Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет математики і інформатики
Кафедра прикладної математики

До захисту допущено
кафедрою прикладної математики, протокол №5 від 12 червня 2026 р.

завідувач кафедри
прикладної математики
доктор фіз.-мат. наук, професор *Валерій КОРОБОВ*

КВАЛІФІКАЦІЙНА РОБОТА

здобувача першого (бакалаврського) рівня вищої освіти

«Моделювання та кластеризація поведінкових патернів користувачів у цифрових системах»

Спеціальність 113 Прикладна математика
Освітня програма Прикладна математика

Здобувач Михайло АРТЬОМОВ

Науковий керівник доктор філософії,
доцент кафедри прикладної математики
Валерія КАРЄВА

Харків – 2026

АНОТАЦІЯ

Артьомов М.П. Моделювання та кластеризація поведінкових патернів користувачів у цифрових системах. Кваліфікаційна робота на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 113 Прикладна математика. – Харківський національний університет імені В.Н. Каразіна, Міністерства освіти і науки України, Харків, 2026.

Мета роботи. Метою дослідження є розробка та впровадження методів кластеризації поведінкових патернів користувачів у цифрових системах для автоматизованого профілювання клієнтів на основі різнотипових даних їхньої активності.

Об'єкт та предмет дослідження. Об'єктом дослідження є процес опрацювання різнотипових даних користувачів електронної комерції. Предметом дослідження виступають методи та засоби кластеризації поведінкових патернів для подальшої класифікації клієнтів.

Методи проведення дослідження. У роботі використано методи кластеризації (k-means, ієрархічна кластеризація, Density-Based Spatial Clustering of Applications with Noise, Gaussian Mixture Model), методи зменшення розмірності (Principal Component Analysis), методи нормалізації даних (Z-score), а також методи класифікації (логістична регресія, Random Forest). Аналіз результатів здійснювався за допомогою метрик: Silhouette score, Davies-Bouldin index, Calinski-Harabasz index – для методів кластеризації, а також accuracy, precision, recall та F1-score – для методів класифікації.

Отримані результати та їхня новизна. Запропоновано підхід до моделювання поведінкових профілів користувачів на основі інтегральних індикаторів «Активність», «Незадоволення» та «Цінність покупки», сформованих із використанням кореляційного аналізу та методу головних компонент (PCA). За результатами порівняльного аналізу встановлено, що метод k-means є найбільш ефективним для досліджуваного набору даних. Виділено три поведінкові сегменти користувачів: клієнти з високим ризиком

відтоку, преміум клієнти та активні лояльні клієнти. Для автоматизованого віднесення нових користувачів до визначених сегментів побудовано модель логістичної регресії, яка забезпечила високу точність класифікації (99,87 %). Отримані результати можуть бути використані для сегментації клієнтів, персоналізації сервісів та підтримки прийняття управлінських рішень у системах електронної комерції (зосередити увагу на клієнтах групи ризику, запровадити спеціальні сервіси для ВІП-клієнтів тощо).

Загальні висновки. Запропонований підхід дозволяє автоматично визначати тип клієнта за його поведінковими характеристиками з високою точністю. Розроблену модель може бути використано в системах електронної комерції для персоналізації маркетингових стратегій, покращення обслуговування клієнтів та підвищення їхньої лояльності.

Ключові слова: попередня обробка даних, кластеризація, профілювання користувачів, silhouette score, k-means, логістична регресія.

ABSTRACT

Artomov Mykhailo. Modeling and clustering of user behavioral patterns in digital systems. Qualification thesis for the first (bachelor's) level of higher education in the Specialty 113 Applied Mathematics. – V.N. Karazin Kharkiv National University, Ministry of Education and Science of Ukraine, Kharkiv, 2026.

Purpose of the work. The aim of the research is to develop and implement methods for clustering user behavioral patterns in digital systems for automated customer profiling based on heterogeneous data of their activity.

Object and subject of research. The object of research is the processing of heterogeneous data of e-commerce users. The subject of research is the methods and tools for clustering behavioral patterns for subsequent customer classification.

Research methods. The work employs clustering methods (k-means, hierarchical clustering, Density-Based Spatial Clustering of Applications with Noise, Gaussian Mixture Model), dimensionality reduction methods (Principal Component Analysis), data normalization methods (Z-score), as well as classification methods (logistic regression, Random Forest). The analysis of results was carried out using the following metrics: Silhouette score, Davies-Bouldin index, Calinski-Harabasz index – for clustering methods, as well as accuracy, precision, recall and F1-score – for classification methods.

Obtained results and their novelty. A user behavior profiling approach based on the integral indicators "Activity", "Dissatisfaction", and "Purchase Value" was proposed. These indicators were formed using correlation analysis and Principal Component Analysis (PCA). Based on a comparative analysis of clustering methods, k-means was identified as the most effective approach for the studied dataset. Three behavioral user segments were identified: at-risk customers, premium customers, and active loyal customers. To automatically assign new users to the identified segments, a logistic regression model was developed, achieving a high classification accuracy of 99.87%. The obtained results can be applied to customer segmentation, service personalization, and decision support in e-commerce systems, including customer

retention strategies and the development of specialized services for high-value customers.

General conclusions. The proposed approach allows for automatically determining the customer type based on their behavioral characteristics with high accuracy. The developed model can be used in e-commerce systems for personalizing marketing strategies, improving customer service and increasing their loyalty.

Keywords: data preprocessing, clustering, user profiling, silhouette score, k-means, logistic regression.

ЗМІСТ

ВСТУП

Персоналізований підхід до клієнта є ключовим напрямом розвитку сучасних сервісів обслуговування. Шляхом аналізу цифрового сліду, який користувач залишив під час відвідувань додатку, можна скласти профіль його поведінки та розробити індивідуальний підхід до нього.

Для виявлення прихованих закономірностей у поведінці клієнтів застосовуються методи машинного навчання. Методи кластеризації та класифікації широко використовуються для сегментації клієнтів, аналізу поведінкових патернів та побудови рекомендаційних систем [3, 4, 8, 11, 12]. Кластеризація дозволяє групувати користувачів зі схожими патернами поведінки, що дає змогу виділити типові портрети. Наприклад, серед клієнтів платформи можна виділити активних та лояльних, проблемних тощо.

Подальша класифікація дозволяє автоматично відносити нового користувача до одного з виявлених типів на основі його перших дій у системі, що відкриває можливості для миттєвої персоналізації обслуговування. Подібний підхід активно використовується у сучасних системах аналізу поведінки споживачів та електронної комерції [10, 14, 15].

Практична цінність такого підходу полягає у можливості автоматизувати прийняття рішень щодо кожної групи клієнтів, завчасно побачити ризикові групи, мати підґрунтя для розробки стратегії розвитку бізнесу. Користувач, своєю чергою, відчуває відповідне ставлення до нього та схильний частіше звертатися за персоналізованими послугами.

Саме актуальністю цього підходу зумовлений вибір теми даної кваліфікаційної роботи. Вона присвячена дослідженню теоретичних основ та створенню моделі кластеризації та класифікації користувачів.

Мета і завдання дослідження. *Метою* даної кваліфікаційної роботи є розробка методів кластеризації поведінкових патернів користувачів у цифрових системах для автоматизованого профілювання клієнтів на основі різнотипових даних їхньої активності.

Об'єкт дослідження – процес опрацювання різнотипових даних користувачів електронної комерції.

Предмет дослідження – методи та засоби кластеризації поведінкових патернів для подальшої класифікації клієнтів.

Завдання дослідження:

– провести аналіз існуючих методів кластеризації даних та визначити їхні переваги та недоліки в контексті задачі профілювання користувачів;

– виконати попередню обробку даних (заповнення пропусків, видалення дублікатів та викидів, нормалізацію);

– здійснити кореляційний аналіз та зменшення розмірності даних за допомогою методу головних компонент (PCA);

– реалізувати кластеризацію користувачів наступними методами: k-means, ієрархічна кластеризація, DBSCAN та GMM;

– провести порівняльний аналіз якості кластеризації за метриками Silhouette, Davies-Bouldin та Calinski-Harabasz;

– інтерпретувати отримані кластери;

– побудувати класифікаційну модель для прогнозування належності нового користувача до кластера.

Структура кваліфікаційної роботи. Кваліфікаційна робота складається з анотації, змісту, вступу, огляду літератури, трьох розділів, висновків, переліку використаних джерел та додатків з кодом програми**.**

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ ТА ТЕОРЕТИЧНИХ ОСНОВ

Розуміння типових патернів поведінки дозволяє підприємствам персоналізувати обслуговування, підвищувати лояльність клієнтів та оптимізувати бізнес-процеси. Для вирішення цих завдань широко застосовуються методи машинного навчання, зокрема кластеризація для групування користувачів зі схожими характеристиками та класифікація для автоматичного віднесення нових клієнтів до визначених груп.

Цей розділ присвячений поглибленому аналізу не тільки відповідних методів, а й метрикам оцінювання якості кластеризації, які дозволяють математично обґрунтувати результати роботи програми.

1.1. Поняття поведінкових патернів та їх роль в аналізі користувачів

Поведінкові патерни – це регулярні послідовності дій або навігаційних рішень, які користувачі виконують під час роботи з цифровою системою. Вони відображають типові маршрути переміщення між сторінками, частоту звернення до певних розділів, тривалість сесій, а також специфічні дії, такі як додавання товарів до кошика, написання відгуків або звернення до служби підтримки. Аналіз цих патернів дозволяє виявити приховані закономірності в поведінці, які неможливо побачити при поверхневому розгляді окремих дій.

Аналіз поведінкових патернів базується на трьох основних типах даних [1]:

- контентних (інформація на сторінках);
- структурних (гіперпосилання між сторінками);
- лог-файли серверів.

Саме останній тип даних є найбільш інформативним для виявлення патернів, оскільки він фіксує кожен клік користувача, включаючи IP-адресу, час запиту, тип браузера тощо. Ці дані після відповідної попередньої обробки

стають основою для аналізу. Роль поведінкових патернів в аналізі користувачів є багатоаспектною.

По-перше, вони дозволяють сегментувати користувачів на групи зі схожими звичками за допомогою методів кластеризації. Об’єднання користувачів у кластери на основі послідовностей запитів дає змогу створювати узагальнені профілі, які відображають типову поведінку [2].

По-друге, патерни використовуються для передбачення майбутніх дій користувача (наприклад, знаючи послідовність попередніх кліків, система може рекомендувати наступні сторінки або товари). Це особливо важливо в електронній комерції, де підвищення конверсії напряму залежить від персоналізації пропозицій.

По-третє, аналіз відхилень від типових патернів дає змогу виявляти аномалії, такі як шахрайські дії або технічні помилки.

Окрему увагу в дослідженнях приділяється проблемі узагальнення патернів. Оскільки навігаційні шляхи окремих користувачів рідко бувають ідентичними, використовують процедуру генералізації URL, яка полягає у видаленні частини сегментів з кінця шляху [2]. Це дозволяє працювати з більш абстрактними зонами сайту, а не з конкретними сторінками, що робить виявлені патерни більш стійкими до незначних варіацій у поведінці.

Таким чином, вивчення поведінкових патернів є ключовим інструментом для автоматизації профілювання користувачів. Коректна інтерпретація отриманих результатів дозволяє не лише розуміти поточну поведінку аудиторії, а й прогнозувати її майбутні дії, що відкриває широкі можливості для персоналізації сервісів, покращення навігації та підвищення загальної задоволеності користувачів.

1.2. Методи кластеризації даних

Кластеризація є одним із основних методів машинного навчання без вчителя, який дозволяє групувати об'єкти за ступенем їхньої схожості без попередньої інформації про належність до певних класів [3, 4]. У контексті аналізу поведінки користувачів кластеризація дає змогу виділити типові групи клієнтів зі схожими патернами активності, що є основою для подальшого профілювання та персоналізації сервісів.

1.2.1. k-means

Метод к-середніх (далі k-means) є одним з найбільш поширених та обчислювально ефективних методів кластеризації, який належить до центроїдних алгоритмів [3, 4] . Основна ідея методу полягає в розбитті множини об'єктів на заздалегідь визначену кількість кластерів k таким чином, щоб об'єкти всередині одного кластера були максимально близькими один до одного, а об'єкти з різних кластерів – максимально віддаленими.

Алгоритм роботи k-means [5]:

1. Ініціалізація – випадковим чином обираються k початкових центроїдів;
2. Призначення об’єктів до кластерів – кожен об’єкт x_i відноситься до того кластера, центроїд якого є найближчим за обраною метрикою (за замовчуванням – евклідова відстань):

$$C_i = \{x_i : \forall l,\ 1 \le l \le k\ \ ||x_i - \mu_j||^2 \le ||x_i - \mu_l||^2,\ 1 \le j \le k\},$$

де C_i – множина об’єктів j-го кластера, а μ_j – центроїд j-ого кластеру.

3. Перерахунок центроїдів – після перерозподілу об'єктів обчислюються нові центроїди як середні арифметичні всіх об'єктів, що належать до відповідного кластера:

$$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i,\ \ 1 \le j \le k;$$

4. Перевірка збіжності – кроки 2-3 повторюються доти, доки центроїди не перестануть змінюватися (або зміни не стануть меншими за заданий поріг, або до досягнення максимальної кількості ітерацій).

Серед переваг k-means можна виділити простоту реалізації, високу швидкодію (складність алгоритму лінійна) та масштабованість для великих наборів даних.

Важливими недоліками є чутливість до викидів (якої в роботі запобігли за допомогою попередньої обробки даних) та чутливість до початкового набору центроїдів (цієї проблеми дозволяє уникнути вбудована в бібліотеку Scikit-learn реалізація KMeans, а саме гіперпараметр n_init, який відповідає за кількість незалежних запусків алгоритму [16]).

1.2.2. Ієрархічна кластеризація

Теоретичні основи ієрархічної кластеризації та її застосування до неоднорідних даних детально розглянуті в роботі [5]. Ієрархічна кластеризація будує вкладену систему кластерів, яка може бути представлена у вигляді дерева. Відстань між двома об'єднаними кластерами $A,\ B$ залежить від обраного методу зв'язку.

Наведемо алгоритм роботи ієрархічної кластеризації [5]:

1. Ініціалізація – кожен об'єкт вважається окремим кластером;
2. Пошук найближчих кластерів – обчислюються відстані між усіма парами кластерів за обраним методом зв'язку;
3. Об'єднання – два найближчих кластери об'єднуються в один;
4. Повторення – кроки 2-3 повторюються, доки всі об'єкти не опиняться в одному кластері.

Нехай:

- $A,\ B$ – два кластери;
- $a \in A,\ b \in B$ – об’єкти в кластерах;

- $|A|$, $|B|$ – кількість об’єктів у кластерах;
- μ_A, μ_B – центроїди кластерів A та B;
- $||a - b||$ – евклідова відстань між точками.

Тоді методи зв’язку можна описати наступним чином:

1. Single linkage – відстань між двома найближчими точками кластерів:
$$d(A, B) = min\,||a - b||,\ a \in A,\ b \in B;$$
2. Complete linkage – відстань між двома найбільш віддаленими точками кластерів:
$$d(A, B) = max\,||a - b||,\ a \in A,\ b \in B;$$
3. Average linkage – середня відстань між усіма парами точок з різних кластерів:
$$d(A, B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} ||a - b||;$$
4. Ward linkage – мінімізація збільшення внутрішньокластерної дисперсії при об’єднанні кластерів:
$$d(A, B) = \frac{2 \cdot |A| \cdot |B|}{|A| + |B|} \cdot ||\mu_A - \mu_B||^2.$$

На відміну від методу k-means, ієрархічна кластеризація не потребує заздалегідь заданої кількості кластерів, а також немає “випадковості”, тобто при фіксованих параметрах результат завжди однаковий.

Головним недоліком є квадратична обчислювальна складність алгоритму, яка робить метод дуже витратним з обчислювальної точки зору.

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

DBSCAN – це метод кластеризації на основі щільності, який групує точки, що знаходяться близько одна до одної, та позначає ізольовані точки як шум. Алгоритм DBSCAN є одним із найбільш відомих методів кластеризації на

основі щільності та залишається актуальним для виявлення кластерів довільної форми та шумових об'єктів [6].

Серед гіперпараметрів можна виділити наступні:

- ε-neighborhood – множина точок, що знаходяться на відстані не більше ε від точки p:

$$N_{\varepsilon}(p) = \{q \in X : ||p - q|| \leq \varepsilon\};$$

- Core point – точка, яка має не менше $MinPts$ сусідів у межах радіуса ε:

$$|N_{\varepsilon}(p)| \geq MinPts.$$

Таким чином, кластером у DBSCAN називається множина, кожна пара точок якої є зв’язними за щільністю (тобто якщо існує така точка-ядро, з якої досяжні обидві точки p та q). Точки, які не належать до жодного з кластерів, позначаються як шум.

Як ієрархічна кластеризація, метод DBSCAN не потребує задавати кількість кластерів. Основною перевагою є стійкість до викидів, бо алгоритм самостійно виділяє шуми.

До недоліків можна віднести погану кластеризацію дуже щільних даних (як-от у випадку нашого датасету).

1.2.4. Gaussian Mixture Model (GMM)

Gaussian Mixture Models належать до ймовірнісних моделей машинного навчання та ґрунтуються на припущенні про суміш багатовимірних нормальних розподілів [7]. GMM належить до методів м’якої кластеризації, тобто кожен об’єкт не має жорсткого призначення до одного кластера (як у розглянутих раніше алгоритмах), навпаки, він належить до кожного кластера з певною ймовірністю.

Для знаходження параметрів GMM використовується ітеративний EM-алгоритм (Expectation-Maximization):

1. Expectation – обчислюється ймовірність того, що точка x_i належить до j -ого кластера:

$$\gamma_{ij} = \frac{\pi_j \cdot N(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_l \cdot N(x_i \mid \mu_l, \Sigma_l)},$$

де μ_j – середнє арифметичне j-ого кластеру, Σ_j – коваріаційна матриця j-ого кластеру, π_j – вага j-ого кластера, $N(x_i \mid \mu_j, \Sigma_j)$ – густина гаусового розподілу j-ого кластера;

2. Maximization – оновлюються параметри моделі на основі обчислених γ_{ij} (тобто відбувається оновлення ваг, середніх та коваріаційних матриць).

Теоретичні аспекти побудови ймовірнісних моделей детально викладені у роботах [8, 9]. М’яка кластеризація дозволяє працювати з даними, що перекриваються, а також кластери можуть мати різну форму та орієнтацію, що допомагає підтвердити або скасувати гіпотезу щодо оптимальної форми, отриманої під час реалізації інших методів.

До недоліків можна віднести швидкість виконання алгоритму (k-means є найшвидшим, що буде підтверджено на практиці) та саму структуру м’якої кластеризації, яка може бути недоцільною для деяких задач кластеризації.

1.3. Метрики оцінювання кластеризації

Оцінка якості кластеризації є важливим етапом аналізу даних, бо вона дозволяє математично обґрунтувати оптимальну кількість кластерів та порівнювати різні методи за конкретними параметрами. Використання декількох метрик оцінювання кластеризації рекомендується сучасними дослідженнями для комплексного аналізу якості сегментації [10].

1.3.1. Silhouette score

Silhouette score – це метрика, яка оцінює, наскільки добре кожен об'єкт належить своєму кластеру порівняно з іншими кластерами. Значення метрики лежить у діапазоні від −1 до +1, де більше значення вказує на кращу кластеризацію.

Для кожного об’єкта і обчислюються наступні величини:

1. Середня відстань до точок свого кластеру:

$$a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, j \neq i} ||x_i - x_j||, 1 \leq i \leq k,$$

де C_i – кластер, до якого належить об’єкт i, а $|C_i|$ – кількість об’єктів у цьому кластері.

2. Середня відстань до точок сусіднього кластеру:

$$b(i) = min \frac{1}{|C|} \sum_{j \in C} ||x_i - x_j||, \text{ де } C \neq C_i.$$

Тоді (силуетний коефіцієнт для об’єкта i):

$$s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}.$$

Загальний силуетний коефіцієнт:

$$S = \frac{1}{n} \sum_{i=1}^{n} s(i).$$

1.3.2. Davies–Bouldin index

Davies–Bouldin index – це метрика, яка оцінює якість кластеризації на основі співвідношення внутрішньокластерної подібності та міжкластерної відмінності. На відміну від силуетного коефіцієнта, менше значення індекса Девіса-Болдіна вказує на кращу кластеризацію.

Для кожного C_i обчислюється середня відстань від точок кластеру до його центроїда:

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} ||x - \mu_i||, 1 \leq i \leq k,$$

де $|C_i|$ – кількість точок у кластері C_i, а μ_i - центроїд кластеру C_i.

Для кожної пари кластерів обчислюється відстань між центроїдами:

$$M_{ij} = ||\mu_i - \mu_j||, 1 \leq i \leq k, 1 \leq j \leq k.$$

Індекс Девіса-Болдіна для k кластерів:

$$DB = \frac{1}{k} \sum_{i=1}^{k} max\left(\frac{S_i + S_j}{M_{ij}}\right), \text{ де } j \neq i, 1 \leq i \leq k, 1 \leq j \leq k.$$

1.3.3. Calinski–Harabasz index

Calinski–Harabasz index – це метрика, яка оцінює якість кластеризації на основі співвідношення міжкластерної дисперсії та внутрішньокластерної дисперсії.

Наведемо формулу розрахунку індексу:

1. Міжкластерна дисперсія – сума квадратів відстаней від центроїдів кластерів до загального центру, зважена на розміри кластерів:

$$B = \sum_{j=1}^{k} |C_j| \cdot ||\mu_j - \mu||^2,$$

де C_j – j-й кластер, μ_j – його центроїд, k – кількість кластерів, а μ – середнє арифметичне всіх даних.

2. Внутрішньокластерна дисперсія – сума квадратів від точок до центроїдів їхніх кластерів:

$$W = \sum_{j=1}^{k} \sum_{i \in C_j} ||x_i - \mu_j||^2,$$

Індекс Калінського-Харабаса:

$$CH = \frac{B}{W} \cdot \frac{n-k}{k-1}$$

Зауважимо, що саме сукупність трьох метрик дозволяє отримати повноцінну картину якості кластеризації, оскільки кожна з них оцінює різні аспекти розбиття даних.

Silhouette score вимірює, наскільки кожен об'єкт відповідає своєму кластеру порівняно з сусідніми, надаючи інформацію про індивідуальну якість віднесення точок. Davies–Bouldin index аналізує співвідношення між компактністю кластерів та відстанню між ними, акцентуючи увагу на найгірших парах. Calinski–Harabasz index оцінює глобальне співвідношення міжкластерної та внутрішньокластерної дисперсії, показуючи, наскільки добре кластери відокремлені в цілому.

1.4. Методи класифікації даних

Після виявлення кластерів користувачів виникає задача автоматичного віднесення нового користувача до одного з цих кластерів на основі його поведінкових характеристик. Це завдання вирішується за допомогою логістичної регресії та/або Random Forest – класичних методів класифікації даних. Логістична регресія та ансамблеві методи класифікації належать до найбільш поширених алгоритмів машинного навчання для побудови прогнозних моделей [8, 11, 12].

На відміну від кластеризації, де аналізуються дані без попередніх міток, класифікація використовує розмічену навчальну вибірку, де для кожного об'єкта відомо, до якого кластеру він належить. Мета класифікації: побудувати модель, яка за вхідними ознаками передбачатиме клас нового, раніше невідомого, об'єкта.

1.4.1. Логістична регресія

Логістична регресія – це метод класифікації, який використовується для передбачення ймовірності належності об'єкта до одного з кількох класів. У даній роботі вона застосовується для багатокласової класифікації (тобто три або більше кластерів). Логістична регресія є базовим методом статистичного навчання та широко використовується для задач багатокласової класифікації [8, 11].

Для випадку $k > 2$ використовується мультиноміальна логістична регресія:

$$p(y = k|x) = \frac{e^{\omega_i^T x + b_i}}{\sum_{j=1}^{k} e^{\omega_j^T x + b_j}},$$

де $p(y = k|x)$ – ймовірність того, що об’єкт x належить до класу k, ω_i та b_i – параметри для i-ого кластера.

Навчання логістичної регресії полягає у знаходженні параметрів ω та b, які максимізують функцію правдоподібності:

$$L(\omega, b) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} log\,(p_{ik}),$$

де y_{ij} – індикатор, чи належить i-ий об’єкт до класу j, p_{ik} – передбачена ймовірність належності.

Серед переваг логістичної регресії можна виділити простоту реалізації та та швидкодію, до недоліків можна віднести чутливість до викидів та потребу у нормалізації даних (обидві проблеми будуть вирішені під час обробки даних).

1.4.2. Random Forest

Random Forest – це ансамблевий метод класифікації, який будує велику кількість дерев рішень під час навчання та видає результат як голосування

більшості дерев. Метод випадкового лісу забезпечує високу якість класифікації навіть для складних нелінійних залежностей між ознаками [8, 9].

Опишемо алгоритм роботи методу випадкового лісу:

1. Генерація навчальних вибірок – для кожного дерева випадково вибирається певна кількість прикладів з вихідної навчальної вибірки (приблизно дві третини), а решта використовується для оцінки;
2. Побудова дерева – у кожному вузлі випадково вибирається m ознак, серед них знаходиться найкраще розбиття, а вузол розбивається на два дочірніх. Відповідний процес повторюється до досягнення критерію зупинки;
3. Передбачення – результат визначається шляхом мажоритарного голосування або усереднення результатів, отриманих від усіх дерев.

Метод Random Forest є повільнішим за логістичну регресію, проте до його переваг можна віднести адаптивність до великої кількості кластерів, стійкість до перенавчання та високу точність класифікації.

1.4.3. Метрики оцінювання класифікації

Для оцінки якості класифікації використовуються метрики, які базуються на елементах матриці помилок (confusion matrix): True Positive (TP) – правильно передбачені позитивні, True Negative (TN) – правильно передбачені негативні, False Positive (FP) – хибнопозитивні, False Negative (FN) – хибнонегативні.

Наведемо відповідні метрики:

- $Accuracy$ – частка правильних передбачень серед усіх:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN};$$

- $Precision$ – частка дійсно позитивних серед усіх передбачених як позитивні:

$$Precision = \frac{TP}{TP+FP};$$

- $Recall$ – частка реальних позитивних, які модель знайшла:

$$Recall = \frac{TP}{TP+FN};$$

- $F1$-score – середнє гармонійне між $Precision$ та $Recall$:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}.$$

У даній роботі відповідна комбінація з чотирьох метрик використовується не тільки для порівняльного аналізу логістичної регресії та методу випадкового лісу, а й для перевірки якості роботи моделей безпосередньо.

Висновки до розділу 1

У даному розділі було розглянуто теоретичні складові ключових понять, які будуть використані під час реалізації методів кластеризації та класифікації. Також чималу увагу було приділено метрикам оцінювання для розуміння результатів роботи програми не тільки за допомогою візуалізації, а й за допомогою суворих статистичних показників.

РОЗДІЛ 2. МЕТОДИКА ДОСЛІДЖЕННЯ

У даному розділі описується послідовність етапів обробки даних, яка передує безпосередній кластеризації та класифікації поведінкових патернів користувачів. Метою обробки є формування узагальнених поведінкових індикаторів на основі вихідних ознак набору даних. В якості датасету для кваліфікаційної роботи було обрано знеособлені дані користувачів платформи електронної комерції.

2.1. Попередня обробка даних

Перед тим, як реалізовувати обробку даних, треба зрозуміти структуру датасету, з яким ми маємо справу:

```
Data columns (total 25 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Age                             47505 non-null  float64
 1   Gender                          50000 non-null  object
 2   Country                         50000 non-null  object
 3   City                            50000 non-null  object
 4   Membership_Years                50000 non-null  float64
 5   Login_Frequency                 50000 non-null  int64
 6   Session_Duration_Avg            46601 non-null  float64
 7   Pages_Per_Session               47000 non-null  float64
 8   Cart_Abandonment_Rate           50000 non-null  float64
 9   Wishlist_Items                  46000 non-null  float64
 10  Total_Purchases                 50000 non-null  float64
 11  Average_Order_Value             50000 non-null  float64
 12  Days_Since_Last_Purchase        47000 non-null  float64
 13  Discount_Usage_Rate             46500 non-null  float64
 14  Returns_Rate                    45509 non-null  float64
 15  Email_Open_Rate                 47472 non-null  float64
 16  Customer_Service_Calls          49832 non-null  float64
 17  Product_Reviews_Written         46500 non-null  float64
 18  Social_Media_Engagement_Score   44000 non-null  float64
 19  Mobile_App_Usage                45000 non-null  float64
 20  Payment_Method_Diversity        47500 non-null  float64
 21  Lifetime_Value                  50000 non-null  float64
 22  Credit_Balance                  44500 non-null  float64
 23  Churned                         50000 non-null  int64
 24  Signup_Quarter                  50000 non-null  object
dtypes: float64(19), int64(2), object(4)
```

Рис.1. Зведена інформація про дані

Обраний датасет складається з 50 000 рядків та 25-ти стовпців, серед яких 21 мають числові значення, а інші 4 – категоріальні. Аналіз показує значну кількість пропущених значень, проте першочергово необхідно проаналізувати від’ємні значення, бо специфіка даних вказує на те, що негативних значень не повинно бути. Відповідні рядки вирішено видалити, якщо їх кількість не є значною.

```
import numpy as np

numeric_cols = customer_data.select_dtypes(include=[np.number]).columns.tolist()

rows_with_negatives = ((customer_data[numeric_cols] < 0) & (customer_data[numeric_cols].notna())).any(axis=1).sum()

customer_data = customer_data[~((customer_data[numeric_cols] < 0) & (customer_data[numeric_cols].notna())).any(axis=1)]

print(f"Рядків з від'ємними значеннями: {rows_with_negatives}")
print(f"Поточний розмір датасету: {customer_data.shape}")
```

```
Рядків з від'ємними значеннями: 40
Поточний розмір датасету: (49960, 25)
```

Рис.2. Видалення від’ємних значень

Наступним кроком розглянемо кількість пропущених значень:

```
Пропусків до обробки:
Age                              2492
Session_Duration_Avg             3397
Pages_Per_Session                2997
Wishlist_Items                   3993
Days_Since_Last_Purchase         2998
Discount_Usage_Rate              3497
Returns_Rate                     4486
Email_Open_Rate                  2527
Customer_Service_Calls            168
Product_Reviews_Written          3500
Social_Media_Engagement_Score    5994
Mobile_App_Usage                 4997
Payment_Method_Diversity         2498
Credit_Balance                   5495
dtype: int64
Поточний розмір датасету: (49960, 25)
```

Рис.3. Кількість пропущених значень

Кількість таких рядків виявилася значною, тому безпосереднє видалення не є коректним. Враховуючи подальшу нормалізацію та специфіку роботи з методами кластеризації та класифікації, всі пропущені значення були заповнені медіанами відповідних стовпців. Заповнення пропущених значень медіаною є

одним із рекомендованих підходів підготовки даних для алгоритмів машинного навчання [11, 12].

Наступним кроком датасет було перевірено на кількість дублікатів:

```
initial_count = len(customer_data)
customer_data = customer_data.drop_duplicates(keep='first')
print(f"Видалено дублікатів: {initial_count - len(customer_data)}")
```

```
Видалено дублікатів: 0
```

Рис.4. Перевірка на дублікати

Для виявлення та видалення викидів було використано метод Z-оцінки (стандартної нормалізації). Використання Z-нормалізації є стандартною практикою під час підготовки даних для алгоритмів кластеризації та класифікації [11, 12]. Для кожної числової колонки обчислювалась Z-оцінка:

$$Z = \frac{x-\mu}{\sigma},$$

де μ – середнє значення стовпця, σ – стандартне відхилення.

Слід зазначити, що для поведінкових даних застосування Z-методу потребує обережності, оскільки активні користувачі (з високою частотою входів, довгими сесіями або великими чеками) можуть статистично виглядати як викиди, хоча насправді вони є важливим цільовим сегментом бізнесу. Тому в даній роботі було обрано помірний поріг $|Z| > 3.5$, що дозволяє видалити лише екстремальні аномалії. В результаті нормалізації було видалено 1985 рядків (близько 4% від початкового обсягу), що є прийнятним рівнем втрати, враховуючи специфіку даних.

Нормалізовані дані було збережено в окремий датасет для подальшого кореляційного аналізу.

2.2. Кореляційний аналіз

Для того, щоб виявити поведінкові індикатори для застосування методів кластеризації та класифікації, було використано кореляційний аналіз. Метою проведеного аналізу є зменшення розмірності простору ознак та формування інтегральних поведінкових індикаторів шляхом об'єднання висококорельованих ознак, що забезпечує підвищення інтерпретованості результатів дослідження.

Зменшення розмірності та побудова інтегральних показників на основі корельованих ознак широко застосовується в задачах аналізу даних та машинного навчання [8, 11].

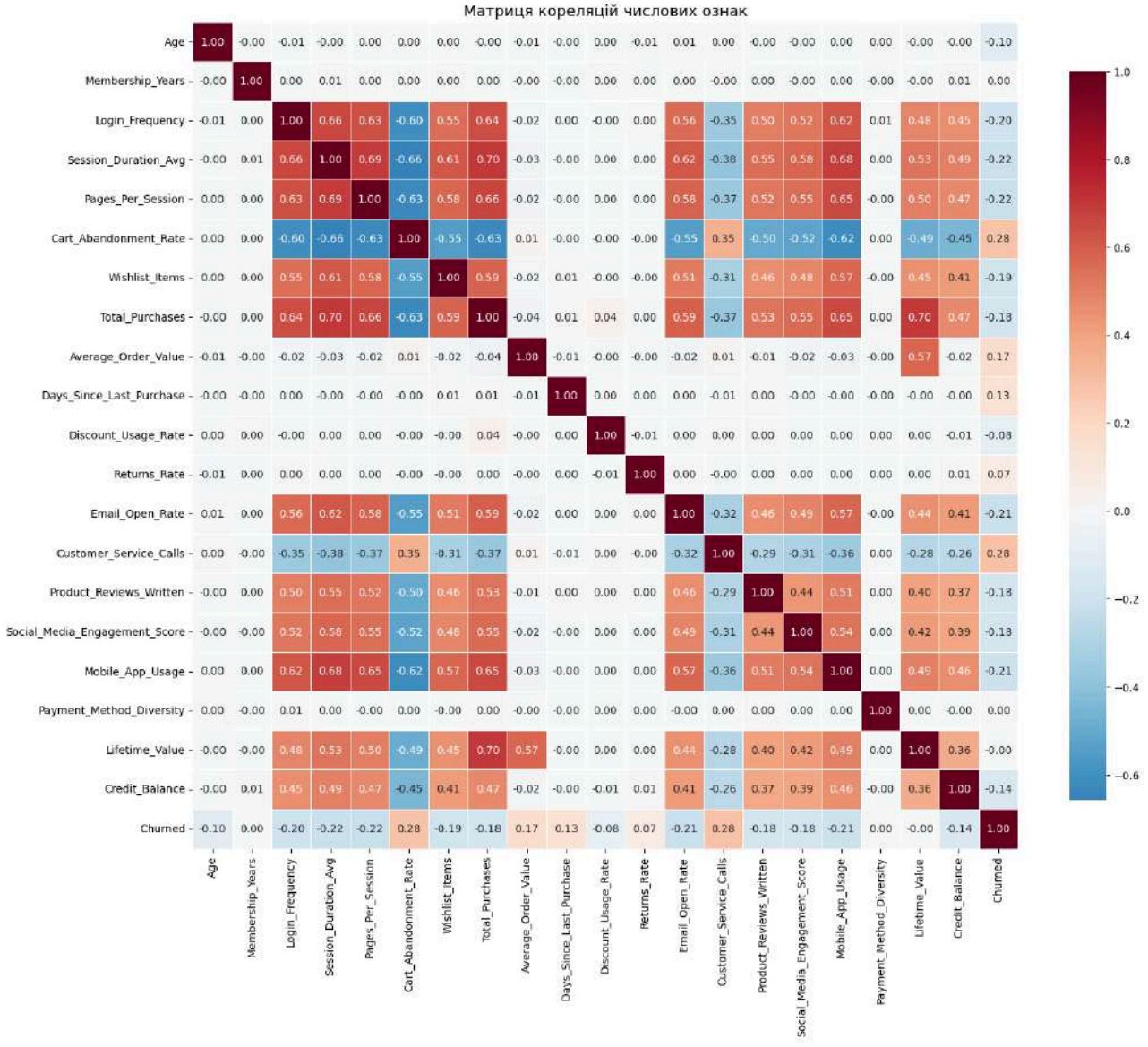

Рис.5. Матриця кореляцій числових ознак

Зауважимо, що категоріальні дані (Гендер, Країна, Місто та Чверть року, в якій людина погодилась на розсилку) було відкинуто, бо вони не впливають на моделювання поведінки користувачів.

Переходячи до результатів з рисунку 5 доцільним є об'єднання наступних висококорельованих поведінкових індикаторів:

```
activity_cols = [
    'Login_Frequency',
    'Session_Duration_Avg',
    'Pages_Per_Session',
    'Mobile_App_Usage',
    'Email_Open_Rate',
    'Product_Reviews_Written',
    'Social_Media_Engagement_Score',
    'Wishlist_Items'
]
```

Поведінковий індикатор назвали "Активність", бо вона безпосередньо описує взаємодію користувача з сервісами.

Для визначення вагових коефіцієнтів було застосовано метод головних компонент (Principal Component Analysis, PCA) – статистичний метод зменшення розмірності даних, який перетворює велику кількість корельованих ознак у головні компоненти, тобто об'єднані некорельовані змінні. PCA є одним із найпоширеніших методів зменшення розмірності та побудови латентних факторів у багатовимірних даних [8, 11]. Основна ідея PCA полягає у знаходженні таких напрямків у просторі ознак, вздовж яких дисперсія даних є максимальною. Математично PCA зводиться до обчислення власних векторів та власних чисел коваріаційної матриці даних.

Результати роботи PCA для нашої задачі:

```
Login_Frequency          : 0.3634
Session_Duration_Avg     : 0.3887
Pages_Per_Session        : 0.3757
```

Mobile_App_Usage : 0.3701

Email_Open_Rate : 0.3433

Product_Reviews_Written : 0.3145

Social_Media_Engagement_Score : 0.3265

Wishlist_Items : 0.3395

Отримані результати свідчать про рівноцінний внесок кожної з восьми ознак до формуваннія узагальненого поведінкового індикатору.

За тими самими міркуваннями отримали другий збірний поведінковий індикатор – “Незадоволення”, для якої висновки виявились ідентичними з першою:

Коефіцієнти для кожної ознаки:

Cart_Abandonment_Rate : 0.7071

Customer_Service_Calls : 0.7071

Оскільки всі висококорельовані ознаки були об’єднані, для повноти аналізу було вирішено додати третій поведінковий індикатор, а саме “Цінність покупки”, до якої увійшла “Average_Order_Value”. Подальше збільшення до чотирьох і більше поведінкових індикаторів призводить до погіршення кластеризації, через недостатню корельованість окремих ознак.

Як остаточну перевірку було складено матрицю кореляцій між обраними поведінковими індикаторами:

КОРЕЛЯЦІЯ МІЖ КАТЕГОРІЯМИ

	Activity	Dissatisfaction	Purchase_Value
Activity	1.0000	-0.7103	-0.0285
Dissatisfaction	-0.7103	1.0000	0.0158
Purchase_Value	-0.0285	0.0158	1.0000

Рис.6. Матриця кореляцій між збірними поведінковими індикаторами

Отримані результати підтверджують доцільність вибору саме таких поведінкових індикаторів, бо вони попарно між собою не є висококорельовними, а разом охоплюють важливі складові поведінки користувача.

Висновки до розділу 2

В результаті попередньої обробки даних було отримано датасет із 47 975 рядків, придатний для аналізу, причому кореляційний аналіз підтвердив логічність об'єднання ознак у три збірні поведінкові індикатори: Activity, Dissatisfaction, Purchase_Value. Зауважимо, що кожний поведінковий індикатор було нормалізовано після перерахунку вагових коефіцієнтів за допомогою PCA.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

У цьому розділі буде реалізована кластеризація та класифікація за допомогою методів, розглянутих раніше. Відповідні алгоритми були реалізовані у середовищі Python за допомогою бібліотеки Scikit-learn, Pandas та NumPy [13, 16–18]. Як вхідні ознаки використовувалися інтегральні поведінкові індикатори, сформовані на попередньому етапі дослідження. Порівняльний аналіз методів здійснювався за допомогою метрик оцінювання якості кластеризації та класифікації, а також за показниками часу виконання відповідних алгоритмів. Для наочної демонстрації були використані pairplots та heatmap для методів, візуалізація яких має сенс.

3.1. Проведення кластеризації

3.1.1. k-means

Порівняння алгоритмів кластеризації за допомогою декількох метрик відповідає сучасним підходам до сегментації клієнтів у цифрових системах [10, 15].

Таблиця 1. k-means до інтегральних поведінкових індикаторів

k	Silhouette	Davies-Bouldin	Calinski-Harabasz	Time (sec)
2	0.3399	1.1373	29578.80	0.110
3	0.3220	1.0772	27958.79	0.149
4	0.2791	1.0788	26233.99	0.323
5	0.2694	1.0832	24158.81	0.439

Для методу k-means найкращий баланс між якістю кластеризації та змістовною інтерпретацією досягається при k = 3. Незважаючи на дещо вищі значення Silhouette та Calinski-Harabasz для k = 2 (0.3399 та 29578.80 відповідно), три кластери дозволяють виділити три чітко відмінні поведінкові групи користувачів, що має більшу практичну цінність в рамках бізнес-інтерпретації. Крім того, падіння метрик при збільшенні k до 4 та 5 є закономірним і свідчить про те, що подальше збільшення кількості кластерів призводить до надмірної деталізації без суттєвого покращення якості розбиття.

На рисунку наведено результати візуалізації трьох кластерів:

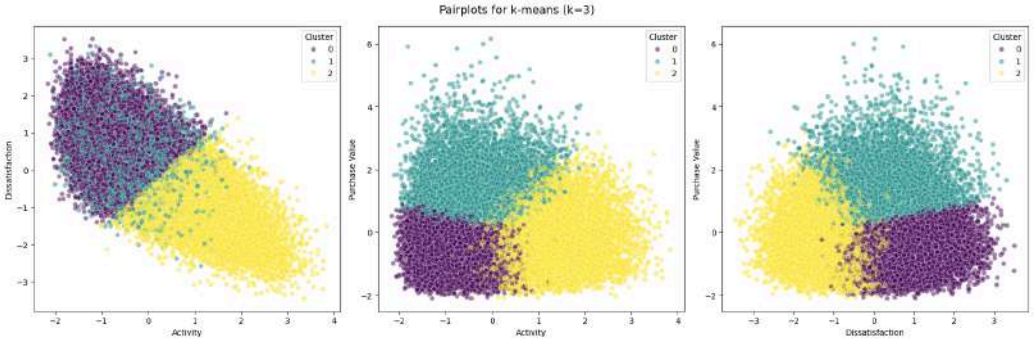

Рис.7. Pairplots для методу k-means (k = 3)

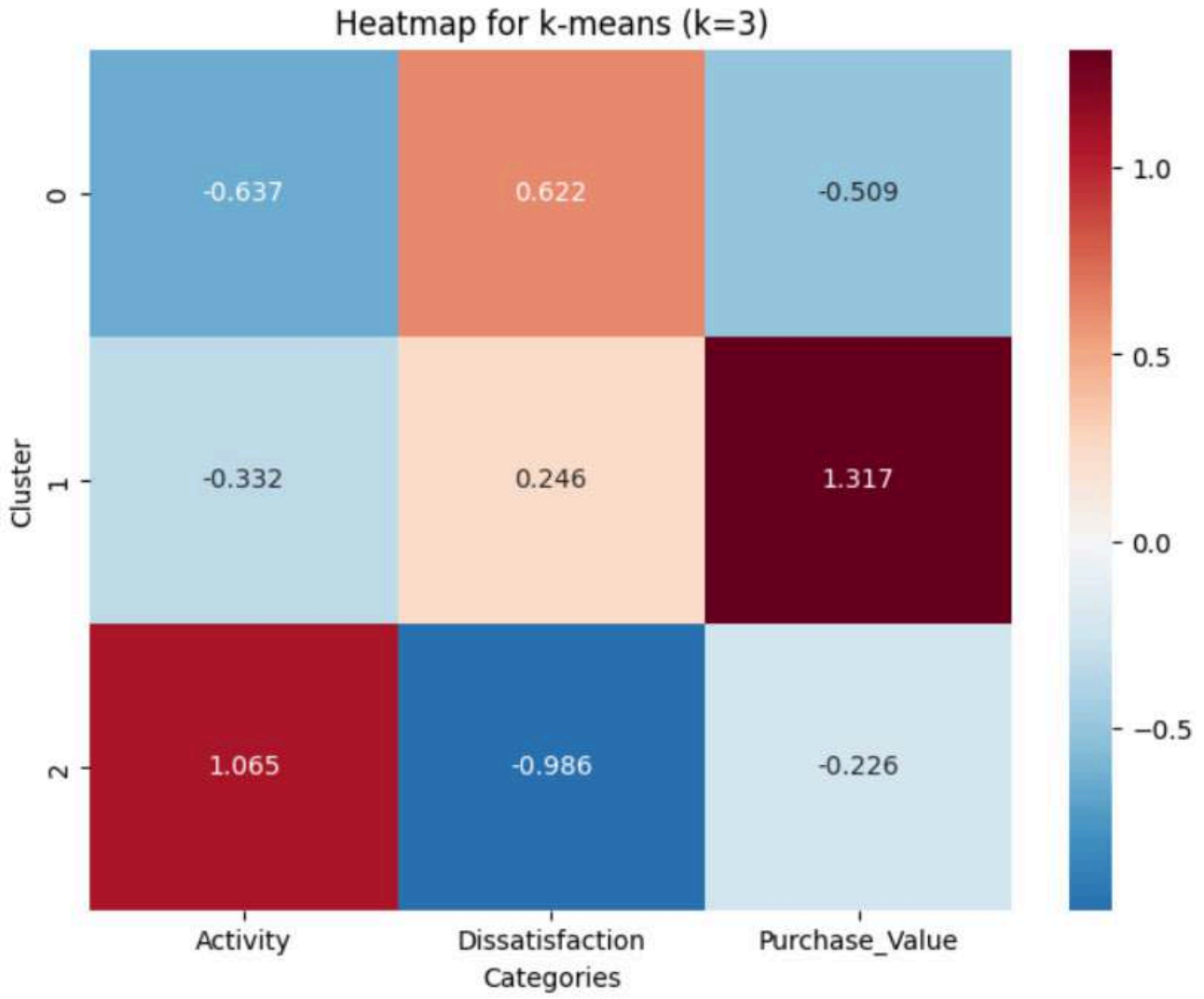

Рис.8. Heatmap для методу k-means (k = 3)

Отримали наступне розбиття для трьох кластерів:

```
Кластер 0: 21077 користувачів, 43.93%
Кластер 1: 10899 користувачів, 22.72%
Кластер 2: 15999 користувачів, 33.35%
```

Рис.9. Розбиття користувачів за k-means (k = 3)

3.1.2. Ієрархічна кластеризація

Таблиця 2. Hierarchical clustering до інтегральних поведінкових індикаторів

k	Silhouette	Davies-Bouldin	Calinski-Harabasz	Time (sec)
2	0.2822	1.2812	9501.09	17.057
3	0.2310	1.2417	8457.84	16.584
4	0.2209	1.1689	8393.89	16.677
5	0.2107	1.3290	8105.39	16.562

Проблема реалізації ієрархічної кластеризації для великих обсягів даних полягає в квадратичній алгоритмічній складності, порівняно з лінійною для інших трьох методів. Результати для таблиці 2 були отримані з випадкової вибірки розміру 20 000. Навіть для неї час на виконання кластеризації зайняв 16.5-17 секунд, значення метрик оцінювання якості не є вражаючими, порівняно з методом k-means.

3.1.3. DBSCAN

Проблема реалізації методу DBSCAN для нашого датасету полягає у щільності даних, яка продемонстрована на рисунку 6. Тестування були виконані для наступних значень параметрів:

eps_values = [0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.8];

min_samples_values = [3, 5, 10, 15, 20, 30, 50].

Аналізуючи результати, наведені у додатку з кодом програми, можемо зробити висновок:

По-перше, DBSCAN створює один дуже великий кластер, який складає більше ніж 80% всіх даних;

По-друге, алгоритм відкидає чималу кількість об’єктів в “Шум”.

По-третє, виконання методу займає в 10 разів більше часу, ніж k-means.

Таким чином, використання методу DBSCAN для нашого датасету не є доречним через щільність даних, з якою алгоритм погано працює. Всі негативні ефекти такі як диспропорційність кластерів, відкидання великої кількості даних до “Шуму” та час на виконання роботи суттєво не залежать від значень гіперпараметрів, як було продемонстровано у додатку.

3.1.4. GMM

Таблиця 3. Gaussian Mixture Model до інтегральних поведінкових індикаторів

k	Silhouette	Davies-Bouldin	Calinski-Harabasz	Time (sec)
2	0.3262	1.1555	27680.47	0.200
3	0.3201	1.0618	26404.20	0.269
4	0.2721	1.1082	25204.76	0.346
5	0.2646	1.0862	23228.84	0.585

Аналізуючи результати GMM для різних значень кількості кластерів, найкращі результати досягаються для трьох кластерів. Хоча силуетний коефіцієнт є дещо вищим для двох кластерів (0.3262 та 0.3201 відповідно), індекс Девіса–Болдіна є найнижчим саме при трьох кластерах (1.0618), що свідчить про кращу відокремленість кластерів. Крім того, три кластери забезпечують змістовну інтерпретацію, а подальше збільшення кількості кластерів до чотирьох або п'яти призводить до помітного зниження силуетного коефіцієнта (0.2721 та 0.2646 відповідно) та індексу Калінського–Харабаса (25204.76 та 23228.84 відповідно).

Отже, для GMM оптимальними є три кластери, що узгоджується з результатами k-means.

Наведемо візуалізацію для трьох кластерів:

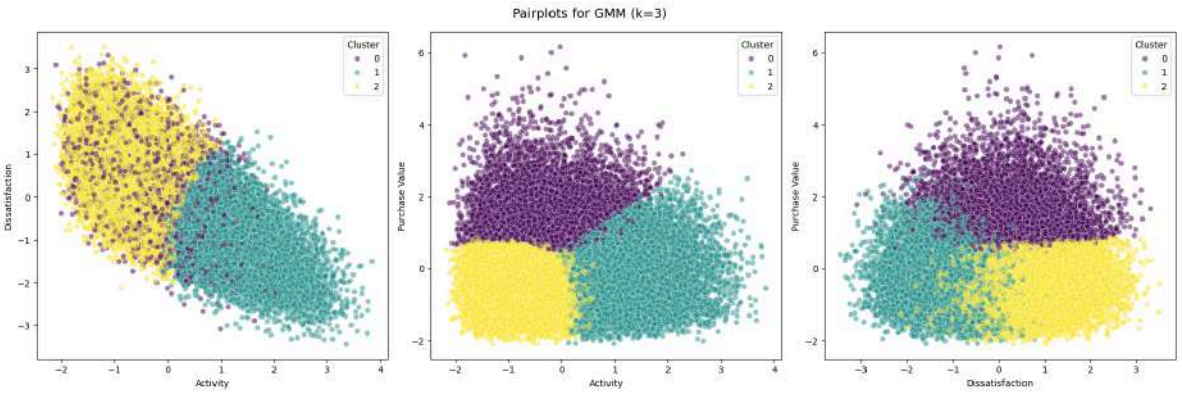

Рис.10. Pairplots для методу GMM (k = 3)

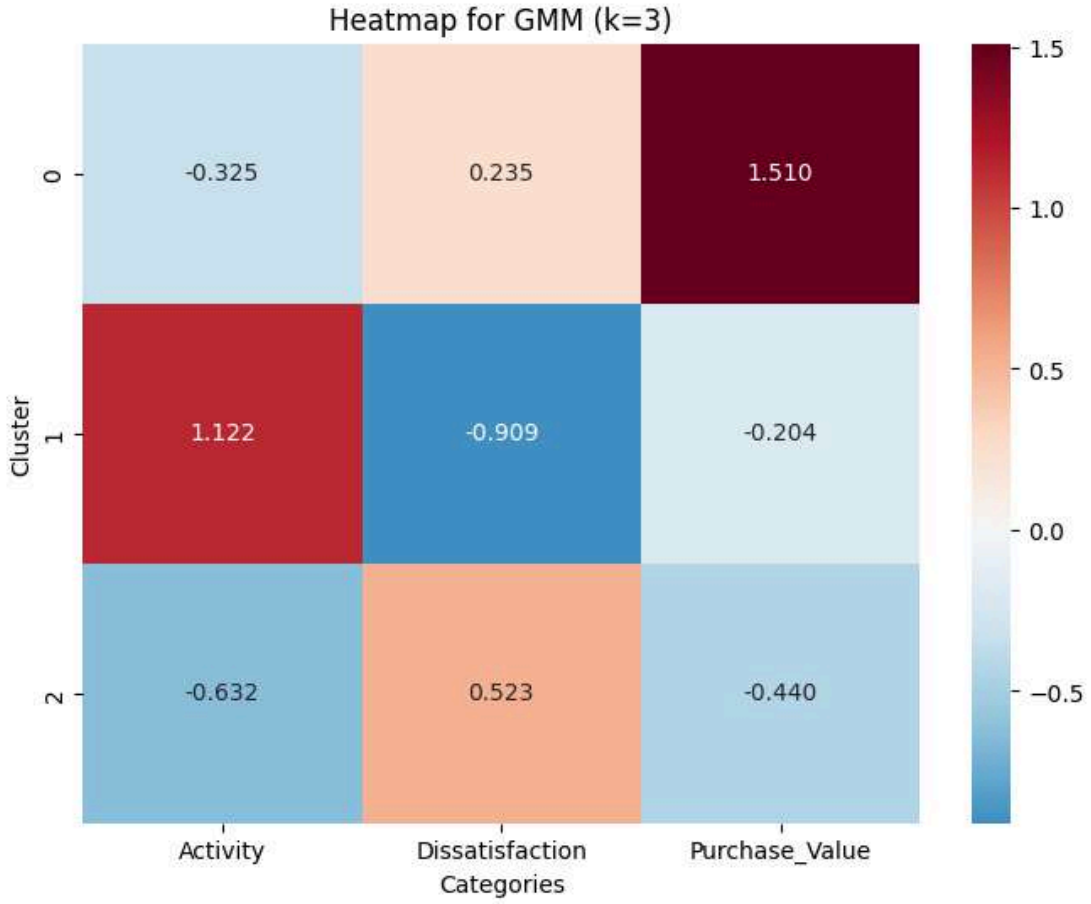

Рис.11. Heatmap для методу GMM (k = 3)

Отримали наступне розбиття для трьох кластерів:

```
Кластер 0: 8919 користувачів, 18.59%
Кластер 1: 15722 користувачів, 32.77%
Кластер 2: 23334 користувачів, 48.64%
```

Рис.12. Розбиття користувачів за GMM (k = 3)

В результаті проведення кластеризації чотирма методами виявили, що ієрархічна кластеризація не є оптимальним варіантом для нашого датасету через квадратичну обчислювальну складність, метод DBSCAN не забезпечив задовільної якості кластеризації через щільність даних, методи k-means та GMM продемонстрували, що три кластери є оптимальним вибором для нашої задачі не тільки з точки зору метрик оцінювання якості, а й з точки зору швидкодії.

Для подальшої класифікації будемо використовувати метод k-means через те, що він є в два рази швидшим за GMM.

3.2. Інтерпретація поведінкових груп

Отримані сегменти узгоджуються з результатами сучасних досліджень поведінки споживачів та задачами клієнтської сегментації в електронній комерції [10, 14, 15].

На основі аналізу середніх значень трьох поведінкових індикаторів (Activity, Dissatisfaction, Purchase_Value) для кожного кластера були визначені наступні типи користувачів:

1. Клієнти з високим ризиком відтоку (Кластер 0: 21077 користувачів, 43.93% від загальної кількості) – цей кластер характеризується низькою активністю (Activity = -0.637), високим рівнем незадоволення (Dissatisfaction = 0.622) та низькою цінністю покупки (Purchase_Value = -0.509). Користувачі цієї групи рідко взаємодіють з платформою, часто кидають кошик та звертаються до служби підтримки, при цьому приносять невеликий прибуток. Така поведінка свідчить про високий ризик відтоку, тому ця низка користувачів потребує першочергової уваги: покращення сервісу, зменшення кількості звернень до підтримки та стимулювання активності;
2. Преміум клієнти (Кластер 1: 10899 користувачів, 22.72% від загальної кількості) – цей кластер має низьку активність (Activity = -0.332),

підвищене незадоволення (Dissatisfaction = 0.246) та значно вищий за середній чек (Purchase_Value = 1.317). Користувачі цієї групи приносять високий прибуток, але через незадоволення можуть піти до конкурентів. Вони є найціннішими, але водночас найризикованішими клієнтами, для яких рекомендовано персональне обслуговування, оперативне вирішення проблеми та надання ексклюзивних пропозицій;

3. Активні лояльні клієнти (Кластер 2: 15999 користувачів, 33.35% від загальної кількості) – цей кластер має високу активність (Activity = 1.065), низьке незадоволення (Dissatisfaction = -0.986) та дещо нижчий за середній чек (Purchase_Value = -0.226). Користувачі цієї групи є лояльною та активною аудиторією, яка задоволена сервісом, але має потенціал для збільшення середнього чека. Для них доцільно запровадити програми лояльності, персоналізовані пропозиції.

3.3. Проведення класифікації

Обидва методи класифікації навчалися на вибірці, що складала 80% даних, кластери були попередньо отримані з методу k-means для k = 3.

Результати роботи наведемо у вигляді зведеної таблиці та Confusion Matrix:

Таблиця 4. Логістична регресія vs Random Forest

Метод	Кластер	Precision	Recall	F1-score	Accuracy	Time (sec)
Логістична регресія	0	1.00	1.00	1.00	0.9987	0.227
	1	1.00	1.00	1.00		
	2	1.00	1.00	1.00		

Random Forest	0	0.99	0.99	0.99	0.9905	4.457
	1	0.99	0.99	0.99		
	2	0.99	0.99	0.99		

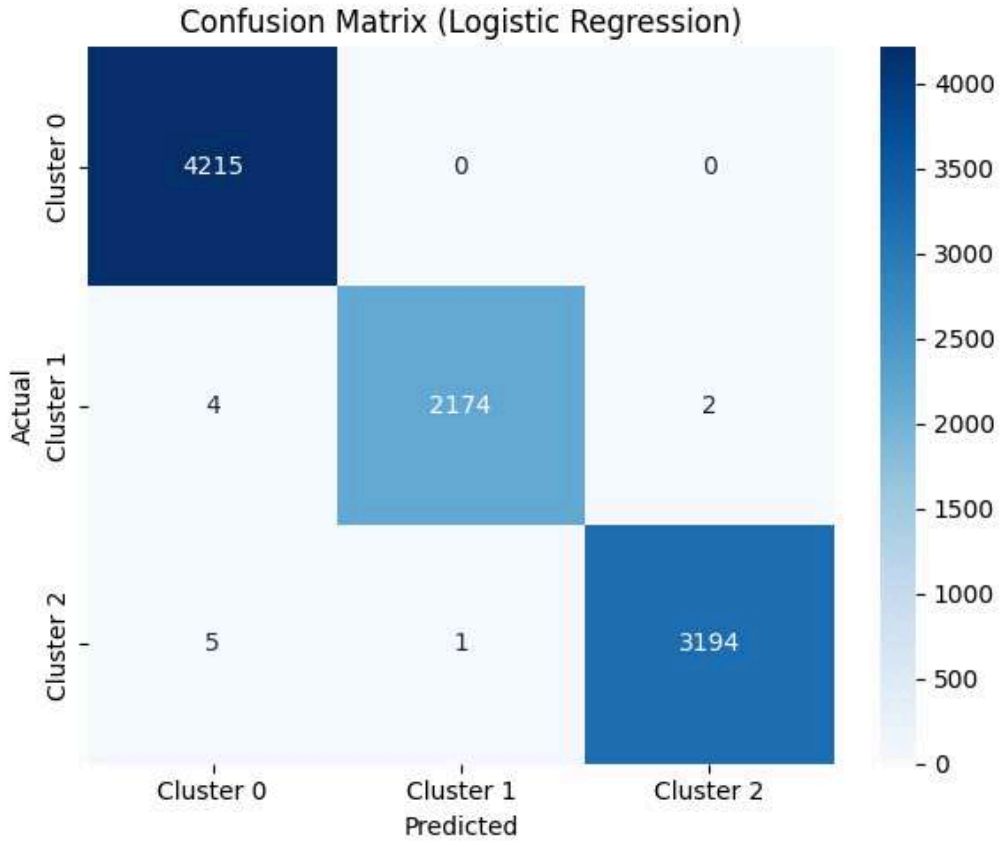

Рис.13. Confusion Matrix для логістичної регресії

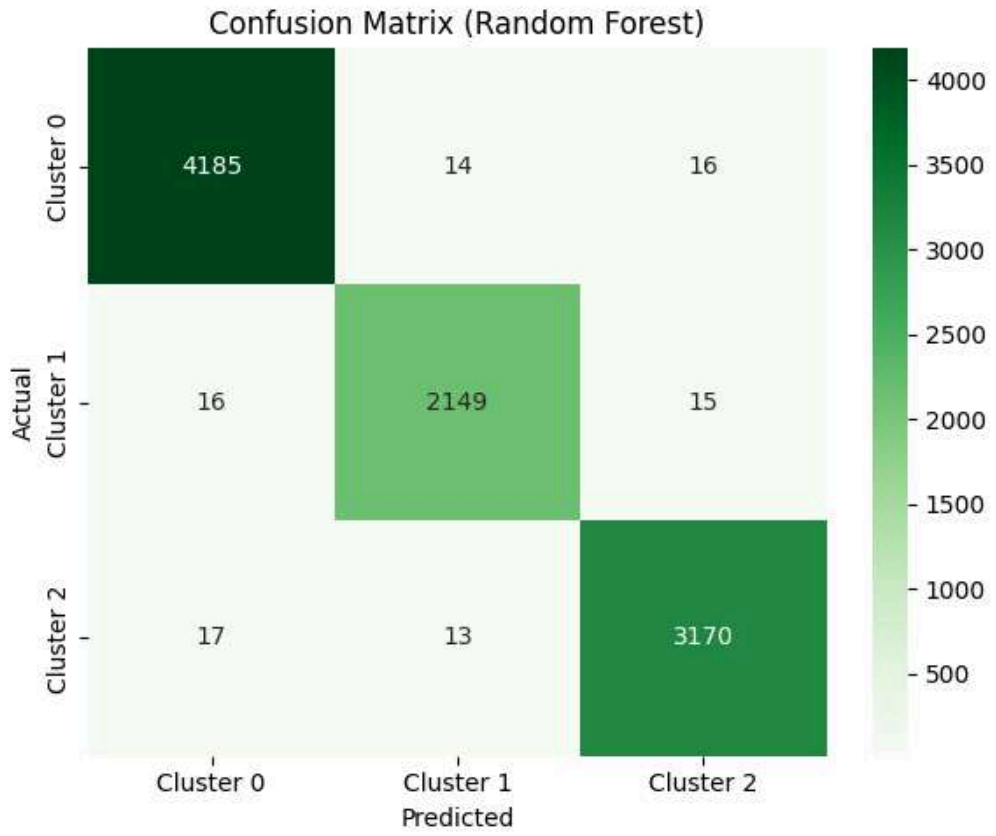

Рис.14. Confusion Matrix для Random Forest

Зауважимо, що моделі тренувалися на 20% даних, тому й відповідні значення у Confusion Matrix.

Побудована модель логістичної регресії продемонструвала практично ідеальну якість класифікації: accuracy становить 99.87%, а precision, recall та F1-score для кожного з трьох кластерів дорівнюють 1.00. Час навчання моделі склав лише 0.227 секунди, що робить її придатною для використання в реальних системах. Висока точність пояснюється тим, що модель навчалася відтворювати сегменти, попередньо сформовані методом k-means на тих самих поведінкових ознаках.

Модель Random Forest показала дещо нижчу якість класифікації: accuracy становить 99.05%, а precision, recall та F1-score для всіх трьох кластерів дорівнюють 0.99. Хоча цей результат також є дуже високим, модель припускається незначної кількості помилок при віднесенні користувачів до

кластерів. Крім того, час навчання Random Forest (4.457 с) є значно більшим порівняно з логістичною регресією.

Незважаючи на те, що обидві моделі показали високу якість класифікації, логістична регресія є кращим вибором для даної задачі з кількох причин: забезпечення вищої точності (99.87% проти 99.05%), а також швидкість навчання (0.227 с проти 4.457 с). Логістична регресія є простішою та більш інтерпретованою моделлю, що дозволяє аналізувати вплив кожної ознаки на прийняття рішення.

Таким чином, для задачі прогнозування належності нового користувача до кластеру доцільно обрати саме логістичну регресію.

Висновки до розділу 3

Шляхом порівняльного аналізу методів кластеризації (k-means, ієрархічна кластеризація, DBSCAN, GMM) було встановлено, що k-means з кількістю кластерів k = 3 є оптимальним для виявлення поведінкових патернів користувачів, дозволяючи виділити три змістовні групи: клієнти з високим ризиком відтоку, преміум клієнти та активні лояльні клієнти. Для автоматичного віднесення нового користувача до одного з виявлених кластерів було навчено модель логістичної регресії, яка показала майже ідеальну якість класифікації (accuracy = 99.87%) та високу швидкодію (0.227 с).

Таким чином, запропонований підхід дозволяє в реальному часі визначати тип клієнта за його поведінковими характеристиками та обирати персоналізовану бізнес-стратегію для кожної групи.

ВИСНОВКИ ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

В результаті виконання кваліфікаційної роботи було досліджено та опановано методи кластеризації та класифікації поведінкових патернів користувачів електронної комерції, а саме:

- Проведено попередню обробку даних: видалення негативних значень, заповнення пропущених значень медіаною, видалення дублікатів, виявлення викидів методом Z-score з порогом 3.5 та стандартної нормалізації, що дозволило підготувати якісний датасет для подальшого аналізу (вихідний датасет з 50 000 рядків після очищення склав 47 975 рядків – видалено 4% даних);
- На основі кореляційного аналізу було сформовано три збірні поведінкові індикатори: «Активність» (об'єднує 8 ознак активності), «Незадоволення» (об'єднує 2 ознаки негативного досвіду) та «Цінність покупки» (середній чек). Застосування методу головних компонент (PCA) дозволило визначити вагові коефіцієнти для кожної ознаки: для поведінкового індикатору «Активність» коефіцієнти знаходяться в діапазоні 0.31–0.39, що свідчить про рівномірний внесок усіх восьми ознак, а для поведінкового індикатору «Незадоволення» обидва коефіцієнти дорівнюють 0.7071;
- Проведено порівняльний аналіз чотирьох методів кластеризації: k-means, ієрархічної кластеризації, DBSCAN, GMM – за трьома метриками: Silhouette score, Davies-Bouldin index та Calinski-Harabasz index. Встановлено, що k-means з кількістю кластерів k = 3 є оптимальним, оскільки забезпечує найкращий баланс між якістю кластеризації (Silhouette = 0.3220, Davies-Bouldin = 1.0772, Calinski-Harabasz = 27958.79) та змістовною інтерпретацією. Метод DBSCAN не забезпечив задовільної якості кластеризації через щільну структуру даних, а ієрархічна кластеризація потребує значно більше обчислювальних ресурсів. Метод GMM продемонстрував результати схожі до k-means,

проте перевагу було надано останньому через показник швидкодії (0.269 та 0.149 с відповідно);

- В результаті інтерпретації результатів кластеризації було виділено три поведінкові групи користувачів: клієнти з високим ризиком відтоку (низька активність, високе незадоволення, низький чек), преміум клієнти (низька активність, підвищене незадоволення, високий чек), активні лояльні клієнти (висока активність, низьке незадоволення, середній чек). Отримані профілі дозволяють розробити персоналізовані бізнес-стратегії для кожної з груп.
- Побудовано класифікаційні моделі для прогнозування належності нового користувача до кластера. Модель логістичної регресії забезпечила високу точність віднесення користувачів до попередньо сформованих поведінкових сегментів: accuracy 99.87%, precision = recall = f1-score = 1.00 для всіх кластерів, а час навчання склав лише 0.227 секунди. Random Forest показав дещо гірші результати: accuracy 99.05%, precision = recall = f1-score = 0.99, час навчання склав 4.457 секунди. Таким чином, для практичного впровадження рекомендовано використовувати логістичну регресію як більш швидку та точну модель.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Srivastava J., Cooley R., Deshpande M., Tan P.N. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. – University of Minnesota, 2000.

[2] Arbelaitz O., Gurrutxaga I., Lojo A., Muguerza J., Pérez J.M., Perona I. Adaptation of the User Navigation Scheme using Clustering and Frequent Pattern Mining Techniques for Profiling, 2012.

[3] Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2012.

[4] Aggarwal C. C. Data Mining: The Textbook. Springer, 2015.

[5] Ткачик О. А. Методи та засоби кластеризації різнотипових даних. – Національний університет «Львівська політехніка», 2023.

[6] Schubert E., Sander J., Ester M., Kriegel H.-P., Xu X. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. ACM Transactions on Database Systems. 2017.

[7] Murphy K. P. Probabilistic Machine Learning: An Introduction. MIT Press, 2022.

[8] Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2009.

[9] Bishop C. M., Bishop H. Deep Learning: Foundations and Concepts. Springer, 2024.

[10] Sakina N., Arun A. P., R P., Prabhu H V., Gupta P. K. Optimizing Customer Segmentation: A Comparative Analysis of Clustering Algorithms Using Evaluation Metrics. 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS). 2024.

[11] James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. Springer, 2021.

[12] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. O'Reilly Media, 2022.

[13] Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.

[14] Castro-González S, No-Perez A. Artificial Intelligence Applications in Consumer Behaviour Analysis: A Systematic Review, Mapping Trends and Challenges. Acta Informatica Pragensia. 2026.

[15] Zhang Y., Ju Yi. Analysis of customer shopping behavior on E-commerce platform supported by fuzzy clustering algorithm. Discov Computing 29, 281. 2026.

[16] Офіційна документація бібліотеки Scikit-learn.

[17] Офіційна документація бібліотеки Pandas.

[18] Офіційна документація бібліотеки NumPy.

ДОДАТКИ

```python
#Завантаження бібліотек і початкового датасету
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.preprocessing import StandardScaler

from google.colab import files
uploaded = files.upload()
file_name = list(uploaded.keys())[0]
customer_data = pd.read_csv(file_name)

customer_data.info()
```

```python
#Видалення рядків з від'ємними значеннями
numeric_cols = customer_data.select_dtypes(include=[np.number]).columns.tolist()

rows_with_negatives = ((customer_data[numeric_cols] < 0) & (customer_data[numeric_cols].notna())).any(axis=1).sum()
customer_data = customer_data[~((customer_data[numeric_cols] < 0) & (customer_data[numeric_cols].notna())).any(axis=1)]

print(f"\nРядків з від'ємними значеннями: {rows_with_negatives}")
print(f"Поточний розмір датасету: {customer_data.shape}")
```

```python
#Заповнення пропусків медіанами
print("\nПропусків до обробки:")
print(customer_data.isnull().sum()[customer_data.isnull().sum() > 0])

numeric_cols = customer_data.select_dtypes(include=[np.number]).columns
for col in numeric_cols:
    if customer_data[col].isnull().sum() > 0:
        median_val = customer_data[col].median()
        customer_data[col] = customer_data[col].fillna(median_val)

print(f"Поточний розмір датасету: {customer_data.shape}")
```

```python
#Видалення дублікатів
initial_count = len(customer_data)
customer_data = customer_data.drop_duplicates(keep='first')
print(f"\nВидалено дублікатів: {initial_count - len(customer_data)}")
```

```
#Видалення викидів методом Z-score (поріг 3.5)
print(f"\nПоточний розмір датасету: {customer_data.shape}")

customer_data = customer_data.reset_index(drop=True)
numeric_cols = customer_data.select_dtypes(include=[np.number]).columns

mask = np.ones(len(customer_data), dtype=bool)

for col in numeric_cols:
    z_scores = np.abs(stats.zscore(customer_data[col]))
    mask = mask & (z_scores <= 3.5)

initial_count = len(customer_data)
customer_data = customer_data[mask].copy()

print(f"Видалено рядків: {initial_count - len(customer_data)}")
print(f"Поточний розмір датасету: {customer_data.shape}")
```

```
#Нормалізація даних
print(f"\nПоточний розмір датасету: {customer_data.shape}")

numeric_cols = customer_data.select_dtypes(include=[np.number]).columns.tolist()
print(f"Числові колонки для нормалізації ({len(numeric_cols)} шт.): {numeric_cols[:5]}...")

scaler = StandardScaler()
customer_data_scaled = scaler.fit_transform(customer_data[numeric_cols])
customer_data[numeric_cols] = customer_data_scaled

print(f"Розмір датасету після нормалізації: {customer_data.shape}")

customer_data.to_csv('customer_data_normalized.csv', index=False)
```

```python
#Побудова матриці кореляцій
import matplotlib.pyplot as plt
import seaborn as sns

customer_data = pd.read_csv('customer_data_normalized.csv')

numeric_cols = customer_data.select_dtypes(include=[np.number]).columns.tolist()

correlation_matrix = customer_data[numeric_cols].corr()

plt.figure(figsize=(16, 14))
sns.heatmap(correlation_matrix,
            annot=True,
            fmt='.2f',
            cmap='RdBu_r',
            center=0,
            square=True,
            linewidths=0.5,
            cbar_kws={"shrink": 0.8})

plt.title('Матриця кореляцій числових ознак', fontsize=14)
plt.tight_layout()
plt.show()
```

```python
#Створення трьох категорій за допомогою PCA
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

activity_cols = [
    'Login_Frequency',
    'Session_Duration_Avg',
    'Pages_Per_Session',
    'Mobile_App_Usage',
    'Email_Open_Rate',
    'Product_Reviews_Written',
    'Social_Media_Engagement_Score',
    'Wishlist_Items'
]

activity_cols = [col for col in activity_cols if col in customer_data.columns]

pca_activity = PCA(n_components=1)
pca_activity.fit(customer_data[activity_cols])

activity_weights = pca_activity.components_[0]

print("Поведінковий індикатор 1: Активність")
print("\nКоефіцієнти для кожної ознаки:")
for col, weight in zip(activity_cols, activity_weights):
    print(f"  {col:35} : {weight:8.4f}")

customer_data['Activity'] = customer_data[activity_cols].dot(activity_weights)

dissatisfaction_cols = ['Cart_Abandonment_Rate', 'Customer_Service_Calls']
dissatisfaction_cols = [col for col in dissatisfaction_cols if col in customer_data.columns]

pca_dissatisfaction = PCA(n_components=1)
pca_dissatisfaction.fit(customer_data[dissatisfaction_cols])


dissatisfaction_weights = pca_dissatisfaction.components_[0]

print("\nПоведінковий індикатор 2: Незадоволення")
print("\nКоефіцієнти для кожної ознаки:")
for col, weight in zip(dissatisfaction_cols, dissatisfaction_weights):
    print(f"  {col:35} : {weight:8.4f}")

customer_data['Dissatisfaction'] = customer_data[dissatisfaction_cols].dot(dissatisfaction_weights)

customer_data['Purchase_Value'] = customer_data['Average_Order_Value']

print("\nПоведінковий індикатор 3: Цінність покупки")
print("  Average_Order_Value (без змін)")

scaler = StandardScaler()
customer_data[['Activity', 'Dissatisfaction', 'Purchase_Value']] = scaler.fit_transform(
    customer_data[['Activity', 'Dissatisfaction', 'Purchase_Value']]
)

print("\nКОРЕЛЯЦІЯ МІЖ КАТЕГОРІЯМИ")
categories = ['Activity', 'Dissatisfaction', 'Purchase_Value']
print(customer_data[categories].corr().round(4))
```

```python
#Реалізація методу k-means
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score
import pandas as pd
import time

customer_data = pd.read_csv('customer_data_categories_normalized.csv')

categories = ['Activity', 'Dissatisfaction', 'Purchase_Value']
X = customer_data[categories]

print("k | Silhouette | Davies-Bouldin | Calinski-Harabasz | Time (sec)")

for k in range(2, 6):
    start_time = time.time()

    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X)

    exec_time = time.time() - start_time

    sil = silhouette_score(X, labels)
    db = davies_bouldin_score(X, labels)
    ch = calinski_harabasz_score(X, labels)

    print(f"{k} | {sil:.4f}     | {db:.4f}          | {ch:.2f}         | {exec_time:.3f}")
```

```python
#Розбиття користувачів за k-means (k = 3)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
customer_data['Cluster'] = kmeans.fit_predict(X)

cluster_means = customer_data.groupby('Cluster')[['Activity', 'Dissatisfaction', 'Purchase_Value']].mean()

for i in range(3):
    size = (customer_data['Cluster'] == i).sum()
    percentage = size / len(customer_data) * 100
    print(f"Кластер {i}: {size} користувачів, {percentage:.2f}%")
```

```
#Pairplots для методу k-means (k = 3)
import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(1, 3, figsize=(18, 6))

sns.scatterplot(data=customer_data, x='Activity', y='Dissatisfaction',
                hue='Cluster', palette='viridis', alpha=0.5, ax=axes[0])
axes[0].set_xlabel('Activity')
axes[0].set_ylabel('Dissatisfaction')

sns.scatterplot(data=customer_data, x='Activity', y='Purchase_Value',
                hue='Cluster', palette='viridis', alpha=0.5, ax=axes[1])
axes[1].set_xlabel('Activity')
axes[1].set_ylabel('Purchase Value')

sns.scatterplot(data=customer_data, x='Dissatisfaction', y='Purchase_Value',
                hue='Cluster', palette='viridis', alpha=0.5, ax=axes[2])
axes[2].set_xlabel('Dissatisfaction')
axes[2].set_ylabel('Purchase Value')

plt.suptitle('Pairplots for k-means (k=3)', fontsize=14)
plt.tight_layout()
plt.show()
```

```
  #Heatmap для методу k-means (k = 3)
  plt.figure(figsize=(8, 6))
  sns.heatmap(cluster_means, annot=True, fmt='.3f', cmap='RdBu_r', center=0,
              xticklabels=['Activity', 'Dissatisfaction', 'Purchase_Value'],
              yticklabels=['0', '1', '2'])
  plt.xlabel('Categories')
  plt.ylabel('Cluster')
  plt.title('Heatmap for k-means (k=3)')
  plt.show()
```

```python
#Реалізация ієрархічної кластеризації для вибірки n = 20 000
from sklearn.cluster import AgglomerativeClustering

customer_data = pd.read_csv('customer_data_categories_normalized.csv')

categories = ['Activity', 'Dissatisfaction', 'Purchase_Value']

X_sample = customer_data[categories].sample(n=20000, random_state=42)

print("k | Silhouette | Davies-Bouldin | Calinski-Harabasz | Time (sec)")

for k in range(2, 6):
    start_time = time.time()

    hier = AgglomerativeClustering(n_clusters=k, linkage='ward')
    labels = hier.fit_predict(X_sample)

    exec_time = time.time() - start_time

    sil = silhouette_score(X_sample, labels)
    db = davies_bouldin_score(X_sample, labels)
    ch = calinski_harabasz_score(X_sample, labels)

    print(f"{k}  | {sil:.4f}    | {db:.4f}        | {ch:.2f}        | {exec_time:.3f}")
```

```python
#Реалізація методу DBSCAN
from sklearn.cluster import DBSCAN

customer_data = pd.read_csv('customer_data_categories_normalized.csv')

categories = ['Activity', 'Dissatisfaction', 'Purchase_Value']
X = customer_data[categories]

eps_values = [0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.8]
min_samples_values = [3, 5, 10, 15, 20, 30, 50]

print("\neps | min_samples | К-ть кластерів | Шум | % в осн. кластері | Silhouette | Time (sec)")

for eps in eps_values:
    for min_samples in min_samples_values:
        start_time = time.time()

        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        labels = dbscan.fit_predict(X)

        exec_time = time.time() - start_time

        n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
        n_noise = list(labels).count(-1)

        unique, counts = np.unique(labels, return_counts=True)
        main_cluster_pct = (max(counts) / len(labels)) * 100

        sil = silhouette_score(X, labels)

        print(f"{eps:.2f}  | {min_samples:^11} | {n_clusters:^9} | {n_noise:^5} | {main_cluster_pct:^15.1f} | {sil:.4f}     | {exec_time:.3f}"
```

```python
#Реалізнація GMM
from sklearn.mixture import GaussianMixture

customer_data = pd.read_csv('customer_data_categories_normalized.csv')

categories = ['Activity', 'Dissatisfaction', 'Purchase_Value']
X = customer_data[categories]

gmm_warmup = GaussianMixture(n_components=2, random_state=42)
gmm_warmup.fit_predict(X)

print("k | Silhouette | Davies-Bouldin | Calinski-Harabasz | Time (sec)")

for k in range(2, 6):
    start_time = time.time()

    gmm = GaussianMixture(n_components=k, random_state=42)
    labels = gmm.fit_predict(X)

    exec_time = time.time() - start_time

    sil = silhouette_score(X, labels)
    db = davies_bouldin_score(X, labels)
    ch = calinski_harabasz_score(X, labels)

    print(f"{k}  | {sil:.4f}    | {db:.4f}        | {ch:.2f}        | {exec_time:.3f}")
```

```python
#Розбиття користувачів за GMM (k = 3)
gmm = GaussianMixture(n_components=3, random_state=42)
customer_data['Cluster'] = gmm.fit_predict(X)

cluster_means = customer_data.groupby('Cluster')[['Activity', 'Dissatisfaction', 'Purchase_Value']].mean()

for i in range(3):
    size = (customer_data['Cluster'] == i).sum()
    percentage = size / len(customer_data) * 100
    print(f"Кластер {i}: {size} користувачів, {percentage:.2f}%")
```

```
#Pairplots для методу GMM (k = 3)
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

sns.scatterplot(data=customer_data, x='Activity', y='Dissatisfaction',
                hue='Cluster', palette='viridis', alpha=0.5, ax=axes[0])
axes[0].set_xlabel('Activity')
axes[0].set_ylabel('Dissatisfaction')

sns.scatterplot(data=customer_data, x='Activity', y='Purchase_Value',
                hue='Cluster', palette='viridis', alpha=0.5, ax=axes[1])
axes[1].set_xlabel('Activity')
axes[1].set_ylabel('Purchase Value')

sns.scatterplot(data=customer_data, x='Dissatisfaction', y='Purchase_Value',
                hue='Cluster', palette='viridis', alpha=0.5, ax=axes[2])
axes[2].set_xlabel('Dissatisfaction')
axes[2].set_ylabel('Purchase Value')

plt.suptitle('Pairplots for GMM (k=3)', fontsize=14)
plt.tight_layout()
plt.show()
```

```
#Heatmap для методу GMM (k = 3)
plt.figure(figsize=(8, 6))
sns.heatmap(cluster_means, annot=True, fmt='.3f', cmap='RdBu_r', center=0,
            xticklabels=['Activity', 'Dissatisfaction', 'Purchase_Value'],
            yticklabels=['0', '1', '2'])
plt.xlabel('Categories')
plt.ylabel('Cluster')
plt.title('Heatmap for GMM (k=3)')
plt.show()
```

```python
#Реалізація логістичної регресії
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

customer_data = pd.read_csv('customer_data_categories_normalized.csv')

categories = ['Activity', 'Dissatisfaction', 'Purchase_Value']
X = customer_data[categories]

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
y = kmeans.fit_predict(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

start_time = time.time()

logreg = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000, random_state=42)
logreg.fit(X_train, y_train)

exec_time = time.time() - start_time

y_pred = logreg.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Час навчання: {exec_time:.3f} сек")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Cluster 0', 'Cluster 1', 'Cluster 2']))
```

```python
#Візуалізація результатів роботи логістичної регресії за допомогою Confusion Matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Cluster 0', 'Cluster 1', 'Cluster 2'],
            yticklabels=['Cluster 0', 'Cluster 1', 'Cluster 2'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Logistic Regression)')
plt.tight_layout()
plt.show()
```

```python
#Реалізація Random Forest
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

customer_data = pd.read_csv('customer_data_categories_normalized.csv')

categories = ['Activity', 'Dissatisfaction', 'Purchase_Value']
X = customer_data[categories]

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
y = kmeans.fit_predict(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

start_time = time.time()

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

exec_time = time.time() - start_time

y_pred = rf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Час навчання: {exec_time:.3f} сек")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Cluster 0', 'Cluster 1', 'Cluster 2']))
```

```python
#Візуалізація результатів роботи Random Forest за допомогою Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens',
            xticklabels=['Cluster 0', 'Cluster 1', 'Cluster 2'],
            yticklabels=['Cluster 0', 'Cluster 1', 'Cluster 2'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Random Forest)')
plt.tight_layout()
plt.show()
```