ДОДАТОК А

Посібник користувача

ЗМІСТ

ВСТУП

Даний посібник користувача створений для ознайомлення з функціональними можливостями програмного продукту Routers, який розроблено з метою оптимізації побудови маршрутів переміщення вантажів.

Основна задача програми – забезпечення ефективного управління логістичними процесами шляхом інтеграції сучасних методів кластеризації даних та технологій автоматизації. Це дозволяє мінімізувати витрати ресурсів, часу та підвищити продуктивність підприємства.

Routers призначений для:

- побудови оптимальних маршрутів перевезення вантажів з урахуванням географічних даних, обмежень та пріоритетів;
- автоматизації аналізу логістичних даних і прийняття рішень;
- візуалізації маршрутів та кластерів на інтерактивній карті;
- підтримки оперативного планування та управління логістикою на підприємстві.

У посібнику представлено:

- опис призначення додатку та його ключових функцій;
- вимоги до апаратного та програмного забезпечення для коректної роботи програми;
- інструкції з налаштування та використання основних функцій;
- рекомендації для ефективної роботи з програмою.

Routers розроблено на основі сучасних алгоритмів машинного навчання та методів кластеризації, що забезпечує високу точність моделювання маршрутів. У разі виникнення питань або труднощів під час роботи з додатком користувачі можуть звернутися до даного посібника для отримання необхідної інформації.

Цей документ створений для того, щоб зробити взаємодію з програмою максимально зручною, забезпечуючи чіткі покрокові рекомендації для її використання.

А. 1 Призначення додатку

Додаток Routers призначений для автоматизації процесу побудови оптимальних маршрутів перевезення вантажів на основі логістичних даних підприємства. Основна мета програми – забезпечення ефективного управління логістикою за допомогою сучасних технологій аналізу даних і автоматизації.

Програма виконує такі функції:

– оптимізація маршрутів, використання алгоритмів машинного навчання для створення оптимальних шляхів перевезення з урахуванням географічних, часових і ресурсних обмежень;

– кластеризація вантажів і точок доставки: аналіз даних для групування пунктів призначення, що дозволяє мінімізувати загальну протяжність маршрутів;

– інтерактивна візуалізація, відображення побудованих маршрутів на карті для зручності навігації та контролю;

– інтеграція даних, сумісність із внутрішніми інформаційними системами підприємства для автоматичного завантаження і оновлення логістичних даних;

– моніторинг і аналіз, надання детальної статистики про виконані маршрути, витрати часу та ресурсів.

Додаток Routers є гнучким і масштабованим рішенням, яке підходить для підприємств різних масштабів, від малого бізнесу до великих логістичних компаній. Його застосування дозволяє зменшити витрати на транспортування, підвищити ефективність використання ресурсів та покращити якість обслуговування клієнтів.

А. 2 Технічні вимоги

Для коректної роботи додатку Routers необхідно дотримуватися наступних технічних вимог:

а) апаратне забезпечення:

1) процесор: не менше 2.4 ггц (4 ядра або більше);

2) оперативна пам’ять: мінімум 8 гб (рекомендовано 16 гб для роботи з великими обсягами даних);

3) місце на диску: мінімум 10 гб вільного простору для встановлення програми та збереження даних;

4) відеокарта: інтегрована або дискретна з підтримкою opengl 3.0 або вище (рекомендовано для інтерактивної візуалізації);

5) монітор: роздільна здатність не менше 1920x1080 пікселів;

б) програмне забезпечення:

1) операційна система: windows 10/11, macos 11+ або linux;

2) програмні компоненти:

– python 3.8 або новіше (з необхідними бібліотеками, які вказуються у документації додатку);

– база даних mysql або postgresql (залежно від налаштувань підприємства);

– браузер google chrome або microsoft edge (для інтерактивної візуалізації маршрутів);

3) додаткові інструменти:

– інстальовані бібліотеки для машинного навчання та кластеризації (наприклад, scikit-learn, numpy, pandas);

– інструменти для роботи з картографічними даними (google maps api або openstreetmap);

в)мережеві вимоги:

1) інтернет-з’єднання: стабільне з мінімальною швидкістю 10 мбіт/с для доступу до картографічних сервісів і оновлення даних;

2) локальна мережа: підтримка tcp/ip-протоколу для інтеграції з внутрішніми інформаційними системами підприємства;

3) додаткові вимоги:

– обліковий запис google api: для використання інтеграції з google maps (за необхідності);

– сумісність даних: підтримуються формати csv, json, xml для імпорту та експорту логістичних даних;

– захист даних: встановлення антивірусного програмного забезпечення та налаштування брандмауера для забезпечення безпеки під час роботи.

Дотримання цих вимог забезпечить стабільну та продуктивну роботу додатку Routers. У разі виникнення технічних труднощів рекомендується звертатися до служби технічної підтримки.

А. 3 Опис основних функцій

Програма запускається з головного вікна, головне вікно програми – це вікно авторизації, рис. А.3.1.

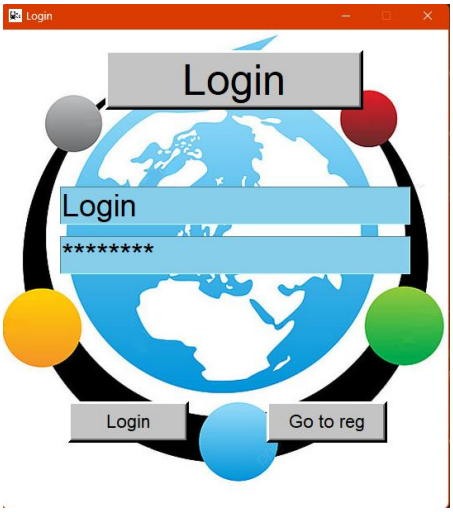


Рисунок А.3.1 – Результат роботи програми (вікно авторизації)

Далі є можливість зареєструватись, треба натиснути на кнопку «Go to reg» та перейти до вікна реєстрації, рисунок А.3.2.

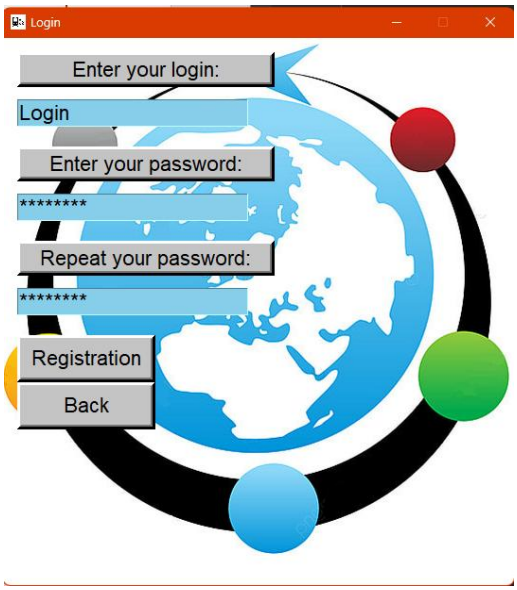


Рисунок А.3.2 – Результат роботи програми (вікно реєстрації)

У вікні реєстрації потрібно ввести свій логін, пароль та підтвердити пароль. Введений логін повинен мати лише латинські літери та цифри, якщо спробувати ввести інші символи, то отримаєте відповідне повідомлення. Також стоїть обмеження на введення символів, дозволяється ввести до 10 символів, рис. А.3.3.

Рисунок А.3.3 – Результат роботи програми (помилка введення логіну)

Таке ж саме обмеження використовується для паролю, тобто дозволяється вводити лише латинські символи та цифри довжиною не більше 10 символів, рисунок А.3.4.

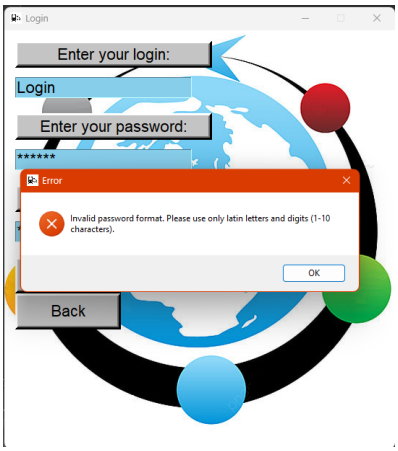


Рисунок А.3.4 – Результат роботи програми (помилка введення паролю)

Після введення паролю, необхідно його повторити, якщо паролі відрізняються, то виведеться відповідне повідомлення, рисунок А.3.5.

Рисунок А.3.5 – Результат роботи програми (помилка «паролі не співпадають»)

Якщо коректно заповнити всі поля, але використовувати логін, що вже існує то знов виведеться повідомлення, рис. А.3.6.

Далі, після реєстрації логіну і паролю, можна перейти до авторизації, де відбувається перевірка, чи існує введений логін, рис. А.3.7.

Якщо ввести існуючий вже логін та пароль, то виведеться повідомлення про успішний вхід в систему та відкриється вікно кластеризації, рис. А.3.8.

Рисунок А.3.6 – Результат роботи програми (помилка «логін вже існує»)

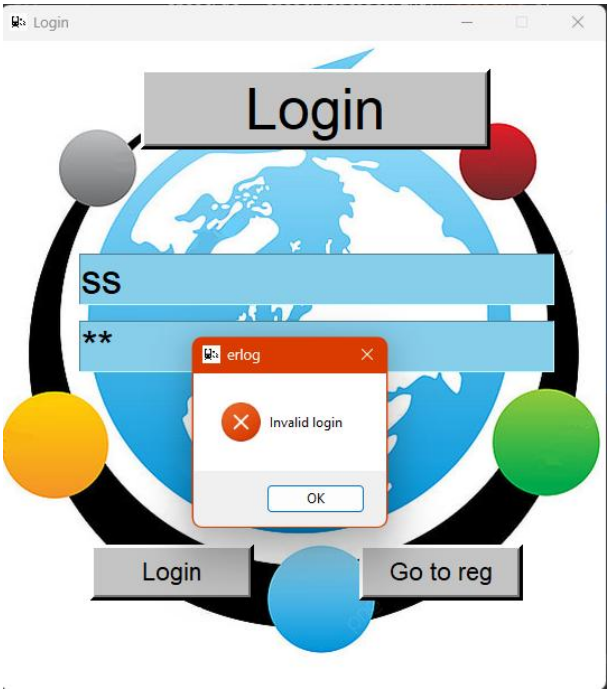


Рисунок А.3.7 – Результат роботи програми (помилка «логіну не існує»)

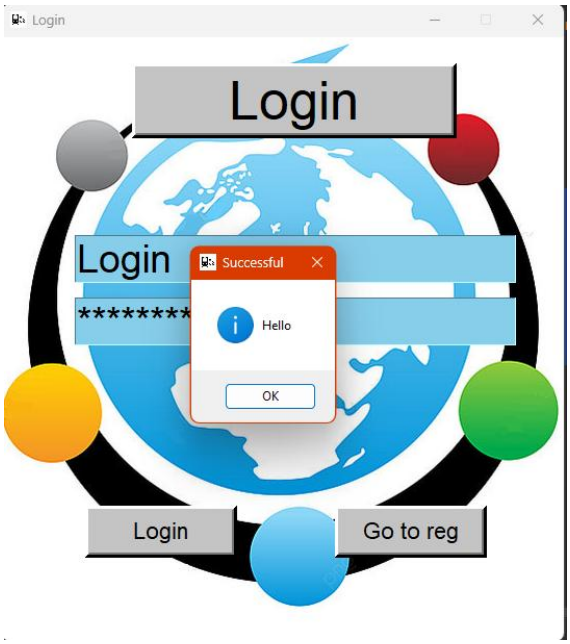


Рисунок А.3.8 – Результат роботи програми (повідомлення «успішний вхід»)

Результатом роботи програми є виведення на екран мапи з прокластеризованими даними і побудованими за ними маршрутами, рис. А.3.9.

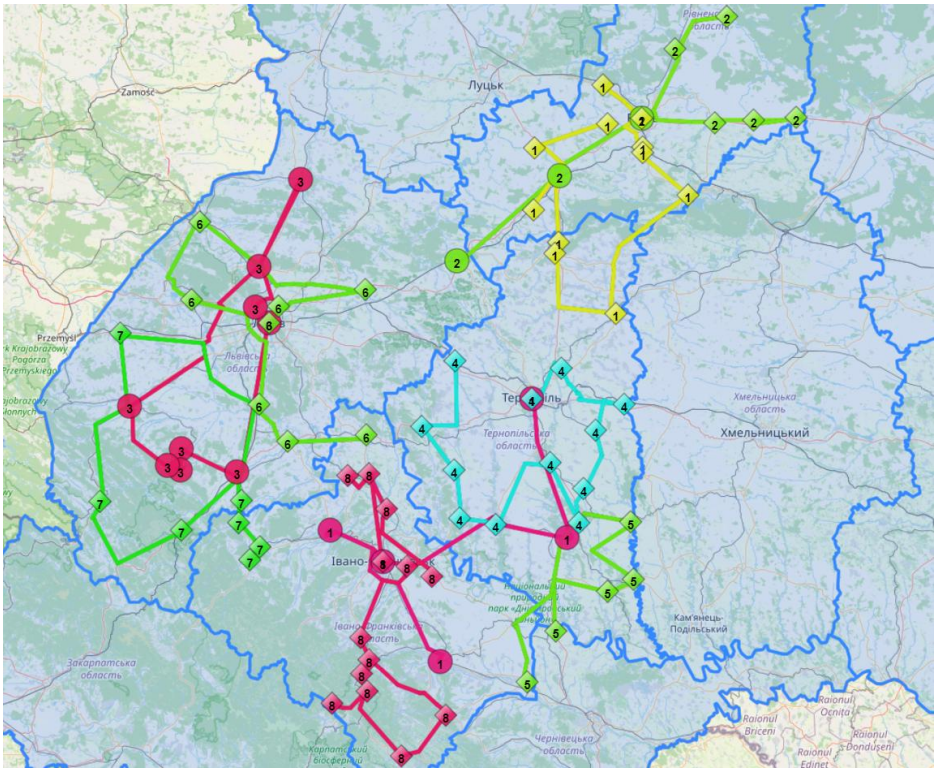


Рисунок А.3.9 – Побудова маршрутів

А.4 Рекомендації для роботи

Щоб забезпечити ефективну роботу з додатком Routers, перед початком роботи переконайтеся, що всі технічні вимоги виконані, а програмне середовище налаштоване відповідно до вказаних параметрів. Завантажте актуальні дані про точки доставки, маршрути та обмеження у підтримуваних форматах (CSV, JSON) та перевірте їх на наявність помилок чи неповноти. Це дозволить уникнути збоїв під час виконання обчислень і забезпечить точність отриманих результатів.

Для виконання кластеризації оберіть найбільш відповідний алгоритм залежно від специфіки ваших даних. Наприклад, K-Means підходить для рівномірного розподілу точок, DBSCAN краще працює з даними нерівномірної густини, а Agglomerative чи Spectral є оптимальними для складних ієрархічних структур. Перед запуском кластеризації налаштуйте параметри, такі як кількість кластерів або радіус охоплення, щоб забезпечити відповідність отриманих груп логістичним вимогам вашого підприємства.

Після завершення кластеризації використовуйте відповідну функцію для побудови маршрутів. Перегляньте отримані результати на інтерактивній карті, щоб перевірити їх відповідність вашим вимогам. У разі потреби можна вручну редагувати окремі маршрути, якщо автоматичний алгоритм не врахував усі деталі. Збережіть результати роботи для подальшого використання у вигляді звітів або експортуйте їх у форматах, сумісних із зовнішніми навігаційними системами.

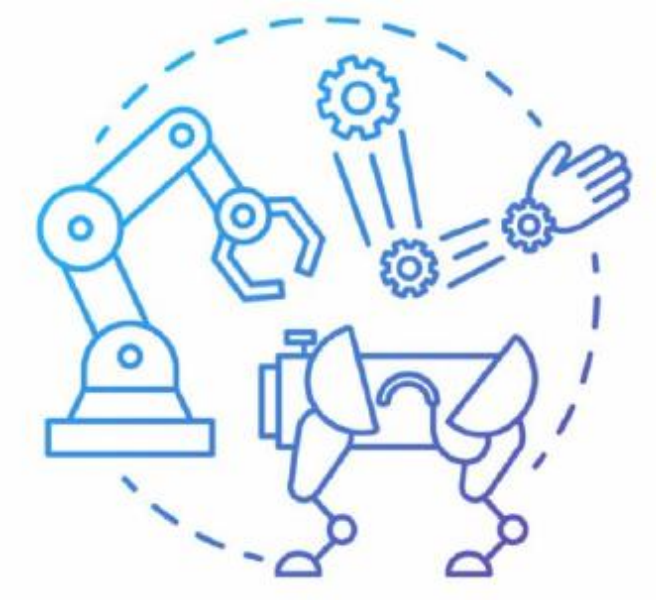
Рекомендується регулярно оновлювати базу даних, додаючи нові точки доставки чи оновлюючи параметри існуючих. Це дозволить завжди працювати з актуальною інформацією та уникати помилок у процесі планування. Важливо також уникати внесення змін до бази під час виконання кластеризації або побудови маршрутів, щоб не порушити хід обчислень.

Дотримання цих рекомендацій дозволить повністю реалізувати можливості додатку Routers, забезпечуючи високий рівень автоматизації та оптимізації логістичних процесів.

ДОДАТОК Б

Апробація результатів кваліфікаційної роботи

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(КІТАР)



МАТЕРІАЛИ

І Всеукраїнської конференції
«Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки»
(Computer-integrated technologies, automation and robotics)
CITAR`24
16-17 травня 2024

[електронне видання]

Харків 2024

УДК: 005:004.896:62-65:338.3

Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки 2024: матеріали І-ої Всеукраїнської конференції, Харків, 16-17 травня 2024.: тези доповідей / [редкол. І.Ш. Невлюдов (відповідальний редактор)].-Харків: [електронний друк], 2024. – 163 с.

У збірник включені тези доповідей, які присвячені сучасним автоматизованим технологіям Industry 4.0 та їх впровадження; інформаційні управляючі системи технологічного призначення; математичні методи в системах автоматизації; розробка та програмування в робототехніці; штучний інтелект та машинне навчання в автоматизації; інтеграція технологій у виробництві та промисловості; сенсорні технології та взаємодія людини з роботами в Industry 5.0; ефективність використання роботизованих систем у виробництві; етика та правові аспекти в робототехніці; Інтернет речей та Інтегровані системи в комп'ютерно-інтегрованих технологіях, автоматизації та робототехніки; технологічні виклики та інновації у світі робототехніки.

Редакційна колегія: І.Ш. Невлюдов, В.В. Євсєєв.

Computer-integrated technologies, automation and robotics 2024: Proceedings of I st All-Ukrainian Conference, Kharkiv, May 16-17, 2024: Thesises of Reports / [Ed. I.Sh. Nevlyudov (chief editor).] .- Kharkiv .: [electronic version], 2024. - 163 p.

The collection includes abstracts devoted to modern automated technologies of Industry 4.0 and their implementation; information control systems for technological purposes; mathematical methods in automation systems; development and programming in robotics; artificial intelligence and machine learning in automation; integration of technologies in production and industry; sensor technologies and human interaction with robots in Industry 5.0; efficiency of using robotic systems in production; ethics and legal aspects in robotics; Internet of Things.

Editorial board: Igor.Sh. Nevludov, Vladyslav.V. Yevsieiev

ЗМІСТ

ОЦІНКА ЯКОСТІ КЛАСТЕРИЗАЦІЇ КООРДИНАТ ГЕОГРАФІЧНИХ ОБ'ЄКТІВ РІЗНИМИ МЕТОДАМИ

Дідик П.Ю., Максимова С.С.
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: pavlo.doidyk@nure.ua

Анотація: У даній статті було розглянуто оцінку результати кластеризації населених пунктів різними методами, а саме K-means, DBSCAN, Agglomerative, Spectral, Berch. Проведено оцінку якості кожного методу кластеризації за допомогою відповідних критеріїв. Дано рекомендації щодо підбирання вхідних даних для кластеризації.

Ключові слова: кластеризація, метод, оцінка якості

ASSESSMENT OF THE QUALITY OF CLUSTERING GEOGRAPHIC OBJECTS' COORDINATES USING VARIOUS METHODS.

Didyk P., Maksimova S.
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky Ave. 14

Annotation: In this article, an evaluation of clustering results for settlements using various methods, namely K-means, DBSCAN, Agglomerative, Spectral, and Berch, has been discussed. The quality assessment of each clustering method was conducted using respective criteria. Recommendations regarding the selection of input data for clustering have been provided.

Key words: clustering, method, quality assessment

Цілеспрямована оцінка якості кластеризації географічних об'єктів за допомогою різних методів є критично важливою для широкого кола дисциплін, включаючи геоінформатику, геодезію, маркетинг та інші сфери. Розуміння ефективності кластеризаційних методів у відношенні до географічних даних може сприяти поліпшенню аналізу просторової інформації та прийняттю стратегічних рішень у сферах, де важлива локалізація та геопросторові взаємозв'язки.

У контексті Індустрії 4.0, кластеризація географічних об'єктів відіграє ключову роль у вирішенні різних завдань, пов'язаних з оптимізацією процесів, аналізом даних та прийняттям рішень. Одним з основних аспектів є використання географічних даних для побудови ефективних стратегій розташування виробничих підприємств, складів та інфраструктури. Кластеризація географічних об'єктів дозволяє ідентифікувати регіони або зони з концентрацією підприємств та ресурсів, що сприяє ефективному управлінню логістичними та постачальними ланцюгами, а також забезпечує оптимальне розташування виробничих об'єктів для зниження витрат та збільшення продуктивності.

Крім того, кластеризація географічних об'єктів важлива для аналізу та прогнозування тенденцій розвитку ринків та регіонів. Шляхом групування різних географічних зон за подібними ознаками та параметрами, можна отримати інсайти щодо споживчих звичок, попиту на ринку, а також виявити можливі потенційні ризики або перспективи для бізнесу. Це дозволяє підприємствам бути більш адаптивними та гнучкими у відповіді на змінні умови ринку.

Кластеризація - це процес групування схожих об'єктів у підмножини, які називаються кластерами. Метою кластеризації є виявлення природних структур або патернів в даних шляхом групування об'єктів, які мають схожі характеристики або властивості.

Кластеризація географічних об'єктів може бути використана для вирішення практичних завдань, таких як планування маршрутів доставки, розміщення нових об'єктів, оптимізація логістичних та постачальних ланцюгів і багато іншого. Кластеризація надає можливість систематизувати та аналізувати географічні дані для ефективного прийняття управлінських та стратегічних рішень.

Аналізуючи дані, отримані при кластеризації різних географічних об’єктів України за допомогою різних методів кластеризації та вивчаючи графіки з оцінками різних підходів, можна провести оцінку ефективності цих методів та зробити висновки для кожного з них.

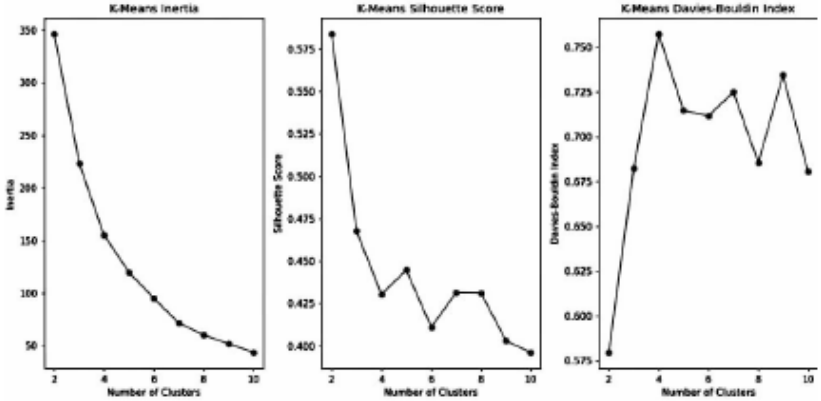


Рисунок 1 – Результат аналізу кластеризації методом K-means

а) Інерція свідчить про компактність точок всередині кластерів, тому оптимальним варіантом є вибір меншої інерції;

б) Вище значення коефіцієнта силуету вказує на більш точне відокремлення кластерів;

в) Найменше значення індексу Девіса-Боулдіна вказує на краще розділення кластерів;

Загалом, аналізуючи всі три методи оцінки, можна вважати, що оптимальною кількістю кластерів для даної задачі є 5.

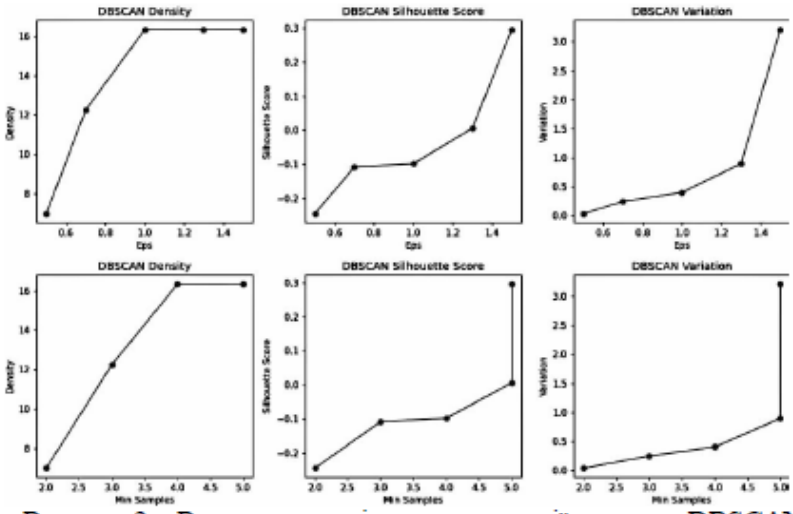


Рисунок 2 – Результат аналізу кластеризації методом DBSCAN

а) З поданих даних видно, що значення Density (eps) збільшується поступово, тоді як Density (min samples) залишається постійним. При eps=1.3 та min_samples=4 спостерігається

однакове значення для Silhouette Score та Variation, що робить ці параметри можливо оптимальними;

б) Негативні значення Silhouette Score, такі як -2.5 та -0.98, свідчать про низьку якість кластеризації.

в) Дані eps=1.5 та min_samples=5 мають найвище значення Variation, що може свідчити про значну зміну внутрішньокластерних відстаней.

Загалом, при аналізі всіх трьох методів оцінки можна вважати, що оптимальними значеннями є eps=1.3 та min_samples=4.

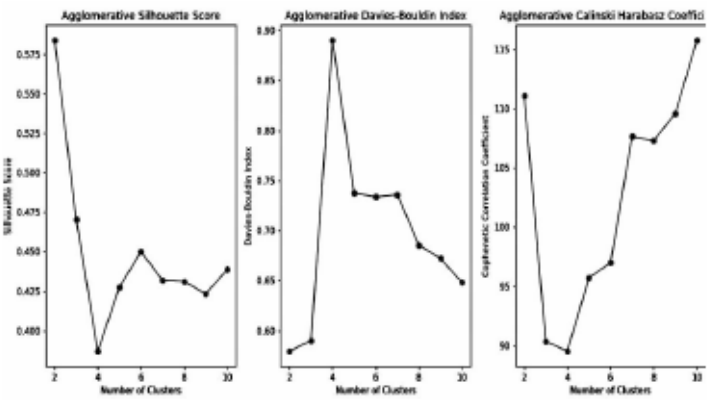


Рисунок 3 – Результат аналізу кластеризації методом Agglomerative

а) Вищі значення Silhouette Score свідчать про вищу якість кластеризації;

б) Нижчі значення Davies-Bouldin Index свідчать про кращу якість кластеризації;

в) Вищі значення Calinski-Harabasz Index вказують на більш компактні та роздільні кластери;

Загалом, на основі цих показників розгляд може бути спрямований на 2 або 3 кластери як оптимальний вибір для цього набору даних..

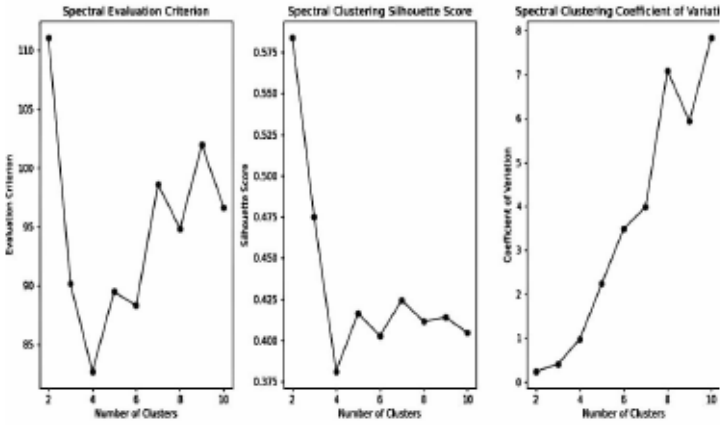


Рисунок 4 – Результат аналізу кластеризації методом Spectral

а) за цими даними, ми можемо побачити, що значення критерію оцінки зменшується при збільшенні кількості кластерів від 2 до 4, а потім починає зростати;

б) За цими даними, ми бачимо, що значення Silhouette максимальне при 2 кластерах і плавно знижується зі збільшенням кількості кластерів;

в) за цими даними, ми бачимо, що значення коефіцієнта варіації збільшується зі збільшенням кількості кластерів;

Загалом, з урахуванням усіх трьох метрик, можна вважати оптимальним значенням кластерів 2 або 3. Рекомендується перевірити обидва варіанти та порівняти результати для визначення найкращого.

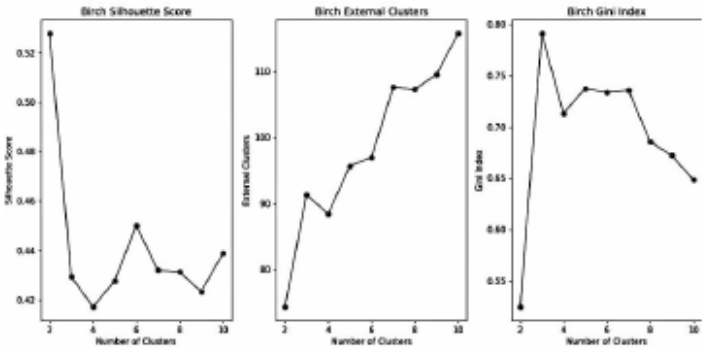


Рисунок 5 – Результат аналізу кластеризації методом Birch

а) Silhouette Score: ця метрика коливається від -1 до 1, де значення, що наближаються до 1, свідчать про вищу якість кластеризації;

б) External clusters: ця метрика вимірює зовнішню якість кластеризації, де більші значення вказують на кращу якість;

в) Gini Index: ця метрика коливається від 0 до 1, де значення, що наближаються до 0, вказують на вищу якість кластеризації;

Залежно від конкретних вимог або контексту задачі, ви можете обрати оптимальну кількість кластерів для методу Birch. За результатами оцінки, 10, 9 та 8 кластерів виявляються перспективними варіантами. Рекомендую спробувати ці значення і провести додатковий аналіз результатів для вибору найкращого варіанту.

ВИСНОВКИ. Аналіз показав, що кожен метод має свої переваги та обмеження, і вибір оптимального методу залежить від конкретних умов задачі. Результати дослідження підкреслили важливість глибокого розуміння кластеризаційних методів для вдосконалення аналізу географічних даних та розробки стратегій управління ресурсами та інфраструктурою у контексті Індустрії 4.0.

ЛІТЕРАТУРА

1. Карпенко, А. А., Шевченко, В. І., & Григорович, А. П. (2019). Методи оцінки ефективності алгоритмів кластеризації в аналізі даних. Кібернетика і системний аналіз, 55(4), 161-176.

2. Лазарєв, О. В., & Попов, Є. П. (2020). Оцінка точності кластеризації текстових даних за допомогою внутрішніх критеріїв. Інформаційні технології та комп'ютерна інженерія, 4(46), 56-67.

3. Гончар, І. М., & Шуляк, В. В. (2018). Аналіз методів оцінки якості кластеризації в біологічних дослідженнях. Біоінформатика та комп'ютерна біологія, 9(1), 35-42.

4. Коваленко, Л. О., & Лисенко, Т. О. (2021). Використання валідаційних індексів для визначення кількості кластерів у задачі кластеризації даних. Наукові записки НаУКМА, 244, 58-65.

5. Літвінова, Т. В., & Іванов, О. О. (2019). Порівняльний аналіз метрик для оцінки якості кластеризації. Вісник НТУУ "КПІ". Серія Інформатика, управління та обчислювальна техніка, (78), 15-22.

6. Attar, H., & et al.. (2022). Control System Development and Implementation of a CNC Laser Engraver for Environmental Use with Remote Imaging. Computational Intelligence and Neuroscience, 2022, Article ID 9140156, https://doi.org/10.1155/2022/9140156.

7. Attar, H., & et al.. (2022). Zoomorphic Mobile Robot Development for Vertical Movement Based on the Geometrical Family Caterpillar. Computational Intelligence and Neuroscience, 2022, Article ID 3046116, https://doi.org/10.1155/2022/3046116.

8. Al-Sharo, Y., Abu-Jassar, A., Lyashenko, V., Yevsieiev, V., Maksymova, S. A Robo-hand prototype design gripping device within the framework of sustainable development, Indian Journal of Engineering, 20 2023 e37ije1673. https://doi.org/10.54905/disssi.v20i54.e37ije1673

9. Lyashenko, V., Abu-Jassar, A.T., Yevsieiev, V., Maksymova, S. Automated Monitoring and Visualization System in Production, Int. Res. J. Multidiscip. Technovation, 5(6) 2023 09-18. https://doi.org/10.54392/irjmt2362

10. Невлюдов І. Ш. BEAM робототехніка : навч. посіб. / І. Ш. Невлюдов, В. В. Євсєєв, С. С. Максимова ; Харків. нац. ун-т радіоелектроніки, кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР). – Кривий Ріг : Видавець Чернявський Д. О., 2024. – 276 с. – ISBN 978-617-8045-79-1

11. Nevliudov, I., Yevsieiev, V., Maksymova, S., Klymenko, O. (2024), "The "load balancing" and "adaptive task completion" algorithms implementation on a pharmaceutical sorting conveyor line", Innovative Technologies and Scientific Solutions for Industries, No. 1 (24), P. 14–24. DOI: https://doi.org/10.30837/ITSSI.2023.23.000

12. Amer Abu-Jassar, Vladyslav Yevsieiev, & Svitlana Maksymova. (2024). The Optical Flow Method and Graham's Algorithm Implementation Features for Searching for the Object Contour in the Mobile Robot's Workspace. Journal of Universal Science Research, 2(3), 64–75.

13. Svitlana Maksymova, Vladyslav Yevsieiev, & Amer Abu-Jassar. (2024). Gripping Device Development: Some Aspects. Journal of Universal Science Research, 2(1), 150–158.

14. Svitlana Maksymova, & Vladyslav Yevsieiev. (2024). Coin Counting Device Kinematic Diagram Development. Journal of Universal Science Research, 2(1), 159–168.

15. Svitlana Maksymova, Vladyslav Yevsieiev, & Amer Abu-Jassar. (2024). The Bipedal Robot a Kinematic Diagram Development. Journal of Universal Science Research, 2(1), 6–17.

16. Svitlana Maksymova, Vladyslav Yevsieiev, & Amer Abu-Jassar. (2024). Gripping Device Development: Some Aspects. Journal of Universal Science Research, 2(1), 150–158.

17. Yevsieiev, V. Comparative Analysis of the Characteristics of Mobile Robots and Collaboration Robots Within INDUSTRY 5.0. / V. Yevsieiev, D. Gurin // In the VI International Scientific and Theoretical Conference, September 8, 2023. Chicago, USA. P.92-94

18. A Small-Scale Manipulation Robot a Laboratory Layout Development / Yevsieiev V., Starodubcev N., Maksymova S., Stetsenko K. // International independent scientific journal, №47, 2023. P.18-28.

ДОДАТОК В

Висвітлення результатів кваліфікаційної роботи

МІНІСТЕРСТВО ОБОРОНИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПОВІТРЯНИХ СИЛ ІМЕНІ ІВАНА КОЖЕДУБА

Системи обробки інформації

Щоквартальний збірник наукових праць

Випуск 1 (176)

Заснований у березні 1996 року

У збірнику відображено результати досліджень з розробки нових інформаційних технологій як для рішення традиційних задач збору, обробки та відображення даних, так і для побудови систем обробки інформації в різних проблемних галузях.

Збірник призначений для наукових працівників, викладачів, докторантів, ад'юнктів, аспірантів, а також курсантів та студентів старших курсів відповідних спеціальностей.

Засновник і видавець:
Харківський національний університет Повітряних Сил імені Івана Кожедуба

61023, м. Харків-23, а/с 11800

Телефон:
+38 (057) 704-91-97
+38 (068) 118-78-50

E-mail:
journal-hnups@ukr.net

Інформаційний сайт:
journal-hnups.com.ua

Харків • 2024

Затверджений до друку вченою радою
Харківського національного університету Повітряних Сил імені Івана Кожедуба
(протокол від 21.05.2024 р. № 6).

Включений до категорії "Б" Переліку наукових фахових видань України,
технічні та військові науки за спеціальностями 122, 123, 125, 126, 172, 253, 255, 272, 275
(накази Міністерства освіти і науки України від 17.03.2020 р. № 409 та від 02.07.2020 р. № 886).

Ідентифікатор медіа – R30-03672 (рішення від 21.03.2024 р. № 938).

Усі статті, що публікуються в збірнику, проходять обов'язкове рецензування,
яке здійснюється за анонімною формою як для авторів, так і для рецензентів (подвійне сліпе рецензування).
Унікальність текстів публікацій перевіряється за допомогою системи пошуку ознак плагіату Unicheck.
За достовірність викладених фактів, цитат та інших відомостей відповідальність несе автор.

ISSN 1681-7710

MINISTRY OF DEFENCE OF UKRAINE
IVAN KOZHEDUB KHARKIV NATIONAL
AIR FORCE UNIVERSITY

Information Processing Systems

Issue 1 (176)

Quarterly scientific publication

Founded in March, 1996

The scientific works "Information Processing Systems" was founded in 1996 and became the first information platform for Ukrainian specialists in the field of data processing system. Its objective is to publish research on the development of information technologies and management systems, information processing in complex technical and organizational systems both for solving traditional problems such as acquisition, processing and data reporting, and for building communication, measurement, educational, information protection, cyber security and other systems.

Founder and publisher:
Ivan Kozhedub Kharkiv National Air Force University

Address: post office box 11800, Kharkiv, 61023, Ukraine

Phone: +38 (057) 704-91-97
+38 (068) 118-78-50

E-mail:
journal-hnups@ukr.net

Website:
journal-hnups.com.ua

Kharkiv • 2024

Academic Council of Ivan Kozhedub Kharkiv National Air Force University
confirmed for printing (Record No. 6 dated May, 21, 2024).

The Publication is inscribed to the category "Б" of the List of Scientific Professional Publications of Ukraine,
Technical and Military Sciences by specialties 122, 123, 125, 126, 172, 253, 255, 272, 275
(Orders of Ministry of Education and Science of Ukraine No. 409 dated March, 17, 2020
and No. 886 dated July, 2, 2020).

Media ID – R30-03672 (Decision No. 938 dated March, 21, 2024).

All the articles that are published must be peer reviewed.
It is conducted anonymous both for authors and reviewers (double blind peer review).
The uniqueness of the texts of publications is checked with using the Unicheck plagiarism signs search system.
The authors take responsibilities for the reliability of stated facts, quotations and other statements.

ЗМІСТ
CONTENTS

UDC 004.8 DOI: 10.30748/soi.2024.176.14

O. Chorna, P. Didyk, S. Titov, O. Titova

Kharkiv National University of Radio Electronics, Kharkiv

USAGE OF CLUSTERING ALGORITHMS FOR AUTOMATING ROUTE PLANNING IN TRANSPORTATION ROUTING TASKS

This scientific paper explores the comprehensive evaluation of clustering results applied to the geographical settlements of Ukraine. Diverse clustering methods, including K-means, DBSCAN, Agglomerative, Spectral, and Birch, were employed to analyze the spatial distribution of settlements. The assessment of each clustering method involved the application of relevant quality criteria, contributing to a thorough understanding of their performance in the context of Ukrainian settlements. The findings from this study offer valuable insights into the strengths and limitations of each clustering approach, facilitating informed decision-making in the selection of an appropriate method based on specific geographical characteristics. Additionally, the paper provides practical recommendations for optimizing the input data utilized in the clustering process, enhancing the overall efficacy of settlement analysis methodologies. This research contributes to the advancement of clustering techniques tailored to geographical datasets, with potential implications for urban planning, regional development, and geographic information systems.

Keywords: *clustering, vehicle routing problem, pick-up and delivery problem, clustering evaluation.*

Introduction

Problem statement. For numerous years, research efforts have extensively focused on transport routing and logistics issues, a substantial aspect of contemporary inquiry. The urgency of these problems escalates with the increasing interconnectedness, openness, and lack of barriers in the modern world. The rapid dissemination of information results in global awareness of humanitarian pleas for aid from ordinary individuals and public organizations situated in crisis zones during disasters or challenging conditions. The vigor of the volunteer movement, coupled with horizontal connections among people, facilitates the aggregation of essential humanitarian assistance in small increments from various contributors. This aid can then be dispatched to multiple recipients situated in different sectors of a disaster or conflict zone. A specialized field known as the Pick-up and Delivery Vehicle Routing Problem is dedicated to addressing transportation challenges of this nature.

This encompasses a broad category of optimization challenges that mirror diverse real-world scenarios. Examples include the imperative to meet specific time constraints for delivering goods, taking into account general or specific limitations on truck loading, as well as constraints on the sequence of loading and unloading vehicles. What defines this category is the multidimensional nature of the input data, the necessity for decision-making amid uncertainty, and the demand for swiftly generating solutions. It's noteworthy that the importance of obtaining an approximate solution with a specified level of accuracy can vary depending on the particular problem's scope. Some situations may prioritize acquiring a solution that is roughly accurate, allowing the use of any number of computing and time resources. In contrast, other scenarios may be time-sensitive, emphasizing the need for a heuristic acceptable solution that expedites the essential cargo delivery (Fig. 1).

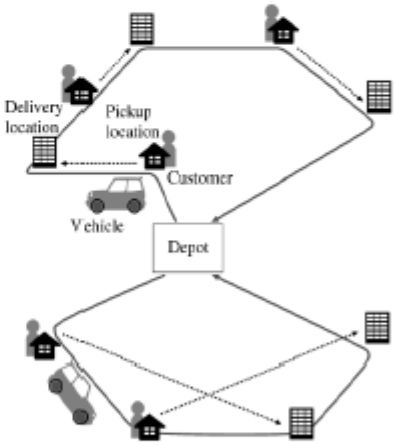


Fig. 1. Example of a one-to-one Pickup and Delivery Problem solution
Source: [1].

In instances where multiple trucks are engaged in addressing the pick-up and delivery challenge, involving goods from numerous senders to multiple recipients, a viable approach involves clustering the points slated for visitation [2–4]. Subsequently, these clusters can be allocated among the various vehicles. Various clustering methods can be employed to generate clusters corresponding to each vehicle. The solution obtained can then be further evaluated based on a specific clustering approach.

Analysis of the latest research and publications. In the realm of machine learning, three distinct ap-

proaches are employed based on the nature of the available data: supervised learning, semi-supervised learning, and unsupervised learning [5–9]. Supervised learning involves utilizing labeled data, where known outputs serve as definitive values for the corresponding inputs [10–13]. This can be likened to having explicit knowledge of car prices determined by features such as make, model, style, drivetrain, and other attributes. Semi-supervised learning, on the other hand, deals with a substantial dataset where some data points are labeled, while the majority remain unlabeled [14–16]. This approach accommodates real-world datasets where acquiring expert labeling for every data point proves economically impractical. A pragmatic workaround involves integrating a combination of both supervised and unsupervised learning techniques. Unsupervised learning is characterized by a dataset entirely devoid of labels, rendering unknown any inherent patterns within the data. In this scenario, the algorithm is entrusted with the task of autonomously identifying any discernible patterns.

Clustering algorithms [17–19] play a pivotal role in unsupervised learning, presenting one of the methodologies available for tackling unsupervised learning problems [4; 20]. Approaches to constructing clusters can vary significantly in terms of how objects are grouped within the same cluster and how the search for them is efficiently conducted. The capability to form clusters exists based on factors such as the distance between objects, density plots in the data space, intervals, or specific statistical distributions. The selection of a specific approach hinges on the characteristics of the dataset and its intended use.

It's important to note that cluster analysis is not a fully automated process; rather, it is iterative. Experimentation with data processing methods and model parameters is often necessary until the desired properties and results are achieved. The ambiguity in solving clustering problems arises for several reasons. Firstly, there is no universal criterion for assessing clustering quality. While there are numerous effective criteria and algorithms available for constructing high-quality clusters, outcomes can vary. Secondly, the number of clusters is typically not predetermined and is subjectively determined by a criterion. Thirdly, the results of clustering are significantly influenced by the choice of the metric ρ, a subjective decision made by a skilled individual in the field.

The purpose of the research is to compare the effectiveness of different clustering algorithms for solving complex transport logistics problems. The problem of routing vehicles for receiving and delivering humanitarian aid is considered. This involves utilizing clustering algorithms, particularly in unsupervised learning, to efficiently allocate resources and optimize delivery routes amid the dynamic and uncertain nature of humanitarian crises.

Summary of the main material

1. Choosing clustering methods with distinct usage specifics

Application of clustering methods in data analysis to geospatial data, such as latitude and longitude, allows not only identify groups with similar profiles but also pinpoint members within a specified location or region [21; 22]. These geographical patterns offer practical solutions to various location-based challenges, including demand forecasting, personalized branch marketing, route optimization, and fraud prevention.

Outlined below are three of the most extensively employed machine learning algorithms for clustering geospatial data.

1. Partition-Based – this algorithm partitions the data into k subsets by establishing virtual or actual centroids and measuring their distance from the remaining datapoints. Notable examples in this category encompass the widely recognized K-means algorithm and its variations, including K-mean, K-modes, K-medoids, and Fuzzy C-means.

2. Hierarchical – this algorithm assesses the similarity between datapoints within a hierarchical structure to accomplish clustering. The hierarchical structure can be either agglomerative (bottom-up) or divisive (top-down) based on the chosen approach. BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) and DIANA (Divisive Analysis) fall within this category.

3. Density-Based – this algorithm primarily utilizes the distance to the nearest point to delineate regions with higher concentration from those with lower density. The most widely utilized algorithm in this category is DBSCAN (Density-based spatial clustering of applications with noise).

In the context of the aforementioned considerations, the current research was methodically structured to encompass a diverse array of results. The objective was to compare the effectiveness of various clustering algorithms, thereby adopting distinct approaches to clustering methods. Specifically, a selection was made from the family of partition-based algorithms, with the standard K-means algorithm chosen as the allegorical "starting point". This choice aimed to ensure the comparability of our research with numerous other scientific papers that employ K-means as a benchmark.

Correspondingly, for the other broad groups of clustering techniques, two algorithms were selected from each category. The hierarchical family of clustering algorithms is represented by agglomerative clustering and the BIRCH algorithm. The density-based approach is implemented using DBSCAN. The fifth and final method chosen for this study is spectral clustering. In cases where the identification of connected graph components with no cut edges is trivial, a spectral implementation of DBSCAN is linked to spectral cluster-

ing. However, due to computational performance considerations, the original DBSCAN algorithm is often preferred over its spectral implementation [23].

2. Clustering method effectiveness evaluation

Assessing the efficacy and precision of clustering methods involves gauging their success in effectively grouping similar objects within a dataset. This evaluation is crucial in numerous machine learning and data analysis tasks that necessitate the clustering of similar objects [24; 25].

Various metrics are available for appraising clustering methods, and these include:

– the similarity metric measures the degree of similarity between clusters. This metric can be calculated using various formulas, such as the Jaccard index, the Randy index, the Fowler index, and others;

– the stability metric measures the stability of clusters when the clustering method is repeated on the same data. It can be calculated using indicators such as the Jaccard index, the Cohen index, the Randy index, and others;

– internal metrics are used to evaluate the quality of clusters within one run of the clustering method. For example, the silhouette metric, Kalinin-Harabas metric, Davis-Baldwin metric, and others can be used for this purpose;

– external metrics are used to evaluate the quality of clusters based on external data. For example, the quality of clusters can be evaluated by comparing them with the classification of objects in a "gold standard" or other set of correct classifications. One example of such external metrics is the Adjusted Rand Index (ARI) metric, which measures the degree of agreement between the clustering and the "gold standard". Other external metrics, such as the F-mer metric, can be used to assess the quality of clusters when a "gold standard" is not available.

3. Data preparation and features of the program interface

In current research it was decided to use all of the abovementioned clustering methods for real life data, such as location of the main population centers of Ukraine. Data about cities were stored in a MySQL database where each record has the following structure: idCity (city identifier), Name (city name), x (latitude) and y (longitude). The program takes information about coordinates from the database in the format of latitude and longitude (x and y), which is then used for clustering.

Also in the program interface are the following buttons corresponding to data clustering functions: "Cluster K-Means", "Cluster DBSCAN", "Cluster Agglomerative", "Cluster Spectral" and "Birch Cluster". Accordingly, when pressed, each button opens a window with the result of clustering by its method using the create_ukraine_map function. The function create_ukraine_map is responsible for creating a window with the result of clustering on the map of Ukraine. This feature uses the folium library to create interactive maps. The create_ukraine_map function uses data about the coordinates of cities from the data list and their corresponding clusters from the clusters list to create a map of Ukraine with marked markers that represent the clusters.

In addition, the program interface contains five options responsible for checking the quality of clustering by one or another method. The result of pressing these buttons will be the display of graphs showing the results of the analysis of the corresponding clustering method. With the help of the obtained results, you can analyze a separate clustering method, determine the optimality of the input parameters, and speculate on recommendations for further choosing a clustering method.

4. Results interpretation

The analysis of the clustering quality evaluation results of Ukrainian settlements using the K-means method includes the study of three key parameters. From the inertia graph, it can be concluded that if the number of clusters increases from 2 to 10, the inertia continuously decreases (Fig. 2).

This parameter indicates the compactness of the points inside the clusters, and therefore, choosing a smaller inertia is considered preferable.

From the analysis of the graph, it can be determined that the acceptable number of clusters from the point of view of inertia is 5–7, since the graph changes the slope sharply and then slowly decreases. On the graph of the silhouette coefficient, it can be seen that with an increase in the number of clusters from 2 to 10, the values of the silhouette coefficient fluctuate. A higher value of the silhouette coefficient indicates a more accurate separation of clusters. The highest value of the silhouette coefficient is achieved with the number of clusters of 2, but values of 4 and 5 are also acceptable. The last parameter to consider is the Davis-Bouldin index graph, on which the index values fluctuate as the number of clusters increases from 2 to 10. The minimum value of the Davis-Bouldin index indicates a better separation of clusters. The minimum value of the Davis-Bouldin index is reached when the number of clusters is 5, 6, 8, 10. Summarizing the results of the three evaluation methods, it can be assumed that the optimal number of clusters when using the K-means method for this problem is 5. The DBSCAN method involves two crucial parameters for consideration. The first parameter is ε (epsilon or Eps) which establishes the maximum distance allowed between two points within the same cluster. The second parameter is the minimum samples (Min Samples), dictating the minimum number of data points necessary to constitute a distinct cluster (Fig. 3).

When considering the clustering quality evaluation results of major town cities of Ukraine by the DBSCAN method, the following was found.

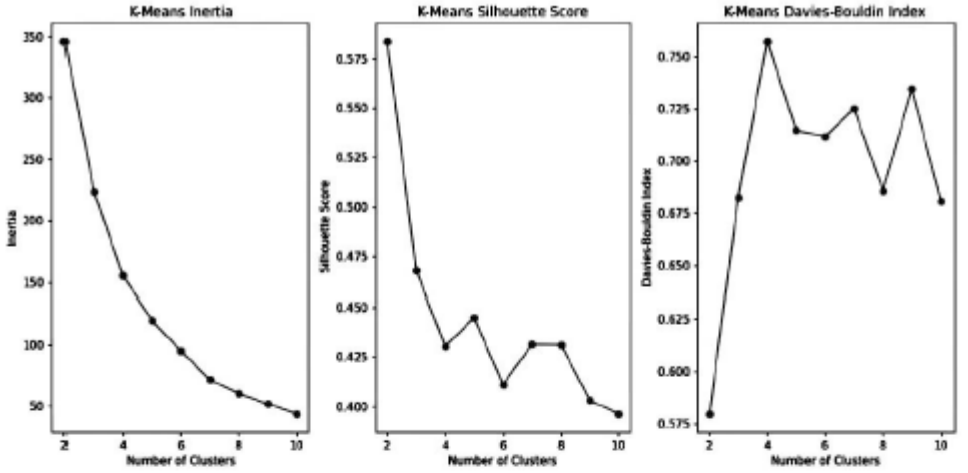


Fig. 2. The result of K-means clustering analysis
Source: developed by the authors.

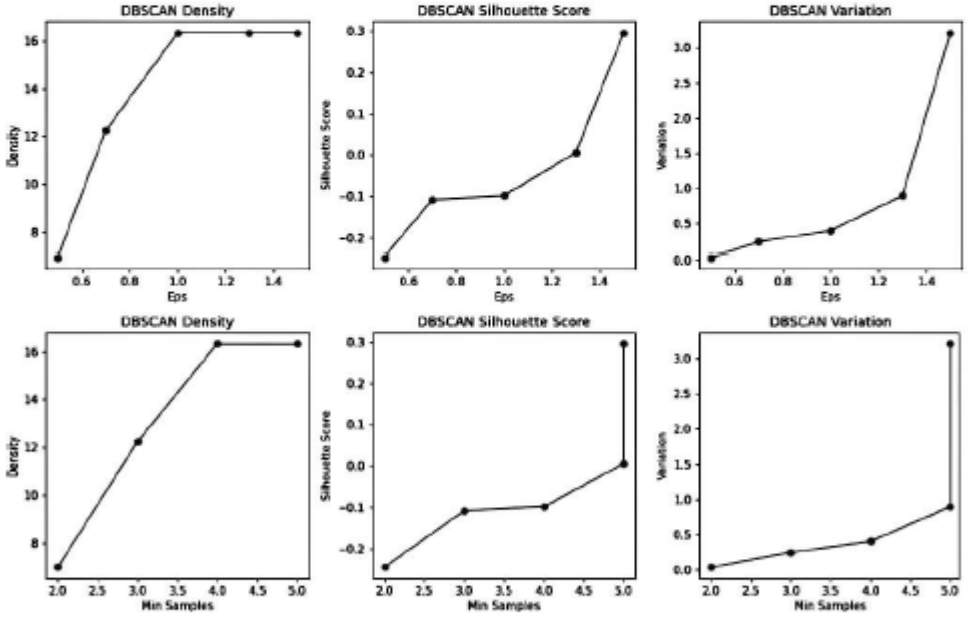


Fig. 3. The result of the clustering analysis by the DBSCAN method
Source: developed by the authors.

The values of the Density(Eps) plot show a gradual increase in Eps, while Density(Min Samples) remains constant. At Eps=1.3 and Min Samples=4, the same values were noted for Silhouette Score and Variation, which may make these parameters optimal. It is important to consider other values, as each case may require an individual approach. Regarding Silhouette Score(Eps) and Silhouette Score(Min Samples) plots, negative values such as –2.5 and –0.98 indicate poor clustering quality. However, Eps=1.3 has a Silhouette Score close to zero (0.008), which may indicate more acceptable clustering. For Variation(Eps) and Variation(Min Samples) plots, Variation values indicate the change in intracluster distances. Data at Eps=1.5 and Min Samples=5 have the

highest Variation value, which may indicate a significant change in intracluster distances. Summarizing the results of the three evaluation methods, it can be considered that when applying this method to the clustering of geographical data on the settlements of Ukraine, it is possible to give preference to choosing the values Eps=1.3 and Min Samples=4.

The analysis of the results of the quality assessment of data clustering using the agglomerative method revealed the following (Fig. 4). It is known that higher values of the Silhouette Score parameter indicate a better quality of clustering. The maximum value of this parameter is reached with 2 clusters, after which there is a sharp decrease in the value of the parameter, the minimum of which is reached with four clusters. When the number of clusters is more than 4, an increase and stabilization of the value at the level of (0.45–0.425) is observed. On the Davies-Bouldin Index graph, lower values indicate better clustering quality. In this case, it is also possible to observe a substantial increase in the indicator with 4 clusters and a slow stabilization with an increase in the number of clusters. Regarding the Calinski-Harabasz Index plot, higher values indicate more compact and separated clusters. The maximum value is reached at 10 clusters, but this may cause the risk of overtraining or insufficient generalization of the model. When choosing the optimal number of clusters, it is important to maintain a balance between the number of clusters and the accuracy of clustering. Summarizing, it can be considered that the optimal choice for this dataset is either an extremely small number of clusters, i.e., 2 or 3, or a substantially larger number of clusters.

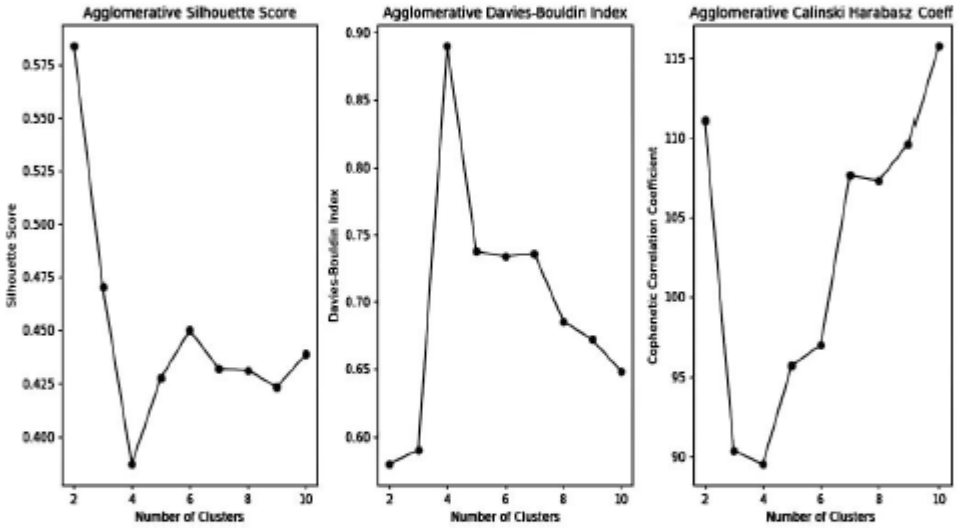


Fig. 4. The result of the clustering analysis by the Agglomerative method
Source: developed by the authors.

When using the Spectral method for clustering the cities of Ukraine, the following patterns can be determined (Fig. 5). The Evaluation Criterion graph shows that the value of the evaluation criterion first decreases when the number of clusters increases from 2 to 4, and then begins to increase. The optimal value of clusters can be more than 4, as point where the criterion reaches the minimum value before increasing. The Silhouette score and Variation graphs indicate that the acceptable value of these coefficients is achieved with the minimum possible number of clusters. Summarizing, we can conclude that in the case of using the Spectral method on similar data, it will be necessary to fine-tune this method, taking into account the behavior of the evaluation parameters. One of the parameters, namely the Evaluation Criterion, tends to increase the number of clusters, while the other two parameters have a substantial deterioration with the increase in the number of clusters.

Analyzing the results of Birch clustering quality assessment (Fig. 6), the following was found: the Silhouette Score graph indicates that the highest values of this metric are achieved with the number of clusters 2 (0.525), 6 (0.45) and 10 (0.44). The External clusters graph, which measures the external quality of clustering, shows that the highest values of this metric are achieved with the number of clusters 10 (117), 9 (110) and 8 (108). The Gini Index graph, which indicates the quality of clustering, shows that the lowest values of this metric are achieved with the number of clusters 10 (0.65), 9 (0.67) and 8 (0.68).

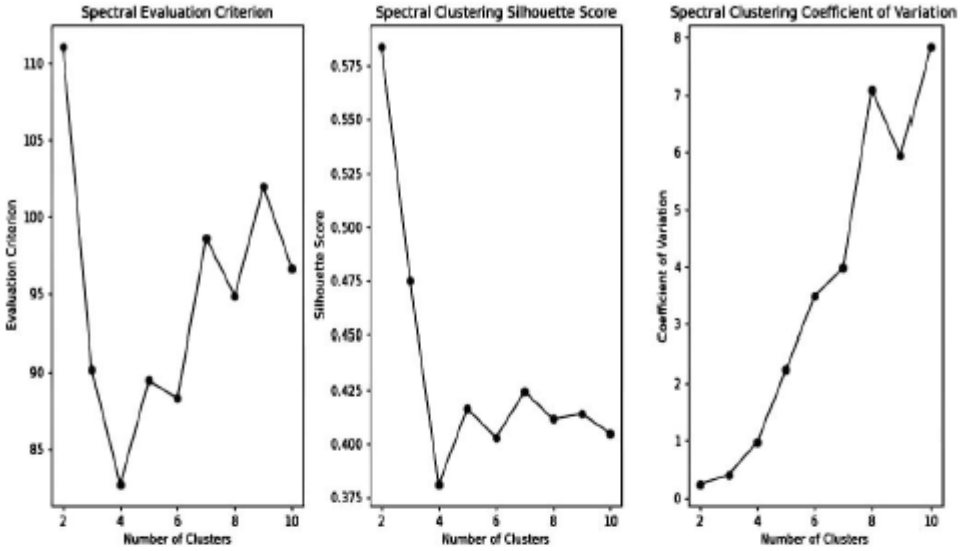


Fig. 5. The result of the clustering analysis by the Spectral method
Source: developed by the authors.

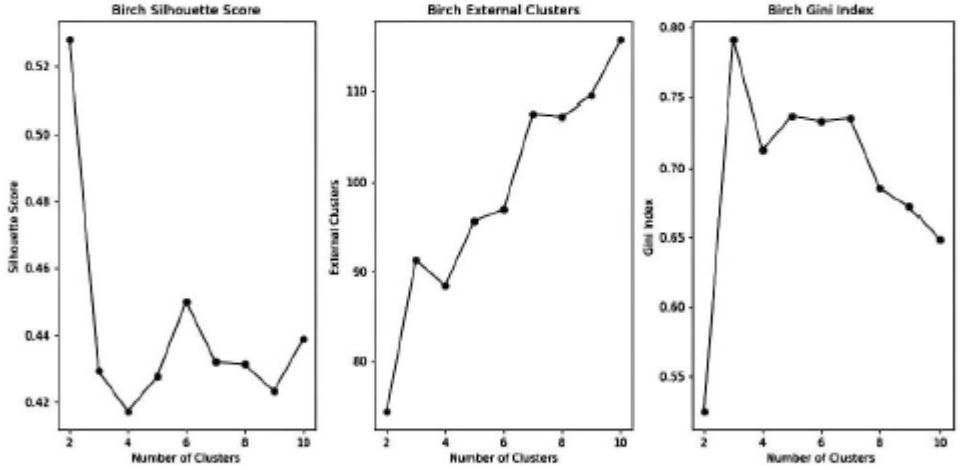


Fig. 6. The result of the clustering analysis by the Birch method
Source: developed by the authors.

Conclusions

The comprehensive evaluation of clustering results applied to the geographical settlements of Ukraine has provided valuable insights into the performance of various clustering methods, including K-means, DBSCAN, Agglomerative, Spectral, and Birch. Each clustering method was systematically assessed using relevant quality criteria, such as inertia, silhouette coefficient, and Davies-Bouldin index, offering a nuanced understanding of their effectiveness in grouping spatial data. The findings suggest that the choice of clustering method should be tailored to specific geographical characteristics and the objectives of the analysis, with considerations for factors like compactness of clusters, separation of data points, and computational efficiency. Practical recommendations have been provided for optimizing input data and selecting appropriate clustering techniques, which can enhance the efficacy of settlement analysis methodologies for applications in urban plan-

ning, regional development, and geographic information systems. The evaluation of clustering methods highlights the importance of considering both internal and external metrics to gauge clustering quality accurately, ensuring robust and reliable results in real-world applications. The study underscores the significance of fine-tuning clustering algorithms and parameters based on the behavior of evaluation parameters, acknowledging the iterative nature of the clustering process and the need for adaptive approaches. Analysis of the results demonstrates that different clustering methods may exhibit varying performance across different datasets, emphasizing the need for careful consideration and experimentation to identify the most suitable approach for a given problem. The insights gained from this research contribute to advancing clustering techniques tailored to geographic datasets, fostering advancements in data-driven decision-making processes and optimization strategies in transportation routing tasks and beyond. Future research directions may include exploring hybrid clustering approaches, incorporating domain-specific knowledge, and addressing challenges related to scalability and robustness in real-world applications. Overall, this study underscores the importance of leveraging clustering algorithms for automating route planning in transportation routing tasks, offering valuable contributions to the fields of machine learning, geographic information science, and logistics optimization.

References

1. Sakakibara, K., Tamaki, H. and Nishikawa, I. (2007), Autonomous distributed approaches for pickup and delivery problems with time windows, *In Proceedings of the SICE Annual Conference 2007*, IEEE, pp. 2639–2642. http://doi.org/10.1109/SICE.2007.4421437.

2. Dupas R., Grebennik, I., Litvinchev, I., Romanova, T. and Chorna, O. (2020), Solution strategy for one-to-one pickup and delivery problem using the cyclic transfer approach, *EAI Endorsed Transactions on Energy Web*, Vol. 20, No. 27, pp. 1–9. http://doi.org/10.4108/eai.13-7-2018.164110.

3. Grebennik, I., Chorna, O. and Urniaieva, I. (2022), Distribution of Permutations with Different Cyclic Structure in Mathematical Models of Transportation Problems, *2022 12th International Conference on Advanced Computer Information Technologies* (*ACIT*), *IEEE*, pp. 18–21. http://doi.org/10.1109/ACIT54803.2022.9913183.

4. Naeem, S., Ali, A., Anam, S. and Ahmed, M.M. (2023), An Unsupervised Machine Learning Algorithms: Comprehensive Review, *International Journal of Computing and Digital Systems*, Vol. 13, No. 1, pp. 911–921. http://doi.org/10.12785/ijcds/130172.

5. Sharifani, K. and Amini, M. (2023), Machine Learning and Deep Learning: A Review of Methods and Applications, *World Information Technology and Engineering Journal*, Vol. 10, No. 07, pp. 3897–3904.

6. Sarker, I.H. (2021), Machine Learning: Algorithms, Real-World Applications and Research Directions, *SN Computer Science*, Vol. 2, Art. 160. https://doi.org/10.1007/s42979-021-00592-x.

7. Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021), Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, *Journal of Big Data*, No. 8, Art. 53. https://doi.org/10.1186/s40537-021-00444-8.

8. Jhaveri, R.H., Revathi, A., Ramana, K., Raut, R. and Dhanaraj, R.K. (2022), A Review on Machine Learning Strategies for Real-World Engineering Applications, *Mobile Information Systems*, Vol. 2022, Art. 1833507, 26 p. https://doi.org/10.1155/2022/1833507.

9. Ray, S. (2019), A Quick Review of Machine Learning Algorithms, *In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, IEEE, pp. 35–39. https://doi.org/10.1109/COMITCon.2019.8862451.

10. Saraswat, P. (2022), Supervised Machine Learning Algorithm: A Review of Classification Techniques. In: García Márquez, F.P. (eds), *International Conference on Intelligent Emerging Methods of Artificial Intelligence & Cloud Computing. IEMAICLOUD 2021*, Smart Innovation, Systems and Technologies, vol 273, Springer, Cham, pp. 477–482. https://doi.org/10.1007/978-3-030-92905-3_58.

11. Mohalder, R.N., Hossain, Md.A. and Hossain, N. (2024), Classifying the supervised machine learning and comparing the performances of the algorithms, *International Journal of Advanced Research*, No. 12(1), pp. 422–438. https://doi.org/10.21474/IJAR01/18138.

12. Gupta, V., Mishra, V.K., Singhal, P. and Kumar, A. (2022), An Overview of Supervised Machine Learning Algorithm, *In Proceedings of the 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*, IEEE, pp. 87–92. https://doi.org/10.1109/SMART55829.2022.10047618.

13. Uddin, S., Khan, A., Hossain, M.E. and Moni, M.A. (2019), Comparing different supervised machine learning algorithms for disease prediction, *BMC Medical Informatics and Decision Making*, No. 19, Art. 281. https://doi.org/10.1186/s12911-019-1004-8.

14. Van Engelen, J.E. and Hoos, H.H. (2020), A survey on semi-supervised learning, *Machine Learning*, Vol. 109, pp. 373–440. https://doi.org/10.1007/s10994-019-05855-6.

15. Lee, V.L.S., Gan, K.H., Tan, T.P. and Abdullah, R. (2019), Semi-supervised Learning for Sentiment Classification using Small Number of Labeled Data, *Procedia Computer Science*, Vol. 161, pp. 577–584. https://doi.org/10.1016/j.procs.2019.11.159.

16. Huo, H., Rong, Z., Kononova, O., Sun, W., Botari, T., He, T., Tshitoyan, V. and Ceder, G. (2019), Semi-supervised machine-learning classification of materials synthesis procedures, *npj Computational Materials*, No. 5, Art. 62. https://doi.org/10.1038/s41524-019-0204-1.

17. Ezugwu, A.E., Ikotun, A.M., Oyelade, O.O., Abualigah, L., Agushaka, J.O., Eke, C.I. and Akinyelu, A.A. (2022), A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects, *Engineering Applications of Artificial Intelligence*, Vol. 110, Art. 104743. https://doi.org/10.1016/

j.engappai.2022.104743.

18. Pitafi, S., Anwar, T. and Sharif, Z. (2023), A Taxonomy of Machine Learning Clustering Algorithms, Challenges, and Future Realms, *Applied Sciences*, Vol. 13, No. 6, Art. 3529. https://doi.org/10.3390/app13063529.

19. Li, Y. and Wu, H. (2012), A Clustering Method Based on K-Means Algorithm, *Physics Procedia*, Vol. 25, pp. 1104–1109. https://doi.org/10.1016/j.phpro.2012.03.206.

20. Yang, X., Wang, Z., Zhang, H., Ma, N., Yang, N., Liu, H., Zhang, H. and Yang, L. (2022), A Review: Machine Learning for Combinatorial Optimization Problems in Energy Areas, *Algorithms*, Vol. 5, No. 6, Art. 205. https://doi.org/10.3390/a15060205.

21. Wu, X., Cheng, C., Zurita-Milla, R. and Song, C. (2020). An overview of clustering methods for geo-referenced time series: from one-way clustering to co- and tri-clustering, *International Journal of Geographical Information Science*, Vol. 34, No. 9, pp. 1822–1848. https://doi.org/10.1080/13658816.2020.1726922.

22. Mohammed, Z., Hanae, C. and Larbi, C. (2020), Comparative study on machine learning algorithms for early fire forest detection system using geodata, *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 10, No. 5, pp. 5507–5513. https://doi.org/10.11591/ijece.v10i5.pp5507-5513.

23. Xuan, H., Wu, L. and Ye, Y. (2019), A review on dimensionality reduction techniques, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 33, No. 10, Art. 1950017. https://doi.org/10.1142/S0218001419500174.

24. Wang, H.-Y., Wang, J.-S. and Wang, G. (2022), A Survey of Fuzzy Clustering Validity Evaluation Methods, *Information Sciences*, Vol. 618, pp. 270–297. https://doi.org/10.1016/j.ins.2022.11.010.

25. Shutaywi, M. and Kachouie, N.N. (2021), Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering, *Entropy*, Vol. 23, No. 6, Art. 759. https://doi.org/10.3390/e23060759.

Список літератури

1. Sakakibara K., Tamaki H., Nishikawa I. Autonomous distributed approaches for pickup and delivery problems with time windows. *SICE Annual Conference 2007* : conference paper. IEEE, 2007. P. 2639–2642.

2. Dupas R., Grebennik I., Litvinchev I., Romanova T., Chorna O. Solution strategy for one-to-one pickup and delivery problem using the cyclic transfer approach. *EAI Endorsed Transactions on Energy Web*. 2020. Vol. 20. No. 27. P. 1–9.

3. Grebennik I., Chorna O., Urniaieva I. Distribution of Permutations with Different Cyclic Structure in Mathematical Models of Transportation Problems. *2022 12th International Conference on Advanced Computer Information Technologies (ACIT)* : conference paper. IEEE, 2022. P. 18–21.

4. Naeem S., Ali A., Anam S., Ahmed M. M. An Unsupervised Machine Learning Algorithms: Comprehensive Review. *International Journal of Computing and Digital Systems*. 2023. Vol. 13. No. 1. P. 911–921.

5. Sharifani K., Amini M. Machine Learning and Deep Learning: A Review of Methods and Applications. *World Information Technology and Engineering Journal*. 2023. Vol. 10. No. 07. P. 3897–3904.

6. Sarker I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*. 2021. Vol. 2. Art. 160.

7. Alzubaidi L., Zhang J., Humaidi A. J., Al-Dujaili A., Duan Y., Al-Shamma O., Santamaría J., Fadhel M. A., Al-Amidie M., Farhan L. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*. 2021. No. 8, Art. 53.

8. Jhaveri R. H., Revathi A., Ramana K., Raut R., Dhanaraj R. K. A Review on Machine Learning Strategies for Real-World Engineering Applications. *Mobile Information Systems*. 2022. Vol. 2022. Art. 1833507. 26 p.

9. Ray S. A Quick Review of Machine Learning Algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)* : conference paper. IEEE, 2019. P. 35–39.

10. Saraswat P. Supervised Machine Learning Algorithm: A Review of Classification Techniques. In: García Márquez, F.P. (Eds.) *International Conference on Intelligent Emerging Methods of Artificial Intelligence & Cloud Computing. IEMAICLOUD 2021*. Smart Innovation, Systems and Technologies, vol 273. Cham : Springer, 2022. P. 477–482.

11. Mohalder R. N., Hossain Md. A., Hossain N. Classifying the supervised machine learning and comparing the performances of the algorithms. *International Journal of Advanced Research*. 2024. No. 12(1). P. 422–438.

12. Gupta V., Mishra V. K., Singhal P., Kumar A. An Overview of Supervised Machine Learning Algorithm. *11th International Conference on System Modeling & Advancement in Research Trends (SMART)* : conference paper. IEEE, 2022. P. 87–92.

13. Uddin S., Khan A., Hossain M. E., Moni M. A. Comparing different supervised machine learning algorithms for disease prediction. *BMC Medical Informatics and Decision Making*. 2019. No. 19. Art. 281.

14. Van Engelen J. E., Hoos H. H. A survey on semi-supervised learning. *Machine Learning*. 2020. Vol. 109. P. 373–440.

15. Lee V. L. S., Gan K. H., Tan T. P., Abdullah R. Semi-supervised Learning for Sentiment Classification using Small Number of Labeled Data. *Procedia Computer Science*. 2019. Vol. 161. P. 577–584.

16. Huo H., Rong Z., Kononova O., Sun W., Botari T., He T., Tshitoyan V., Ceder G. Semi-supervised machine-learning classification of materials synthesis procedures. *npj Computational Materials*. 2019. No. 5. Art. 62.

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

18. Pitafi S., Anwar T., Sharif Z. A Taxonomy of Machine Learning Clustering Algorithms, Challenges, and Future Realms. *Applied Sciences*. 2023. Vol. 13. No. 6. Art. 3529.

19. Li Y., Wu H. A Clustering Method Based on K-Means Algorithm. *Physics Procedia*. 2012. Vol. 25. P. 1104–1109.

20. Yang X., Wang Z., Zhang H., Ma N., Yang N., Liu H., Zhang H., Yang L. A Review: Machine Learning for Combinatorial Optimization Problems in Energy Areas. *Algorithms*. 2022. Vol. 5. No. 6. Art. 205.

21. Wu X., Cheng C., Zurita-Milla R., Song C. An overview of clustering methods for geo-referenced time series: from one-way clustering to co- and tri-clustering. *International Journal of Geographical Information Science*. 2020. Vol. 34. No. 9. P. 1822–1848.

22. Mohammed Z., Hanae C., Larbi C. Comparative study on machine learning algorithms for early fire forest detection sys-

tem using geodata. *International Journal of Electrical and Computer Engineering (IJECE)*. 2020. Vol. 10. No. 5. P. 5507–5513.

23. Xuan H., Wu L., Ye Y. A review on dimensionality reduction techniques. *International Journal of Pattern Recognition and Artificial Intelligence*. 2019. Vol. 33. No. 10. Art. 1950017.

24. Wang H.-Y., Wang J.-S., Wang G. A Survey of Fuzzy Clustering Validity Evaluation Methods. *Information Sciences*. 2022. Vol. 618. P. 270–297.

25. Shutaywi M., Kachouie N. N. Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering. *Entropy*. 2021. Vol. 23. No. 6. Art. 759.

Received by Editorial Board 27.02.2024
Signed for Printing 20.05.2024

Відомості про авторів:

Чорна Ольга Сергіївна
кандидат технічних наук
доцент
Харківського національного університету радіоелектроніки,
Харків, Україна
https://orcid.org/0000-0001-6745-8137

Дідик Павло Юрійович
магістр
Харківського національного університету радіоелектроніки,
Харків, Україна
https://orcid.org/0009-0007-3725-8995

Тітов Сергій Володимирович
кандидат технічних наук доцент
доцент
Харківського національного університету радіоелектроніки,
Харків, Україна
https://orcid.org/0000-0003-0910-4415

Тітова Олена Вітольдіївна
кандидат технічних наук доцент
доцент
Харківського національного університету радіоелектроніки,
Харків, Україна
https://orcid.org/0000-0001-8894-2040

Information about the authors:

Olga Chorna
PhD in Engineering
Associate Professor
of Kharkiv University of Radio Electronics,
Kharkiv, Ukraine
https://orcid.org/0000-0001-6745-8137

Pavlo Didyk
Master
of Kharkiv University of Radio Electronics,
Kharkiv, Ukraine
https://orcid.org/0009-0007-3725-8995

Serhii Titov
PhD in Engineering Associate Professor
Associate Professor
of Kharkiv University of Radio Electronics,
Kharkiv, Ukraine
https://orcid.org/0000-0003-0910-4415

Olena Titova
PhD in Engineering Associate Professor
Associate Professor
of Kharkiv University of Radio Electronics,
Kharkiv, Ukraine
https://orcid.org/0000-0001-8894-2040

ВИКОРИСТАННЯ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ АВТОМАТИЗАЦІЇ ПЛАНУВАННЯ МАРШРУТІВ У ЗАДАЧАХ МАРШРУТИЗАЦІЇ ПЕРЕВЕЗЕНЬ

О.С. Чорна, П.Ю. Дідик, С.В. Тітов, О.В. Тітова

Стаття присвячена комплексній оцінці результатів кластеризації, застосованої до географічних населених пунктів України. Для аналізу просторового розподілу населених пунктів були використані різні методи кластеризації, включаючи K-mean, DBSCAN, агломеративний, спектральний та метод BIRCH. Оцінка кожного методу кластеризації передбачала застосування відповідних критеріїв якості, що сприяло глибокому розумінню їхньої ефективності в контексті українських населених пунктів. Результати дослідження дають цінну інформацію про сильні та слабкі сторони кожного методу кластеризації, що сприяє прийняттю обґрунтованих рішень при виборі відповідного методу з урахуванням конкретних географічних характеристик. В роботі надаються практичні рекомендації щодо оптимізації вхідних даних, які використовуються в процесі кластеризації, що підвищує загальну ефективність методології аналізу розрахунків. Це дослідження сприяє розвитку методів кластеризації, адаптованих до географічних наборів даних, з потенційними наслідками для міського планування, регіонального розвитку та географічних інформаційних систем. Отримані результати свідчать про те, що вибір методу кластеризації повинен бути адаптований до конкретних географічних характеристик і цілей аналізу, з урахуванням таких факторів, як компактність кластерів, відокремлення точок даних і обчислювальна ефективність. Надано практичні рекомендації щодо оптимізації вхідних даних і вибору відповідних методів кластеризації, які можуть підвищити ефективність методології аналізу поселень для застосування в містобудуванні, регіональному розвитку та геоінформаційних системах. Оцінка методів кластеризації підкреслює важливість врахування як внутрішніх, так і зовнішніх показників для точного визначення якості кластеризації, що забезпечує надійність і достовірність результатів у реальних умовах. Дослідження підкреслює важливість тонкого налаштування алгоритмів кластеризації та параметрів на основі поведінки оціночних параметрів, визнаючи ітеративний характер процесу кластеризації та потребу в адаптивних підходах. Аналіз результатів демонструє, що різні методи кластеризації можуть демонструвати різну ефективність на різних наборах даних, що підкреслює необхідність ретельного розгляду та експериментів для визначення найбільш ефективного підходу для конкретної проблеми. Майбутні напрямки досліджень можуть включати вивчення гібридних підходів до кластеризації, включення знань про конкретні галузі та вирішення проблем, пов'язаних з масштабованістю і надійністю в реальних застосуваннях.

Ключові слова: *кластеризація, проблема маршрутизації транспортних засобів, проблема забирання та доставки, оцінка кластеризації.*

ДОДАТОК Г

Текст програми

```
from tkinter import *
from tkinter import messagebox
from tkinter import ttk  # Для випадаючих списків (Combobox)
from PIL import ImageTk, Image
import pickle
import re
from sklearn.cluster import (KMeans, DBSCAN, AgglomerativeClustering, SpectralClustering, Birch)
import numpy as np
import mysql.connector
import folium
import webview
import json
import itertools
import googlemaps
from sklearn.metrics import (silhouette_score, davies_bouldin_score,
                             calinski_harabasz_score)
from scipy.spatial.distance import pdist
import os
import colorsys
import random

# Глобальні словники для збереження результатів
cluster_results = {
    "km": {"clusters_with_ramp": None, "clusters_without_ramp": None, "data_with_ramp": None,
           "data_without_ramp": None},
    "db": {"clusters_with_ramp": None, "clusters_without_ramp": None, "data_with_ramp": None,
           "data_without_ramp": None},
    "ag": {"clusters_with_ramp": None, "clusters_without_ramp": None, "data_with_ramp": None,
           "data_without_ramp": None},
    "sp": {"clusters_with_ramp": None, "clusters_without_ramp": None, "data_with_ramp": None,
           "data_without_ramp": None},
    "br": {"clusters_with_ramp": None, "clusters_without_ramp": None, "data_with_ramp": None, "data_without_ramp":
None}
}
button_disabled_color = {"bg": "#555555", "fg": "#cccccc"}  # Сірий
button_enabled_color = {"bg": "#918DA2", "fg": "#ffffff"}  # Синій
os.environ["LOKY_MAX_CPU_COUNT"] = "4"
city_entry = 'None'
latitude_entry = None
longitude_entry = None
root = Tk()
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'root1234',
    'database': 'database'
}
gmaps = googlemaps.Client(key='AIzaSyBzoIZur_pkncvPbYPHsRkYHgtU3s9i86o')
manual_params_with_ramp = {
    "km": None,
    "db": None,
    "ag": None,
    "sp": None,
    "br": None,
}

manual_params_without_ramp = {
    "km": None,
    "db": None,
    "ag": None,
    "sp": None,
    "br": None,
```

```
}

try:
    route = gmaps.directions("Kyiv, Ukraine", "Lviv, Ukraine", mode="driving", departure_time="now")
    print(route)
except Exception as e:
    print("Error:", e)


def build_route_with_google_maps(cluster_points, api_key, color):
    print(f"Total points for routing: {len(cluster_points)}")  # Debugging output
    """
    Будує маршрут через точки кластера, розбиваючи їх на підгрупи, якщо їх більше 25.
    """
    gmaps = googlemaps.Client(key=api_key)
    routes = []

    # Розбиваємо точки на підгрупи
    subgroups = list(split_points(cluster_points, max_waypoints=23))
    print(f"Number of subgroups created: {len(subgroups)}")  # Debugging output

    for idx, subgroup in enumerate(subgroups):
        try:
            print(f"Processing subgroup {idx + 1}/{len(subgroups)} with {len(subgroup)} points")  # Debugging output

            # Якщо підгрупа містить одну точку, будуємо маршрут від неї до себе (для коректності API)
            if len(subgroup) == 1:
                start_point = end_point = subgroup[0]
                route = gmaps.directions(
                    origin=f"{start_point[0]},{start_point[1]}",
                    destination=f"{end_point[0]},{end_point[1]}",
                    mode="driving"
                )
                if route:
                    routes.append((route, color))
                    print(f"Subgroup {idx + 1} (single point) processed successfully.")
                else:
                    print(f"Subgroup {idx + 1} (single point) returned no route.")
                continue

            # Початкова і кінцева точки
            start_point = subgroup[0]
            end_point = subgroup[-1]
            waypoints = subgroup[1:-1]

            # Запит до Google Maps API
            route = gmaps.directions(
                origin=f"{start_point[0]},{start_point[1]}",
                destination=f"{end_point[0]},{end_point[1]}",
                waypoints=[f"{point[0]},{point[1]}" for point in waypoints],
                optimize_waypoints=True,
                mode="driving"
            )
            if route:
                routes.append((route, color))
                print(f"Subgroup {idx + 1} processed successfully.")
            else:
                print(f"Subgroup {idx + 1} returned no route.")
        except Exception as e:
            print(f"Error building route for subgroup {idx + 1}: {e}")

    print(f"Total routes generated: {len(routes)}")  # Debugging output
```

```
    return routes


def build_routes_only(algorithm, api_key):
    global cluster_results
    if cluster_results[algorithm]["clusters_with_ramp"] is None or cluster_results[algorithm][
        "clusters_without_ramp"] is None:
        messagebox.showwarning("Warning", f"No clusters available for {algorithm}. Please run clustering first.")
        return

    if algorithm in cluster_results:
        results = cluster_results[algorithm]
        clusters_with_ramp = results["clusters_with_ramp"]
        clusters_without_ramp = results["clusters_without_ramp"]
        data_with_ramp = results["data_with_ramp"]
        data_without_ramp = results["data_without_ramp"]

        if clusters_with_ramp is not None and clusters_without_ramp is not None:
            def build_routes_for_clusters(cluster_dict, colors):
                routes = []
                for cluster_id, points in cluster_dict.items():
                    if len(points) == 0:
                        print(f"Cluster {cluster_id} has no points. Skipping.")
                        continue
                    print(f"Building routes for cluster {cluster_id} with {len(points)} points")  # Debugging output
                    color = colors[cluster_id % len(colors)]
                    cluster_routes = build_route_with_google_maps(points, api_key, color)
                    routes.append(cluster_routes)
                return routes

            # Перетворення кластерів у словники
            cluster_dict_with_ramp = {}
            for i, cluster_id in enumerate(clusters_with_ramp):
                if cluster_id not in cluster_dict_with_ramp:
                    cluster_dict_with_ramp[cluster_id] = []
                cluster_dict_with_ramp[cluster_id].append([float(data_with_ramp[i][2]), float(data_with_ramp[i][3])])

            cluster_dict_without_ramp = {}
            for i, cluster_id in enumerate(clusters_without_ramp):
                if cluster_id not in cluster_dict_without_ramp:
                    cluster_dict_without_ramp[cluster_id] = []
                cluster_dict_without_ramp[cluster_id].append(
                    [float(data_without_ramp[i][2]), float(data_without_ramp[i][3])])


            # Генерація кольорів для кластерів
            total_clusters = len(set(clusters_with_ramp)) + len(set(clusters_without_ramp))
            all_colors = generate_bright_colors(total_clusters)

            # Розподіл кольорів
            colors_with_ramp = all_colors[:len(set(clusters_with_ramp))]
            colors_without_ramp = all_colors[len(set(clusters_with_ramp)):]  # Решта кольорів

            # Будуємо маршрути
            routes_with_ramp = build_routes_for_clusters(cluster_dict_with_ramp, colors_with_ramp)
            routes_without_ramp = build_routes_for_clusters(cluster_dict_without_ramp, colors_without_ramp)

            # Візуалізація маршрутів
            create_map_with_routes(
                data_with_ramp, clusters_with_ramp, routes_with_ramp, colors_with_ramp,
                data_without_ramp, clusters_without_ramp, routes_without_ramp, colors_without_ramp
            )
```

```
            print(f"Маршрути побудовано для {algorithm}.")
        else:
            print(f"Немає даних для побудови маршрутів для {algorithm}.")
    else:
        print(f"Алгоритм {algorithm} не знайдено в результатах кластеризації.")


def create_map_with_routes(data_with_ramp, clusters_with_ramp, routes_with_ramp, colors_with_ramp,
                           data_without_ramp, clusters_without_ramp, routes_without_ramp, colors_without_ramp):
    """
    Відображає маршрути і кластери для даних з рампою і без рампи.
    """
    # Центр карти
    map_center = [48.379433, 31.165580]  # Центр України
    route_map = folium.Map(location=map_center, zoom_start=6, tiles='OpenStreetMap')

    # Завантаження GeoJSON для кордонів України
    try:
        with open('Ukraine.geojson') as file:
            geojson_data = json.load(file)
        folium.GeoJson(geojson_data, name='Ukraine borders').add_to(route_map)
    except FileNotFoundError:
        print("Файл Ukraine.geojson не знайдено. Карта буде відображена без кордонів.")

    # Візуалізація маршрутів
    def visualize_routes(routes):
        for route_group in routes:
            for route, color in route_group:
                for leg in route[0]["legs"]:
                    for step in leg["steps"]:
                        start = (step["start_location"]["lat"], step["start_location"]["lng"])
                        end = (step["end_location"]["lat"], step["end_location"]["lng"])

                        # Додаємо лінію маршруту з тонким чорним обідком
                        folium.PolyLine([start, end], color="black", weight=4, opacity=1).add_to(route_map)
                        folium.PolyLine([start, end], color=color, weight=4.5, opasity=0.9).add_to(route_map)

    visualize_routes(routes_with_ramp)
    visualize_routes(routes_without_ramp)

    # Додавання точок кластерів
    def add_cluster_markers(data_group, clusters, color_palette, shape):
        for city, cluster_id in zip(data_group, clusters):
            color = color_palette[cluster_id % len(color_palette)]
            if shape == "circle":
                folium.CircleMarker(
                    location=(city[2], city[3]),
                    radius=13,
                    color="black",
                    fill=True,
                    fill_color=color,
                    fill_opacity=0.9,
                    weight=0.5,
                    popup=f"Cluster: {cluster_id + 1}"
                ).add_to(route_map)
                # Додавання цифр поверх кружечків
                folium.Marker(
                    location=(city[2], city[3]),
                    icon=folium.DivIcon(
                        html=f"<div style='font-size: 13px; color: black; font-weight: bold; text-align: center;'>{cluster_id}</div>"
                    )
```

```
            ).add_to(route_map)
        elif shape == "square":
            folium.RegularPolygonMarker(
                location=(city[2], city[3]),
                number_of_sides=4,
                radius=12,
                color="black",
                fill=True,
                fill_color=color,
                fill_opacity=0.9,
                weight=0.5,
                popup=f"Cluster: {cluster_id + 1}"
            ).add_to(route_map)
            # Додавання цифр поверх квадратів
            folium.Marker(
                location=(city[2], city[3]),
                icon=folium.DivIcon(
                    html=f"<div style='font-size: 13px; color: black; font-weight: bold; text-align: center;'>{cluster_id}</div>"
                )
            ).add_to(route_map)


    # Додаємо кластери з рампою (кружечки)
    add_cluster_markers(data_with_ramp, clusters_with_ramp, colors_with_ramp, "circle")
    # Додаємо кластери без рампи (квадратики)
    add_cluster_markers(data_without_ramp, clusters_without_ramp, colors_without_ramp, "square")

    # Збереження і відображення карти
    route_map.save('map_with_routes.html')
    webview.create_window("Map with Routes", "map_with_routes.html", width=800, height=600, resizable=True)
    webview.start()


def split_points(points, max_waypoints=23):
    """
    Розділяє список точок на підгрупи, кожна з яких містить не більше max_waypoints точок.
    """
    for i in range(0, len(points), max_waypoints):
        yield points[i:i + max_waypoints]


def create_ukraine_map(data, clusters_km, clusters_db, clusters_ag, clusters_sp, clusters_br, show_cluster_names):
    map_center = [48.379433, 31.165580]
    map_clusters = folium.Map(location=map_center, zoom_start=6, tiles='OpenStreetMap')
    # Завантаження даних про кордони України з файлу Ukraine.geojson
    with open('Ukraine.geojson') as file:
        geojson_data = json.load(file)
    # Визначення палітри кольорів
    colors = ['red', 'green', 'blue', 'pink', 'brown', 'yellow', 'orange', 'violet', 'black', 'purple']
    folium.GeoJson(geojson_data, name='geojson').add_to(map_clusters)

    # Перевірка кожного кластера на None та створення пустого списку, якщо потрібно
    clusters_km = [] if clusters_km is None else clusters_km
    clusters_db = [] if clusters_db is None else clusters_db
    clusters_ag = [] if clusters_ag is None else clusters_ag
    clusters_sp = [] if clusters_sp is None else clusters_sp
    clusters_br = [] if clusters_br is None else clusters_br
    color = 'yellow'
    # Перебір усіх даних і всіх типів кластерів
    for city, cluster_km, cluster_db, cluster_ag, cluster_sp, cluster_br in itertools.zip_longest(data, clusters_km,
                                                                                                  clusters_db,
                                                                                                  clusters_ag,
```

```
                                                    clusters_sp,
                                                    clusters_br):
        lat, lon = city[2:4]
        cluster_list = []  # Список кластерів для відображення при наведенні на маркер

        if cluster_km is not None:
            color = colors[cluster_km % len(colors)]  # Вибір кольору з палітри за індексом кластеру
            if show_cluster_names:
                cluster_list.append(f"K-Means: {cluster_km + 1}")
            else:
                cluster_list.append(str(cluster_km + 1))  # Використовувати лише цифру без назви кластера
        if cluster_db is not None:
            color = colors[cluster_db % len(colors)]  # Вибір кольору з палітри за індексом кластеру
            if show_cluster_names:
                cluster_list.append(f"DBSCAN: {cluster_db + 1}")
            else:
                cluster_list.append(str(cluster_db + 1))  # Використовувати лише цифру без назви кластера
        if cluster_ag is not None:
            color = colors[cluster_ag % len(colors)]  # Вибір кольору з палітри за індексом кластеру
            if show_cluster_names:
                cluster_list.append(f"Agglomerative:</span>{cluster_ag + 1}")
            else:
                cluster_list.append(str(cluster_ag + 1))  # Використовувати лише цифру без назви кластера
        if cluster_sp is not None:
            color = colors[cluster_sp % len(colors)]  # Вибір кольору з палітри за індексом кластеру
            if show_cluster_names:
                cluster_list.append(f"Spectral: {cluster_sp + 1}")
            else:
                cluster_list.append(str(cluster_sp + 1))  # Використовувати лише цифру без назви кластера
        if cluster_br is not None:
            color = colors[cluster_br % len(colors)]  # Вибір кольору з палітри за індексом кластеру
            if show_cluster_names:
                cluster_list.append(f"Birch: {cluster_br + 1}")
            else:
                cluster_list.append(str(cluster_br + 1))  # Використовувати лише цифру без назви кластера

        folium.CircleMarker(location=[lat, lon], radius=10, color=color, fill=True, fill_color=color).add_to(
            map_clusters)

        # Відцентрування цифри в середині маркера
        style = "font-size: 12pt; text-align: center; display: flex; justify-content: center; align-items: center; width: 100%;
height: 100%;"
        folium.Marker(location=[lat, lon],
                      icon=folium.DivIcon(html=f"<div style='{style}'>{'<br>'.join(cluster_list)}</div>")).add_to(
            map_clusters)

    map_clusters.save('cluster_map.html')
    webview.create_window("Clustered Data", "cluster_map.html", width=800, height=600, resizable=True)
    webview.start()


def open_ukraine_map():
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()
    query = "SELECT * FROM data"
    cursor.execute(query)
    data = cursor.fetchall()
    cursor.close()
    conn.close()
    create_ukraine_map(data, None, None, None, None, None, False)
```

```
def get_data_from_database():
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()
        query = "SELECT * FROM data where oblast_idoblast = 23 or oblast_idoblast = 18 or oblast_idoblast = 20 or
oblast_idoblast = 22"
        # query = "SELECT * FROM data"
        cursor.execute(query)
        data = cursor.fetchall()
        cursor.close()
        conn.close()
        return data

    except mysql.connector.Error as error:
        messagebox.showerror("Помилка підключення до бази даних:", error)
        return


def add_data_to_database():
    city = city_entry.get()
    latitude = latitude_entry.get()
    longitude = longitude_entry.get()

    if city and latitude and longitude:
        try:
            conn = mysql.connector.connect(**db_config)
            cursor = conn.cursor()
            query = "SELECT MAX(idCity) FROM data"
            cursor.execute(query)
            result = cursor.fetchone()
            if result[0]:
                max_id = result[0] + 1
            else:
                max_id = 1
            insert_query = f"INSERT INTO data (idCity, Name, x, y) VALUES ({max_id}, '{city}', {latitude}, {longitude})"
            cursor.execute(insert_query)
            conn.commit()
            cursor.close()
            conn.close()
            messagebox.showinfo("Success", "Data added successfully!")
        except mysql.connector.Error as error:
            messagebox.showerror("Error", f"Error while adding data to database: {error}")
    else:
        messagebox.showwarning("Incomplete Fields", "Please fill in all fields.")


def delete_data_from_database():
    city = city_entry.get()

    if city:
        try:
            conn = mysql.connector.connect(**db_config)
            cursor = conn.cursor()
            delete_query = f"DELETE FROM data WHERE Name = '{city}'"
            cursor.execute(delete_query)
            conn.commit()
            cursor.close()
            conn.close()
            messagebox.showinfo("Success", "Data deleted successfully!")
        except mysql.connector.Error as error:
            messagebox.showerror("Error", f"Error while deleting data from database: {error}")
    else:
```

```
        messagebox.showwarning("Missing City", "Please enter a city name.")


def add_geographical_features(data, rivers, mountains):
    """
    Додає географічні ознаки до даних.
    :param data: Масив координат точок [[lat1, lon1], [lat2, lon2], ...]
    :param rivers: Список координат річок [[lat1, lon1], [lat2, lon2], ...]
    :param mountains: Список координат гір [[lat1, lon1], [lat2, lon2], ...]
    :return: Масив даних з новими ознаками
    """
    enhanced_data = []
    for point in data:
        # Відстань до найближчої річки
        river_distances = [np.linalg.norm(np.array(point) - np.array(river)) for river in rivers]
        min_river_distance = min(river_distances)

        # Відстань до найближчої гори
        mountain_distances = [np.linalg.norm(np.array(point) - np.array(mountain)) for mountain in mountains]
        min_mountain_distance = min(mountain_distances)

        # Додаємо нові ознаки
        enhanced_data.append(list(point) + [min_river_distance, min_mountain_distance])

    return np.array(enhanced_data)


rivers = [
    [50.4501, 30.5234],  # Дніпро, Київ
    [48.6239, 35.2256],  # Дніпро, Дніпро
    [46.4775, 30.7326],  # Дністер, Одеса
    [47.8390, 35.1383],  # Самара, Запоріжжя
    [48.2732, 25.9358],  # Прут, Чернівці
    [49.0000, 31.2000],  # Рос, Центральна Україна
    [48.7000, 37.6000],  # Сіверський Донець, Донбас
    [49.8400, 30.1300],  # Тетерів, Житомир
    [47.8388, 35.1396],  # Дніпро, Запоріжжя
    [46.6524, 32.6178],  # Дніпро, Херсон
    [46.6332, 32.6149],  # Інгулець, Херсон
    [48.7808, 24.1900],  # Дністер, Івано-Франківськ
    [48.9245, 24.7118],  # Бистриця Надвірнянська, Надвірна
    [48.8790, 24.6925]  # Бистриця Солотвинська, Івано-Франківськ
]

mountains = [
    [48.1607, 24.4995],  # Говерла
    [48.1726, 24.5794],  # Петрос
    [48.4011, 23.6054],  # Піп Іван
    [48.4477, 24.3663],  # Довбушанка
    [47.9958, 24.2061],  # Менчул
    [48.0326, 24.4236],  # Бребенескул
    [47.9606, 23.7243],  # Стримба
    [48.5603, 23.2234],  # Красна Гора
    [48.2667, 24.4167],  # Туркул
    [48.1720, 24.6640]  # Хом'як
]


def evaluate_data(rivers, mountains):
    """
    Оцінює кластеризацію для двох груп даних: з рампою і без.
    Зберігає результати в окремі текстові файли та визначає оптимальні параметри для кожної групи.
```

```
"""
# Отримання даних із бази
data = get_data_from_database()
if not data:
    print("Не вдалося отримати дані з бази даних.")
    return {}, {}

# Розподіл даних на дві групи
data_with_ramp = [row for row in data if row[-1] == 2 or row[-1] == 3]  # З рампою або комбіновані
data_without_ramp = [row for row in data if row[-1] == 1 or row[-1] == 3]  # Без рампи або комбіновані
# Вагові коефіцієнти для метрик
metric_weights = {
    "kmeans": {
        'silhouette': 2,
        'calinski_harabasz': 2,
        'davies_bouldin': -1,
        'density_variation': 1,
        'inertia': -2
    },
    "dbscan": {
        'silhouette': 1,
        'calinski_harabasz': 1,
        'davies_bouldin': -2,
        'density_variation': 2,
        'inertia': 0
    },
    "agglomerative": {
        'silhouette': 2,
        'calinski_harabasz': 2,
        'davies_bouldin': -1,
        'density_variation': 1,
        'inertia': 0
    },
    "spectral": {
        'silhouette': 2,
        'calinski_harabasz': 1,
        'davies_bouldin': -1,
        'density_variation': 1,
        'inertia': 0
    },
    "birch": {
        'silhouette': 1,
        'calinski_harabasz': 2,
        'davies_bouldin': -1,
        'density_variation': 1,
        'inertia': -1
    }
}

def calculate_density_variation(data, clusters):
    """
    Обчислює варіативність щільності кластерів.
    """
    unique_clusters = set(clusters)
    if -1 in unique_clusters:
        unique_clusters.remove(-1)  # Виключаємо точки шуму (кластер -1)

    if not unique_clusters:  # Якщо немає жодного кластера
        return 0.0

    densities = []
    for cluster in unique_clusters:
```

```
            cluster_points = data[clusters == cluster]
            if len(cluster_points) > 1:
                distances = pdist(cluster_points)  # Всі парні відстані
                mean_distance = np.mean(distances)
                densities.append(1 / mean_distance if mean_distance > 0 else 0)
            else:
                densities.append(0)  # Якщо кластер складається з однієї точки

        return np.var(densities)

    def select_best_params(results, weights):
        best_score = float('-inf')
        best_params = None

        for params, metrics, clusters in results:
            if any(list(clusters).count(c) == 1 for c in set(clusters)):
                continue  # Пропустити результати з кластерами розміром 1

            # Розрахунок загального балу
            score = sum(
                weights.get(metric, 0) * value
                for metric, value in metrics.items() if metric in weights
            )

            if score > best_score:
                best_score = score
                best_params = params

        return best_params

    def evaluate_single_group(data_group, algorithm):
        """
        Оцінює кластеризацію для однієї групи даних і підбирає метрики залежно від алгоритму.
        """
        if not data_group:
            return {}, []

        numeric_data = np.array([[float(coord) for coord in row[2:4]] for row in data_group])
        numeric_data = add_geographical_features(numeric_data, rivers, mountains)

        # # Вибір метрик залежно від алгоритму
        # metrics = {
        #     "kmeans": ["silhouette", "calinski_harabasz", "inertia"],
        #     "dbscan": ["silhouette", "density_variation", "davies_bouldin"],
        #     "agglomerative": ["silhouette", "calinski_harabasz", "cophenetic"],
        #     "spectral": ["silhouette", "davies_bouldin", "spectral_gap"],
        #     "birch": ["silhouette", "calinski_harabasz", "cluster_diameter"]
        # }

        param_grid = {
            'kmeans': {
                'n_clusters': range(2, 16),
                'n_init': [10, 20],
                'random_state': [42]
            },
            'dbscan': {
                'eps': np.arange(0.2, 1.6, 0.1),
                'min_samples': range(2, 6)
            },
            'agglomerative': {
                'n_clusters': range(2, 16)
            },
```

```
        'spectral': {
            'n_clusters': range(2, 16),
            'random_state': [42],
            'affinity': ['nearest_neighbors', 'rbf']
        },
        'birch': {
            'n_clusters': range(2, 16)
        }
    }

    results = []
    for param_comb in itertools.product(*param_grid[algorithm].values()):
        param_dict = dict(zip(param_grid[algorithm].keys(), param_comb))

        try:
            # Ініціалізація моделі
            if algorithm == 'kmeans':
                model = KMeans(**param_dict)
            elif algorithm == 'dbscan':
                model = DBSCAN(**param_dict)
            elif algorithm == 'agglomerative':
                model = AgglomerativeClustering(**param_dict)
            elif algorithm == 'spectral':
                model = SpectralClustering(**param_dict)
            elif algorithm == 'birch':
                model = Birch(**param_dict)
            else:
                continue

            # Навчання моделі
            clusters = model.fit_predict(numeric_data)

            if len(set(clusters)) < 2:  # Мінімум 2 кластери
                continue

            # Обчислення обраних метрик
            metric_results = {}
            if "silhouette" in metric_weights[algorithm]:
                metric_results["silhouette"] = silhouette_score(numeric_data, clusters)
            if "calinski_harabasz" in metric_weights[algorithm]:
                metric_results["calinski_harabasz"] = calinski_harabasz_score(numeric_data, clusters)
            if "davies_bouldin" in metric_weights[algorithm]:
                metric_results["davies_bouldin"] = davies_bouldin_score(numeric_data, clusters)
            if "density_variation" in metric_weights[algorithm]:
                metric_results["density_variation"] = calculate_density_variation(numeric_data, clusters)
            if "inertia" in metric_weights[algorithm] and hasattr(model, 'inertia_'):
                metric_results["inertia"] = model.inertia_

            results.append((param_dict, metric_results, clusters))

        except Exception:
            continue

    # Вибір найкращих параметрів
    best_params = select_best_params(results, metric_weights[algorithm])
    return best_params, results

# Запис усіх результатів у два файли (з рампою і без рампи)
with open("cluster_results_with_ramp.txt", "w") as file_with_ramp, open("cluster_results_without_ramp.txt",
                                                                        "w") as file_without_ramp:
    summary_with_ramp = []
    summary_without_ramp = []
```

```
        for algo in ["kmeans", "dbscan", "agglomerative", "spectral", "birch"]:
            best_params_with_ramp, results_with_ramp = evaluate_single_group(data_with_ramp, algo)
            best_params_without_ramp, results_without_ramp = evaluate_single_group(data_without_ramp, algo)

            file_with_ramp.write(f"Алгоритм: {algo}\n")
            for params, metrics, clusters in results_with_ramp:
                file_with_ramp.write(f"Параметри: {params}, Метрики: {metrics}\n")
            summary_with_ramp.append((algo, best_params_with_ramp))
            file_with_ramp.write("\n")

            file_without_ramp.write(f"Алгоритм: {algo}\n")
            for params, metrics, clusters in results_without_ramp:
                file_without_ramp.write(f"Параметри: {params}, Метрики: {metrics}\n")
            summary_without_ramp.append((algo, best_params_without_ramp))
            file_without_ramp.write("\n")

        # Додавання підсумкової таблиці до файлів
        file_with_ramp.write("_____\n")
        file_with_ramp.write("Алгоритм | Найкращі параметри\n")
        file_with_ramp.write("_____\n")
        for algo, params in summary_with_ramp:
            file_with_ramp.write(f"{algo} | {params}\n")

        file_without_ramp.write("_____\n")
        file_without_ramp.write("Алгоритм | Найкращі параметри\n")
        file_without_ramp.write("_____\n")
        for algo, params in summary_without_ramp:
            file_without_ramp.write(f"{algo} | {params}\n")

    return summary_with_ramp, summary_without_ramp


def add_geographical_features(data, rivers, mountains):
    """
    Додає географічні ознаки до даних.
    :param data: Масив координат точок [[lat1, lon1], [lat2, lon2], ...]
    :param rivers: Список координат річок [[lat1, lon1], [lat2, lon2], ...]
    :param mountains: Список координат гір [[lat1, lon1], [lat2, lon2], ...]
    :return: Масив даних з новими ознаками
    """
    enhanced_data = []
    for point in data:
        # Відстань до найближчої річки
        river_distances = [np.linalg.norm(np.array(point) - np.array(river)) for river in rivers]
        min_river_distance = min(river_distances)

        # Відстань до найближчої гори
        mountain_distances = [np.linalg.norm(np.array(point) - np.array(mountain)) for mountain in mountains]
        min_mountain_distance = min(mountain_distances)

        # Додаємо нові ознаки
        enhanced_data.append(list(point) + [min_river_distance, min_mountain_distance])

    return np.array(enhanced_data)


def cluster_data(algorithm, best_params_with_ramp, best_params_without_ramp):
    """
    Універсальна функція кластеризації з врахуванням наявності рампи.
    Підтримує різні алгоритми кластеризації.
    """
```

```
print ('aldo', algorithm)
data = get_data_from_database()  # Отримуємо дані з бази
if not data:
    print("Не вдалося отримати дані з бази даних.")
    # Повертаємо порожні результати, щоб уникнути помилок
    return [], [], [], []
    # Мапа скорочених назв алгоритмів на повні
ALGORITHM_KEYS = {
    "km": "kmeans",
    "db": "dbscan",
    "ag": "agglomerative",
    "sp": "spectral",
    "br": "birch"
}
# Отримати повну назву алгоритму
full_algorithm_name = ALGORITHM_KEYS.get(algorithm, algorithm)
print(f"Використовується алгоритм: {full_algorithm_name}")
# Розподіл даних
data_with_ramp = [row for row in data if row[-1] == 2 or row[-1] == 3]  # З рампою або комбіновані
data_without_ramp = [row for row in data if row[-1] == 1 or row[-1] == 3]  # Без рампи або комбіновані
print("best params1", best_params_with_ramp)
print("best params2", best_params_without_ramp)
print("Отримані дані з бази:")
print(f"Дані з рампою: {len(data_with_ramp)} записів")
print(f"Дані без рампи: {len(data_without_ramp)} записів")

# Функція для отримання параметрів для алгоритму
def get_algorithm_params(algorithm_name, params_list):
    for algo, params in params_list:
        if algo == algorithm_name:
            return params
    return {}  # Якщо алгоритм не знайдено

def cluster_group(data_group, algorithm, params):
    if data_group:
        lat_lon_data = np.array([[float(coord) for coord in row[2:4]] for row in data_group])

        # Вивід вхідних даних для кластеризації
        print(f"\n[Алгоритм: {algorithm.upper()}]")
        print(f"Вхідні параметри: {params}")
        print(f"Перших 5 точок даних: {lat_lon_data[:5]}")
        print(f"Кількість точок для кластеризації: {lat_lon_data.shape[0]}")

        # Ініціалізація моделі
        if algorithm == "kmeans":
            model = KMeans(**params)
        elif algorithm == "dbscan":
            model = DBSCAN(**params)
        elif algorithm == "agglomerative":
            model = AgglomerativeClustering(**params)
        elif algorithm == "spectral":
            model = SpectralClustering(**params)
        elif algorithm == "birch":
            model = Birch(**params)
        else:
            raise ValueError(f"Unknown algorithm: {algorithm}")

        # Виконання кластеризації
        clusters = model.fit_predict(lat_lon_data)
        print(f"Кількість унікальних кластерів: {len(set(clusters))}")
        print(f"Кластери: {clusters}")
```

```
            return clusters
        else:
            print(f"[Алгоритм: {algorithm.upper()}] Немає даних для кластеризації.")
            return []

    def create_map_with_shapes(data_with_ramp, data_without_ramp, clusters_with_ramp, clusters_without_ramp,
                               colors_with_ramp, colors_without_ramp):
        """
        Відображає кластери на мапі з різними формами та кольорами.
        """
        map_center = [48.379433, 31.16558]  # Центр України
        map_clusters = folium.Map(location=map_center, zoom_start=6, tiles='OpenStreetMap')

        # Завантаження даних про кордони України
        try:
            with open('Ukraine.geojson') as file:
                geojson_data = json.load(file)
            folium.GeoJson(geojson_data, name='geojson').add_to(map_clusters)
        except FileNotFoundError:
            print("Файл Ukraine.geojson не знайдено. Перевірте шлях до файлу.")

        def add_cluster_points(data_group, clusters, marker_shape, color_palette):
            for city, cluster_id in zip(data_group, clusters):
                color = color_palette[cluster_id - 1] if cluster_id > 0 else "gray"  # Шум сірий
                if marker_shape == "circle":
                    folium.CircleMarker(
                        location=(city[2], city[3]),
                        radius=13,
                        color="black",
                        fill=True,
                        fill_color=color,
                        fill_opacity=0.9,
                        weight=0.5,
                        popup=f"Cluster: {cluster_id}"
                    ).add_to(map_clusters)
                    # Додавання цифр поверх кружечків
                    folium.Marker(
                        location=(city[2], city[3]),
                        icon=folium.DivIcon(
                            html=f"<div style='font-size: 13px; color: black; font-weight: bold; text-align:
center;'>{cluster_id}</div>"
                        )
                    ).add_to(map_clusters)
                elif marker_shape == "square":
                    folium.RegularPolygonMarker(
                        location=(city[2], city[3]),
                        number_of_sides=4,
                        radius=12,
                        color="black",
                        fill=True,
                        fill_color=color,
                        fill_opacity=0.9,
                        weight=0.5,
                        popup=f"Cluster: {cluster_id}"
                    ).add_to(map_clusters)
                    # Додавання цифр поверх квадратів
                    folium.Marker(
                        location=(city[2], city[3]),
                        icon=folium.DivIcon(
                            html=f"<div style='font-size: 13px; color: black; font-weight: bold; text-align:
center;'>{cluster_id}</div>"
                        )
```

```
            ).add_to(map_clusters)

        # Додавання кластерів
        add_cluster_points(data_with_ramp, clusters_with_ramp, "circle", colors_with_ramp)
        add_cluster_points(data_without_ramp, clusters_without_ramp, "square", colors_without_ramp)

        # Збереження та відображення карти
        map_clusters.save('cluster_map_with_shapes.html')
        webview.create_window("Clustered Data", "cluster_map_with_shapes.html", width=800, height=800, resizable=True)
        webview.start()

    # Отримуємо параметри для кластеризації
    params_with_ramp = get_algorithm_params(full_algorithm_name, best_params_with_ramp)
    params_without_ramp = get_algorithm_params(full_algorithm_name, best_params_without_ramp)
    print(f"Параметри для {full_algorithm_name} (з рампою): {params_with_ramp}")
    print(f"Параметри для {full_algorithm_name} (без рампи): {params_without_ramp}")

    # Кластеризація для обох груп
    clusters_with_ramp = cluster_group(data_with_ramp, full_algorithm_name, params_with_ramp)
    clusters_without_ramp = cluster_group(data_without_ramp, full_algorithm_name, params_without_ramp)

    # Перенумерація кластерів
    def renumber_clusters(clusters):
        unique_clusters = sorted(set(clusters) - {-1})  # Ігноруємо шум (-1)
        cluster_mapping = {old: new + 1 for new, old in enumerate(unique_clusters)}
        cluster_mapping[-1] = 0  # Шум залишається 0
        return [cluster_mapping[c] for c in clusters]

    clusters_with_ramp = renumber_clusters(clusters_with_ramp)

    # Окрема нумерація для кластерів без рампи
    def renumber_clusters_separately(clusters):
        unique_clusters = sorted(set(clusters) - {-1})  # Ігноруємо шум (-1)
        cluster_mapping = {old: new + 1 for new, old in enumerate(unique_clusters)}
        cluster_mapping[-1] = 0  # Шум залишається 0
        return [cluster_mapping[c] for c in clusters]

    clusters_without_ramp = renumber_clusters_separately(clusters_without_ramp)

    # Генерація кольорів
    num_clusters_with_ramp = len(set(clusters_with_ramp)) - (1 if 0 in clusters_with_ramp else 0)
    num_clusters_without_ramp = len(set(clusters_without_ramp)) - (1 if 0 in clusters_without_ramp else 0)
    total_clusters = num_clusters_with_ramp + num_clusters_without_ramp

    # Генеруємо кольори для всіх кластерів
    all_colors = generate_bright_colors(total_clusters)

    # Розділяємо кольори для груп
    colors_with_ramp = all_colors[:num_clusters_with_ramp]
    colors_without_ramp = all_colors[num_clusters_with_ramp:]

    # Створення карти з кластерами
    create_map_with_shapes(
        data_with_ramp, data_without_ramp,
        clusters_with_ramp, clusters_without_ramp,
        colors_with_ramp, colors_without_ramp
    )

    return clusters_with_ramp, clusters_without_ramp, data_with_ramp, data_without_ramp


def generate_bright_colors(num_colors):
```

```
    """
    Генерує яскраві унікальні кольори в HEX-форматі.
    """
    colors = set()
    max_attempts = num_colors * 20  # Збільшуємо кількість спроб для уникнення повторів

    for _ in range(max_attempts):
        if len(colors) >= num_colors:
            break

        # Генеруємо псевдовипадковий hue
        hue = random.uniform(0, 1)
        saturation = 0.8
        lightness = 0.5
        rgb = colorsys.hls_to_rgb(hue, lightness, saturation)
        hex_color = '#{:02x}{:02x}{:02x}'.format(int(rgb[0] * 255), int(rgb[1] * 255), int(rgb[2] * 255))

        # Перевіряємо, чи колір унікальний
        if hex_color not in colors:
            colors.add(hex_color)

    # Додаємо резервні кольори, якщо базових не вистачило
    fallback_colors = ["#FF0000", "#00FF00", "#0000FF", "#FFFF00", "#FF00FF", "#00FFFF", "#FFFFFF", "#000000"]
    while len(colors) < num_colors:
        for fallback_color in fallback_colors:
            if fallback_color not in colors:
                colors.add(fallback_color)
            if len(colors) >= num_colors:
                break

    # Якщо навіть після резерву кольорів недостатньо
    if len(colors) < num_colors:
        raise ValueError(f"Недостатньо доступних кольорів. Згенеровано {len(colors)} з {num_colors}.")

    return list(colors)




def update_clusters(algorithm, best_params_with_ramp, best_params_without_ramp):
    global cluster_results, manual_params_with_ramp, manual_params_without_ramp

    # Визначаємо параметри для кластеризації
    params_to_use_with_ramp = (
        best_params_with_ramp if manual_params_with_ramp[algorithm] is None else
manual_params_with_ramp[algorithm]
    )
    params_to_use_without_ramp = (
        best_params_without_ramp if manual_params_without_ramp[algorithm] is None else manual_params_without_ramp[
            algorithm]
    )
    print(params_to_use_with_ramp)
    print(params_to_use_without_ramp)
    # Виконуємо кластеризацію
    try:
        clusters_with_ramp, clusters_without_ramp, data_with_ramp, data_without_ramp = cluster_data(
            algorithm, params_to_use_with_ramp, params_to_use_without_ramp
        )

        if clusters_with_ramp is not None and clusters_without_ramp is not None:
            cluster_results[algorithm]["clusters_with_ramp"] = clusters_with_ramp
            cluster_results[algorithm]["clusters_without_ramp"] = clusters_without_ramp
```

```
            cluster_results[algorithm]["data_with_ramp"] = data_with_ramp
            cluster_results[algorithm]["data_without_ramp"] = data_without_ramp

            enable_route_buttons(algorithm)

        else:
            messagebox.showerror("Error", f"Не вдалося виконати кластеризацію для {algorithm}.")
    except Exception as e:
        messagebox.showerror("Error", f"Помилка кластеризації для {algorithm}: {e}")




def click_reg():
    width = 540
    height = 580
    frame1 = Frame(root)
    frame1.place(x=0, y=0, relheight=1, relwidth=1)
    img = Image.open("bg.webp")
    img = img.resize((width, height), Image.LANCZOS)
    bg_img = ImageTk.PhotoImage(img)
    bg_label = Label(frame1, image=bg_img)
    bg_label.image = bg_img
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)
    inputlog_lable = Label(frame1, text="Enter your login:", font="Arial 15", bg="grey77",
                           width=20, relief=RAISED, bd=5)
    inputlog_lable.place(x=15, y=15)
    temp_log_in1 = lambda name: log_in1.delete(0, "end")
    log_in1 = Entry(frame1, bg="SkyBlue", width=20, font="Arial 15")
    log_in1.insert(0, "Login")
    log_in1.bind("<FocusIn>", temp_log_in1)
    log_in1.place(x=15, y=65)
    inputpas_lable = Label(frame1, text="Enter your password:", font="Arial 15", bg="grey77",
                           width=20, relief=RAISED, bd=5)
    inputpas_lable.place(x=15, y=115)
    temp_pas_in1 = lambda name: pas_in1.delete(0, "end")
    pas_in1 = Entry(frame1, bg="SkyBlue", width=20, font="Arial 15", show="*")
    pas_in1.insert(0, "Password")
    pas_in1.bind("<FocusIn>", temp_pas_in1)
    pas_in1.place(x=15, y=165)
    inputpas1_lable = Label(frame1, text="Repeat your password:", font="Arial 15", bg="grey77",
                            width=20, relief=RAISED, bd=5)
    inputpas1_lable.place(x=15, y=215)
    temp_pas_in2 = lambda name: pas_in2.delete(0, "end")
    pas_in2 = Entry(frame1, bg="SkyBlue", width=20, font="Arial 15", show="*")
    pas_in2.insert(0, "Password")
    pas_in2.bind("<FocusIn>", temp_pas_in2)
    pas_in2.place(x=15, y=265)
    but_reg2 = Button(frame1, text='Registration',
                      command=lambda: save(log_in1.get(), pas_in1.get(), pas_in2.get()),
                      activebackground="grey40",
                      font='Arial 15', bg='grey77', width=10, relief=RAISED, bd=5)
    but_reg2.place(x=15, y=315)
    but_back = Button(frame1, text='Back', command=lambda: main_win(), activebackground="grey40",
                      font='Arial 15', bg='grey77', width=10, relief=RAISED, bd=5)
    but_back.place(x=15, y=365)


def click_log(log_in, pas_in):
    try:
        f = open("login.txt", "rb")
```

```
        login_pass_save = pickle.load(f)
        f.close()

        if log_in in login_pass_save:
            if pas_in == login_pass_save[log_in]:
                messagebox.showinfo("Successful", "Hello", type=messagebox.OK)
                root.withdraw()
                open_main_win()
            else:
                messagebox.showerror("er", "Invalid password")
        else:
            messagebox.showerror("erlog", "Invalid login")
    except FileNotFoundError:
        messagebox.showerror("erlog", "Invalid login")


# Глобальна змінна для зберігання найкращих параметрів
best_params = {}


# Функція для оцінки всіх алгоритмів і збереження результатів
def evaluate_and_store():
    """
    Проводить оцінку кластеризації для даних з рампою і без рампи та зберігає результати.
    """
    global best_params_with_ramp, best_params_without_ramp

    # Виклик оновленої функції evaluate_data
    best_params_with_ramp, best_params_without_ramp = evaluate_data(rivers, mountains)

    print("Оцінка завершена. Оптимальні параметри збережено.")

    # Відображення результатів
    print("Результати кластеризації з рампою:")
    for algo, params in best_params_with_ramp:
        print(f"Алгоритм: {algo}, Найкращі параметри: {params}")

    print("\nРезультати кластеризації без рампи:")
    for algo, params in best_params_without_ramp:
        print(f"Алгоритм: {algo}, Найкращі параметри: {params}")

    # Автоматичне відкриття згенерованих файлів
    try:
        os.startfile("cluster_results_with_ramp.txt")
        os.startfile("cluster_results_without_ramp.txt")
    except AttributeError:
        os.system("open cluster_results_with_ramp.txt")  # Для macOS/Linux
        os.system("open cluster_results_without_ramp.txt")


def open_main_win():
    global main_window, but_km, but_db, but_ag, but_sp, but_br
    global but_km_routes, but_db_routes, but_ag_routes, but_sp_routes, but_br_routes
    global button_disabled_color, button_enabled_color

    width = 1855
    height = 975
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x = (screen_width // 2) - (width // 2)
    y = (screen_height // 2) - (height // 2)-50
    global main_window
```

```
main_window = Toplevel(root)
main_window.title("Головне вікно")
main_window.geometry(f"{width}x{height}+{x}+{y}")
main_window.resizable(width=False, height=False)

frame = Frame(main_window)
frame.place(x=0, y=0, relheight=1, relwidth=1)

img = Image.open("bg_main.webp")
img = img.resize((width, height), Image.LANCZOS)
bg_img = ImageTk.PhotoImage(img)

bg_label = Label(frame, image=bg_img)
bg_label.image = bg_img
bg_label.place(x=0, y=0, relwidth=1, relheight=1)

# Завантаження зображення
param_image = Image.open("bg_param.webp")  # Замініть шлях до зображення
# param_image = param_image.resize((width, height), Image.LANCZOS)
param_image = param_image.resize((620, 430), Image.LANCZOS)  # Розмір секції 500x465
param_photo = ImageTk.PhotoImage(param_image)

# Додати секцію для введення параметрів
frame_manual_params = Frame(frame, bg="white", relief=RAISED, bd=2)
frame_manual_params.place(x=70, y=30, width=620, height=430)

# Вставити зображення в секцію
param_label = Label(frame_manual_params, image=param_photo, bg="white")
param_label.image = param_photo  # Зберігаємо посилання, щоб уникнути видалення
param_label.place(x=0, y=0, relwidth=1, relheight=1)

# # Додати заголовок поверх зображення
# Label(frame_manual_params, text="Введення параметрів вручну",
#       font='Arial 12 bold', bg="white").place(x=10, y=10)

# Додати елементи керування поверх зображення
add_manual_param_controls(frame_manual_params)

def add_info_tooltip(widget, text):
    def show_tooltip(event):
        tooltip = Toplevel(widget)
        tooltip.wm_overrideredirect(True)
        tooltip.geometry(f"+{event.x_root + 10}+{event.y_root + 10}")
        Label(tooltip, text=text, bg="lightyellow", relief="solid", borderwidth=1, font="Arial 10").pack()
        widget.tooltip = tooltip

    def hide_tooltip(event):
        if hasattr(widget, 'tooltip'):
            widget.tooltip.destroy()
            del widget.tooltip

    widget.bind("<Enter>", show_tooltip)
    widget.bind("<Leave>", hide_tooltip)

# City Entry
city_label = Label(frame, text="Місто:", font='Arial 12', bg='white')
city_label.place(x=1480, y=60)

global city_entry
city_entry = Entry(frame, font='Arial 12', width=20)
city_entry.place(x=1480, y=90)
```

```
# Latitude Entry
latitude_label = Label(frame, text="Широта:", font='Arial 12', bg='white')
latitude_label.place(x=1480, y=150)

global latitude_entry
latitude_entry = Entry(frame, font='Arial 12', width=20)
latitude_entry.place(x=1480, y=180)

# Longitude Entry
longitude_label = Label(frame, text="Довгота:", font='Arial 12', bg='white')
longitude_label.place(x=1480, y=210)

global longitude_entry
longitude_entry = Entry(frame, font='Arial 12', width=20)
longitude_entry.place(x=1480, y=240)

add_button = Button(frame, text='Додати', command=add_data_to_database,
                    font='Arial 12', width=19, relief=RAISED, bd=5, state="normal", bg=button_enabled_color["bg"],
                    fg=button_enabled_color["fg"])
add_button.place(x=1480, y=290)

delete_button = Button(frame, text='Видалити', command=delete_data_from_database,
                       font='Arial 12', width=19, relief=RAISED, bd=5, state="normal",
                       bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])
delete_button.place(x=1480, y=340)



but_evaluate_all = Button(frame, text='Провести оцінку алгоритмів',
                          command=lambda: evaluate_and_enable_clustering_buttons(),
                          font='Arial 14', width=22, relief=RAISED, bd=5, state="normal",
                          bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])
but_evaluate_all.place(x=70, y=870)
add_info_tooltip(but_evaluate_all, "Аналізує всі підходи для побудови маршрутів.")

but_km = Button(frame, text='Метод K-Means',
                command=lambda: update_clusters("km", best_params_with_ramp, best_params_without_ramp),
                font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled", bg=button_disabled_color["bg"],
                fg=button_disabled_color["fg"])
but_km.place(x=70, y=730)
add_info_tooltip(but_km, "Оптимізує маршрути, розподіляючи їх на групи за близькістю точок.")


but_db = Button(frame, text='Метод DBSCAN',
                command=lambda: update_clusters("db", best_params_with_ramp, best_params_without_ramp),
                font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled", bg=button_disabled_color["bg"],
                fg=button_disabled_color["fg"])
but_db.place(x=430, y=730)
add_info_tooltip(but_db, "Розраховує маршрути для густих груп точок.")

but_ag = Button(frame, text='Метод Agglomerative',
                command=lambda: update_clusters("ag", best_params_with_ramp, best_params_without_ramp),
                font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled", bg=button_disabled_color["bg"],
                fg=button_disabled_color["fg"])
but_ag.place(x=770, y=730)
add_info_tooltip(but_ag, "Групує точки для маршрутів на основі ієрархічного підходу.")

but_sp = Button(frame, text='Метод Spectral',
                command=lambda: update_clusters("sp", best_params_with_ramp, best_params_without_ramp),
                font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled", bg=button_disabled_color["bg"],
                fg=button_disabled_color["fg"])
but_sp.place(x=1110, y=730)
```

```
    add_info_tooltip(but_sp, "Оптимізує маршрути на основі спектрального розкладу даних.")

    but_br = Button(frame, text='Метод Birch',
                    command=lambda: update_clusters("br", best_params_with_ramp, best_params_without_ramp),
                    font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled", bg=button_disabled_color["bg"],
                    fg=button_disabled_color["fg"])
    but_br.place(x=1450, y=730)
    add_info_tooltip(but_br, "Розподіляє маршрути для великої кількості точок.")

    open_ukraine_map_button = Button(frame, text='Мапа України',
                                     command=open_ukraine_map,
                                     font='Arial 14', width=22, relief=RAISED, bd=5, state="normal",
                                     bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])
    open_ukraine_map_button.place(x=1450, y=870)
    add_info_tooltip(open_ukraine_map_button, "Відкриває мапу з усіма містами України, що додані до програми")

    but_km_routes = Button(frame, text='Побудувати маршрути K-Means',
                           command=lambda: build_routes_only("km", "AIzaSyBzoIZur_pkncvPbYPHsRkYHgtU3s9i86o"),
                           font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled",
                           bg=button_disabled_color["bg"], fg=button_disabled_color["fg"])
    but_km_routes.place(x=70, y=800)
    add_info_tooltip(but_km_routes, "Будує оптимальні маршрути для обраного підходу K-Means.")

    but_db_routes = Button(frame, text='Побудувати маршрути DBSCAN',
                           command=lambda: build_routes_only("db", "AIzaSyBzoIZur_pkncvPbYPHsRkYHgtU3s9i86o"),
                           font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled",
                           bg=button_disabled_color["bg"], fg=button_disabled_color["fg"])
    but_db_routes.place(x=430, y=800)
    add_info_tooltip(but_db_routes, "Будує маршрути для густих груп точок DBSCAN.")

    but_ag_routes = Button(frame, text='Побудувати маршрути Agglomerative',
                           command=lambda: build_routes_only("ag", "AIzaSyBzoIZur_pkncvPbYPHsRkYHgtU3s9i86o"),
                           font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled",
                           bg=button_disabled_color["bg"], fg=button_disabled_color["fg"])
    but_ag_routes.place(x=770, y=800)
    add_info_tooltip(but_ag_routes, "Побудова маршрутів на основі ієрархічної кластеризації.")

    but_sp_routes = Button(frame, text='Побудувати маршрути Spectral',
                           command=lambda: build_routes_only("sp", "AIzaSyBzoIZur_pkncvPbYPHsRkYHgtU3s9i86o"),
                           font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled",
                           bg=button_disabled_color["bg"], fg=button_disabled_color["fg"])
    but_sp_routes.place(x=1110, y=800)
    add_info_tooltip(but_sp_routes, "Будує маршрути з використанням спектрального аналізу.")

    but_br_routes = Button(frame, text='Побудувати маршрути Birch',
                           command=lambda: build_routes_only("br", "AIzaSyBzoIZur_pkncvPbYPHsRkYHgtU3s9i86o"),
                           font='Arial 14', width=22, relief=RAISED, bd=5, state="disabled",
                           bg=button_disabled_color["bg"], fg=button_disabled_color["fg"])
    but_br_routes.place(x=1450, y=800)
    add_info_tooltip(but_br_routes, "Будує маршрути з використанням методу для великої кількості Birch.")


def unlock_buttons():
    # Список кнопок, які потрібно розблокувати
    buttons = [but_km, but_db, but_ag, but_sp, but_br]
    for button in buttons:
        button.config(state="normal", bg=button_enabled_color["bg"],
                      fg=button_enabled_color["fg"])  # Активні кнопки – яскраві


def evaluate_and_enable_clustering_buttons():
    global best_params_with_ramp, best_params_without_ramp
```

```
    # Викликаємо основну функцію оцінки
    evaluate_and_store()
    unlock_buttons()  # Розблокування кнопок


def enable_route_buttons(algorithm):
    global cluster_results
    if cluster_results[algorithm]["clusters_with_ramp"] is not None and cluster_results[algorithm][
        "clusters_without_ramp"] is not None:
        # Розблоковуємо відповідні кнопки для маршрутів
        if algorithm == "km":
            but_km_routes.config(state="normal", bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])
        elif algorithm == "db":
            but_db_routes.config(state="normal", bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])
        elif algorithm == "ag":
            but_ag_routes.config(state="normal", bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])
        elif algorithm == "sp":
            but_sp_routes.config(state="normal", bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])
        elif algorithm == "br":
            but_br_routes.config(state="normal", bg=button_enabled_color["bg"], fg=button_enabled_color["fg"])

    else:
        messagebox.showwarning("Warning", f"Clustering for {algorithm} is not completed. Please run clustering first.")


def main_win():
    width = 540
    height = 580
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x = (screen_width // 2) - (width // 2)
    y = (screen_height // 2) - (height // 2)
    root.title("Login")
    root.geometry(f"{width}x{height}+{x}+{y}")
    root.resizable(width=False, height=False)
    root.iconbitmap("icon.ico")
    frame = Frame(root)
    frame.place(x=0, y=0, relheight=1, relwidth=1)
    # Завантаження та зміна розміру зображення
    img = Image.open("bg.webp")
    img = img.resize((width, height), Image.LANCZOS)
    bg_img = ImageTk.PhotoImage(img)
    # Створення зображення на фоні frame
    bg_label = Label(frame, image=bg_img)
    bg_label.image = bg_img
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)
    but_log = Button(frame, text='Login', font='Arial 15',
                     command=lambda: click_log(log_in.get(), pas_in.get()),
                     bg='grey77', width=10, relief=RAISED, bd=5, activebackground="grey40")  #
    but_log.place(x=80, y=450)
    but_reg = Button(frame, text='Go to reg', command=click_reg, activebackground="grey40",
                     font='Arial 15', bg='grey77', width=10, relief=RAISED, bd=5)  #
    but_reg.place(x=320, y=450)
    log_label = Label(frame, text="Login", font="Arial 35", bg="grey77",
                      width=10, relief=RAISED, bd=5)
    log_label.place(x=125, y=25)
    temp_log_in = lambda name: log_in.delete(0, "end")
    log_in = Entry(frame, bg="SkyBlue", width=20, font="Arial 25")
    log_in.insert(0, "Login")
    log_in.bind("<FocusIn>", temp_log_in)
    log_in.place(x=70, y=190)
    temp_pas_in = lambda name: pas_in.delete(0, "end")
```

```
    pas_in = Entry(frame, bg="SkyBlue", width=20, font="Arial 25", show="*")
    pas_in.insert(0, "Password")
    pas_in.bind("<FocusIn>", temp_pas_in)
    pas_in.place(x=70, y=250)


def save(reg_log, pas, reg_pas):
    try:
        f = open("login.txt", "rb")
        login_pass_save = pickle.load(f)
        f.close()
    except FileNotFoundError:
        login_pass_save = {}
    if not re.match("^[a-zA-Z0-9]{1,10}$", reg_log):
        messagebox.showerror("Error",
                             "Invalid username format. Please use only latin letters and digits (1-10 characters).")
        return

    if reg_log in login_pass_save:
        messagebox.showerror("Error", "Username already exists")
        return

    if not re.match("^[a-zA-Z0-9]{1,10}$", reg_pas):
        messagebox.showerror("Error",
                             "Invalid password format. Please use only latin letters and digits (1-10 characters).")
        return

    if pas != reg_pas:
        messagebox.showerror("Error", "Passwords do not match")
        return

    login_pass_save[reg_log] = reg_pas
    f = open("login.txt", "wb")
    pickle.dump(login_pass_save, f)
    f.close()
    main_win()


# Додати поля введення і кнопки в GUI
def add_manual_param_controls(frame):
    global kmeans_n_clusters_with_ramp, kmeans_n_init_with_ramp, kmeans_n_clusters_without_ramp,
kmeans_n_init_without_ramp
    global dbscan_eps_with_ramp, dbscan_min_samples_with_ramp, dbscan_eps_without_ramp,
dbscan_min_samples_without_ramp
    global agglomerative_n_clusters_with_ramp, agglomerative_n_clusters_without_ramp
    global spectral_n_clusters_with_ramp, spectral_affinity_with_ramp, spectral_n_clusters_without_ramp,
spectral_affinity_without_ramp
    global birch_n_clusters_with_ramp, birch_n_clusters_without_ramp


    col_offset = 0
    row_offset = 1

    # Додати заголовок "Налаштування параметрів вручну"
    Label(frame, text="Налаштування параметрів вручну", font="Arial 14 bold", bg="#C0C0C0", fg="#000000") \
        .grid(row=0, column=0, columnspan=5, pady=10)

    def add_info_icon(row, col, text):
        """Додає іконку інформації з підказкою при наведенні."""
        icon = Label(frame, text="i", font="Arial 12 bold", fg="blue", cursor="hand2")
        icon.grid(row=row, column=col, padx=5)
```

```
        # Функція для створення підказки
        def show_tooltip(event):
            tooltip = Toplevel(frame)
            tooltip.wm_overrideredirect(True)  # Прибираємо рамку
            tooltip.geometry(f"+{event.x_root + 10}+{event.y_root + 10}")
            Label(tooltip, text=text, bg="lightyellow", relief="solid", borderwidth=1, font="Arial 10").pack()
            icon.tooltip = tooltip

        # Функція для видалення підказки
        def hide_tooltip(event):
            if hasattr(icon, 'tooltip'):
                icon.tooltip.destroy()
                del icon.tooltip

        # Прив'язуємо події до іконки
        icon.bind("<Enter>", show_tooltip)
        icon.bind("<Leave>", hide_tooltip)

    # Випадаючі списки для K-Means (з рампою і без рампи)
    Label(frame, text="Вхідні дані великогабарит K-Means", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    kmeans_n_clusters_with_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    kmeans_n_clusters_with_ramp.grid(row=row_offset, column=col_offset + 1)
    kmeans_n_clusters_with_ramp.set("2")
    kmeans_n_init_with_ramp = ttk.Combobox(frame, values=[10, 20], width=10)
    kmeans_n_init_with_ramp.grid(row=row_offset, column=col_offset + 2)
    kmeans_n_init_with_ramp.set("10")
    add_info_icon(row_offset, col_offset + 3, "Кількість груп впливає на деталізацію маршрутів. Більша кількість груп
забезпечує більше варіантів, але може створювати надмірну роздрібленість. Другий параметр впливає на точність
вимірювань, більша кількість - вища точність")

    row_offset += 1
    Label(frame, text="Вхідні дані малогабарит K-Means", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    kmeans_n_clusters_without_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    kmeans_n_clusters_without_ramp.grid(row=row_offset, column=col_offset + 1)
    kmeans_n_clusters_without_ramp.set("2")
    kmeans_n_init_without_ramp = ttk.Combobox(frame, values=[10, 20], width=10)
    kmeans_n_init_without_ramp.grid(row=row_offset, column=col_offset + 2)
    kmeans_n_init_without_ramp.set("10")
    add_info_icon(row_offset, col_offset + 3, "Кількість груп впливає на деталізацію маршрутів. /nБільша кількість
груп забезпечує більше варіантів, але може створювати надмірну роздрібленість. Другий параметр впливає на
точність вимірювань, більша кількість - вища точність")
    Button(frame, text="Застосувати", command=lambda: apply_manual_params("km")).grid(row=row_offset - 1,
                                                                                       column=col_offset + 4)
    Button(frame, text="Відновити", command=lambda: reset_to_optimal_params("km")).grid(row=row_offset,
                                                                                        column=col_offset + 4)

    row_offset += 1

    # Випадаючі списки для DBSCAN
    Label(frame, text="Вхідні дані великогабарит DBSCAN", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    dbscan_eps_with_ramp = ttk.Combobox(frame, values=[round(x, 1) for x in np.arange(0.2, 1.6, 0.1)], width=10)
    dbscan_eps_with_ramp.grid(row=row_offset, column=col_offset + 1)
    dbscan_eps_with_ramp.set("0.2")
    dbscan_min_samples_with_ramp = ttk.Combobox(frame, values=list(range(2, 6)), width=10)
    dbscan_min_samples_with_ramp.grid(row=row_offset, column=col_offset + 2)
    dbscan_min_samples_with_ramp.set("2")
    add_info_icon(row_offset, col_offset + 3, "Радіус охоплення визначає, які точки будуть включені до маршруту.
Мінімальна кількість точок задає, скільки їх потрібно для створення маршруту.")
```

```
    row_offset += 1
    Label(frame, text="Вхідні дані малогабарит DBSCAN", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    dbscan_eps_without_ramp = ttk.Combobox(frame, values=[round(x, 1) for x in np.arange(0.2, 1.6, 0.1)], width=10)
    dbscan_eps_without_ramp.grid(row=row_offset, column=col_offset + 1)
    dbscan_eps_without_ramp.set("0.2")
    dbscan_min_samples_without_ramp = ttk.Combobox(frame, values=list(range(2, 6)), width=10)
    dbscan_min_samples_without_ramp.grid(row=row_offset, column=col_offset + 2)
    dbscan_min_samples_without_ramp.set("2")
    add_info_icon(row_offset, col_offset + 3, "Радіус охоплення визначає, які точки будуть включені до маршруту.
Мінімальна кількість точок задає, скільки їх потрібно для створення маршруту.")
    Button(frame, text="Застосувати", command=lambda: apply_manual_params("db")).grid(row=row_offset - 1,
                                                                                       column=col_offset + 4)
    Button(frame, text="Відновити", command=lambda: reset_to_optimal_params("db")).grid(row=row_offset,
                                                                                        column=col_offset + 4)

    row_offset += 1

    # Випадаючі списки для Agglomerative
    Label(frame, text="Вхідні дані великогабарит Agglomerative", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    agglomerative_n_clusters_with_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    agglomerative_n_clusters_with_ramp.grid(row=row_offset, column=col_offset + 1)
    agglomerative_n_clusters_with_ramp.set("2")
    add_info_icon(row_offset, col_offset + 3, "Задайте кількість груп для створення компактних маршрутів.")

    row_offset += 1
    Label(frame, text="Вхідні дані малогабарит Agglomerative", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    agglomerative_n_clusters_without_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    agglomerative_n_clusters_without_ramp.grid(row=row_offset, column=col_offset + 1)
    agglomerative_n_clusters_without_ramp.set("2")
    add_info_icon(row_offset, col_offset + 3, "Задайте кількість груп для створення компактних маршрутів.")
    Button(frame, text="Застосувати", command=lambda: apply_manual_params("ag")).grid(row=row_offset - 1,
                                                                                       column=col_offset + 4)
    Button(frame, text="Відновити", command=lambda: reset_to_optimal_params("ag")).grid(row=row_offset,
                                                                                        column=col_offset + 4)

    row_offset += 1

    # Випадаючі списки для Spectral
    Label(frame, text="Вхідні дані великогабарит Spectral", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    spectral_n_clusters_with_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    spectral_n_clusters_with_ramp.grid(row=row_offset, column=col_offset + 1)
    spectral_n_clusters_with_ramp.set("2")
    spectral_affinity_with_ramp = ttk.Combobox(frame, values=["nearest_neighbors", "rbf"], width=15)
    spectral_affinity_with_ramp.grid(row=row_offset, column=col_offset + 2)
    spectral_affinity_with_ramp.set("nearest_neighbors")
    add_info_icon(row_offset, col_offset + 3, "Параметри впливають на вибір методів обробки для побудови
маршрутів.")

    row_offset += 1
    Label(frame, text="Вхідні дані малогабарит Spectral", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    spectral_n_clusters_without_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    spectral_n_clusters_without_ramp.grid(row=row_offset, column=col_offset + 1)
    spectral_n_clusters_without_ramp.set("2")
    spectral_affinity_without_ramp = ttk.Combobox(frame, values=["nearest_neighbors", "rbf"], width=15)
    spectral_affinity_without_ramp.grid(row=row_offset, column=col_offset + 2)
    spectral_affinity_without_ramp.set("nearest_neighbors")
    add_info_icon(row_offset, col_offset + 3, "Параметри впливають на вибір методів обробки для побудови
```

```
маршрутів.")

    Button(frame, text="Застосувати", command=lambda: apply_manual_params("sp")).grid(row=row_offset - 1,
                                                                                       column=col_offset + 4)
    Button(frame, text="Відновити", command=lambda: reset_to_optimal_params("sp")).grid(row=row_offset,
                                                                                        column=col_offset + 4)

    row_offset += 1

    # Випадаючі списки для Birch
    Label(frame, text="Вхідні дані великогабарит Birch", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    birch_n_clusters_with_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    birch_n_clusters_with_ramp.grid(row=row_offset, column=col_offset + 1)
    birch_n_clusters_with_ramp.set("2")
    add_info_icon(row_offset, col_offset + 3, "Цей параметр відповідає за обробку великої кількості точок для
маршруту.")

    row_offset += 1
    Label(frame, text="Вхідні дані малогабарит Birch", font="Arial 10 bold", bg="#C0C0C0",
fg="#000000").grid(row=row_offset, column=col_offset, pady=5, padx=5)
    birch_n_clusters_without_ramp = ttk.Combobox(frame, values=list(range(2, 16)), width=10)
    birch_n_clusters_without_ramp.grid(row=row_offset, column=col_offset + 1)
    birch_n_clusters_without_ramp.set("2")
    add_info_icon(row_offset, col_offset + 3, "Цей параметр відповідає за обробку великої кількості точок для
маршруту.")

    Button(frame, text="Застосувати", command=lambda: apply_manual_params("br")).grid(row=row_offset - 1,
                                                                                       column=col_offset + 4)
    Button(frame, text="Відновити", command=lambda: reset_to_optimal_params("br")).grid(row=row_offset,
                                                                                        column=col_offset + 4)




def apply_manual_params(algorithm):
    global manual_params_with_ramp, manual_params_without_ramp

    try:
        if algorithm == "km":
            manual_params_with_ramp["km"] = [('kmeans', {
                "n_clusters": int(kmeans_n_clusters_with_ramp.get()),
                "n_init": int(kmeans_n_init_with_ramp.get()),
                "random_state": 42
            })]
            manual_params_without_ramp["km"] = [('kmeans', {
                "n_clusters": int(kmeans_n_clusters_without_ramp.get()),
                "n_init": int(kmeans_n_init_without_ramp.get()),
                "random_state": 42
            })]

        elif algorithm == "db":
            manual_params_with_ramp["db"] = [('dbscan', {
                "eps": float(dbscan_eps_with_ramp.get()),
                "min_samples": int(dbscan_min_samples_with_ramp.get())
            })]
            manual_params_without_ramp["db"] = [('dbscan', {
                "eps": float(dbscan_eps_without_ramp.get()),
                "min_samples": int(dbscan_min_samples_without_ramp.get())
            })]

        elif algorithm == "ag":
```

```
            manual_params_with_ramp["ag"] = [('agglomerative', {
                "n_clusters": int(agglomerative_n_clusters_with_ramp.get())
            })]
            manual_params_without_ramp["ag"] = [('agglomerative', {
                "n_clusters": int(agglomerative_n_clusters_without_ramp.get())
            })]

        elif algorithm == "sp":
            manual_params_with_ramp["sp"] = [('spectral', {
                "n_clusters": int(spectral_n_clusters_with_ramp.get()),
                "random_state": 42,
                "affinity": spectral_affinity_with_ramp.get()
            })]
            manual_params_without_ramp["sp"] = [('spectral', {
                "n_clusters": int(spectral_n_clusters_without_ramp.get()),
                "random_state": 42,
                "affinity": spectral_affinity_without_ramp.get()
            })]

        elif algorithm == "br":
            manual_params_with_ramp["br"] = [('birch', {
                "n_clusters": int(birch_n_clusters_with_ramp.get())
            })]
            manual_params_without_ramp["br"] = [('birch', {
                "n_clusters": int(birch_n_clusters_without_ramp.get())
            })]

        else:
            raise ValueError(f"Невідомий алгоритм: {algorithm}")

        messagebox.showinfo("Success", f"Параметри для {algorithm} успішно застосовані!")
    except Exception as e:
        messagebox.showerror("Error", f"Помилка застосування параметрів для {algorithm}: {e}")


def reset_to_optimal_params(algorithm):
    global manual_params_with_ramp, manual_params_without_ramp

    try:
        manual_params_with_ramp[algorithm] = None
        manual_params_without_ramp[algorithm] = None

        # Оновлення GUI
        if algorithm == "km":
            kmeans_n_clusters_with_ramp.set("2")
            kmeans_n_init_with_ramp.set("10")
            kmeans_n_clusters_without_ramp.set("2")
            kmeans_n_init_without_ramp.set("10")

        elif algorithm == "db":
            dbscan_eps_with_ramp.set("0.2")
            dbscan_min_samples_with_ramp.set("2")
            dbscan_eps_without_ramp.set("0.2")
            dbscan_min_samples_without_ramp.set("2")

        elif algorithm == "ag":
            agglomerative_n_clusters_with_ramp.set("2")
            agglomerative_n_clusters_without_ramp.set("2")

        elif algorithm == "sp":
            spectral_n_clusters_with_ramp.set("2")
            spectral_affinity_with_ramp.set("nearest_neighbors")
```

```
            spectral_n_clusters_without_ramp.set("2")
            spectral_affinity_without_ramp.set("nearest_neighbors")

        elif algorithm == "br":
            birch_n_clusters_with_ramp.set("2")
            birch_n_clusters_without_ramp.set("2")

        else:
            raise ValueError("Unknown algorithm")

        messagebox.showinfo("Success", f"Оптимальні параметри для {algorithm} успішно відновлені!")
    except Exception as e:
        messagebox.showerror("Error", f"Помилка відновлення оптимальних параметрів для {algorithm}: {e}")


main_win()
root.mainloop()
```

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК. 507150.006 ПЗ	Пояснювальна записка	70 с., ф. А4
		Графічні документи	
		Презентація	у форматі
			*.ppt

					ГЮИК. 507150.006 ВД
Змін.	Арк.	№ документа	Підп.	Дата	
Розроб.		Дідик П.Ю.			Літ. | Аркуш | Аркушів
Перевір.		Максимова С.С.			1 | 1
Н. контр.		Бронніков А.І.			ХНУРЕ
Затвер.		Невлюдов І.Ш			Кафедра КІТАР