICDM'04).* Brighton: UK. https://doi.org/10.1109/ICDM.2004.10087
11. Boyko, N., Hetman, S., & Kots, I. (2021). Comparison of Clustering Algorithms for Revenue and Cost Analysis. In *Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2021).* Lviv: Ukraine. Retrieved from https://ceur-ws.org/Vol-2870/paper136.pdf
12. Procopiuc, C. M., Jones, M., Agarwal, P. K., & Murali, T. M. (4–6 June, 2002). A Monte Carlo algorithm for fast projective clustering. In *ACM SIGMOD Intern. conf. on management of data.* Madison, Wisconsin: USA.
13. Boyko, N. (2016). Application of mathematical models for improvement of "cloud" data processes organization. *Mathematical Modeling and Computing,* 3(2), 111–119. https://doi.org/10.23939/mmc2016.02.111
14. Slamet, C., Rahman, A., Ramdhani, M. A., & Darmalaksana, W. (2016). Clustering the verses of the Holy Qur'an using K-means algorithm. *Asian Journal of Information Technology,* 15(24), 5159–5162. http://doi.org/10.3923/ajit.2016.5159.5162
15. Boyko, N., Kmetyk-Podubinska, K., & Andrusiak, I. (2021). Application of Ensemble Methods of Strengthening in Search of Legal Information. *Lecture Notes on Data Engineering and Communications Technologies,* 77, 188–200. https://doi.org/10.1007/978-3-030-82014-5_13
16. Bekiros, S., Nguyen, D. K., Sandoval, J. L., & Uddin, G. S. (2017). Information diffusion, cluster formation and entropy-based network dynamics in equity and commodity markets. *European Journal of Operational Research,* 256(3), 945–961. https://doi.org/10.1016/j.ejor.2016.06.052

Одержано 14.02.2025