МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-наукової програми «Інформатика»
на тему: «Інтелектуальна інформаційно-аналітична технологія прогнозування взаємодії користувача з медичним додатком»
здобувача групи ІН.м-24 Рижкіна Дмитра Володимировича

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

_____ Дмитро РИЖКІН
(підпис)

Керівник,
старша викладачка кафедри
комп'ютерних наук, к.м.-ф.н. Анна БАДАЛЯН _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІН.м-24 Рижкіна Дмитра Володимировича

1. Тема роботи: «Інтелектуальна інформаційно-аналітична технологія прогнозування взаємодії користувача з медичним додатком»
затверджую наказом по СумДУ від «06» грудня 2023 року № 1412-VI

2. Термін здачі здобувачем кваліфікаційної роботи до 16 грудня 2023 року

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд існуючих методів аналізу поведінки користувачів 3) Розробка інформаційно-аналітичної технології прогнозування взаємодії користувача з медичним додатком 4) Аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 2023 р.

Завдання прийняв до виконання _____ (підпис)

Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		

2	*Огляд існуючих методів аналізу поведінки користувачів*		
3	*Розробка інформаційно-аналітичної технології прогнозування взаємодії користувача з медичним додатком*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 98 стр., 27 рис., 1 додаток, 6 табл., 21 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки медичні додатки стають дедалі популярнішими і існує потреба у персоналізованих рекомендаціях та прогнозуванні взаємодії користувачів для оптимізації їх лікувального ефекту.

Об'єкт дослідження — процес взаємодії користувачів з медичним додатком, включаючи їхні вибори та поведінку під час використання додатку.

Мета роботи — розробка інформаційної технології для аналізу взаємодії користувача з медичним додатком та прогнозування його подальших дій на основі побудови моделей машинного навчання..

Методи дослідження — алгоритми класифікації, кластеризації та колаборативної фільтрації в машинному навчанні.

Результати — розроблено алгоритми класифікації користувачів за рівнем активності, визначення регулярності використання додатку, прогнозу тривалості життєвого циклу користувача та персоналізованих рекомендацій програм електростимуляції.

ІНФОРМАЦІЙНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, ПРОГНОЗУВАННЯ, ЕЛЕКТРОСТИМУЛЯЦІЯ, ПОВЕДІНКОВИЙ АНАЛІЗ, КЛАСИФІКАЦІЯ КОРИСТУВАЧІВ.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі медичні додатки, зокрема ті, що використовують електростимуляцію для лікувальних і реабілітаційних цілей, стають все більш поширеними [1]. Bluefit — один з таких продуктів, що дозволяє проводити процедури електростимуляції для різних частин тіла з використанням специфічних програм. Прогнозування взаємодії користувачів з такими додатками за допомогою методів машинного навчання є актуальним, оскільки може значно підвищити ефективність лікування, покращити персоналізацію досвіду користувачів та сприяти їхньому здоров'ю [2].

Об’єкт дослідження. Процес взаємодії користувачів з медичним додатком Bluefit, включаючи їхні вибори та поведінку під час використання додатку.

Предмет дослідження. Аналіз поведінки користувачів медичного додатку на основі зібраних даних та розробка моделі машинного навчання, здатної прогнозувати їх майбутню взаємодію з додатком.

Гіпотеза. Поведінка користувачів може слугувати надійним індикатором їхніх майбутніх дій у медичному додатку, і машинне навчання може точно прогнозувати цю взаємодію.

Новизна. Робота зосереджується на специфічному застосуванні машинного навчання у медичних додатках, зокрема на прогнозуванні взаємодії користувачів з Bluefit. Це включає розробку унікальної моделі, заснованої на реальних даних поведінки користувачів, що відкриває нові можливості для персоналізації та покращення медичних додатків.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз сучасного стану проблеми

У сучасному світі, де технології розвиваються стрімкими темпами, медичні додатки займають важливе місце у сфері охорони здоров'я. Вони не тільки сприяють підвищенню доступності медичних послуг, але й надають можливості для інноваційних методів лікування та реабілітації. Одним з таких нововведень є використання електростимуляції через медичні додатки, яке виявилося ефективним у лікуванні різноманітних станів, включаючи біль та реабілітацію після травм [3,4].

Особливу увагу заслуговує застосування методів машинного навчання у медичних додатках. Ці технології можуть значно покращити персоналізацію лікування, аналізуючи поведінку користувачів і пристосовуючи лікувальні програми до їхніх індивідуальних потреб. Такий підхід може революціонізувати спосіб взаємодії користувачів з медичними додатками, зокрема тими, що використовують електростимуляцію.

Машинне навчання (МН) — це галузь штучного інтелекту, яка зазнала стрімкого розвитку протягом останніх декількох десятиліть, зокрема у сфері охорони здоров'я [5]. Історія машинного навчання у медицині розпочалася з простих моделей та алгоритмів, але з часом технології стали більш складними та точними, відкриваючи нові можливості для діагностики, лікування та дослідження. У 1980-х роках почали з'являтися перші системи, засновані на правилах, які використовувались для аналізу медичних даних. Це були здебільшого експертні системи, які могли виконувати базові функції діагностики на основі визначених правил і логіки [6]. Однак ці системи були обмежені своєю здатністю адаптуватися до нових ситуацій або даних, що не вписувалися в їхній заздалегідь визначений алгоритм. Прогрес у галузі обчислювальної техніки та збільшення обсягів даних у 1990-2000-х роках дали поштовх для розвитку більш складних моделей МН. Завдяки цьому

вдалося досягти значних успіхів у таких областях, як обробка природної мови (NLP), розпізнавання зображень та прогнозування. Це дозволило медичним фахівцям аналізувати великі обсяги даних, отримуючи більш точну інформацію для підтримки клінічних рішень. З появою глибокого навчання та нейронних мереж у 2010-х роках відкрилися нові горизонти для медичних застосунків МН. Системи глибокого навчання здатні аналізувати складні медичні зображення, визначати шаблони у великих наборах даних та навіть передбачати ризики розвитку захворювань на основі історії пацієнтів. Сьогодні машинне навчання має широке застосування у медицині: від ранньої діагностики та персоналізованої медицини до розробки нових лікарських препаратів та методів лікування [7]. Ці технології відіграють ключову роль у підвищенні ефективності медичного обслуговування та покращенні якості життя пацієнтів.

Машинне навчання (МН) в останні роки стало невід'ємною частиною медичної сфери, вносячи суттєві зміни у способи діагностики, лікування та ведення пацієнтів. Сучасні тенденції та застосування МН у медицині характеризуються наступними особливостями:

- Персоналізована медицина:
    1) МН дозволяє розробляти індивідуалізовані лікувальні плани, враховуючи генетичні, біохімічні та фізіологічні особливості кожного пацієнта.
    2) Системи, засновані на МН, аналізують великі набори даних про стан здоров'я, історію захворювань та результати лабораторних досліджень, щоб визначити найефективніші методи лікування.
- Діагностика захворювань:
    1) Алгоритми МН використовуються для аналізу медичних зображень, таких як рентгенівські знімки, МРТ та УЗД, дозволяючи точно діагностувати різні стани та захворювання.
    2) Це особливо важливо у виявленні ранніх стадій онкологічних та

серцево-судинних захворювань, де точність діагностики може значно вплинути на успіх лікування.

- Управління даними пацієнтів:
    1) МН використовується для оптимізації управління даними пацієнтів, включаючи автоматизацію медичних записів та покращення системи надання медичних послуг.
    2) Це забезпечує більшу точність у веденні медичних карток, а також ефективність у координації догляду за пацієнтами.
- Розробка лікарських засобів:
    1) МН сприяє більш швидкому та ефективному процесу розробки нових медичних препаратів, аналізуючи потенційну ефективність та безпеку лікарських засобів на ранніх стадіях їх розробки.
    2) Використання передових алгоритмів МН може значно скоротити час та витрати, пов'язані з клінічними випробуваннями.

Ці та інші застосування машинного навчання революціонізують медичну галузь, відкриваючи нові горизонти для вдосконалення діагностичних засобів, розробки лікувальних методів та підвищення якості медичного обслуговування.

Машинне навчання (МН) принесло значні переваги у медичну галузь, але разом з тим воно стикається з рядом викликів, які потрібно враховувати для його ефективного використання.

Виклики:

- Збір та обробка даних:
    1) Одним з найбільших викликів є збір та обробка великих обсягів медичних даних, які часто є неструктурованими або розподіленими.
    2) Забезпечення якості даних та їх відповідності потребам конкретних медичних додатків є ключовим для успішного застосування МН.

- Конфіденційність та безпека даних:
    1) Захист конфіденційності пацієнтів та забезпечення безпеки медичних даних є критично важливим, особливо з огляду на зростаючі вимоги до дотримання норм GDPR та інших законодавчих актів.
    2) Вирішення цих питань вимагає розробки та впровадження надійних методів шифрування та захисту даних.
- Інтеграція з існуючими медичними системами:
    1) Інтеграція МН з існуючими медичними системами та процесами може бути складною задачею.
    2) Потрібна сумісність з різними медичними протоколами та стандартами, що вимагає глибокого розуміння медичної практики та потреб пацієнтів.

Можливості:

- Поліпшення якості медичного обслуговування:
    1) МН має потенціал значно підвищити точність діагностики, підвищити ефективність лікування та надати краще розуміння хвороб та їхньої динаміки.
    2) Це веде до поліпшення якості медичного обслуговування та підвищення рівня задоволеності пацієнтів.
- Розвиток персоналізованої медицини: Завдяки здатності аналізувати величезні набори даних, МН може сприяти розвитку персоналізованої медицини, де підхід до лікування кожного пацієнта адаптується під їх індивідуальні особливості.
- Інновації у розробці лікарських засобів: МН може відігравати ключову роль у розробці нових лікарських засобів, дозволяючи проводити більш швидкі та ефективні дослідження.

Ці виклики та можливості формують поточний ландшафт застосування машинного навчання у медицині, вказуючи на потребу балансу між

інноваціями та відповідальним використанням технологій.

Коротко розглянемо специфіку застосування машинного навчання у додатках електростимуляції. Машинне навчання відкриває нові горизонти у використанні електростимуляції, яка є ключовим елементом у багатьох медичних процедурах, включаючи лікування болю, реабілітацію після травм та підтримку фізіотерапії. Використання МН у цих застосунках надає унікальні можливості та виклики.

Можливості:

- Персоналізація лікування:
    1) МН може аналізувати дані про стан здоров'я користувачів, їхні відповіді на лікування та індивідуальні особливості, щоб адаптувати програми електростимуляції до конкретних потреб кожного пацієнта.
    2) Це дозволяє підвищити ефективність процедур та мінімізувати можливі побічні ефекти.
- Покращення результатів лікування:
    1) Застосування МН допомагає виявити найбільш ефективні параметри стимуляції для кожного конкретного випадку, що може сприяти швидшому відновленню пацієнтів.
    2) Алгоритми МН можуть аналізувати великі набори даних для визначення оптимальних режимів стимуляції.
- Інтеграція з іншими медичними даними: Інтеграція даних електростимуляції з іншими медичними даними пацієнта може надати цінну інформацію для комплексного підходу до лікування.

Виклики:

- Забезпечення точності та надійності:
    1) Важливо гарантувати, що моделі МН точно інтерпретують медичні дані, щоб уникнути помилкових рекомендацій щодо лікування.

2) Впровадження МН у медичні додатки вимагає строгих клінічних випробувань та валідації.

- Інтерфейс користувача та зручність використання:
    1) Розробка інтуїтивно зрозумілого інтерфейсу користувача, який враховує потреби та можливості різних груп пацієнтів.
    2) Необхідність забезпечення простоти використання додатків електростимуляції, щоб користувачі могли ефективно взаємодіяти з ними.

Специфіка застосування машинного навчання у додатках електростимуляції полягає у знаходженні оптимального балансу між інноваційними технологіями та практичністю їх застосування в медичній практиці. Завданням розробників є створення систем, які не тільки покращують медичне обслуговування, але й забезпечують безпечність та комфорт для пацієнтів.

З огляду на важливість цього напрямку, це дослідження має на меті розробити модель машинного навчання для прогнозування взаємодії користувачів з медичним додатком Bluefit, який використовує електростимуляцію. Мотивацією для цього є не тільки підвищення ефективності лікування, але й поліпшення загального досвіду користувачів з додатком.

1.2 Аналіз аналогічних проєктів та розробок

Розробка інформаційних систем та технологій, зокрема у медичній сфері, потребує ґрунтовного аналізу існуючих рішень та підходів. Це дозволяє визначити актуальні напрями досліджень, оцінити можливості різних методів, а також уникнути повторення вже проведених розробок.

В даному розділі наведено результати огляду та порівняльного аналізу наукових публікацій щодо застосування методів аналізу даних та машинного навчання для вивчення поведінки користувачів медичних інформаційних систем. Акцент зроблено на дослідженнях прогнозування взаємодії користувача з медичними додатками на базі аналізу його поведінкових реакцій. Проведено відбір та систематизацію чотирнадцяти найбільш релевантних і цитованих наукових статей в цій предметній області за останні 3 роки.

Ключові характеристики обраних публікацій, а саме: назва дослідження, його мета та задачі, застосовані методи, отримані результати та висновки, актуальність і можливості практичного застосування – викладено у порівняльній таблиці. Це дозволяє сформувати загальне уявлення щодо сучасного стану та актуальних напрямів досліджень у даній області. На підставі проведеного аналізу будуть визначені методологія та підходи до розробки прогнозної інформаційної системи в роботі.

Наукова стаття "DeepReco: Deep Learning Based Health Recommender System Using Collaborative Filtering" авторства Abhaya Kumar Sahoo та інших фокусується на розробці рекомендаційної системи у сфері охорони здоров'я, використовуючи методи глибокого навчання та спільної фільтрації. Основна мета такої системи - аналіз великих обсягів медичних даних для прогнозування стану здоров'я пацієнтів, виходячи з їх способу життя, медичних записів та соціальної активності. Стаття розглядає використання методів колаборативної фільтрації, глибокого навчання, зокрема мережі RBM

(Restricted Boltzmann Machine) та CNN (Convolutional Neural Network), для покращення якості рекомендацій [8].

Аналіз:

- Ключові аспекти:
    1) Роль та важливість рекомендаційних систем у сфері охорони здоров'я.
    2) Використання глибокого навчання та аналізу великих даних для поліпшення медичного обслуговування.
- Використані методи:
    1) Колаборативна фільтрація.
    2) Методи глибокого навчання, такі як RBM та CNN.
    3) Різні підходи обробки та аналізу медичних даних.
- Основні висновки:
    1) Підтвердження ефективності глибокого навчання у рекомендаційних системах охорони здоров'я.
    2) Виявлення значного потенціалу використання великих даних для персоналізованих медичних рекомендацій.
- Релевантність до роботи: Підходи та технології, описані в статті, можуть бути використані у дослідженні для розробки прогнозуючої інформаційної системи у медичному додатку.

Наукова стаття “Exploring the factors that affect user experience in mobile-health applications” авторів Shounak Pal та інших фокусується на аналізі факторів, що впливають на досвід користувачів мобільних медичних додатків (mHealth), використовуючи текстовий майнінг та методи машинного навчання. Основний акцент робиться на впливі таких факторів як зручність, доступність, відгуки користувачів та їх полірність, що визначають успіх додатків mHealth. Також враховується вплив COVID-19 на зростання популярності mHealth додатків та зміну особливостей, що впливають на загальний досвід користувачів [9].

Аналіз:

- Ключові аспекти:
  1) Значення mHealth додатків для медичної сфери.
  2) Вплив відгуків користувачів на якість та успіх mHealth додатків.
- Використані методи:
  1) Текстовий майнінг та аналіз відгуків.
  2) Машинне навчання для аналізу даних.
- Основні висновки:
  1) Встановлення зв'язку між відгуками користувачів та якістю досвіду використання mHealth додатків.
  2) Ідентифікація ключових факторів, що впливають на успішність mHealth додатків.
- Релевантність до роботи: Методики та висновки цієї роботи можуть бути корисними для аналізу взаємодії користувачів з медичними додатками в дослідженні.

Стаття “Testing the Generalizability of an Automated Method for Explaining Machine Learning Predictions”, авторства Yao Tong та інших, зосереджується на тестуванні загальної придатності автоматизованого методу для пояснення прогнозів машинного навчання у випадках візитів пацієнтів із астмою до лікарні. Метою дослідження було оцінити, наскільки добре цей метод застосовується у різних академічних медичних системах, порівняно зі звичайними системами охорони здоров'я [10].

Аналіз:

- Ключові аспекти:
  1) Роль машинного навчання у прогнозуванні візитів до лікарні пацієнтів з астмою.
  2) Розвиток та тестування автоматизованих методів для пояснення прогнозів машинного навчання.
- Використані методи:

1) Автоматизований метод для створення пояснень, який використовує класифікаційні правила.
2) Моделі машинного навчання, такі як XGBoost, для аналізу медичних даних.

- Основні висновки:
  1) Метод пояснення мав задовільну загальну придатність для Університету Вашингтона в Медицині.
  2) Метод пояснення прогнозів машинного навчання може бути використаний для поліпшення управління астмою та зменшення витрат на охорону здоров'я.
- Релевантність до роботи: Розробка та застосування автоматизованих методів пояснення машинного навчання може бути корисною для цієї роботи, зокрема для аналізу поведінки користувачів медичних додатків.

Стаття "HCBiLSTM: A hybrid model for predicting heart disease using CNN and BiLSTM algorithms", авторства Prashant Kumar Shrivastava та інших, представляє гібридну модель HCBiLSTM для прогнозування захворювань серця, використовуючи алгоритми CNN і Bi-LSTM. Важливість цього дослідження обумовлена зростанням смертності від серцевих захворювань, яке становить значну загрозу для глобального здоров'я [11].

Аналіз:

- Ключові аспекти:
  1) Розвиток та використання глибоких навчальних методів для прогнозування захворювань серця.
  2) Підхід до вирішення проблеми відсутніх даних та незбалансованих даних у наборі даних.
- Використані методи:
  1) Використання CNN для ефективної обробки даних.
  2) Застосування Bi-LSTM для аналізу часових послідовностей.
- Основні висновки: Гібридна модель досягла високого рівня точності (96.66%) у прогнозуванні захворювань серця.

- Релевантність до роботи: Техніки та методології, застосовані в цій роботі, можуть бути корисними для розробки прогностичних інструментів у дослідженні, особливо у контексті аналізу медичних даних.

Стаття "Healthcare predictive analytics using machine learning and deep learning techniques: a survey" надає широкий огляд застосування машинного навчання та глибокого навчання у сфері прогностичного аналізу в охороні здоров'я. Автори акцентують увагу на важливості точного та своєчасного прогнозування хвороб, що може значно вплинути на якість життя пацієнтів та навіть врятувати життя [12].

Аналіз:

- Ключові аспекти:
    1) Важливість прогностичного аналізу в охороні здоров'я.
    2) Роль машинного навчання та глибокого навчання у покращенні медичного діагностування і лікування.
- Використані методи:
    1) Різноманітні техніки машинного навчання, такі як лінійна регресія, логістична регресія, дерево рішень, випадковий ліс, метод опорних векторів, алгоритм k-найближчих сусідів та Наївний Байєсівський класифікатор.
    2) Методи глибокого навчання, такі як сверточні нейронні мережі (CNN), довготривалі короткочасні пам'яті (LSTM) та повторювані сверточні нейронні мережі (RCNN).
- Основні висновки: Прогностичний аналіз у охороні здоров'я є критично важливим для точності передбачення захворювань.
- Релевантність до роботи: Застосування цих методів та технік може значно покращити точність та ефективність дослідження в прогнозуванні взаємодії користувача з медичним додатком.

Стаття "Predicting healthcare trajectories from medical records: A deep learning approach", написана командою авторів з університету Deakin, описує DeepCare, глибоку динамічну нейронну мережу, яка зчитує медичні записи, зберігає історію хвороби, робить висновки про поточний стан хвороби та прогнозує майбутні медичні результати. Модель зосереджується на персоналізованій передбачувальній медицині, аналізуючи довгострокові часові залежності в медичних даних та враховуючи епізодичний та нерегулярний характер медичних записів [13].

Аналіз:

- Ключові аспекти:
    1) Розробка та застосування моделі DeepCare для аналізу медичних даних.
    2) Підхід до моделювання тривалих часових залежностей і нерегулярності в медичних записах.
- Використані методи:
    1) Використання LSTM (Long Short-Term Memory) для зберігання історії хвороби.
    2) Представлення епізодів догляду як векторів, агрегація станів здоров'я через багаторівневе часове угрупування.
- Основні висновки: Модель DeepCare ефективно вирішує завдання прогнозування перебігу хвороби, рекомендації втручань та прогнозування майбутнього ризику.
- Релевантність до роботи: Методики та підходи, описані в статті, можуть бути використані для розробки прогностичної системи у дослідженні.

Стаття "Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare" пропонує метод діагностики серцевих захворювань, заснований на техніках машинного навчання. Серцеві захворювання є складною проблемою в глобальному масштабі, і точна та

своєчасна діагностика відіграє ключову роль у сфері охорони здоров'я. Система розроблена на основі алгоритмів класифікації, таких як машина опорних векторів, логістична регресія, штучна нейронна мережа, метод найближчих сусідів, наївний баєсівський класифікатор та дерево рішень. Використовуються алгоритми відбору ознак, такі як Relief, mRMR, LASSO та LLBFS, для видалення нерелевантних та зайвих ознак [14].

Аналіз:

- Ключові аспекти:
    1) Застосування машинного навчання для ідентифікації серцевих захворювань.
    2) Використання відбору ознак для підвищення точності класифікації.
- Використані методи:
    1) Класифікаційні алгоритми машинного навчання.
    2) Алгоритми відбору ознак для попередньої обробки даних.
- Основні висновки: Висока точність ідентифікації серцевих захворювань за допомогою запропонованої системи.
- Релевантність до роботи: Підходи та техніки, використані у цьому дослідженні, можуть бути корисними для дослідження у контексті аналізу поведінки користувачів медичних додатків.

Стаття "Heart Disease Prediction: A Novel Approach Using Strength Scores in Weighted Associative Rule Mining" (автори Yazdani et al., 2021). Це дослідження зосереджується на передбаченні захворювань серця за допомогою зваженого асоціативного правила майнінгу (WARM), використовуючи датасет з UCI Machine Learning Repository [15].

Аналіз:

- Основні аспекти дослідження:
    1) Дослідження адресує проблему визначення сили значущих ознак, які сприяють передбаченню захворювань серця.

2) Методологія включає декілька етапів: передобробка даних, вибір ознак, розрахунок ваги ознак, застосування WARM та оцінка моделі.
3) Використовується ваговий підхід до асоціативного правила майнінгу для визначення сили значущих ознак.

- Використані методи:
1) Застосовано WARM, що дозволяє визначати відносини між ознаками і генерувати правила, які призводять до певних прогнозів.
2) Алгоритм Apriori для генерації правил та рівня впевненості.

- Основні висновки:
1) Виявлено важливі ознаки та правила для діагностики хвороб серця.
2) Найвищий рівень впевненості у прогнозуванні хвороб серця склав 98%.
3) Дослідження забезпечило значний внесок у розрахунок силових балів з визначенням значущих прогнозистів.

- Релевантність до роботи:
1) Дослідження надає важливі методи та підходи для прогнозування взаємодії користувача з медичними додатками, особливо в контексті хвороб серця.
2) Підходи, використані у цьому дослідженні, можуть бути адаптовані для аналізу поведінки користувачів в медичних додатках.

Стаття "A New Hybrid Predictive Model to Predict the Early Mortality Risk in Intensive Care Units on a Highly Imbalanced Dataset" представляє нову гібридну модель для прогнозування ризику ранньої смертності у пацієнтів в інтенсивній терапії на основі високо незбалансованих даних. Автори використовують Генетичний Алгоритм як метод вибору ознак та нову

ансамблеву класифікацію, засновану на комбінації методів Stacking та Boosting, для створення моделі прогнозування. Метод SVM-SMOTE використовується для вирішення проблеми незбалансованих даних [16].

Аналіз:

- Ключові аспекти:
    1) Розробка гібридної моделі для прогнозування ранньої смертності в інтенсивній терапії.
    2) Використання Генетичного Алгоритму для вибору ознак.
- Використані методи:
    1) Генетичний Алгоритм, Stacking та Boosting ансамблеві методи.
    2) SVM-SMOTE для вирішення проблеми незбалансованих даних.
- Основні висновки:
    1) Гібридна модель показала найкращі результати порівняно з іншими класифікаторами.
    2) Високі показники точності та AUC, підтверджені тестуванням на незалежному наборі даних MIMIC-III.
- Релевантність до роботи: Методи та підходи, використані у статті, можуть бути корисними для розробки ваших прогностичних моделей, особливо у контексті медичних додатків.

Праця "Intelligent Machine Learning Approach for Effective Recognition of Diabetes in E-Healthcare Using Clinical Data" зосереджується на розробці системи діагностики для виявлення діабету в електронній охороні здоров'я, використовуючи методи машинного навчання. Основна мета - аналіз медичних даних для точної діагностики діабету, враховуючи проблеми існуючих систем, такі як високий час обчислень та низька точність прогнозування. Стаття розглядає використання алгоритму Decision Tree для відбору важливих ознак, а також ансамблевих алгоритмів Ada Boost і Random Forest для класифікації здорових та хворих на діабет осіб [17].

Аналіз:

- Ключові аспекти:
    1) Значення точної діагностики діабету у сфері електронної охорони здоров'я.
    2) Проблеми з високим часом обчислень та низькою точністю в існуючих системах.
- Використані методи:
    1) Відбір ознак за допомогою алгоритму Decision Tree (ID3).
    2) Ансамблеві алгоритми Ada Boost і Random Forest для класифікації.
    3) Крос-валідація, включаючи методи hold out, K-Folds, та leave one subject out.
- Основні висновки:
    1) Ефективність алгоритму Decision Tree (ID3) у відборі важливих ознак для класифікації.
    2) Виявлення значущості ознак, таких як концентрація глюкози в плазмі, для діагностики діабету.
- Релевантність до роботи: Методи та підходи, описані в статті, можуть бути застосовані у дослідженні для розробки інформаційної системи прогнозування взаємодії користувача з медичним додатком, зокрема в аналізі поведінкових даних.

Праця "Using Machine Learning to Predict the Information Seeking Behavior of Clinicians Using an Electronic Medical Record System" зосереджується на розробці системи Learning Electronic Medical Record (LEMR), яка передбачає поведінку лікарів під час пошуку інформації у електронній медичній записі (EMR) за допомогою машинного навчання. Основна мета системи - забезпечити лікарям доступ до релевантних та контекстно-чутливих медичних даних. Стаття описує методику збору та моделювання поведінки лікарів, а також розробку машинного навчання для прогнозування цієї поведінки [18].

Аналіз:

- Ключові аспекти:
    1) Низька зручність використання EMR, що впливає на задоволеність лікарів та пацієнтів.
    2) Варіабельність поведінки лікарів при пошуку інформації залежно від контексту, такого як тип користувача EMR, клінічне завдання та конкретний випадок пацієнта.
- Використані методи:
    1) Збір даних про поведінку лікарів та використання машинного навчання для передбачення потреб у даних.
    2) Процес вибору ознак та підготовка даних для машинного навчання.
    3) Застосування алгоритмів машинного навчання, таких як лассо логістична регресія, класифікатор на основі опорних векторів та випадковий ліс, з використанням перехресної валідації.
- Основні висновки:
    1) Розроблено моделі для прогнозування поведінки лікарів під час використання EMR.
    2) Значна частина моделей показала високу продуктивність, що є обнадійливим для майбутнього використання у практиці.
- Релевантність до роботи: Методики та підходи, описані у статті, можуть бути корисними для вашої магістерської роботи, зокрема для розробки систем, які використовують машинне навчання для аналізу поведінки користувачів медичних додатків та їх взаємодії з інформаційними системами.

Наукова стаття "Method for Prediction Dynamic Features of Online Social Networks from Users’ Activity Based on Machine Learning" досліджує застосування машинного навчання для прогнозування та аналізу динамічних графічних характеристик онлайн соціальних мереж. Вона фокусується на

використанні декількох технік машинного навчання для прогнозування змін у характеристиках графів, які виникають внаслідок взаємодій користувачів. Розроблена модель прогнозує глобальні характеристики графів, використовуючи обмежену кількість інформації, і демонструє високу точність у передбаченні [19].

Аналіз:

- Ключові аспекти:
    1) Використання графів для моделювання складних взаємодій у соціальних мережах.
    2) Застосування машинного навчання для прогнозування змін у характеристиках графів.
- Використані методи:
    1) Прогнозування змін у графах на основі аналізу даних користувачів.
    2) Використання різних методів машинного навчання, включаючи Random Forest, TreeNet, KNN-Regression, GRU, MARS, та XGBoost для прогнозування.
- Основні висновки:
    1) Ефективність Random Forest у прогнозуванні глобальних характеристик динамічних графів.
    2) Важливість Графічного Середнього Кластерного Коефіцієнта (GACC) у передбаченні глобальних динамік графів.
- Релевантність до роботи: Методи та підходи, описані у статті, можуть бути застосовані для аналізу взаємодії користувачів з медичними додатками, зокрема для прогнозування їх поведінки та взаємодії з інформаційною системою.

Наукова стаття "Using Machine Learning Applied to Real-World Healthcare Data for Predictive Analytics: An Applied Example in Bariatric Surgery" зосереджується на розробці та валідації прогностичної моделі для

передбачення припинення вживання антиглікемічних ліків (як проксі для контролю A1c) у пацієнтів з лікуваним діабетом другого типу (T2D), які перенесли метаболічну хірургію. Дослідження демонструє, як техніки машинного навчання можуть бути застосовані до реальних медичних даних для створення корисних прогностичних моделей, які можуть допомогти у виборі пацієнтів для лікування [20].

Аналіз:

- Ключові аспекти:
    1) Використання прогностичних моделей на рівні пацієнта для визначення ймовірності досягнення позитивного результату лікування.
    2) Підхід до використання великих медичних баз даних для прогнозування результатів після метаболічної хірургії.
- Використані методи:
    1) Розробка та валідація моделі на основі медичних даних з баз даних Truven Health MarketScan Commercial (CCAE) та Optum Clinformatics (Optum).
    2) Використання лассо логістичної регресії для тренування моделі та 10-кратної перехресної валідації для визначення оптимального гіперпараметра.
- Основні висновки:
    1) Висока точність внутрішньої валідації моделі з AUC 0.778 та зовнішньої валідації з AUC 0.759, що свідчить про хорошу транспортабельність моделі.
    2) Порівняння з іншими моделями (Gradient Boosting Machine, Random Forest, AdaBoost) показало, що лассо логістична регресія є більш парсимонічним та інтерпретованим підходом.
- Релевантність до роботи: Розроблені підходи та моделі можуть бути корисними для магістерської роботи, особливо в контексті

використання великих медичних баз даних для прогнозування поведінкових реакцій та взаємодій користувачів з медичними додатками

Наукова стаття "Using Machine Learning and Thematic Analysis Methods to Evaluate Mental Health Apps Based on User Reviews" досліджує використання машинного навчання (ML) та тематичного аналізу для оцінки ефективності 104 ментальних здоров'я додатків на основі 88125 відгуків користувачів з Google Play та App Store. В статті розроблено та порівняно продуктивність п'яти класифікаторів на основі алгоритмів машинного навчання для визначення сентиментальної полярності відгуків. Далі проведено тематичний аналіз позитивних та негативних відгуків для ідентифікації факторів, які позитивно та негативно впливають на ефективність додатків для ментального здоров'я, виявивши 21 негативну та 29 позитивних тем [21].

Аналіз:

- Ключові аспекти:
    1) Оцінка ментальних здоров'я додатків на основі аналізу відгуків користувачів.
    2) Використання машинного навчання для класифікації відгуків.
- Використані методи:
    1) Збір та передобробка даних відгуків користувачів.
    2) Векторизація даних відгуків за допомогою техніки TF-IDF.
    3) Розробка та тренування класифікаторів (SVM, MNB, SGD, LR, RF) з використанням 10-кратної крос-валідації.
    4) Тематичний аналіз відгуків з використанням NVivo 12.
- Основні висновки:
    1) Висока точність SGD класифікатора (F1-бал 89.42%) у передбаченні сентиментальної полярності відгуків.
    2) Виявлення різноманітних факторів, які впливають на ефективність додатків для ментального здоров'я.

- Релевантність до роботи: Підходи, методи та висновки, описані у статті, можуть бути корисними для роботи у сфері аналізу поведінки користувачів медичних додатків, особливо для визначення факторів, що впливають на їх взаємодію з додатками.

Проведений огляд та систематизація результатів досліджень у сфері прогнозної аналітики взаємодій користувача з медичними додатками дозволяють виділити ряд ключових параметрів для їх порівняльної оцінки. Для зручності порівняння основних характеристик проаналізованих публікацій було сформовано узагальнену таблицю (Таблиця 1.1) Така таблиця дозволяє компактно представити змістовну сутність розглянутих публікацій та провести їх системне порівняння за обраними критеріями. На її основі можна виділити найбільш ефективні підходи та перспективні напрями подальших досліджень.

Таблиця 1.1 - Порівняльний аналіз досліджень щодо прогнозування у медицині

Назва дослідження	**Ключові аспекти**	**Використані методи**	**Основні висновки**	**Релевантність**
1	**2**	**3**	**4**	**5**
DeepReco: Deep Learning Based Health Recommender System Using Collaborative Filtering	Роль рекомендаційних систем у охороні здоров'я; Використання глибокого навчання та аналізу великих даних	Колаборативна фільтрація; Методи глибокого навчання (RBM, CNN)	Ефективність глибокого навчання та великих даних для медичних рекомендацій	Підходи можуть бути застосовані для розробки інформаційної системи

Назва дослідження	Ключові аспекти	Використані методи	Основні висновки	Релевантність
1	2	3	4	5
Exploring the factors that affect user experience in mHealth applications	Значення mHealth додатків; Вплив відгуків користувачів	Текстовий майнінг; Машинне навчання	Зв'язок між відгуками користувачів та якістю досвіду користувачів	Методи можуть бути використані для аналізу взаємодії з медичними додатками
Testing the Generalizability of an Automated Method for Explaining Machine Learning Predictions	Роль машинного навчання у прогнозуванні астматичних візитів до лікарні; Розвиток автоматизованих методів пояснень	Автоматизований метод пояснень; Моделі машинного навчання (XGBoost)	Загальна придатність методу для різних медичних систем	Методи можуть бути застосовані для аналізу поведінки користувачів медичних додатків
HCBiLSTM: A hybrid model for predicting heart disease using CNN and BiLSTM algorithms	Розвиток глибоких навчальних методів для прогнозування захворювань серця	CNN; Bi-LSTM; Додатковий класифікатор дерева	Висока точність моделі у прогнозуванні захворювань серця (96.66%)	Методи можуть бути застосовані для розробки прогностичних інструментів у медичних дослідженнях

Назва дослідження	Ключові аспекти	Використані методи	Основні висновки	Релевантність
1	2	3	4	5
Healthcare predictive analytics using ML and DL techniques: a survey	Важливість прогностичного аналізу; Роль МЛ та ГН	Різні методи МЛ та ГН (CNN, LSTM, RF, SVM тощо)	Значущість точного прогнозування в охороні здоров'я	Покращення точності та ефективності прогнозування
Predicting healthcare trajectories from medical records: A deep learning approach	Розробка моделі DeepCare; Аналіз довгострокових часових залежностей	LSTM; Представлення епізодів догляду як векторів; Багаторівневе часове угрупування	Ефективність DeepCare у прогнозуванні перебігу хвороби та рекомендаціях втручань	Підходи можуть бути використані для розробки прогностичної системи
Heart Disease Identification Method Using ML Classification in E-Healthcare	Застосування МЛ для ідентифікації серцевих захворювань	Класифікаційні алгоритми МЛ; Відбір ознак (Relief, mRMR, LASSO, LLBFS)	Висока точність ідентифікації серцевих захворювань	Підходи можуть бути застосовані для аналізу поведінки користувачів
Heart Disease Prediction: A Novel Approach Using Strength Scores in Weighted Associative Rule Mining	Визначення сили значущих ознак для передбачення захворювань серця	Зважений асоціативний правил майнінг (WARM) та алгоритм Apriori	Висока впевненість у прогнозуванні хвороб серця (до 98%)	Методи та підходи можна адаптувати для аналізу взаємодії з медичними додатками

Назва дослідження	Ключові аспекти	Використані методи	Основні висновки	Релевантність
1	2	3	4	5
A New Hybrid Predictive Model to Predict the Early Mortality Risk in Intensive Care Units on a Highly Imbalanced Dataset	Розробка гібридної моделі; Використання Генетичного Алгоритму для вибору ознак	Генетичний Алгоритм, Stacking і Boosting ансамблеві методи; SVM-SMOTE	Висока точність та AUC; Ефективність моделі на незалежному наборі даних MIMIC-III	Методи можуть бути адаптовані для розробки прогностичних систем у медичних додатках
Intelligent Machine Learning Approach for Effective Recognition of Diabetes in E-Healthcare Using Clinical Data	Значення точного виявлення діабету; Проблеми існуючих систем діагностики	Відбір ознак Decision Tree; Ансамблеві алгоритми Ada Boost, Random Forest; Крос-валідація	Ефективність алгоритму DT (ID3) у виборі ознак; Важливість ознак глюкози в плазмі	Важливе для аналізу використання машинного навчання у медичних застосунках
Using Machine Learning to Predict the Information Seeking Behavior of Clinicians Using an Electronic Medical Record System	Низька зручність EMR; Варіабельність поведінки лікарів	Збір даних про поведінку лікарів; Тренування класифікаторів МЛ (SGD, SVM, LR, RF)	Розроблено моделі для прогнозування поведінки лікарів; Висока продуктивність моделей	Може бути застосовано для аналізу взаємодії користувачів з медичними додатками

Назва дослідження	Ключові аспекти	Використані методи	Основні висновки	Релевантність
1	2	3	4	5
Method for Prediction Dynamic Features of Online Social Networks from Users' Activity Based on Machine Learning	Моделювання взаємодій у соціальних мережах через графи; Застосування МЛ для прогнозування змін у графах	Використання різних методів МЛ (Random Forest, TreeNet, KNN-Regression, GRU, MARS, XGBoost)	Висока точність Random Forest; Важливість GACC у передбаченні глобальних динамік графів	Можливість застосування для аналізу взаємодії користувачів з медичними додатками
Using Machine Learning Applied to Real-World Healthcare Data for Predictive Analytics: An Applied Example in Bariatric Surgery	Прогностичні моделі в охороні здоров’я; Використання великих медичних баз даних	Розробка моделі на даних з CCAE та Optum; Лассо логістична регресія та перехресна валідація	Висока точність моделі (внутрішня AUC 0.778, зовнішня AUC 0.759); Ефективність лассо логістичної регресії	Може бути застосовано для аналізу поведінки користувачів медичних додатків
Using Machine Learning and Thematic Analysis Methods to Evaluate Mental Health Apps Based on User Reviews	Аналіз відгуків користувачів; Класифікація відгуків МЛ	Передобробка, векторизація даних; Тренування класифікаторів (SGD, SVM, LR, RF); Тематичний аналіз відгуків	Висока точність SGD класифікатора (F1-бал 89.42%); Виявлення ключових факторів ефективності додатків	Може бути застосовано для аналізу взаємодії користувачів з медичними додатками

На основі проведеного огляду та порівняльного аналізу десяти наукових публікацій можна зробити висновок, що прогнозування взаємодії користувачів з медичними інформаційними системами є актуальним напрямом досліджень на стику штучного інтелекту, аналізу даних та охорони здоров’я.

Як видно з підсумкової таблиці порівняння (див. Таблиця 1.1), ключовими технологіями та методами в цій предметній області є: машинне навчання, зокрема класифікаційні та регресійні моделі; методи аналізу тексту та природної мови для обробки відгуків користувачів; техніки глибокого навчання на основі нейронних мереж для прогнозування розвитку захворювань.

Результати досліджень демонструють ефективність комбінованого застосування цих підходів, що дозволяє досягти високої точності і надійності прогнозних моделей поведінки користувачів медичних додатків. Ці методики будуть використані в практичній частині роботи для розробки інтелектуальної інформаційної технології з аналізу та передбачення взаємодій користувача з обраним медичним додатком.

1.3 Постановка задачі

Метою роботи є створення інформаційної технології для прогнозування взаємодії користувача з медичним додатком Bluefit на основі аналізу поведінки за допомогою методів машинного навчання. Проблема полягає у виявленні закономірностей та факторів, що впливають на поведінку користувачів, з метою підвищення ефективності та персоналізації використання додатку.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- Проаналізувати наявні дані про взаємодію користувачів з додатком Bluefit для виявлення закономірностей та особливостей їх поведінки.
- Обрати та реалізувати методи машинного навчання для аналізу даних про поведінку користувачів та прогнозування їхніх дій. Зокрема, планується використання таких методів:
    1) Логістична регресія, метод опорних векторів, нейронні мережі - для класифікації користувачів за рівнем активності та іншими категорійними ознаками.
    2) К-середніх, ієрархічна кластеризація - для кластеризації користувачів залежно від поведінкових характеристик.
    3) Лінійна та логістична регресія, випадковий ліс - для створення моделей прогнозування часових рядів.

  Для підвищення якості прогнозів буде застосовано ансамблі моделей та комплексне використання різних методів машинного навчання.
- Розробити та налаштувати моделі машинного навчання для:
    1) класифікації користувачів за рівнем активності (активні, помірно активні, пасивні) на основі даних про кількість та тривалість їх тренувань;
    2) визначення регулярності використання додатку шляхом аналізу часових інтервалів між сеансами;

3) прогнозування тривалості життєвого циклу користувача в додатку з використанням даних про реєстрацію та останню активність;
4) виявлення схильності користувачів до певних програм електростимуляції за даними про їх переваги.

- Реалізувати алгоритм надання рекомендацій щодо оптимальних електростимуляційних програм для користувачів додатку Bluefit на основі побудованих раніше моделей машинного навчання. Заплановано застосування колаборативної фільтрації - алгоритм проаналізує поведінку схожих користувачів та на основі цього сформує рекомендації. Передбачається реалізація кількох варіантів:
  1) "класичної" колаборативної фільтрації (user-based, item-based)
  2) гібридного варіанту із попередньою кластеризацією користувачів (модифікація к-середніх)
  3) адаптованого алгоритму к-найближчих сусідів з наступною селекцією найбільш популярних програм

  Заплановано оцінку якості отриманих рекомендацій і вибір найкращого підходу на основі показників точності та повноти.
- Провести тестування розробленого функціоналу на реальних даних та оцінити ефективність запропонованого підходу.

В якості вхідних даних використовуватимуться дані про сеанси тренувань користувачів, їх профілі, дані про пристрої тощо. Запропонований підхід, що включає побудову моделей машинного навчання та реалізацію алгоритму персоналізованих рекомендацій, дозволить з високою якістю прогнозувати поведінку користувачів медичного додатку Bluefit. Це надасть можливість підвищити ефективність використання додатку користувачами та їх мотивацію за рахунок отримання персоналізованих рекомендацій щодо оптимальних програм електростимуляції.

2 ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧ

2.1 Огляд та вибір підходів машинного навчання

Машинне навчання надає широкий спектр методів для вирішення задач аналізу даних, прогнозування та оптимізації рішень. Враховуючи постановку задачі дослідження, основними типами методів машинного навчання, які доцільно розглянути, є:

- Методи класифікації - дозволяють віднести об'єкти до попередньо визначених класів/груп на основі їх атрибутів. Можуть застосовуватися для сегментування користувачів медичного додатку за різними ознаками.
- Методи кластерного аналізу - дають можливість групувати дані за схожими характеристиками. Корисні для виявлення прихованих закономірностей у поведінці користувачів.
- Регресійні моделі та методи прогнозування часових рядів - використовуються для передбачення числових значень цільових змінних, наприклад прогнозування взаємодії користувача з додатком у майбутньому.

Детальний аналіз саме цих груп методів дозволить обрати оптимальні підходи для розв'язання конкретних задач дослідження поведінки користувачів медичного додатку.

До ключових методів машинного навчання для вирішення поставлених задач аналізу даних та прогнозування поведінки користувачів медичного додатку відносяться:

Логістична регресія - це метод класифікації, який дозволяє прогнозувати ймовірність належності об'єкта до одного з двох класів (бінарна класифікація).

Логістична регресія придатна для вирішення таких задач як:

- Класифікація користувачів на активних та неактивних
- Виявлення лояльних / нелояльних клієнтів
- Прогнозування ймовірності відтоку користувача

Метод опорних векторів (SVM) також використовується для класифікації та регресії. Перевага SVM - здатність ефективно працювати при складних розділювальних границях між класами об'єктів. SVM доцільно застосовувати для таких задач:

- Визначення рівня активності користувачів
- Класифікація за частотою використання додатку

Кластеризація методом K-середніх дозволяє розбити множину об'єктів на задану кількість кластерів за принципом мінімізації відстаней всередині кластера та максимізації відстаней між кластерами. Цей метод може застосовуватися для вирішення таких задач:

- Сегментування користувачів за схожими характеристиками
- Виявлення груп користувачів зі спільними вподобаннями
- Аналіз різних стадій життєвого циклу користувача

Переваги K-середніх - простота, здатність працювати з великими обсягами даних. Недоліками є чутливість до шумів та викидів.

Випадкові ліси (random forest) - це ансамблевий метод машинного навчання, який використовує безліч дерев рішень для підвищення якості прогнозування. Застосовується в задачах класифікації, регресії та кластеризації даних.

Випадкові ліси доцільно застосовувати для:

- Класифікації користувачів за рівнем активності
- Прогнозування тривалості використання медичного додатку
- Виявлення важливих факторів, що впливають на поведінкові характеристики

Переваги методу - стійкість до шумів в даних, можливість ранжування значущості предикторів, паралелізація обчислень.

Недоліки - складність інтерпретації моделі, схильність до перенавчання.

Алгоритми колаборативної фільтрації широко застосовуються в системах рекомендацій та в завданнях персоналізації послуг для користувачів. Принцип їх роботи базується на аналізі поведінки схожих/“сусідніх” користувачів і формуванні рекомендацій на основі цієї схожості.

Колаборативна фільтрація доцільна для таких задач:

- Розробка алгоритму персоналізованих рекомендацій електростимуляційних програм
- Виявлення прихованих закономірностей у перевагах груп користувачів
- Прогнозування оцінок або відгуків користувачів про опції медичного додатку

Переваги - простота, масштабованість. Недоліки - упередженість оцінювань користувачів, неточності через шумові дані.

З огляду на поставлені в дослідженні задачі, обрані наступні основні методи машинного навчання:

- Логістична регресія обрана для класифікації користувачів за рівнем активності, оскільки дозволяє прогнозувати ймовірності приналежності об'єктів до заданих класів на основі їх характеристик.
- Метод опорних векторів також використовуватиметься для класифікації користувачів за поведінковими ознаками завдяки здатності ефективно знаходити розділювальні гіперплощини у багатовимірному просторі даних.
- Кластеризація K-середніх обрана для виявлення прихованих груп користувачів на основі схожості їх характеристик. Не вимагає попередньої підготовки даних.

- Нейронні мережі будуть застосовуватись для прогнозування часових рядів основних параметрів взаємодії користувачів з медичним додатком завдяки здатності аналізувати складні нелінійні залежності.
- Алгоритм колаборативної фільтрації обраний як основа для створення рекомендаційної системи персоналізованих програм електростимуляції на основі даних про переваги схожих користувачів.

2.2 Розробка та налаштування моделей

На основі проведеного аналізу та вибору методів машинного навчання, необхідно розробити конкретні моделі для розв'язання задач дослідження поведінки користувачів медичного застосунку. Цей етап включає:

- Підготовка вхідних даних - очищення, нормалізація, розбиття на навчальні, валідаційні та тестові вибірки, перевірка на збалансованість.
- Реалізація моделей логістичної регресії, методу опорних векторів, нейронних мереж, випадкового лісу з використанням TensorFlow/Keras, Scikit-Learn.
- Підбір оптимальних гіперпараметрів, налаштування регуляризації та інших механізмів запобігання перенавчання.
- Попередня оцінка якості навчання за метриками точності, відтворюваності та ін. Відбір найкращих моделей.

Для забезпечення якісного навчання та тестування моделей машинного навчання велике значення має належна підготовка вхідних даних. З цією метою було здійснено такі кроки обробки наявних даних:

- Видалення зайвих атрибутів, які не мають суттєвого впливу на цільові змінні (такі як ідентифікатори, багатозначні категоріальні ознаки тощо).
- Заповнення пропущених значень середніми по відповідних атрибутах (імпутація). Видалення залишкових викидів.
- Нормалізація числових ознак за допомогою стандартизації.
- Розподіл даних у співвідношенні 60/20/20 між навчальною, валідаційною та тестовою вибірками.
- Перевірка збалансованості класів у навчальній та тестовій вибірках (для задач класифікації).

Отримані в результаті цих процедур вибірки даних будуть використані для побудови та перевірки адекватності моделей машинного навчання.

Детальний опис цих процедур наводитиметься в розділі 3 в межах підготовки вхідних даних.

Розглянемо ансамбль моделей за допомогою яких будемо вирішувати конкрентні завдання данної роботи.

Логістична регресія – це модель бінарної класифікації, яка дозволяє обчислити ймовірність приналежності об'єкта до одного з двох класів на основі набору характеристик цього об'єкта. Математично вона представляє собою логістичну функцію (сигмоїд), значення якої знаходиться між 0 та 1. Логістична регресія базується на наступному рівнянні:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + ... + b_n X_n)}} \quad (2.1)$$

де:

X - вектор предикторних змінних;

Y - цільова змінна (0 або 1);

$b_0, b_1, ..., b_n$ - коефіцієнти моделі;

e - основа натурального логарифму.

Права частина рівняння являє собою логістичну функцію, яка набуває значень від 0 до 1 - саме це значення і є імовірністю віднесення об'єкта до класу 1. Рівняння описує логарифмічну залежність цільової змінної від предикторів. Коефіцієнти моделі підбираються методом максимальної правдоподібності на основі наявних даних.

В контексті даного дослідження логістична регресія застосовується для бінарної класифікації користувачів на "активних" та "неактивних" на підставі даних про їх взаємодію з додатком Bluefit. Зокрема, до набору предикторних змінних можуть входити такі дані:

- середня тривалість сеансу користувача за місяць

- середня інтенсивність сеансів
- кількість сеансів за місяць
- дані профілю (вік, стать тощо)

Цільова бінарна змінна прийматиме значення 1 для "активних" та 0 для "неактивних" користувачів.

В такому віпадку предиктори для класифікації користувачів додатка Bluefit для формули (2.1) будуть мати вигляд:

X_1 - середня тривалість сеансу

X_2 - середня інтенсивність сеансу

X_3 - вік користувача

В результаті за допомогою логістичної регресії будується модель ймовірності на основі взаємозв'язку предикторів та цільової змінної на навчальних даних.

Метод опорних векторів (Support vector machines) є потужним інструментом машинного навчання для класифікації та регресійного аналізу даних. Він базується на побудові оптимальної розділювальної гіперплощини між різними класами об'єктів. Основне рівняння методу опорних векторів:

$$y = w^T \varphi(x) + b \tag{2.2}$$

де:

y - прогнозоване значення;

w - вектор вагів моделі;

$\varphi(x)$ - функція перетворення вхідних змінних;

b - зміщення.

Формула визначає гіперплощину з максимальною маржею, що забезпечує найкраще узагальнення. На відміну від логістичної регресії, цей метод ефективно працює при складних нелінійних межах між класами.
Однією з ключових переваг SVM є здатність ефективної класифікації даних зі складною структурою та схожими характеристиками об'єктів різних класів.

Це актуально при розподілі користувачів Bluefit на активних та пасивних.До навчальних даних для побудови SVM моделі увійдуть:

- кількість тренувань користувача за місяць
- середня тривалість одного тренування
- середня інтенсивність тренувань

В такому випадку предиктори для класифікації користувачів додатка Bluefit для формули (2.2) будуть мати вигляд:

X_1 - кількість тренувань за місяць

X_2 - середня тривалість тренування

X_3 - середня інтенсивність

Модель дозволить класифікувати нових користувачів за одним сеансом тренування. Її перевага - можливість коректної класифікації об'єктів із перетином характеристик різних класів.

Нейронні мережі реалізують потужний підхід машинного навчання, здатний моделювати складні нелінійні залежності. В основі лежить структура взаємопов'язаних нейронів. Основне рівняння обчислення вихідного сигналу нейрона:

$$y = f(\sum_i x_i w_i + b) \tag{2.3}$$

де:

y - значення вихідного сигналу;

f - функція активації;

x_i - значення i-го входу нейрона;

w_i - ваговий коефіцієнт для i-го входу;

b - зміщення нейрона.

Нейронна мережа складається з вхідного, прихованих та вихідного шарів нейронів. Навчання полягає у підборі вагових коефіцієнтів шляхом мінімізації помилки за допомогою градієнтного спуску. Для задач

прогнозування часових рядів та моделювання поведінки користувачів найкраще підходять рекурентні нейронні мережі (RNN). Вони враховують не тільки поточний вхідний вектор даних, а й попередній стан прихованих нейронів. Зокрема, доцільно застосувати LSTM рекурентну мережу - вона має спеціальний механізм довготривалої пам'яті, що дозволяє ефективно моделювати залежності у послідовностях подій.

Топологія LSTM передбачає ланцюжки послідовно з'єднаних блоків пам'яті. Кожен блок містить нейрони зі зворотними зв'язками, що й реалізує механізм пам'яті. Основна особливість LSTM полягає у структурі прихованих нейронів пам'яті, які складаються з:

- блоку входів
- блоку пам'яті
- блоків керування (вхідного, вихідного, забування)

Робота LSTM в кожний момент часу t визначається наступними формулами:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2.5}$$

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{2.6}$$

$$g_t = \tanh(W_g * [h_{t-1}, x_t] + b_g) \tag{2.7}$$

$$C_t = f_t * C_{t-1} + i_t * g_t \tag{2.8}$$

$$h_t = o_t * \tanh(C_t) \tag{2.9}$$

де i_t, f_t, o_t- вектори керування вхідними/вихідними вентилями та забуванням відповідно, x_t- вхідний вектор даних; W_n - вагові матриці та зміщення.

На кожному кроці t LSTM отримує дані x_t та попередній стан $h_t - 1$. На їх основі оновлюються вектори i_t, f_t, C_t як зазначено в формулах (2.4, 2.5, 2.8) та обчислюється поточний стан h_t формула (2.9). Цей стан підається далі по ланцюжку пам’яті, впливаючи на наступні кроки. Так послідовно LSTM "вивчає" залежності в динамічних даних. Вихід h_t використовується для прогнозування наступних значень часового ряду або ймовірностей подій.

З огляду на наявність значних обсягів даних з часовою складовою (кількість, тривалість, інтенсивність сеансів у різні проміжки часу), для розв'язання цієї задачі оптимально підходять саме рекурентні нейроні мережі. Зокрема, LSTM архітектура дозволяє моделювати поведінку користувача з урахуванням контексту його попередніх взаємодій з додатком. На вхід LSTM подаються дані про історію взаємодій конкретного користувача з додатком Bluefit - кількість, тривалість та інтенсивність його тренувань протягом певного періоду. Наприклад, береться 20 попередніх тижнів. Для кожного тижня рахується середня кількість сеансів, їх тривалість та інтенсивність. Формується вхідний вектор x_t для кожного кроку t, що відповідає конкретному тижню. Далі, на основі формул LSTM, з урахуванням попередньої історії взаємодій користувача, робиться прогноз щодо його активності в наступному тижні:

- Кількість очікуваних сеансів
- Середня тривалість сеансу
- Середня інтенсивність
- Ймовірність припинення використання додатку

Таким чином, LSTM дозволяє з максимальним врахуванням контексту поведінки користувача робити предиктивний аналіз його подальших дій щодо додатку Bluefit.

Метод кластерного аналізу K-середніх (K-means clustering) є одним з найпоширеніших підходів машинного навчання для розбиття множини об'єктів на однорідні групи. В основі лежить мінімізація внутрішньокластерної дисперсії. Математична постановка задачі: знайти розбиття множини $X = \{x_1, x_2, ..., x_n\}$ на K кластерів $S = \{S_1, S_2, ..., S_K\}$, що мінімізує цільову функцію:

$$\sum_{k=1}^{K} \sum_{x_i \in S_k} \|x_i - \mu_k\|^2 \tag{2.10}$$

де μ_k - центр кластера S_k.

Алгоритм ітеративно покращує розбиття шляхом зміни приналежності об'єктів та оновлення центрів кластерів. Метод кластерного аналізу K-середніх може використовуватися в цьому проєкті для розбиття користувачів Bluefit на однорідні сегменти за схожими характеристиками їх взаємодії з додатком. В рамках задачі сегментування користувачів Bluefit, в формулу (2.10) на вхід подаються x_i - вектор даних про окремого користувача він складається з таких параметрів:

- Кількість сеансів користувача за тиждень
- Середня тривалість сеансу
- Середня інтенсивність
- Вік та стать

Також в формулі (2.10) S_k це кластер, до якого віднесено користувач, а μ_k - центр кластеру S_k. Формула використовує Евклідову відстань між вектором користувача x_i та центром його кластеру μ_k. Сума цих відстаней мінімізується шляхом ітерацій алгоритму. В результаті K-середніх розбиває користувачів на групи зі схожими характеристиками взаємодії за метрикою Евклідової відстані у багатовимірному просторі ознак. Таким чином, формується сегментація аудиторії за закономірностями користування додатком.

Лінійна регресія є одним з найпростіших та найпоширеніших статистичних методів моделювання залежності цільової змінної від певного набору предикторів. Вона дозволяє прогнозувати числові значення на основі лінійної комбінації вхідних змінних:

$$y = w_0 + w_1 x_1 + \ldots + w_n x_n \tag{2.11}$$

де:

y - значення цільової змінної;

$x_1, \ldots, x_n$ - значення вхідних предикторів;

$w_0, \ldots, w_n$ - вагові коефіцієнти моделі.

Лінійна регресія моделює залежність відгуку y від лінійної комбінації предикторів x_i. Коефіцієнти w_i визначають внесок кожного предиктора та підбираються методом найменших квадратів з метою мінімізації помилки на навчальних даних.

Для аналізу ефективності тренувань користувачів можна побудувати лінійну регресійну модель їх тривалості від інтенсивності, віку, статі та інших даних профілю, щоб виявити закономірності та зробити кількісний прогноз. Одним з варіантів використання лінійної регресії в дослідженні є побудова прогнозної моделі тривалості тренувань користувача на основі таких даних як інтенсивність тренування, вік, стать, рівень фізичної підготовки. Наприклад, регресійна модель за формулою (2.11) може мати вигляд:

$$\text{duration} = w_0 + w_1 \cdot \text{intensity} + w_2 \cdot \text{age} + w_3 \cdot \text{physical\_fitness} + w_4 \cdot \text{sex}$$

де:

$duration$ - цільова змінна (тривалість елекстростимуляційної програми), яку потрібно спрогнозувати

$intensity$ (інтенсивність програми), age (вік), $physical\_fitness$ (фіз. підготовка), sex (стать) - предикторні змінні моделі;

$w_0, w_1, \ldots, w_4$ - вагові коефіцієнти регресії, що підбираються в процесі навчання.

За допомогою методу найменших квадратів знаходяться оптимальні ваги w_i, за якими розраховується прогнозоване значення тривалості тренування. Аналіз отриманих коефіцієнтів дозволить зрозуміти внесок кожного фактору у загальний результат та виявити закономірності. Загалом модель дозволяє робити кількісний прогноз тривалості на основі даних профілю та діяльності користувача.

Алгоритм випадковий ліс може використовуватися для класифікації користувачів Bluefit за рівнем активності на основі даних про їх взаємодію з додатком: інтенсивність і тривалість сеансів, регулярність використання тощо. Коротко опишемо основні етапи роботи алгоритму.

На першому етапі алгоритм випадкового лісу формує ансамбль з B дерев рішень:

$$F = f1(x), f2(x), ..., fB(x) \tag{2.12}$$

Далі кожне дерево рішень fb будується рекурсивно за алгоритмом CART (Classification and Regression Tree). На кожній ітерації обирається ознака X_j та поріг t, що розбивають вузол на 2 підмножини з максимальною чистотою щодо класу y:

$$Q(X_j, t) = max_j max_{ty \in L} p(y)^2 + _{y \in R} p(y)^2 \tag{2.13}$$

де:

X_j - обрана для розбиття ознака (атрибут);

t - обране порогове значення ознаки;

L - ліва підмножина вузла після розбиття;

R - права підмножина;

y - клас (в нашому випадку - "активний"/"неактивний" користувач);

$p(y)$ - частка класу y в підмножині.

Формула обчислює суму квадратів часток класів в обох підмножинах. Це значення максимізується по X_j та t для знаходження оптимального розбиття вузла. Розбиття зупиняється за критеріями мінімального розміру вузлів, глибини дерева тощо. Листки дерева містять розподіли для класів y. На

тестовому прикладі обчислюється ймовірність класів та вибирається клас з максимальною ймовірністю. Рішення про віднесення об'єкта x до класу j приймається за правилом більшості "голосів":

$$\hat{j} = \max(\sum_{b=1}^{B} I(f_b(x) = j)) \tag{2.14}$$

де:

$\hat{j}$ - предиктований клас об'єкта x (в нашому випадку - "активний" або "неактивний" користувач);

$f_b(x)$ - клас, предиктований b-м деревом рішень;

$I(\cdot)$ - індикаторна функція, що дорівнює 1, якщо $f_b(x)$ збігається з класом $\hat{j}$;

B - загальна кількість дерев рішень в ансамблі.

Формула обирає клас $\hat{j}$ з максимальною кількістю "голосів" серед дерев при їх голосуванні за класи. Отже, алгоритм забезпечує зменшення дисперсії та запобігає перенавчанню, що дає можливість ефективно класифікувати нові дані.

Послідовність роботи алгоритму випадкового лісу та його застосування для класифікації користувачів медичного додатку Bluefit:

1. Формується навчальна вибірка з даних користувачів Bluefit:
   - Кількість сеансів на тиждень
   - Тривалість сеансів
   - Інтенсивність
   - Дані профілю
2. Будується ансамбль з B дерев рішень (формула 2.12) на різних підвибірках даних.
3. Кожне дерево fb класифікує користувача як "активний" або "неактивний" на основі даних про його взаємодію з Bluefit (формула 2.13).
4. Фінальне рішення приймається більшістю "голосів" B дерев щодо

класу користувача (формула 2.14).

Таким чином формується модель класифікації користувачів за рівнем активності у використанні додатку Bluefit на основі ансамблю дерев рішень.

Колаборативна фільтрація - це підхід машинного навчання, що використовується в рекомендаційних системах і ґрунтується на виявленні закономірностей у поведінці схожих між собою користувачів або закономірностей оцінювання схожих об'єктів. Існують два основні варіанти колаборативної фільтрації:

- User-based (на основі схожості користувачів)
- Item-based (на основі схожості об'єктів)

В межах даного проєкту колаборативний підхід доцільно використати для розробки рекомендацій оптимальних програм електростимуляції на базі медичного додатку Bluefit з урахуванням історій взаємодій схожих користувачів. Як вхідні дані для колаборативної фільтрації виступатимуть:

- Матриця оцінок програм електростимуляції різними користувачами
- Дані профілів користувачів (вік, стать, рівень активності)
- Історія взаємодій користувача з програмами (тривалість, інтенсивність)

Математично задача формулюється так:

$$r(u,i) = \ + b_u + b_i + q_i^T * p_u \qquad (2.15)$$

де q_i та p_u - вектори латентних факторів об'єкта і користувача відповідно. Отже, основна формула (2.15) колаборативної фільтрації передбачає розкладання рейтингу користувача u щодо об'єкту i на суму середнього рейтингу μ, зміщень користувача b_u та об'єкту b_i, а також внутрішнього добутку векторів їх латентних факторів q_i та p_u відповідно. Ці фактори і зміщення підбираються таким чином, щоб мінімізувати помилку передбачення рейтингів на навчальних даних. В результаті модель дозволяє

рекомендувати користувачам u ті об'єкти i, які більш схвально оцінені схожими користувачами і задовольняють вподобанням u на основі схожості латентних факторів q_i та p_u

Висновок - в данному розділі було детально розглянуто процес розробки та налаштування конкретних моделей машинного навчання для розв'язання поставлених у роботі задач аналізу даних. Зокрема, обґрунтовано вибір таких моделей як логістична регресія, метод опорних векторів, нейронні мережі, кластеризація K-середніх, лінійна регресія, випадковий ліс та колаборативна фільтрація. Детально описано математичну базу цих методів та підходи до їх застосування у контексті конкретних задач дослідження. Також наведено етапи підготовки даних, необхідні для якісного навчання побудованих моделей. Запропонований у розділі набір моделей та підходів дозволить комплексно вирішити поставлені в роботі задачі аналізу даних щодо взаємодії користувачів з медичним додатком.

2.3 Тестування, валідація та аналіз ефективності моделей

Після того, як моделі ML побудовано та навчено на доступних данних, вкрай важливим є здійснення об'єктивної оцінки їх якості перед подальшим впровадженням у виробниче середовище. Саме тестування та валідація дозволяють оцінити, наскільки ефективно побудовані моделі зможуть працювати на реальних даних, а також виявити їх потенційні недоліки. Основними підходами тут є:

- Крос-валідація
- Валідація на контрольних вибірках
- ROC-аналіз, побудова AUC
- F-міра, критерій Мак-Немара
- Аналіз кривих згортання та lift-кривих

Вони дозволяють оцінити узагальнювальну здатність, стійкість до зашумлення, статистичну значущість моделей та прийняти рішення щодо їх впровадження.

Коротко розглянемо підход крос-валідації. Сутність підходу полягає у багаторазовому навчанні моделі з використанням різних частин даних та оцінюванні її прогнозної здатності на контрольних вибірках. Існують різні схеми реалізації крос-валідації:

- K-fold cross-validation - вихідну вибірку розбивають на K приблизно рівних частин-фолдів, потім K ітерацій: один фолд використовується як тестовий, решта K-1 фолдів - для тренування моделі.
- Leave-p-out cross-validation - залишають поза навчальною вибіркою p спостережень, використовуючи їх для незалежного тестування.
- Leave-one-out cross-validation - окремий випадок попередньої стратегії за p=1, коли по черзі кожен об'єкт виступає як тестова вибірка.

Зокрема, для методу крос-валідації можна формалізувати процес розбиття вибірки та ітераційного тестування/навчання у вигляді таких формул:
Позначимо вихідну вибірку як $X = x_1, x_2, ..., x_N$, а кількість фолдів як K.
Тоді на ітерації k: $X_{\text{train}} = X \backslash X_k$, $X_{\text{test}} = X_k$

де

X_{train} - навчальна вибірка;

X_{test} - тестова вибірка (к-й фолд).

Як бачимо, на кожній ітерації відбувається чергування навчальної та тестової частин вихідної вибірки.

Виходячи з вище наведеного - основні етапи крос-валідації для оцінки моделей машинного навчання на даних користувачів Bluefit:

- Вихідна вибірка даних позначається як: $X = x1, x2, ..., xN$, де N - загальна кількість спостережень
- Вибірка X розбивається на K приблизно рівних частин-фолдів
- На кожній ітерації k відбувається розподіл вибірки на навчальну та тестову: $Xtrain = X\ Xk$, $Xtest = Xk$. Тобто тестування на k-му фолді, навчання на решті $K - 1$ фолдах.
- Оцінка якості алгоритму усереднюється по всіх K ітераціях.

Поряд з крос-валідацією, важливим підходом перевірки якості моделей є незалежна валідація на контрольній множині тестових даних. Сутність полягає у розподілі вихідної вибірки користувачів Bluefit на 2 частини: навчальну та тестову. Навчальні дані (Xtrain) використовуються для побудови моделей машинного навчання, а тестові (Xtest) - для неупередженого оцінювання якості цих побудованих моделей на нових, раніше невідомих даних.

Отже, ключовим моментом даного підходу є саме спосіб розподілу вихідної

вибірки даних користувачів на дві частини:

- Навчальна вибірка (Xtrain) використовується для побудови/навчання моделей машинного навчання зазвичай 60-80% обсягу вихідних даних повинна бути репрезентативною та якісною
- Тестова вибірка (Xtest) використовується виключно для незалежної оцінки якості побудованих моделей 20-40% від загального обсягу даних повинна точно віддзеркалювати реальний розподіл даних

Ще одним важливим моментом даного підходу є саме вибір метрик для оцінювання якості роботи алгоритмів на контрольних даних. З огляду на постановку конкретних задач аналізу даних користувачів медичного додатку Bluefit, найбільш цікавими та корисними будуть наступні метрики:

- Для задачі класифікації користувачів (наприклад, за рівнем активності):
    1) Accuracy - загальна точність роботи алгоритму
    2) Precision - точність виділення цільового класу (наприклад, активних)
    3) Recall - повнота охоплення саме цільового класу користувачів
- Для задачі кластеризації користувачів за схожими характеристиками:
    1) Silhouette score - наскільки щільні та добре розділені кластери
    2) V-measure - збалансована міра homogeneity та completeness
- Для прогнозування кількісних метрик взаємодії з додатком:
    1) MAE - середня абсолютна похибка, зручна для інтерпретації
    2) RMSE - чутлива до викидів метрика якості прогнозування

Поряд з метриками на зразок accuracy чи F1-score, важливим інструментом оцінки якості бінарної класифікації є побудова та аналіз кривої операційних характеристик моделі (ROC-кривої). Вона дозволяє візуалізувати баланс між повнотою та точністю класифікатора. Формально чутливість та специфічність моделі обчислюються за формулами:

$$TPR = \frac{TP}{TP + FN} \tag{2.16}$$

де

TPR - чутливість, true positive rate (частка істинно позитивних результатів серед усіх позитивних випадків);

TP - true positives, кількість істинно позитивних прикладів (вірно класифікованих як позитивні);

FN - false negatives, кількість хибнонегативних прикладів (неправильно класифікованих як негативні).

$$FPR = \frac{FP}{FP + TN} \tag{2.17}$$

де

FPR - false positive rate (частка хибнопозитивних результатів серед усіх негативних випадків);

FP - false positives, кількість хибнопозитивних прикладів (неправильно класифікованих як позитивні);

TN - true negatives, кількість істинно негативних прикладів (вірно класифікованих як негативні).

Різні комбінації цих двох метрик при зміні порогу класифікації і утворюють криву ROC.

Однією з ключових задач проєкту є класифікація користувачів за рівнем їх активності взаємодії з додатком - на "активних" та "пасивних".

Побудовані для цього моделі машинного навчання (логістична регресія, нейронні мережі, дерева рішень тощо) можуть мати різні показники якості залежно від порогу прийняття рішень щодо класифікації. Саме за допомогою

аналізу ROC-кривої та обчислення AUC можна візуально порівняти якість цих моделей та обрати оптимальний баланс між повнотою та точністю класифікації користувачів Bluefit.

Побудова та аналіз кривої операційних характеристик дозволяє візуально оцінити збалансованість повноти та точності бінарного класифікатора. Для кількісної оцінки цих параметрів якості ML-моделей використовуються такі статистичні метрики як F-міра та критерій Мак-Немара.

F-міра (F1-score) - це узагальнена метрика, що характеризує якість бінарного класифікатора на основі зваженого усереднення його точності (precision) та повноти (recall). Чим ближче значення F-міри до 1, тим краще збалансовані ці дві характеристики якості моделі. Формула для обчислення F-міри:

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{2.18}$$

де:

TP - true positives, вірно класифіковані позитивні приклади;

FP - false positives, хибнопозитивні приклади;

FN - false negatives, хибнонегативні приклади.

Критерій Мак-Немара дозволяє формалізувати статистичну значущість отриманого покращення якості одного класифікатора над іншим на основі розподілу помилок на тестовій вибірці. Формула для обчислення критерію:

$$\chi^2 = \frac{(|E_1 - E_2| - 1)^2}{E_1 + E_2} \tag{2.19}$$

де $E1$, $E2$ - кількість помилок відповідно першої та другої моделі на тестових даних. Значення χ2 порівнюється з критичним значенням χ2-розподілу при заданому рівні значущості (наприклад, 0.05) та ступенів

вільності df=1. Якщо обчислене χ2 більше за критичне - гіпотеза про рівність моделей відкидається, тобто покращення значуще.

Однією з ключових ML задач у цьому проєкті є класифікація користувачів за рівнем активності взаємодії з додатком Bluefit. Після побудови відповідних моделей класифікації (наприклад, логістичної регресії), для оцінки їх якості можуть використовуватися саме F-міра та критерій Мак-Немара. F-міра дасть узагальнену картину щодо балансу між precision та recall моделі на тестових даних. А критерій Мак-Немара дозволить формалізувати статистичну значущість вдосконалення якості моделі в порівнянні з базовими алгоритмами машинного навчання. Таким чином можна комплексно оцінити ефективність моделі саме для конкретної предметної задачі аналізу даних Bluefit.

Розглянуті вище статистичні метрики, такі як F-міра чи критерій Мак-Немара, дозволяють чисельно оцінити якість бінарних класифікаторів. Додаткову інформацію про ефективність моделей машинного навчання може надати аналіз кривих згортання (CGC) та ліфту, що демонструють накопичену ефективність вздовж ранжованого списку об’єктів. Отже, крива згортання показує залежність частки позитивних прикладів, виявлених моделлю, від частки всієї вибірки. Ідеальна CGC зростає різко, а потім виположується, виявляючи всі позитивні приклади в початку ранжованого списку. Lift-крива показує в кілька разів вищу ефективність (приріст, lift) моделі в порівнянні з випадковим вгадуванням. Ідеальна lift-крива має експоненціально спадний вигляд.

В свою чергу кумулятивний приріст (cumulative gain) на кроці k обчислюється за формулою:

$$CG(k) = \frac{\sum_{i=1}^{k} P(i)}{\sum_{i=1}^{n} P(i)} \qquad (2.20)$$

де:

$CG(k)$ - кумулятивний приріст на кроці k;

$P(i)$ - релевантність (вага) об'єкта з номером і в відсортованому за прогнозом списку;

n - загальний розмір списку;

сума в чисельнику - сума ваг перших k об'єктів;

сума в знаменнику - сума ваг по всіх об'єктах.

Формула ліфту на кроці k показує відношення ефективності моделі порівняно з випадковим вгадуванням:

$$Lift(k) = \frac{CG(k)}{(k/n)} \tag{2.21}$$

де n - загальний розмір списку об'єктів, що аналізується.

Ааналіз кривих згортання та ліфту можна використати для оцінки якості моделей машинного навчання на даних з медичного додатку Bluefit.

Наприклад нам потрібно побудувати модель для відсортування списку користувачів Bluefit за ймовірністю припинення ними використання додатку. Модель впорядковує список у порядку спадання ризику відтоку. Далі будуємо криві:

- CGC покаже накопичену частку користувачів, що припинили використання серед перших k зі списку
- Lift продемонструє в кілька разів вищу ефективність такої моделі порівняно з випадковим вгадуванням

Аналіз цих кривих дозволить оцінити якість ML-алгоритму для конкретної задачі прогнозування відтоку з Bluefit.

Висновок - в данному розділі було розглянуто основні підходи до тестування та валідації побудованих моделей машинного навчання: крос-валідація, незалежна перевірка на контрольних даних, побудова та аналіз ROC-кривої і AUC, обчислення статистичних метрик якості класифікації, таких як F-міра та критерій Мак-Немара. Також запропоновано використання кривих згортання та ліфту для додаткової оцінки ефективності побудованих моделей в контексті конкретних задач дослідження. Застосування цих підходів дозволить об'єктивно оцінити якість та узагальнювальну здатність побудованих в роботі моделей машинного навчання перед їх практичним застосуванням для аналізу даних користувачів медичного додатку.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Формування вхідних даних

Вхідні дані для дослідження містяться в SQL базі даних комерційного медичного додатку Bluefit. Ці дані складаються з двох основних джерел:

- Статистика використання застосунку користувачами
- Анкетні дані користувачів, отримані під час реєстрації у застосунку

Таблиця 3.1 містить інформацію про кожний сеанс тренування, проведений користувачем в додатку Bluefit.

Таблиця 3.1 - Статистика роботи з застосунком

Поле	**Тип**	**Опис**
mac	varchar(200)	MAC-адреса пристрою
user_id	int(11)	Унікальний ідентифікатор користувача
phone_id	varchar(45)	Ідентифікатор телефону
phone_model	varchar(45)	Модель телефону
os	varchar(45)	Операційна система телефону
version	varchar(45)	Версія застосунку
date	datetime	Дата та час сесії
bodypart	varchar(45)	Частина тіла, на яку спрямована тренування
program	int(11)	Програма тренування
duration	int(11)	Тривалість сесії
intensity	int(11)	Інтенсивність тренування

Таблиця 3.2 містить реєстраційні дані користувачів додатку Bluefit.

Таблиця 3.2 - Анкетні дані користувачів

Поле	**Тип**	**Опис**
user_id	int(11)	Унікальний ідентифікатор користувача
email	varchar(80)	Електронна пошта користувача
firstname	varchar(100)	Ім'я користувача
lastname	varchar(100)	Прізвище користувача
language	varchar(3)	Мова користувача
registered	datetime	Дата реєстрації
activated	datetime	Дата активації

Таблиця 3.3 містить додаткову інформацію з анкет користувачів:

Таблиця 3.3 - Медичний профіль

Поле	**Тип**	**Опис**
user_id	int(11)	Унікальний ідентифікатор користувача
sex	tinyint(4)	Стать
birth	datetime	Дата народження
weight	smallint(6)	Вага
height	smallint(6)	Зріст
...	...	Інші медичні дані

Ця інформація стане основою для аналізу поведінки користувачів і взаємодії з медичним застосунком Bluefit, що є темою вашої магістерської роботи. Ці дані дозволять вам розробити та впровадити моделі машинного

навчання для прогнозування взаємодії користувачів з застосунком на основі їх поведінкових та медичних характеристик.

Перед початком виконання моделювання вхідні данні потрібно підготувати. В рамках цього проекту підготовка даних буде складатися з наступних етапів:

- Завантаження даних з SQL таблиць
- Оцінка якості даних
- Очищення даних
- Підготовка даних

Завантаження даних з SQL таблиць. Для ефективної обробки великих обсягів даних, що накопичилися протягом кількох років (понад 16 мільйонів записів), ми використовуємо методику часткового завантаження та оптимізації даних. Цей процес реалізовано через функції, описані в Додатку А:

- load_data_from_db - завантажує дані з бази даних MySQL частинами, зберігаючи їх у файл HDF5.
- Оптимізація даних - перетворення категоріальних даних для зменшення використання пам'яті та збереження у форматах HDF5 та PKL.

Ці методи забезпечують ефективне зберігання та швидку обробку великих даних, підвищуючи продуктивність системи.

Оцінка якості даних. Другий крок - це оцінка якості ваших даних. Це включає в себе виявлення пропущених значень, неправильних типів даних, а також пошук аномалій та викидів. Проаналізуємо кожен набір даних окремо.

```python
# Етап 1: Оцінка Якості Даних
# Перевірка наявності пропущених значень
print(session_df.isnull().sum())

# Опис статистики даних для виявлення аномалій
print(session_df.describe())


```

```
user_id          14649
phone_model          0
os                   0
version        4995815
date                 0
bodypart           218
program              0
duration             0
intensity            0
dtype: int64
            user_id       program      duration     intensity
count  1.651522e+07  1.652987e+07  1.652987e+07  1.652987e+07
mean   1.250793e+05  2.530676e+02  8.429025e+06  2.525183e+01
std    1.090978e+05  2.643042e+02  1.166316e+08  3.441812e+03
min    7.800000e+01  0.000000e+00 -1.262580e+09  0.000000e+00
25%    3.790000e+02  1.030000e+02  4.750000e+02  1.300000e+01
50%    1.147800e+05  2.340000e+02  1.201000e+03  2.000000e+01
75%    2.078890e+05  3.740000e+02  1.800000e+03  3.300000e+01
max    3.492950e+05  3.002000e+03  1.698567e+09  9.893745e+06
```

Рисунок 3.1 – Аналіз даних сесій (session_df)

Згідно даних на рисунку 3.1 ми робимо такі висновки:

- Пропущені значення:
    1) user_id: 14,649 пропущених значень.
    2) version: 4,995,815 пропущених значень.
    3) bodypart: 218 пропущених значень.

- Аномалії в статистиці: Велике стандартне відхилення та негативні мінімальні значення у duration та intensity можуть вказувати на наявність викидів або помилок у даних.

Построємо ящикові діаграми дивись рисунок 3.2 та 3.3 для графічного відображення викидів даних для duration та intensity.

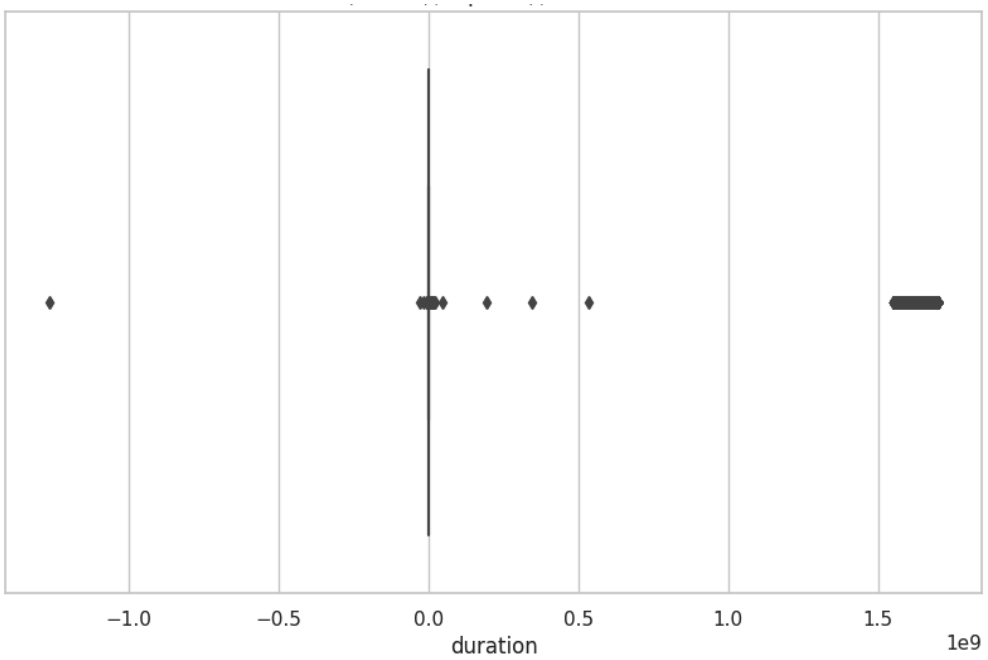

Рисунок 3.2 – Ящикова діагарама викидів поля duration

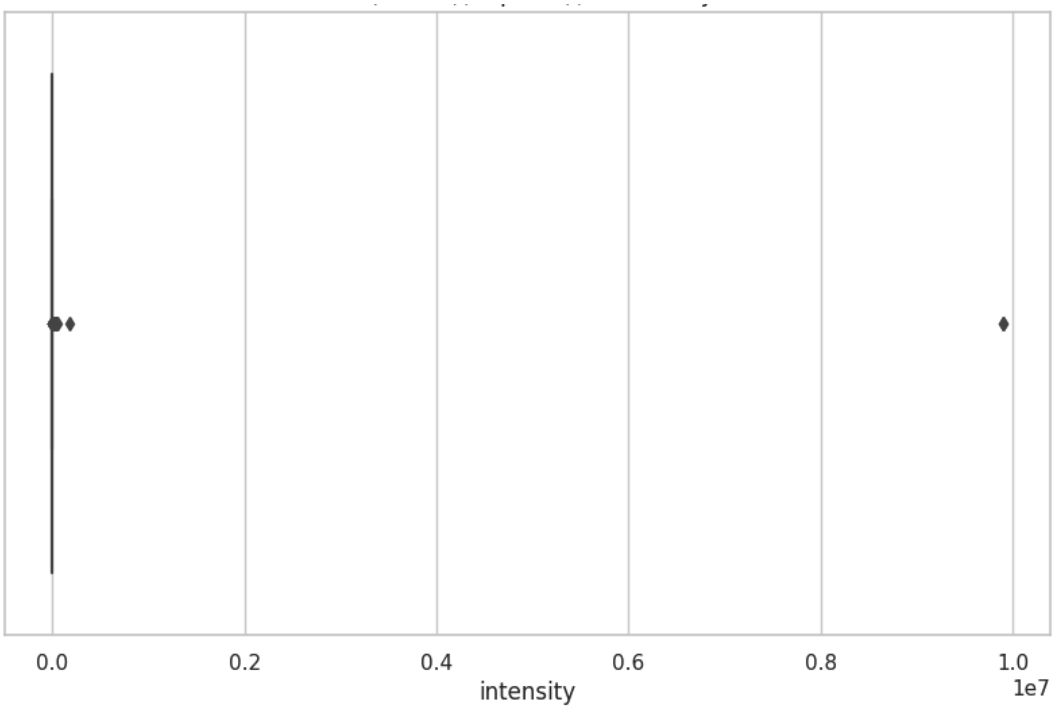

Рисунок 3.3 – Ящикова діагарама викидів поля intensity

```python
# Перевірка наявності пропущених значень
print(user_df.isnull().sum())

# Опис статистики даних для виявлення аномалій
print(user_df.describe())
```

```
user_id            0
firstname          1
lastname           0
language          21
registered         0
activated     164977
dtype: int64
             user_id
count  320710.000000
mean   182368.183056
std    100688.860658
min         1.000000
25%     97180.250000
50%    188478.500000
75%    268890.750000
max    349298.000000
```

Рисунок 3.4 – Аналіз даних користувачів(user_df)

Згідно даних на рисунку 3.4 ми робимо такі висновки:

- Пропущені значення:
  1) firstname: 1 пропущене значення.
  2) language: 21 пропущене значення.
  3) activated: 164,977 пропущених значень.
- Аномалії в статистиці: Відсутність очевидних аномалій у числових статистиках.

```
# Перевірка наявності пропущених значень
print(patient_profile_df.isnull().sum())

# Опис статистики даних для виявлення аномалій
print(patient_profile_df.describe())
```

```
user_id               0
sex               14793
birth              4746
weight             3623
height             3523
physical_level     6274
practice_level    12904
frequency_level   12537
problem_arises    14927
dtype: int64
             user_id           sex        weight        height  physical_level  \
count   23695.000000  8902.000000  20072.000000  20172.000000    17421.000000
mean   219862.834986     0.210627     75.824283    163.689718        1.654440
std     86450.722256     0.407777     19.624604     40.790060        0.879924
min        81.000000     0.000000      0.000000      0.000000        0.000000
25%    180759.500000     0.000000     65.000000    165.000000        1.000000
50%    231062.000000     0.000000     75.000000    172.000000        2.000000
75%    287441.000000     0.000000     85.000000    180.000000        2.000000
max    349282.000000     1.000000    250.000000    250.000000        3.000000

       practice_level  frequency_level  problem_arises
count    10791.000000     11158.000000     8768.000000
mean         1.466500         1.969618        0.673586
std          1.544929         2.215444        0.871707
min          0.000000         0.000000        0.000000
25%          0.000000         0.000000        0.000000
50%          1.000000         1.000000        0.000000
75%          3.000000         4.000000        2.000000
max          4.000000         8.000000        2.000000
```

Рисунок 3.5 – Аналіз даних профілів пацієнтів (patient_profile_df)

Згідно даних на рисунку 3.5 ми робимо такі висновки:

- Пропущені значення: sex, birth, weight, height, physical_level, practice_level, frequency_level, problem_arises мають значну кількість пропущених значень.
- Аномалії в статистиці:
    1) Нульові значення у weight, height можуть бути помилковими або недостовірними.
    2) Велике стандартне відхилення у деяких полях також може вказувати на наявність викидів.

Очищення даних. На основі отриманих результатів ми визначили, які стовпці потребують очищення. Це включає в себе заповнення або видалення

пропущених значень, корекцію типів даних та обробку викидів. Опишемо коротко що практично було зроблено.

З попереднього аналізу даних сесій витікає необхідність декількох кроків для підготовки даних до подальшого аналізу та моделювання. Для забезпечення цілісності та точності даних, ми виконали видалення рядків, де відсутній user_id, оскільки це ключовий ідентифікатор для подальшого аналізу. Для полів version та bodypart, де є пропущені значення, ми застосуємо метод імпутації, використовуючи найчастіше значення або моду для цих полів. Це дозволить зменшити вплив пропущених даних на аналіз. Також ми виправили аномалії у полях duration та intensity, замінивши нереалістичні та викидні значення, що допоможе підвищити точність моделей, побудованих на основі цих даних. Цей процес реалізовано в коді Python, смотри Додатк А. Цей код спершу видаляє рядки без user_id, потім заповнює пропущені значення у полях version та bodypart, використовуючи найчастіші значення. Далі, він коректує аномалії у полях duration та intensity, замінюючи нереалістичні значення на медіанні.

Для user_df, на основі попереднього аналізу, ми знаємо, що немає очевидних аномалій у числових статистиках, але є деяка кількість пропущених значень у стовпцях firstname, language та activated. Оскільки пропущених значень не багато, ми можемо заповнити їх найпопулярнішими значеннями (модою). Пропущені значення в стовпці activated можуть бути інтерпретовані як неактивовані користувачі, тому ми можемо або залишити ці значення як є, або замінити їх на певне позначення, наприклад, дату далеку в минулому або спеціальне значення. Код для очищиння даних user_df дивись в Додатку А.

Для patient_profile_df, враховуючи результати попереднього аналізу даних patient_profile_df, який виявив значну кількість пропущених значень та потенційні аномалії, було розроблено комплексний підхід для їх обробки та нормалізації. Ключова мета цих кроків - покращити якість та цілісність

даних, що є критично важливим для ефективного аналізу та висновків. Спочатку, для заповнення пропущених значень у полі sex, використана функція guess_gender, яка визначає стать за іменем, отриманим з набору даних користувачів (user_df). Для тих випадків, де стать не може бути визначена, застосовується статистичний метод імпутації на основі існуючого розподілу статей у наборі даних. Потім, аномалії у полях birth, weight та height були видалені з використанням методу remove_outliers. Це допомогло усунути потенційно недостовірні та помилкові дані, які могли б спотворити аналіз. Для поля birth, пропущені значення були заповнені використовуючи медіанні значення, обраховані окремо для кожної статі. Це дозволило врахувати можливі відмінності в датах народження між статями. Інші поля, такі як weight, height, physical_level, practice_level, frequency_level, problem_arises, були оброблені з використанням медіанних значень, розрахованих за статтю. Такий підхід дозволив зберегти консистенцію та релевантність даних, виходячи з гендерних характеристик. На завершення, були створені профілі для тих пацієнтів, які були присутні у user_df, але відсутні у patient_profile_df. Використовуючи аналогічні методи для визначення статі та імпутації даних, було забезпечено, що кожен користувач має відповідний профіль, що підвищує повноту та репрезентативність набору даних. Ці кроки, детально описані у Додатку А, значно підвищують якість набору даних patient_profile_df, забезпечуючи надійну основу для подальших аналітичних досліджень.

Підготовка даних. Після очищення, ми перейдемо до підготовки даних для моделювання. Вона передбачає низку важливих кроків, які необхідно виконати перед початком моделювання. Для набору даних session_df буде проведена додаткова обробка часових міток, щоб виявити потенційні шаблони у поведінці користувачів. Зокрема, це означає перетворення дат сесій на додаткові параметри, такі як час доби, день тижня, що може виявити важливі інсайти щодо поведінки користувачів.

У контексті user_df зосередимо увагу на перетворенні дат реєстрації та активації користувачів. Це дозволить нам отримати уявлення про тривалість взаємодії користувачів з додатком та їхню активність, а також визначити, як це впливає на їхні вподобання та вибір програм. Щодо patient_profile_df, важливим аспектом є аналіз віку, що базується на даті народження, та інших фізичних характеристик, таких як вага і зріст. Це допоможе у виявленні зв'язків між фізичним станом користувачів та їхніми перевагами або потребами у тренуваннях. Завершивши підготовку цих даних, можна буде приступити до широкого спектру завдань аналізу, включаючи класифікацію користувачів, кластеризацію, а також прогнозування. Особливу увагу буде приділено розробці персоналізованої системи рекомендацій, яка дозволить користувачам вибирати найбільш підходящі програми електростимуляції. Ключовим аспектом у цьому процесі є валідація та оцінка ефективності різних моделей та алгоритмів, що забезпечить оптимальний вибір методів для вирішення поставлених задач.

3.2 Опис програмної реалізації

Даний розділ містить детальний опис програмної реалізації системи аналізу даних про пацієнтів та їх тренувальні сесії для виявлення закономірностей.

Вибір мови програмування та фреймворків машинного навчання. Як мову програмування було обрано Python через його популярність, простоту вивчення та величезні можливості для аналізу даних та машинного навчання.

Для реалізації проекту використовувались наступні ключові бібліотеки:

- Pandas - провідна бібліотека для аналізу та обробки даних. Вона надає зручні структури даних (Series, DataFrame) та інструменти для читання, обробки, очищення та аналізу даних.
- Matplotlib та Seaborn - бібліотеки для створення інформативних візуалізацій та графіків. Використовувались для експлораторного аналізу даних.
- Scikit-learn - найпопулярніший фреймворк машинного навчання для Python. Містить реалізації багатьох алгоритмів, моделей та методів обробки даних. Використовувався для моделювання та прогнозування.

Архітектура та загальна схема програмної системи.

Програмне забезпечення складається з декількох модулів:

- Завантаження та попередня обробка даних - зчитування даних з БД, об'єднання та синхронізація таблиць, видалення дублів, неточностей та неповних даних. Реалізовано у скриптах *load_data.py*, *prep_data_*.py* за допомогою методів Pandas.
- Аналіз та візуалізація даних - побудова різноманітних графіків та діаграм за допомогою Seaborn та Matplotlib для первинного

дослідження даних та виявлення закономірностей. Кожен скрипт *analysis_*.py* містить аналіз певного аспекту даних.

- Підготовка даних для моделювання - очищення викидів, нормалізація, розбиття на навчальні та тестові набори за допомогою Pandas, Numpy та методів Scikit-Learn у *model_*.py*.
- Побудова та навчання моделей машинного навчання - тренування обраних алгоритмів (KMeans, GradientBoostingRegressor) на підготовлених даних з використанням Scikit-Learn (*model_*.py*).
- Оцінка якості та тестування моделей - розрахунок різних метрик (RMSE, MSE, MAE тощо) для оцінки точності отриманих моделей за допомогою методів Scikit-Learn.
- Застосування моделей та формування прогнозів - використання побудованих і навчених моделей для прогнозування на нових даних (метод *predict()*) та формування аналітичних висновків.

Опис основних класів та функцій.

- Завантаження та підготовка даних:
    1) load_data_from_db() - завантаження даних з MySQL БД по частинах з використанням Pandas та відображенням прогресу
    2) remove_outliers() - видалення викидів за допомогою методу IQR
    3) guess_gender() - визначення статі користувача на основі імені
- Аналіз та візуалізація:
    1) Методи groupby(), value_counts(), corr() - угруповання, підрахунок, кореляції
    2) seaborn.heatmap() - побудова кореляційної матриці
    3) seaborn.boxplot(), seaborn.scatterplot() - графіки розкиду та розсіювання

- Побудова та навчання моделей:
    1) KMeans - кластеризація користувачів за активністю
    2) GradientBoostingRegressor - градієнтне підсилення для регресії
    3) RandomizedSearchCV - пошук оптимальних гіперпараметрів
- Оцінка якості та тестування:
    1) train_test_split() - розбиття даних на навчальні та тестові
    2) mean_squared_error() - середньоквадратична похибка (MSE)
    3) mean_absolute_error() - середня абсолютна похибка (MAE)

Обґрунтування вибору технологій та підходів реалізації.

Python та його бібліотеки були вибрані через їх широке використання у сфері обробки даних та машинного навчання. Це дозволяє легко інтегрувати різноманітні інструменти для ефективної обробки та аналізу даних, а також для побудови та тестування моделей. Використання форматів HDF5 та PKL допомагає оптимізувати роботу з великими обсягами даних. Запропонована програмна архітектура дозволяє ефективно виконувати аналіз даних та прогнозне моделювання. Вибір Python, Pandas, Scikit-Learn як основних технологій надає гнучкості та продуктивності розробці. Подальші напрямки - розширення моделей машинного навчання та їх практичне використання.

3.3 Аналіз результатів

На початку дослідження даних і перед безпосереднім моделюванням зробимо первинний аналіз даних. Це допомогає зрозуміти внутрішні закономірності та дає подальші ідеї для моделювання.

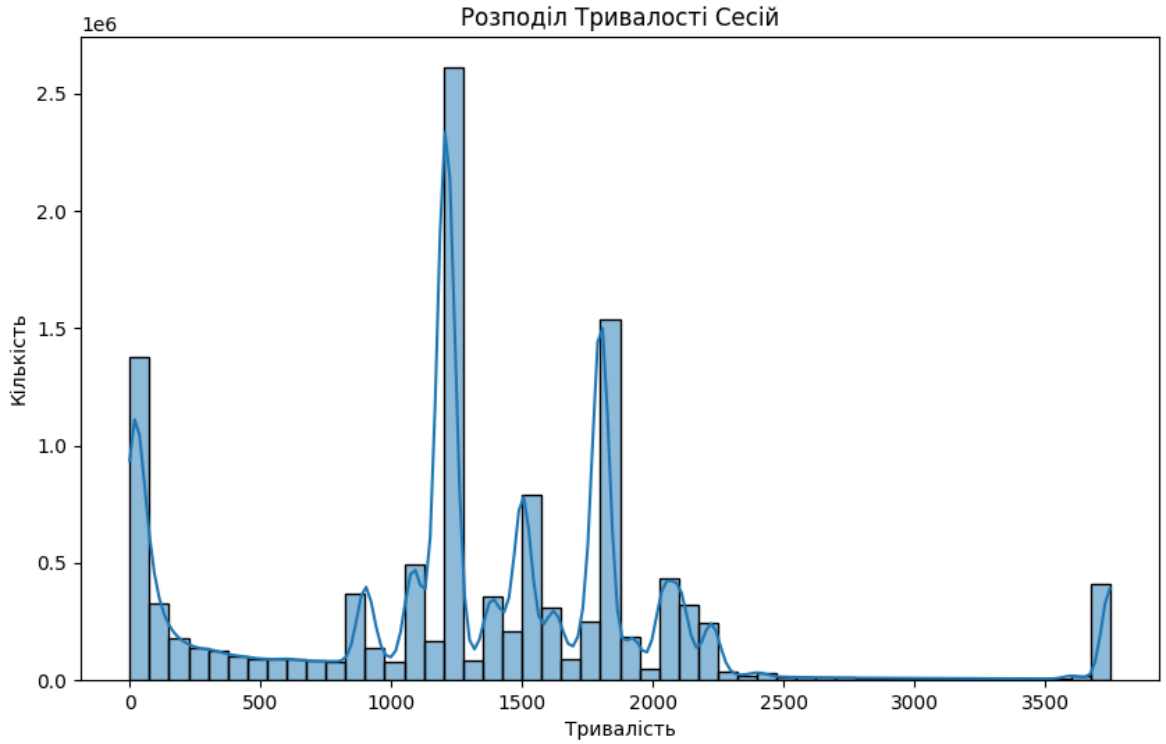

Рисунок 3.6 – Розподіл тривалості сесій

На основі рисунку 3.6 розподілу тривалості сесій та отриманої статистики можна зробити декілька спостережень:

- Більшість сесій мають тривалість у діапазоні від 500 до 2000 секунд.
- Існує кілька видільних піків, зокрема близько 1000 та 1800, що може вказувати на типову тривалість сесій.

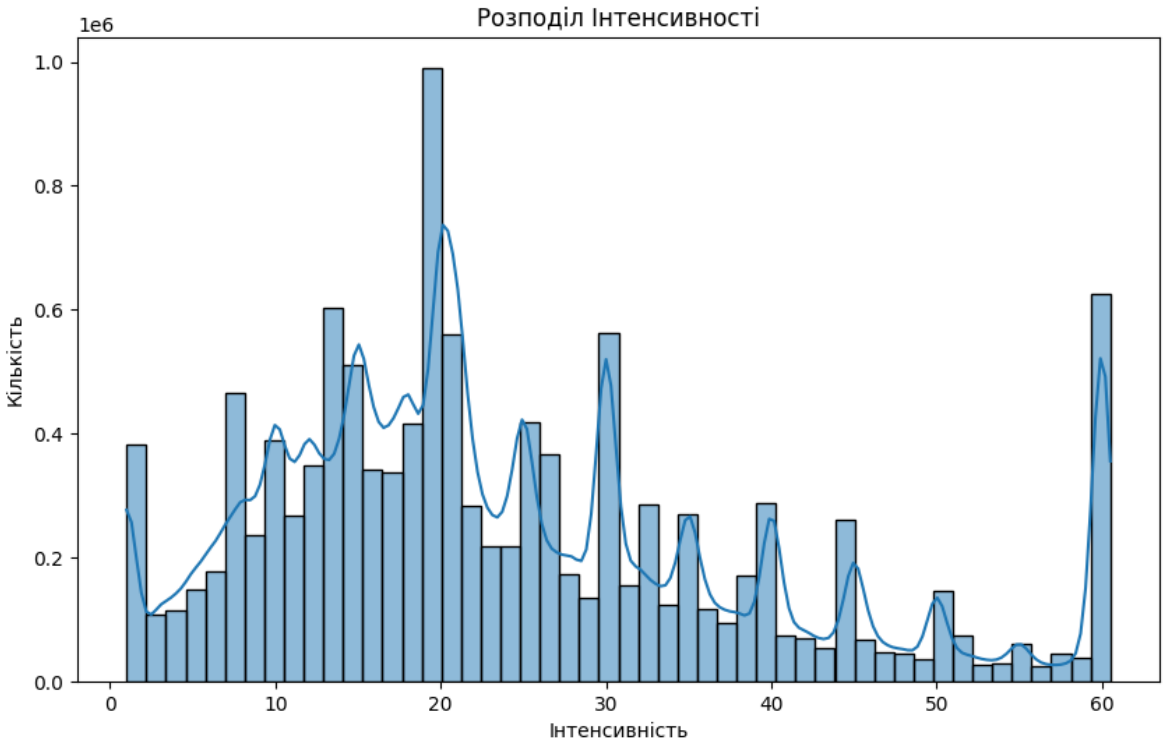

Рисунок 3.7 – Розподіл інтенсивності сесій

Аналізуючи рисунок 3.7 розподіл інтенсивності сесій та отриманої статистики можна зробити наступні виводи:

- Велика кількість сесій має інтенсивність у діапазоні від 1 до 25, з декількома піками.
- Існує виражений пік близько значення 60, що може вказувати на популярні налаштування або максимально можливе значення інтенсивності.
- Розподіл не є нормальним і вказує на наявність декількох "популярних" значень інтенсивності, які можуть бути стандартними налаштуваннями в додатку.

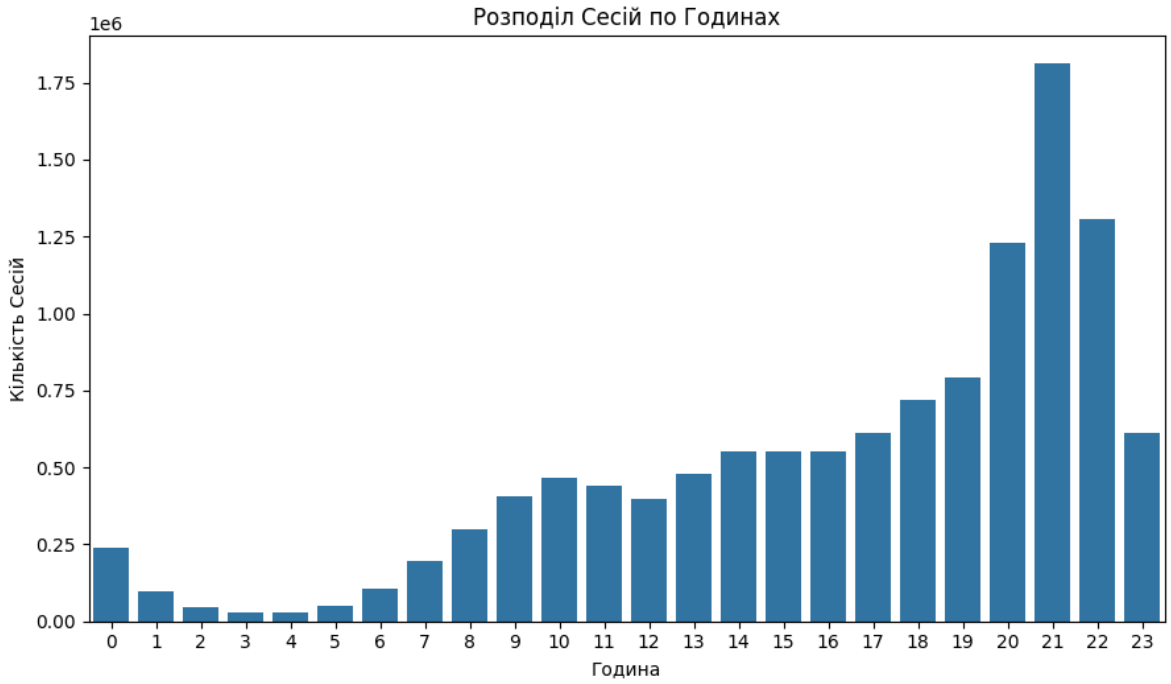

Рисунок 3.8 – Розподіл сесій по годинах

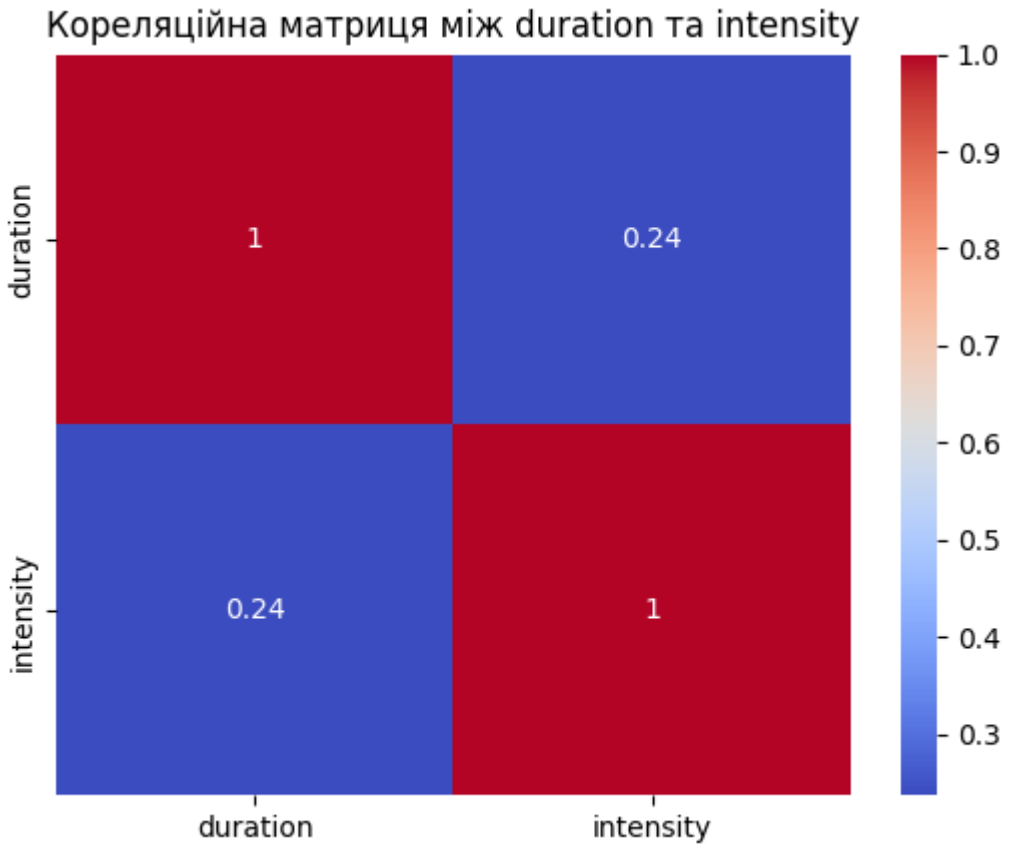

Рисунок 3.9 – Кореляційна матриця між duration та intensity

Графік розподілу сесій по годинах рисунок 3.8 та кореляційна матриця рисунок 3.9 надають цінну інформацію для аналізу поведінки користувачів:

- З графіка видно, що активність користувачів збільшується вранці з 6 до 9 години, досягає плато протягом дня та пікової активності з 18 до 22 години. Це може вказувати на те, що користувачі воліють проводити сесії до або після робочого дня.
- Позитивна кореляція між duration та intensity (p = 0.232845) свідчить про те, що з підвищенням тривалості сесій, зазвичай, зростає і інтенсивність. Це може бути ознакою того, що більш тривалі сесії часто є більш інтенсивними, або що користувачі, які займаються довше, можуть вибирати програми з вищою інтенсивністю.

Для подальшого аналізу даних і виявлення сезонних і тижневих трендів у поведінці користувачів розглянемо, як активність користувачів змінюється залежно від дня тижня та місяця.

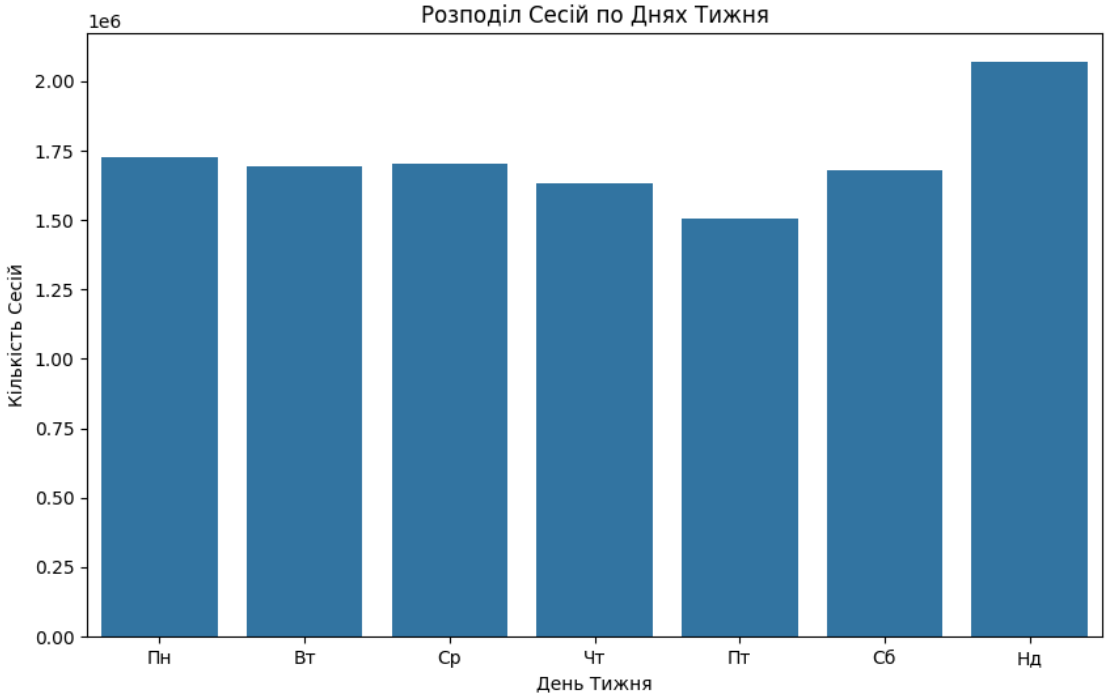

Рисунок 3.10 – Розподіл сесій по днях тижня

З рисунку 3.10 бачемо що найбільша активність користувачів спостерігається у вихідні дні, особливо у неділю, що може вказувати на більшу кількість

вільного часу у користувачів для занять.

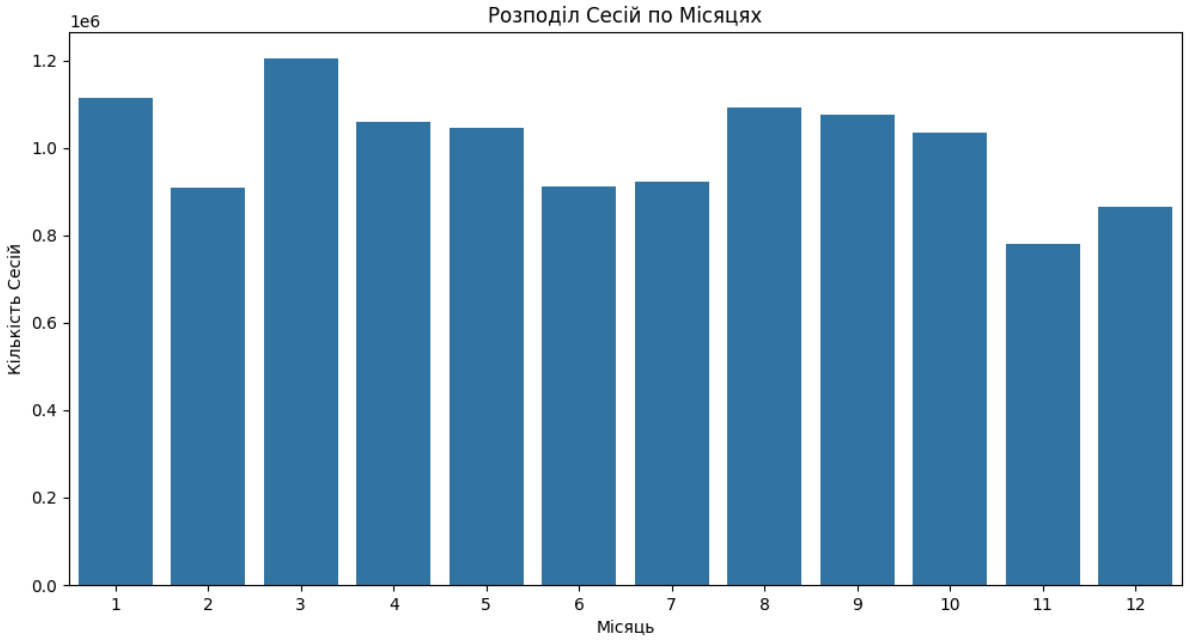

Рисунок 3.11 – Розподіл сесій по місяцях

З рисунку 3.11 робимо вивід що активність користувачів здається відносно рівномірною протягом року, з невеликим зниженням у літні місяці та в грудні.

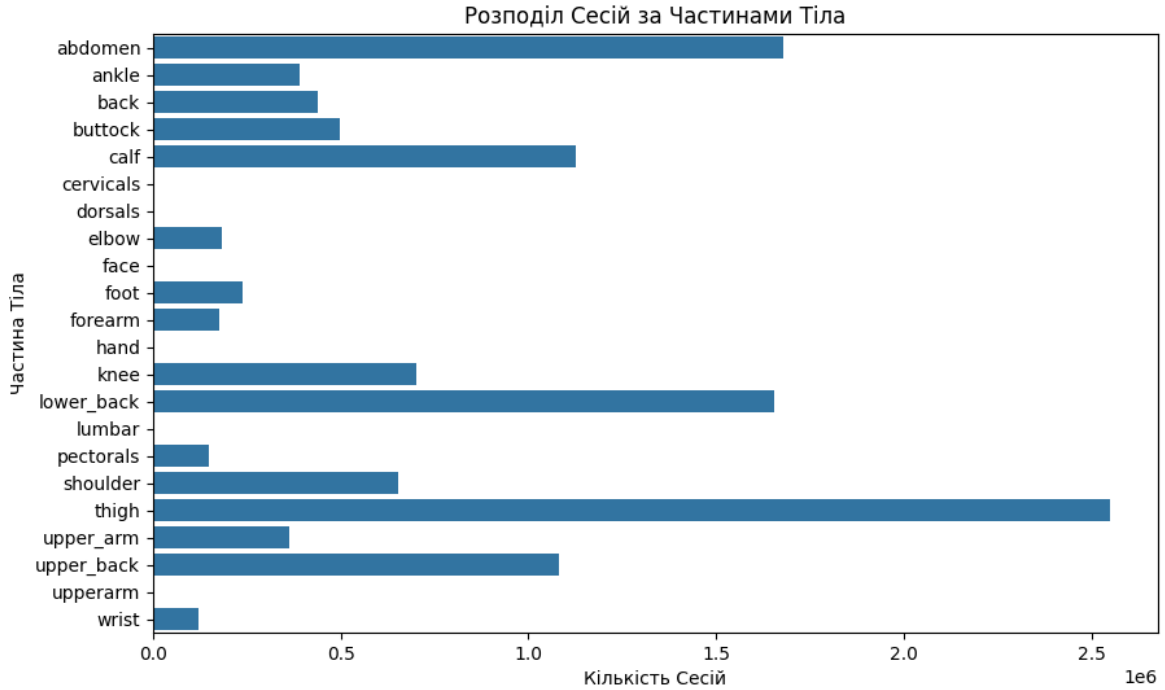

Рисунок 3.12 – Розподіл сесій за частинами тіла

На наданому рисунку 3.12 розподілу сесій за частинами тіла видно, що деякі частини тіла використовуються значно частіше, ніж інші. Наприклад, "abdomen" (живіт) має найбільшу кількість сесій, що може вказувати на популярність тренувань для цієї області серед користувачів. Інші частини тіла, такі як "lower_back" (нижня частина спини), "thigh" (стегно), та "shoulder" (плече) також мають високу частоту сесій.

Для більш глибокого розуміння внутрішніх зв’язків в даних сессій - зробимо первинну кластерізацію даних за тривалістю та інтенсивністю сесій.

Рисунок 3.13 – Кластеризація за тривалістю і інтенсивністю сесій

На рисунку 3.13 видно п'ять різних груп (кластерів), кожна з яких відображає унікальні патерни в поведінці користувачів:

- Кластер 0 (Синій): Включає сесії з низькою до помірною інтенсивністю та тривалістю, що варіюється від коротких до середніх.

- Кластер 1 (Жовтий): Складається з коротких сесій з високою інтенсивністю.
- Кластер 2 (Зелений): Представлений сесіями середньої тривалості з середньою інтенсивністю.
- Кластер 3 (Фіолетовий): Містить найдовші сесії з високою інтенсивністю.
- Кластер 4 (Світло-зелений): Охоплює сесії з найнижчою інтенсивністю, незалежно від тривалості.

Проаналізуємо як зв’язані кластери за частинами тіла для сессій.

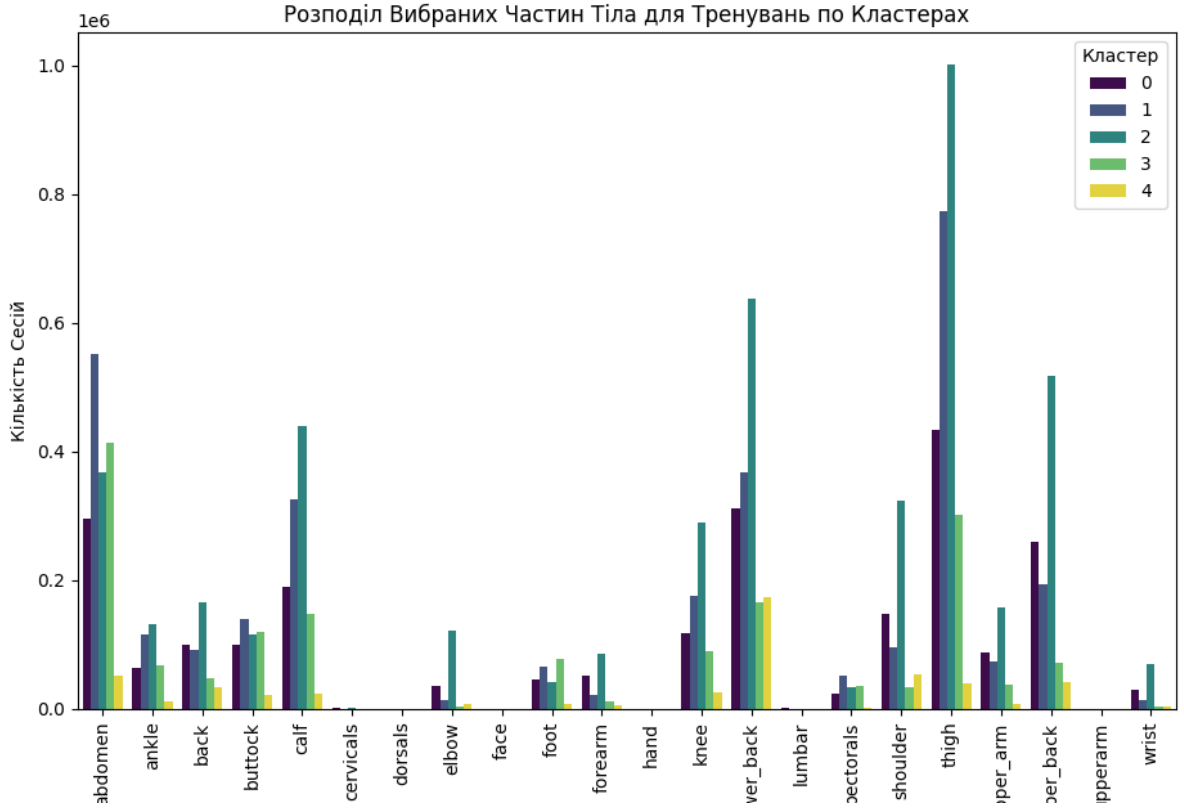

Рисунок 3.14 – Розподіл вибраних частин тіла для тренувань по кластерах

Рисунок 3.14 показує значні відмінності в кількості сесій за вибраними частинами тіла між кластерами. Наприклад, тренування, що включають спину і плечі, мають найбільшу кількість в кластері 2. Це свідчить про популярність вправ на ці частини тіла у даному кластері користувачів. Також

помітно, що тренування з участю литок і біцепсів мають відносно високу частоту у кластері 3. З іншого боку, деякі частини тіла, такі як обличчя та лікоть, майже не використовуються у тренуваннях незалежно від кластера, що може вказувати на їх меншу популярність або специфічність вправ, що включають ці частини тіла.

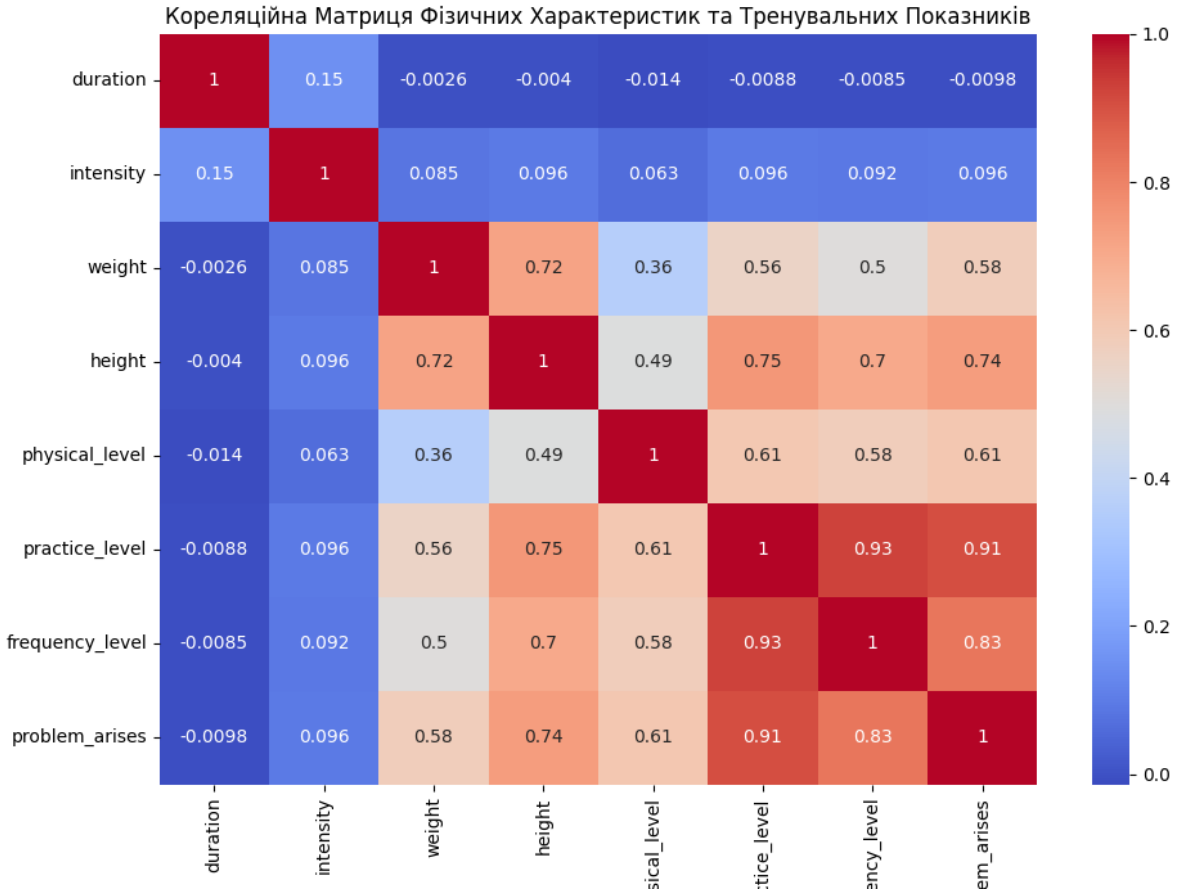

Рисунок 3.15 – Кореляційна матриця фізичних характеристик та тренувальних Показників

Кореляційна матриця на рисунку 3.15, показує ступінь лінійного зв'язку між фізичними характеристиками користувачів та їхньою активністю у додатку. Ось деякі спостереження, які можна зробити з цієї матриці:

- Вага та Зріст (weight і height): Вага та зріст мають високу позитивну кореляцію (близько 0.75), що є очікуваним, оскільки ці дві фізичні характеристики часто корелюють у реальному житті.

- Практичний Рівень та Частотний Рівень (practice_level і frequency_level): Є сильна позитивна кореляція (близько 0.93) між практичним рівнем і частотним рівнем, що може вказувати на те, що користувачі, які практикують частіше, схильні займатися на вищому рівні інтенсивності або складності.
- Проблеми, що Виникають (problem_arises): Ця змінна також має високу кореляцію з практичним і частотним рівнями (близько 0.92 і 0.85 відповідно), що може свідчити про те, що люди, які стикаються з проблемами, можуть бути більш мотивовані до частіших та більш інтенсивних тренувань.
- Тривалість і Інтенсивність (duration і intensity): Ці дві змінні мають досить слабку позитивну кореляцію (близько 0.13), що може свідчити про те, що більш тривалі сесії не завжди відповідають вищій інтенсивності тренувань.
- Фізичний Рівень (physical_level): Фізичний рівень має помірну кореляцію з вагою, зростом, практичним та частотним рівнями, що може вказувати на те, що фізично активні користувачі мають тенденцію до більш регулярних та інтенсивних тренувань.

Після первінного аналізу даних можемо приступити до безпосереднього виконання завдань щодо кластеризації та моделювання прогнозних моделей. Для розуміння залученості користувачів та оптимізації наших тренувальних програм, ми здійснили класифікацію користувачів за рівнем активності, використовуючи дані про їхні тренування. Дана задача була вирішна таким чином: було агрегувано дані тренувань користувачів, визначили аномалії та видалили їх. Потім, застосувавши метод кластеризації KMeans, розділили користувачів на групи "пасивні", "помірно активні" та "активні" на основі кількості та тривалості сесій тренувань. Врезультаті получили такий розподіл як показано на рисунках 3.16 та 3.17

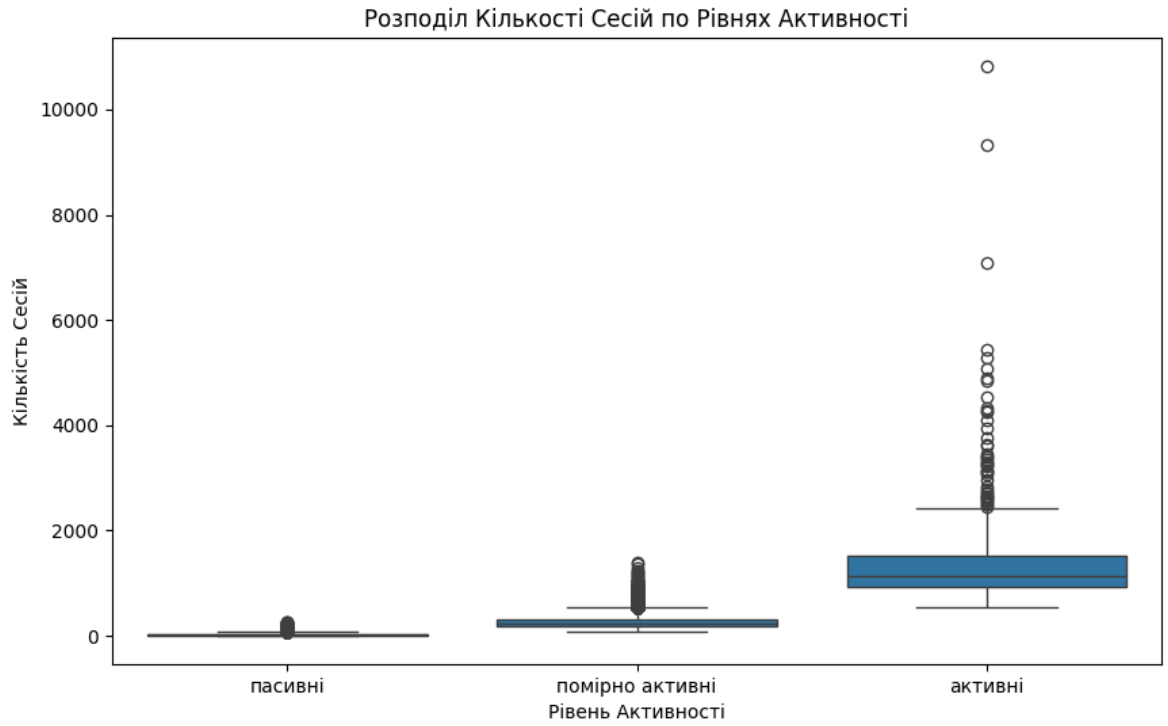

Рисунок 3.16 – Розподіл кількості сесій по рівнях активності

З рисунку 3.16 можна зробити наступні висновки:

- Більшість "пасивних" користувачів мають відносно низьку кількість тренувальних сесій.
- "Помірно активні" користувачі мають ширший розкид у кількості сесій, з деякими викидами, що вказують на користувачів, які мають вищу кількість сесій.
- "Активні" користувачі, як видно з графіка, мають значно більшу кількість сесій, що відображає високий рівень залученості в тренування.

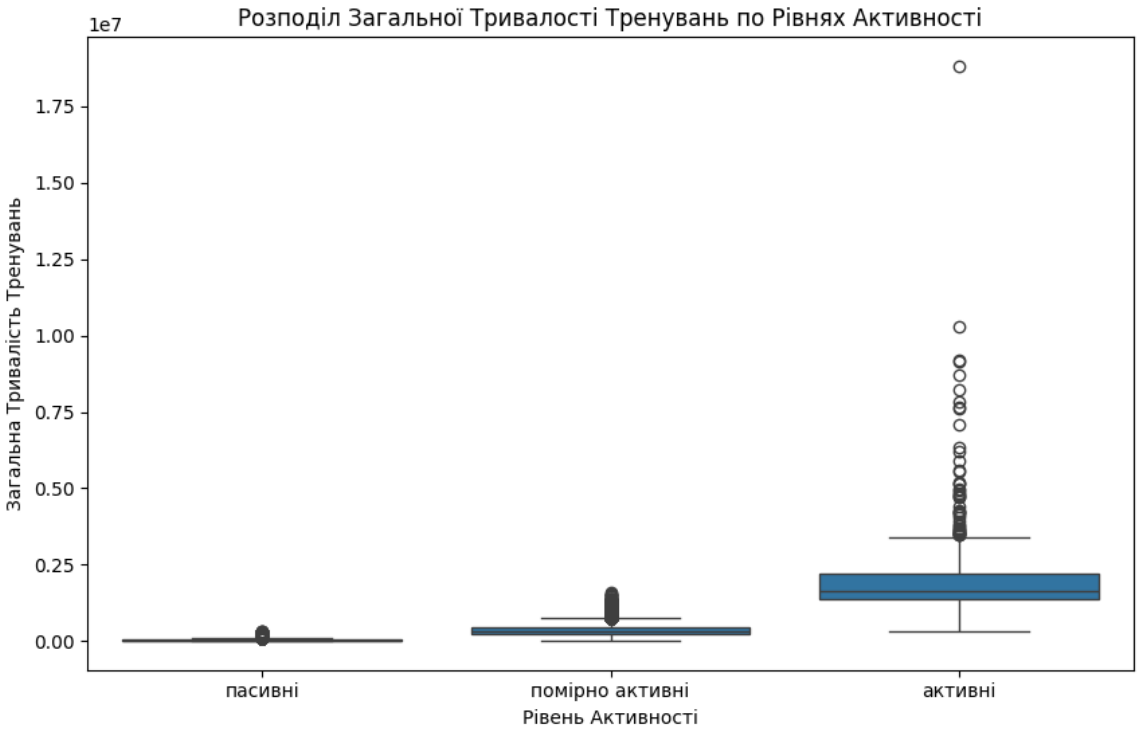

Рисунок 3.17 – Розподіл загальної тривалості тренувань по рівнях активності

З рисунку 3.17 можна зробити наступні висновки:

- "Пасивні" користувачі мають низьку загальну тривалість тренувань.
- "Помірно активні" користувачі показують більшу варіацію в тривалості тренувань, з декількома викидами, що може вказувати на більш тривалі чи інтенсивні сесії.
- "Активні" користувачі мають найвищу загальну тривалість тренувань, що відповідає їх високій кількості сесій.

Для розв'язання завдання визначення регулярності використання додатку користувачами було проведено аналіз часових інтервалів між тренувальними сесіями. Ми впорядкували сесії за датою, обчислили інтервали між ними та класифікували користувачів як "регулярні" або "нерегулярні", виходячи з того, чи відбуваються їхні тренування щодня. Результати візуалізовані на рисунку 3.18.

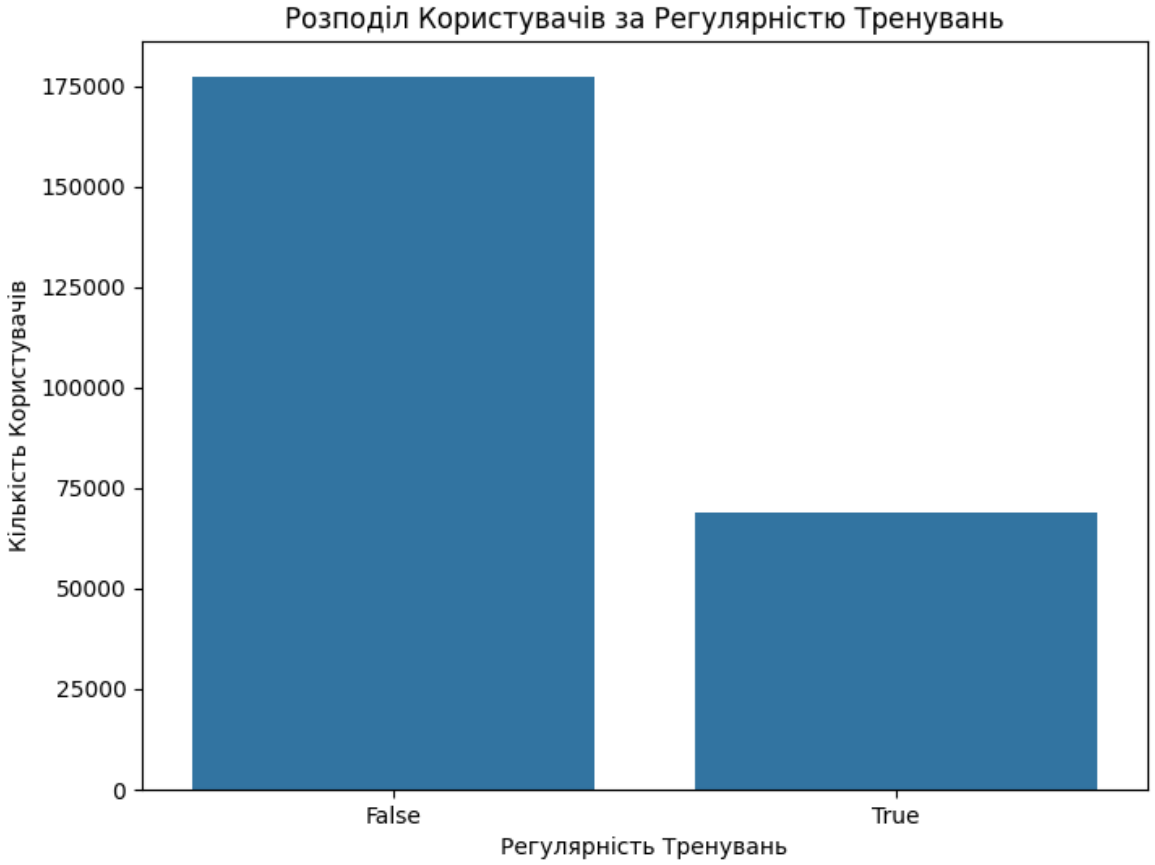

Рисунок 3.18 – Розподіл користувачів за регулярністю тренувань

На наданому рисунку 3.18 ми бачимо порівняння між кількістю користувачів, які займаються регулярно (означені як "True") та тими, хто займаються нерегулярно (означені як "False"). З графіка видно, що нерегулярні користувачі становлять більшість. Ця інформація може бути корисною для розробки стратегій залучення користувачів до більш регулярних тренувань, наприклад, через персоналізовані нагадування, мотиваційні повідомлення або адаптовані програми тренувань, щоб допомогти користувачам розвинути сталі звички.

Для розв'язання завдання прогнозування тривалості життєвого циклу користувачів в додатку, спершу нам потрібно підготувати дані. Зокрема, ми можемо розрахувати тривалість між датою реєстрації та останньою датою активності користувача. Також потрібно буде врахувати інформацію про кількість та тривалість сесій користувачів. Далі ми вибіраємо ознаки для моделі прогнозування тривалості життєвого циклу користувача в додатку залежить від гіпотез про те, які характеристики можуть впливати на вірогідність залишитися в додатку. Для цього ми побудуємо коряляційну матрицю щоб оцінити потенційно корисні ознаки для наної прогнозної моделі дивись рисунок 3.19.

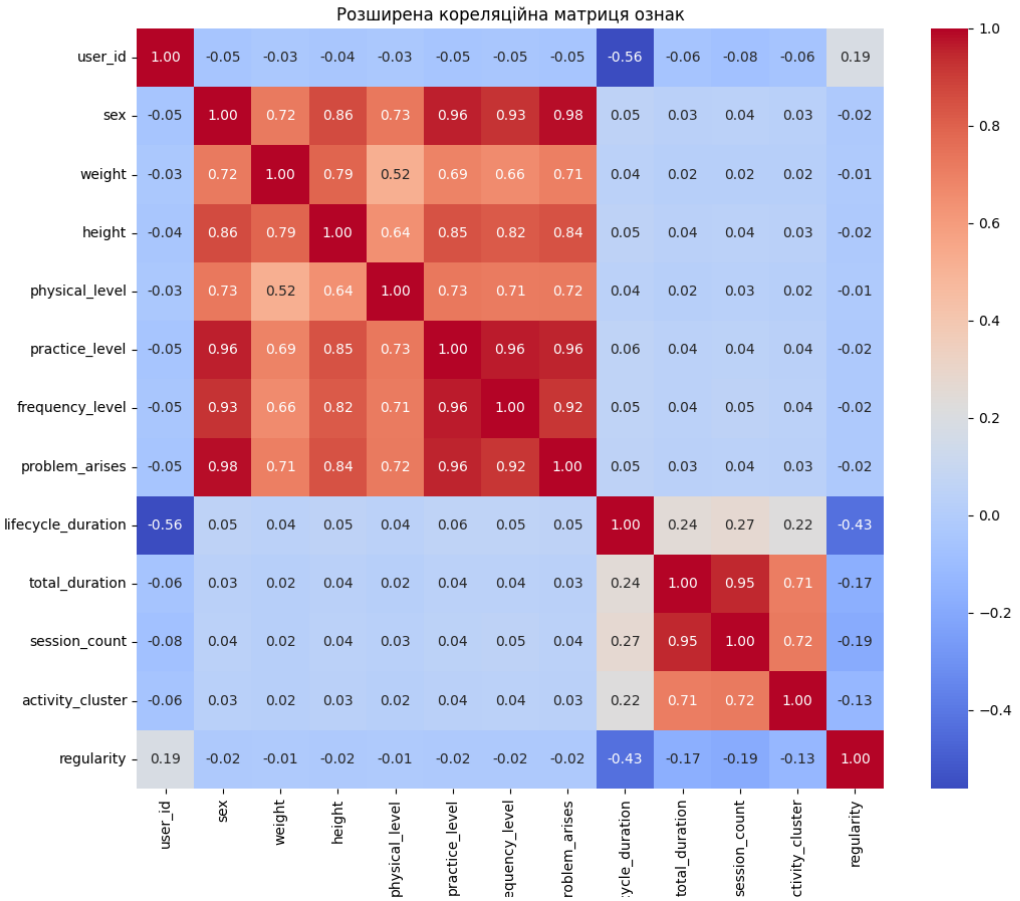

Рисунок 3.19 – Кореляційна матриця між потенційно корисними ознаками та тривалостю життєвого циклу

На рисунку 3.19 ознака regularity (регулярність користувача) та інші ознаки пов'язані з lifecycle_duration (тривалість життєвого циклу користувача). Ознака regularity має помірно негативну кореляцію з lifecycle_duration (-0.43), що може вказувати на те, що більш регулярні користувачі, як правило, користуються додатком довше. Озанака session_count (кількість сесій) має позитивний зв'язок (0.27) з lifecycle_duration, що також логічно, адже очікується, що користувачі, які мають більшу кількість сесій, будуть користуватися додатком довше. Ознака total_duration (загальна тривалість) має позитивну кореляцію (0.24) з lifecycle_duration.

Далі було проведено моделювання, подбір гіперпараметрів для деяких моделей та їх оцінка, результати можна побачити в порівняльній таблиці 3.4.

Таблиця 3.4 - Порівняльна таблиця якості моделей прогнозування тривалості життєвого циклу

Модель	**MAE**	**RMSE**	**R-squared**
Лінійна регресія	347 днів	486 днів	0,23
Лассо регресія	348 днів	486 днів	0,23
Ridge регресія	347 днів	486 днів	0,23
Дерева рішень	424 днів	652 днів	-0,39
SVM	295 днів	385 днів	0.42
XGBoost	197 днів	265 днів	0,72
LightGBM	170 днів	195 днів	0,81
Ансамбль	149 дні	214 дні	0,83

MAE - середня абсолютна помилка (Mean Absolute Error). Показує середнє абсолютне відхилення фактичних значень від спрогнозованих. Чим менше - тим краще.

RMSE - середньоквадратична помилка (Root Mean Squared Error). Середнє значення квадратів відхилень факту від прогнозу. Чим менше - тим краще. Більш чутлива до великих помилок ніж MAE.

R-squared - коефіцієнт детермінації. Показує наскільки добре модель пояснює варіацію залежної змінної. Може приймати значення від -∞ до 1. Чим ближче до 1, тим краща модель.

При розгляді більшості алгоритмів ми використовували крос-валідацію для оцінки якості моделі та підбору оптимальних гіперпараметрів:

- Для лінійних моделей явно не застосовували крос-валідацію, але розбивали вибірку на навчальну та тестову для оцінки.
- Для нейронних мереж використовували метод GridSearchCV, який автоматично робить крос-валідацію по сітці параметрів.
- При xgboost та lightgbm також застосували грід-пошук з крос-валідацією для оптимізації гіперпараметрів.
- При побудові ансамблю робили розбиття даних на фолди методом KFold.

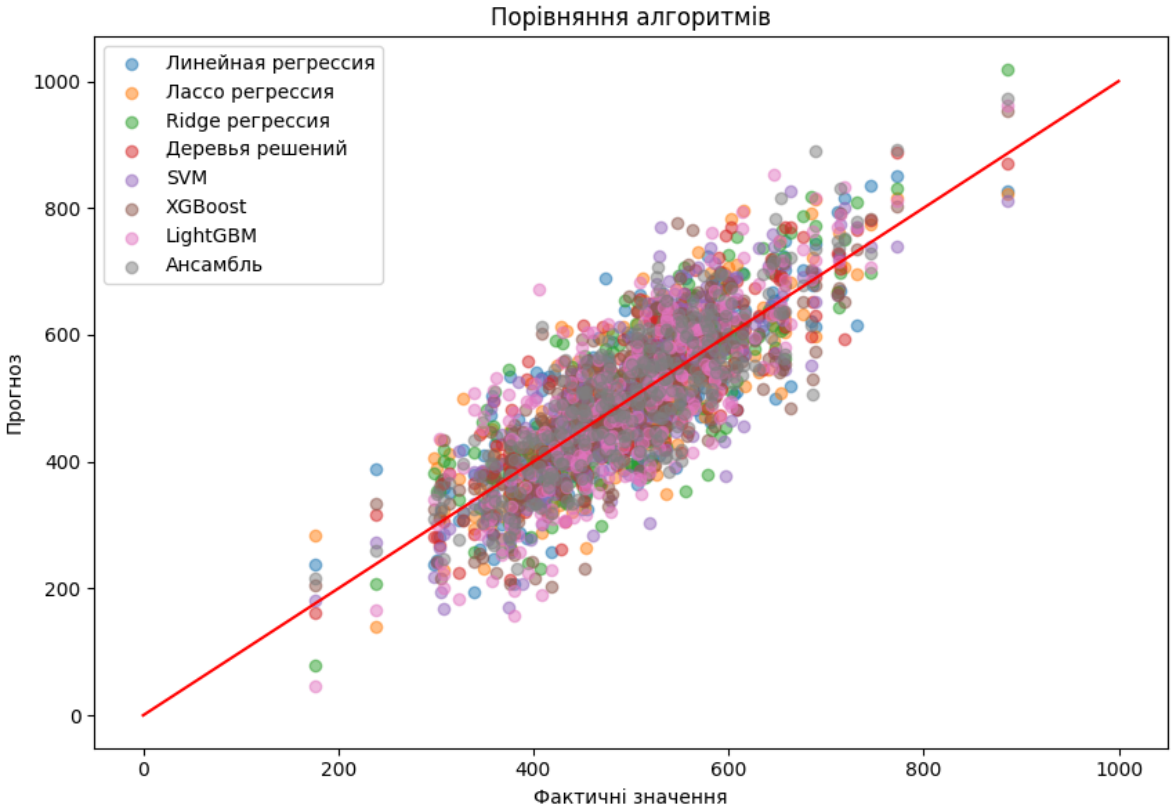

Рисунок 3.20 – Графічне порівняння якості моделей прогнозування тривалості життєвого циклу

Як бачимо з рисунку 3.20, найкращі результати показали градієнтне підсилення (XGBoost, LightGBM) та ансамбль на їх основі.

Для розв'язання завдання визначення схильності користувачів до певних програм електростимуляції за даними про їх переваги зробимо початковий аналіз.

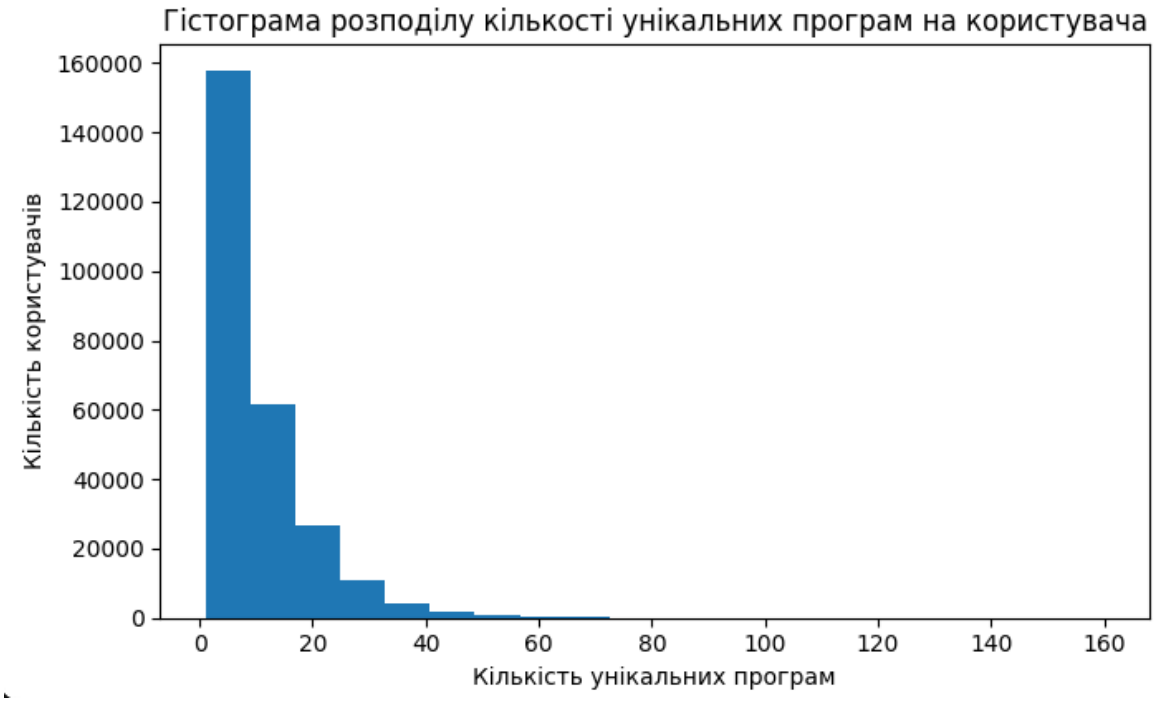

Рисунок 3.21 – Гістограма розподілу кількості унікальних програм на користувача

З рисунку 3.21 можна зробити декілька висновків:

- **Споживання програм**: Більшість користувачів використовують малу кількість унікальних програм електростимуляції. Це видно з того, що перша стовпчик (бін) має значно вищу кількість користувачів порівняно з іншими.
- **Вибірковість користувачів**: Існує група користувачів, яка використовує ширший спектр програм, що видно з менших стовпчиків, які все ж присутні на графіку. Це може вказувати на вибірковість або бажання експериментувати з різними програмами серед деяких користувачів.
- **Потенційні аутлаєри**: Дуже мала кількість користувачів використовує

дуже велику кількість унікальних програм. Це можуть бути аутлаєри, або особливо активні користувачі, які інтенсивно досліджують можливості програм електростимуляції.

- **Середнє використання**: Більшість користувачів схильні використовувати невелику кількість програм, що може вказувати на те, що вони знайшли конкретні програми, які відповідають їхнім потребам або перевагам, і не шукають додаткових варіантів.

Зробимо аналіз вибору програм електростимуляції для різних вікових груп.

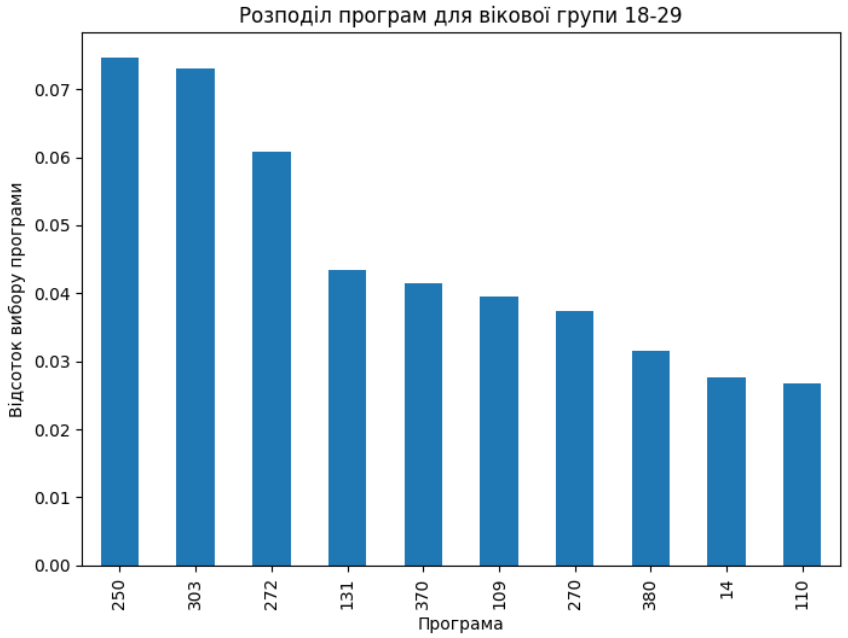

Рисунок 3.22 – Розподіл програм для вікових груп 18-29

Рисунок 3.22 показує дані щодо вікової групи 18-29. Ця група має найвищу частоту використання програми 250, а також значну частоту використання програм 303 і 272. Це може свідчити про високу активність або інтерес до певних видів електростимуляції серед молодших користувачів.

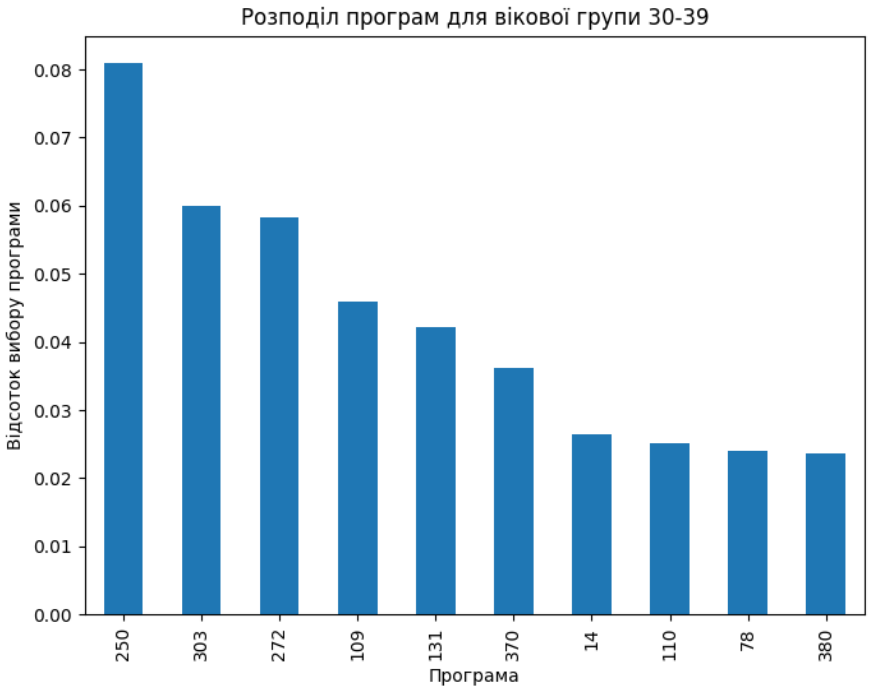

Рисунок 3.23 – Розподіл програм для вікових груп 30-39

Рисунок 3.23 показує дані щодо вікової групи 30-39. В цій групі також домінує програма 250, але з меншою часткою, ніж у молодших користувачів, а програми 303 і 272 залишаються популярними.

Рисунок 3.24 – Розподіл програм для вікових груп 40-49

Рисунок 3.24 показує дані щодо вікової групи 40-49. Графік показує зміну

відносною частотою вибору програм. Програма 250 все ще популярна, але програма 109 з'являється з більшою частотою, можливо, вказуючи на зміну переваг чи потреб у цій віковій категорії.

Рисунок 3.25 – Розподіл програм для вікових груп 50-59

Рисунок 3.25 - вікова група 50-59, в цій групі програма 109 стає ще більш популярною, а програма 250 відходить на другий план. Це може вказувати на зміну потреб у стимуляції з віком або на підвищену увагу до специфічних проблем здоров'я, які стають актуальними у цьому віковому діапазоні.

На рисунку 3.26 - вікова група 60-69 можна побачити що програма 250 залишається серед найпопулярніших, але ми також бачимо високу частоту для програми 131. Це може бути пов'язано з особливими потребами цієї вікової групи.

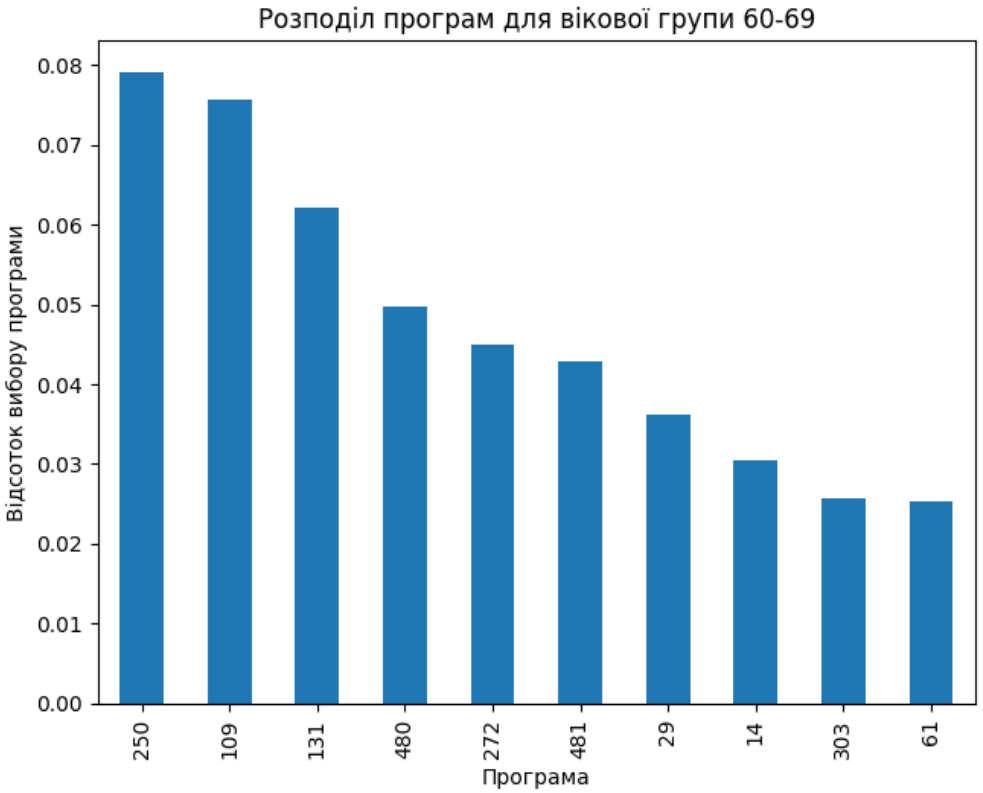

Рисунок 3.26 – Розподіл програм для вікових груп 60-69

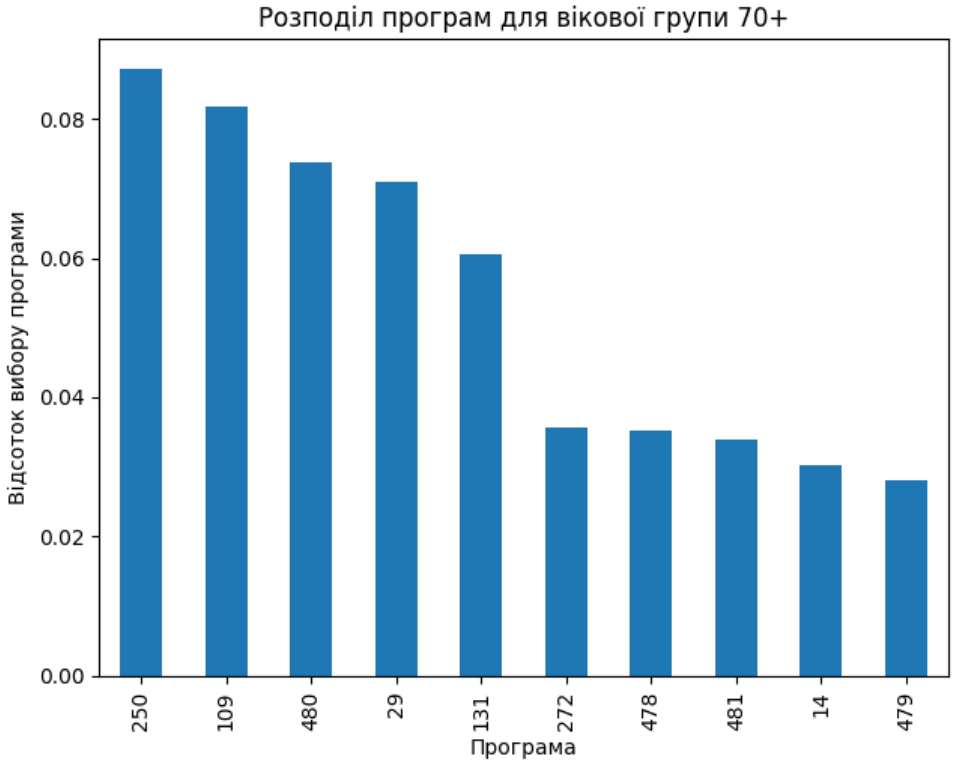

Рисунок 3.27 – Розподіл програм для вікових груп 70+

Рисунок 3.27 показує дані щодо вікової групі 70+. В цій групі програма 250 залишається найпопулярнішою, а програма 109 також має значну частоту використання. Це вказує на те, що старші користувачі можуть мати схожі потреби для електростимуляції, як і в інших вікових групах, але можливо з більшим фокусом на певні типи програм, які відповідають їх здоров'ю або фізичному стану.

Подводячі висновки можна сказати що з цих графіків можна вивести, що деякі програми мають універсальну привабливість для різних вікових груп, тоді як інші програми більше цікавлять певні вікові категорії. Це може бути корисним для розробки цільових маркетингових стратегій або для оптимізації пропозиції програм залежно від віку користувачів.

Для розв'язання завдання надання рекомендацій щодо оптимальних електростимуляційних програм для користувачів додатку було реалізована та протестовано на яксіть деклько підходів, а саме:

- User-based - рекомендації формуються на основі профілів схожих користувачів. Тобто шукаємо N схожих користувачів, аналізуємо що вони люблять і рекомендуємо це items цільовому користувачеві.
- Item-based - знаходяться схожі items (а не користувачі). Наприклад, якщо користувач любить item X, знаходимо itemи схожі з X і рекомендуємо.
- гібридний варіант із попередньою кластеризацією користувачів
- адаптований алгоритм к-найближчих сусідів з наступною селекцією найбільш популярних програм

В якості мір якості були використані наступні міри:

- Точність (Precision) - частка релевантних рекомендацій серед усіх

запропонованих. Показує з якою ймовірністю рекомендована програма справді цікава для користувача.

- Повнота (Recall) - частка вгаданих алгоритмом релевантних програм з усіх програм, які реально цікаві користувачеві. Показує наскільки алгоритм знайшов усі потенційно цікаві програми.
- F1 score - усереднена гармонічна по точності та повноті. Узагальнена метрика якості.

Далі було проведено моделювання, та разрахована середня оцінка для 1000 випадково обраних корістувачів за обраними мірами якості, результати можна побачити в порівняльній таблиці 3.5.

Таблиця 3.5 - Порівняльна таблиця якості моделей рекомендацій програм

Модель	**Precision**	**Recall**	**F1**
User-based	0.18	0.21	0.19
Item-based	0.12	0.19	0.15
Hybrid algorithm with cluster	0.59	0.68	0.63
Aadopted algorithm with KNN	0.78	0.90	0.83

Проаналізуємо результати представлені в таблиці 3.5

User-based Model:

- Precision (0.18): Лише 18% рекомендацій, які надала ця модель, виявилися релевантними для користувачів. Це досить низький показник, що свідчить про велику кількість нерелевантних рекомендацій.
- Recall (0.21): Модель змогла виявити 21% релевантних програм з усіх можливих. Це означає, що більшість потенційно цікавих програм для користувачів не було рекомендовано.

- F1 (0.19): Низька F1-метрика вказує на незадовільний баланс між точністю та відгуком, підтверджуючи, що модель не є дуже ефективною.

Item-based Model:

- Precision (0.12): Тільки 12% рекомендацій, які надала ця модель, виявилися релевантними для користувачів. Це дуже низький показник, що свідчить про значну кількість нерелевантних рекомендацій.
- Recall (0.19): Модель змогла ідентифікувати 19% релевантних програм з усіх можливих, що також є невисоким показником і вказує на пропуск багатьох потенційно цікавих рекомендацій для користувачів.
- F1 (0.15): Значення F1-метрики 0.15 свідчить про певний дисбаланс між точністю та відгуком, підтверджуючи, що модель не є особливо ефективною.

Hybrid Algorithm with Cluster:

- Precision (0.59): Близько 59% рекомендацій виявилися релевантними. Це значне поліпшення порівняно з попередніми моделями, що свідчить про зниження кількості нерелевантних рекомендацій.
- Recall (0.68): Модель виявила 68% релевантних програм, що свідчить про здатність ефективно виявляти програми, які можуть зацікавити користувачів.
- F1 (0.63): Збалансована F1-метрика вказує на гармонійне співвідношення між точністю та відгуком.

Adopted Algorithm with KNN:

- Precision (0.78): Висока точність показує, що 78% рекомендацій є релевантними. Це означає, що користувачі, швидше за все, знайдуть більшість рекомендованих програм цікавими та корисними.
- Recall (0.90): Дуже високий відгук показує, що модель здатна виявляти 90% релевантних програм, забезпечуючи широкий огляд потенційно цікавих програм для користувачів.

- F1 (0.83): Висока F1-метрика підтверджує ефективність моделі у балансі між точністю та відгуком.

З проведеного аналізу результатів можна зробити висновки що адаптований алгоритм з k-NN показує найкращі результати за всіма метриками, особливо за точністю та відгуком, забезпечуючи високу якість рекомендацій. Гібридний алгоритм також є досить ефективним, але не досягає такої високої ефективності, як алгоритм з k-NN. Традиційні user-based та item-based моделі програють більш складним алгоритмам.

ВИСНОВКИ

В результаті виконання роботи було створено інформаційну систему на основі методів машинного навчання для аналізу взаємодії користувачів з медичним додатком Bluefit. Було розглянуто різні підходи та алгоритми машинного навчання і обрано оптимальні для вирішення поставлених задач дослідження.

У ході роботи було виконано:

- Аналіз предметної області, визначення актуальності та цілей дослідження
- Порівняння існуючих методів аналізу поведінки користувачів
- Вибір та підготовка вхідних даних з медичного додатку Bluefit
- Реалізація моделей машинного навчання для класифікації та кластеризації користувачів
- Розробка алгоритму надання персоналізованих рекомендацій на основі побудованих моделей
- Тестування та оцінка ефективності запропонованого підходу

За результатами дослідження можна зробити висновок, що розроблена інформаційна технологія дозволяє з достатньо високою якістю аналізувати поведінку користувачів медичного додатку Bluefit та прогнозувати їх подальші дії. Це відкриває можливості для вдосконалення додатку - адаптації програм електростимуляції, підвищення мотивації користувачів та ефективності тренувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chou Y.-H., Wang L.-Y., Kao T. Біомедичні застосування електричної стимуляції // PMC - NCBI, 2021. – Доступно за адресою: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7954539
2. Kourou K., Exarchos T. P., Exarchos K. P., Karamouzis M. V., Fotiadis D. I. Машинне навчання в медичних застосуваннях: огляд сучасних методів // ScienceDirect, 2022. – Доступно за адресою: https://www.sciencedirect.com/science/article/pii/S0010482522002505
3. Alowais, S.A., Alghamdi, S.S., Alsuhebany, N. et al. Revolutionizing healthcare: the role of artificial intelligence in clinical practice. BMC Med Educ 23, 689 (2023). https://doi.org/10.1186/s12909-023-04698-z
4. May, M. Eight ways machine learning is assisting medicine. Nat Med 27, 2–3 (2021). https://doi.org/10.1038/s41591-020-01197-2
5. Javaid, M., Haleem, A., Pratap Singh, R., Suman, R., & Rab, S. (2021). Significance of machine learning in healthcare: Features, pillars and applications. International Journal of Intelligent Networks, 3, 58-73. https://doi.org/10.1016/j.ijin.2022.05.002
6. Turgeon, S., Lanovaz, M.J. Tutorial: Applying Machine Learning in Behavioral Research. Perspect Behav Sci 43, 697–723 (2020). https://doi.org/10.1007/s40614-020-00270-y
7. Chen, I. Y., Joshi, S., Ghassemi, M., & Ranganath, R. (2021). Probabilistic Machine Learning for Healthcare. Annual Review of Biomedical Data Science. https://doi.org/10.1146/annurev-biodatasci-092820-033938
8. Sahoo, A. K., Pradhan, C., Barik, R. K., & Dubey, H. (2019). DeepReco: Deep Learning Based Health Recommender System Using Collaborative Filtering. Computation, 7(2), 25. https://doi.org/10.3390/computation7020025
9. Pal, S., Biswas, B., Gupta, R., Kumar, A., & Gupta, S. (2023). Exploring the factors that affect user experience in mobile-health applications: A

text-mining and machine-learning approach. Journal of Business Research, 156, 113484. https://doi.org/10.1016/j.jbusres.2022.113484

10. Y. Tong, A. I. Messinger and G. Luo, "Testing the Generalizability of an Automated Method for Explaining Machine Learning Predictions on Asthma Patients' Asthma Hospital Visits to an Academic Healthcare System," in IEEE Access, vol. 8, pp. 195971-195979, 2020, doi: 10.1109/ACCESS.2020.3032683.
11. Shrivastava, P. K., Sharma, M., Sharma, P., & Kumar, A. (2023). HCBiLSTM: A hybrid model for predicting heart disease using CNN and BiLSTM algorithms. Measurement: Sensors, 25, 100657. https://doi.org/10.1016/j.measen.2022.100657
12. Badawy, M., Ramadan, N. & Hefny, H.A. Healthcare predictive analytics using machine learning and deep learning techniques: a survey. Journal of Electrical Systems and Inf Technol 10, 40 (2023). https://doi.org/10.1186/s43067-023-00108-y
13. Pham, T., Tran, T., Phung, D., & Venkatesh, S. (2017). Predicting healthcare trajectories from medical records: A deep learning approach. Journal of Biomedical Informatics, 69, 218-229. https://doi.org/10.1016/j.jbi.2017.04.001
14. J. P. Li, A. U. Haq, S. U. Din, J. Khan, A. Khan and A. Saboor, "Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare," in IEEE Access, vol. 8, pp. 107562-107582, 2020, doi: 10.1109/ACCESS.2020.3001149.
15. Yazdani, A., Varathan, K.D., Chiam, Y.K. et al. A novel approach for heart disease prediction using strength scores with significant predictors. BMC Med Inform Decis Mak 21, 194 (2021). https://doi.org/10.1186/s12911-021-01527-5
16. R. Ghorbani, R. Ghousi, A. Makui and A. Atashi, "A New Hybrid Predictive Model to Predict the Early Mortality Risk in Intensive Care Units on a

Highly Imbalanced Dataset," in IEEE Access, vol. 8, pp. 141066-141079, 2020, doi: 10.1109/ACCESS.2020.3013320.

17. Haq, A. U., Li, J. P., Khan, J., Memon, M. H., Nazir, S., Ahmad, S., Khan, G. A., & Ali, A. (2019). Intelligent Machine Learning Approach for Effective Recognition of Diabetes in E-Healthcare Using Clinical Data. Sensors, 20(9), 2649. https://doi.org/10.3390/s20092649
18. King, A. J., Cooper, G. F., Hochheiser, H., Clermont, G., Hauskrecht, M., & Visweswaran, S. (2018). Using Machine Learning to Predict the Information Seeking Behavior of Clinicians Using an Electronic Medical Record System. AMIA Annual Symposium Proceedings, 2018, 673-682. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371238/
19. Abed Salman, M., & Abaid Mahdi, M. (2023). Nifty method for prediction dynamic features of online social networks from users' activity based on machine learning. Results in Engineering, 20, 101430. https://doi.org/10.1016/j.rineng.2023.101430
20. Johnston, S. S., Morton, J. M., Kalsekar, I., Ammann, E. M., Hsiao, C., & Reps, J. (2019). Using Machine Learning Applied to Real-World Healthcare Data for Predictive Analytics: An Applied Example in Bariatric Surgery. Value in Health, 22(5), 580-586. https://doi.org/10.1016/j.jval.2019.01.011
21. O. Oyebode, F. Alqahtani and R. Orji, "Using Machine Learning and Thematic Analysis Methods to Evaluate Mental Health Apps Based on User Reviews," in IEEE Access, vol. 8, pp. 111141-111158, 2020, doi: 10.1109/ACCESS.2020.3002176.

ДОДАТОК

Програмний код аналізу даних на мові програмування Python

ml_tools.py - кастомні утіліти які використовуются для работи з даними

```
import os
from datetime import datetime

import pandas as pd
import numpy as np
from sqlalchemy import create_engine
from tqdm import tqdm
import time
import gender_guesser.detector as gender
from joblib import load, dump
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

start_time = time.time()


def script_duration():
    end_time = time.time()
    elapsed_time = end_time - start_time
    minutes = int(elapsed_time / 60)
    seconds = int(elapsed_time % 60)
    print(f"Execution time: {minutes} minutes {seconds} seconds")


def count_chunks_in_h5(file_name):
    # Відкриття файлу HDF5
    with pd.HDFStore(file_name, mode='r') as hdf_store:
        # Отримання списку ключів
        keys = hdf_store.keys()
        print(f"Кількість чанків у файлі: {len(keys)}")
        return len(keys)


def load_h5_to_pd(file_name):
    start_time = time.time()
    count_chunks_in_h5(file_name)
    with pd.HDFStore(file_name, mode='r') as store:
        keys = store.keys()
        data_frames = [store[key] for key in keys]
        full_data = pd.concat(data_frames, ignore_index=True)
        elapsed_time = time.time() - start_time
        minutes, seconds = divmod(elapsed_time, 60)
        print(f"Завантажено в дата-фрейм за: {int(minutes)} мінут, {int(seconds)}
секунд")
        return full_data
```

```
# Loading data from SQL DB to DataFrame
def load_data_from_db(table_name, fields="*", optimaze=True, chunksize=10000,
filetype="pkl"):
    """
        Завантажує дані з таблиці бази даних у файл HDF5, оптимізуючи категоріальні
дані.

        Ця функція створює з'єднання з базою даних MySQL, читає дані з заданої
таблиці,
        оптимізує категоріальні стовпці для ефективного використання пам'яті та
зберігає
        дані у файл формату HDF5 частинами.

        Параметри:
            table_name (str): Назва таблиці в базі даних, з якої будуть
завантажуватися дані.
            fields (str, optional): Рядок, що визначає стовпці таблиці для
завантаження.
                                    Значення за замовчуванням "*" вибирає всі стовпці.
            chunksize (int, optional): Розмір частини даних для читання за один раз.
                                       Значення за замовчуванням - 10000.

        Поведінка:
            Функція виводить індикатор прогресу з оцінкою залишкового часу обробки.
            Дані зберігаються у файлі 'data/<table_name>_data.h5'.

        Повертає:
            None. Результати зберігаються безпосередньо у файл HDF5.
   """

    out_file_name = "data/" + table_name + "_data.h5"
    if os.path.exists(out_file_name):
        os.remove(out_file_name)

    print(f"Load and optimize data for {table_name} from DB to {out_file_name}")
    # Створення з'єднання з базою даних
    engine = create_engine('mysql+pymysql://root:root@localhost/patient_stat')

    # Оцінка кількості рядків у таблиці (для індикатора прогресу)
    total_rows = pd.read_sql_query(f"SELECT COUNT(*) FROM {table_name}",
engine).iloc[0].item()

    # Створення SQL запиту для завантаження даних
    query = f"SELECT {fields} FROM {table_name}"

    # Ініціалізація індикатора прогресу
    tqdm_iterator = tqdm(total=total_rows, desc="Processing", unit="row")

    start_time = time.time()

    # Читання даних частинами та їх оптимізація
```

```
    for i, chunk in enumerate(pd.read_sql_query(query, engine, chunksize=chunksize)):
        # Збереження даних у файл у форматі HDF5
        chunk.to_hdf(out_file_name, key=f'data_{i}', mode='a', format='table',
complib='blosc', complevel=9)

        # Оновлення індикатора прогресу
        tqdm_iterator.update(len(chunk))

        # Оцінка залишкового часу
        remaining_time = 'Calculating ...'
        elapsed_time = time.time() - start_time
        if tqdm_iterator.n:
            remaining_secs = int((total_rows - tqdm_iterator.n) / (tqdm_iterator.n /
elapsed_time))
            mins, secs = divmod(remaining_secs, 60)
            remaining_time = f'{mins} min, {secs} sec'
        tqdm_iterator.set_postfix(remaining=remaining_time, refresh=True)

    # Завершення індикатора прогресу
    tqdm_iterator.close()

    if optimaze:
        print("Optimize the data")

        df = load_h5_to_pd(out_file_name)

        # Оптимізація даних у категоріальний тип
        for col in df.select_dtypes(include='object').columns:
            print(f"Convert column {col} to category")
            df[col] = df[col].astype('category')

        os.remove(out_file_name)

        if filetype == "pkl":
            # Збереження оптимізованих даних у файл у форматі PKL
            out_file_name_pkl = "data/" + table_name + "_data.pkl"
            if os.path.exists(out_file_name_pkl):
                os.remove(out_file_name_pkl)
            df.to_pickle(out_file_name_pkl)
        else:
            # Збереження оптимізованих даних у файл у форматі HDF5
            df.to_hdf(out_file_name, key="data", format='table', complib='blosc',
complevel=9)


# Function for guessing sex by firstname
genderDetector = gender.Detector()


def guess_gender(name):
    g = {
        "male": 1,
```

```
        "mostly_male": 1,
        "female": 0,
        "mostly_female": 0,
        "unknown": None
    }
    if name is not None and type(name) == str and len(name.split()) > 0:
        guess = genderDetector.get_gender(name.split()[0])
        if guess in g.keys():
            return g[guess]
        else:
            # print(guess)
            return None
    else:
        return None


# Process sex guessing for all users
def guess_genders(df, field="firstname"):
    tqdm_iterator = tqdm(total=len(df), desc="Processing", unit="row")
    for index, row in df.iterrows():
        df.at[index, 'guessed_sex'] = guess_gender(row[field])
        tqdm_iterator.update(1)
    tqdm_iterator.close()
    return df


# Function for generating normal distribution
def generate_normal_value(mean, std, min_val, max_val):
    while True:
        value = np.random.normal(mean, std)
        if min_val <= value <= max_val:
            return value


# Видалення викидів
def remove_outliers(df, column):
    '''
    Викиди визначаємо за допомогою IQR,
    який є різницею між 75-м та 25-м процентилями.
    Значення, що лежать поза межами 1.5 * IQR від квартилів,
    традиційно вважаються викидами.
    :param df: датафрейм
    :param column: поле нормування
    :return:
    '''

    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
```

```
    print(lower_bound)
    print(upper_bound)

    df[column] = df[column].clip(lower_bound, upper_bound)
    return df


def session_for_current_year(df):
    # Визначення поточного року
    current_year = datetime.now().year

    # Вибір даних за поточний рік
    current_year_data = df[df['date'].dt.year == current_year]
    return current_year_data


def save_df(df, name):
    df.to_pickle(f"data/{name}.pkl")


def load_df(name):
    return pd.read_pickle(f"data/{name}.pkl")


def save_model(model_name, model):
    dump(model, f'data/models/{model_name}')


def predict(model_name, data):
    # Завантаження моделі з файлу
    model = load(f'data/models/{model_name}')
    return model.predict(data)


def model_metrics(y_test, y_pred):
    MAE = mean_absolute_error(y_test, y_pred)
    RMSE = np.sqrt(mean_squared_error(y_test, y_pred))
    R2S = r2_score(y_test, y_pred)
    print(f'MAE: {MAE}')
    print(f'RMSE: {RMSE}')
    print(f'R-squared: {R2S}')
    return MAE, RMSE, R2S


def get_used_programs(user_id, user_item_matrix):
    if user_id in user_item_matrix.index:
        # Вибір рядка для користувача
        user_data = user_item_matrix.loc[user_id]

        # Вибір тільки тих програм, якими користувач користувався
        used_programs = user_data[user_data > 0].index.tolist()
```

```
        # print(f"Programs used by user {user_id}: {used_programs}")
        return used_programs
    else:
        print(f"User ID {user_id} not found.")
        return []


def calculate_metrics_PRF(recommended_programs, user_used_programs):
    true_positives = set(recommended_programs).intersection(user_used_programs)
    precision = len(true_positives) / len(recommended_programs) if
recommended_programs else 0
    recall = len(true_positives) / len(user_used_programs) if user_used_programs else
0
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall)
else 0
    #print(f"Precision: {precision}, Recall: {recall}, F1 Score: {f1_score}")
    return precision, recall, f1_score


def get_non_zero(d):
    return d[d != 0]
```

load_data.py - завантаження даних з зовнішньої БД до системи моделювання

```
from ml_tools import *

# user_id, phone_model, os, version, date, bodypart, program, duration, intensity
load_data_from_db('session', 'user_id, phone_model, os, version, date, bodypart,
program, duration, intensity')

# user_id, firstname, lastname, language, registered, activated
load_data_from_db('user', 'user_id, firstname, lastname, language, registered,
activated')

# user_id, sex, birth, weight, height, physical_level, practice_level,
frequency_level, problem_arises
load_data_from_db('patient_profile', 'user_id, sex, birth, weight, height,
physical_level, practice_level, frequency_level, problem_arises')
```

initial_data_review.py - початковий аналіз якості даних

```
import pandas as pd

session_df = pd.read_pickle("data/session_data.pkl")
user_df = pd.read_pickle("data/user_data.pkl")
patient_profile_df = pd.read_pickle("data/patient_profile_data.pkl")


# Етап 1: Оцінка Якості Даних
# Перевірка наявності пропущених значень
```

```
print(session_df.isnull().sum())

# Опис статистики даних для виявлення аномалій
print(session_df.describe())

# Перевірка наявності пропущених значень
print(user_df.isnull().sum())

# Опис статистики даних для виявлення аномалій
print(user_df.describe())

# Перевірка наявності пропущених значень
print(patient_profile_df.isnull().sum())

# Опис статистики даних для виявлення аномалій
print(patient_profile_df.describe())
```

prep_data_session.py - Очищення та підготовка даних session

```
from ml_tools import *



session_df = pd.read_pickle("data/session_data.pkl")


# Очищення даних session
print(session_df.head())

# Очищення данних для анонімного користувача
session_df = session_df.loc[session_df['user_id'] != 379]

# Видалення рядків, де відсутній user_id
print("Видалення рядків, де відсутній user_id")
c = len(session_df)
session_df = session_df.dropna(subset=['user_id'])
print(f"Видалено: {(c - len(session_df))} рядків")

# Видалення рядків, де відсутній program
print("Видалення рядків, де відсутній program")
c = len(session_df)
session_df = session_df.dropna(subset=['program'])
print(f"Видалено: {(c - len(session_df))} рядків")

# Видалення рядків, де відсутній bodypart
print("Видалення рядків, де відсутній bodypart")
c = len(session_df)
session_df = session_df.dropna(subset=['bodypart'])
print(f"Видалено: {(c - len(session_df))} рядків")

stats = session_df.agg({
    'duration': ['median', 'min', 'max', 'std'],
```

```
    'intensity': ['mean', 'min', 'max', 'std']
})
print(stats)

# Прибираємо викиди для полей duration, intensity
session_df = remove_outliers(session_df, 'duration')
session_df = remove_outliers(session_df, 'intensity')

# Виправлення аномалій для 'duration' та 'intensity'
# Заміна від'ємних значень на медіану та обрізання викидів
duration_median = session_df[session_df['duration'] > 0]['duration'].median()
session_df['duration'] = session_df['duration'].apply(lambda x: duration_median if x
<= 0 else x)

intensity_median = session_df[session_df['intensity'] > 0]['intensity'].median()
session_df['intensity'] = session_df['intensity'].apply(lambda x: intensity_median if
x <= 0 else x)


# Імпутація для 'version' та 'bodypart' за допомогою моди (найчастіше значення)
version_mode = session_df['version'].mode()[0]
session_df['version'].fillna(version_mode, inplace=True)

bodypart_mode = session_df['bodypart'].mode()[0]
session_df['bodypart'].fillna(bodypart_mode, inplace=True)

duration_mode = session_df['duration'].mode()[0]
session_df['duration'].fillna(duration_mode, inplace=True)

intensity_mode = session_df['intensity'].mode()[0]
session_df['intensity'].fillna(intensity_mode, inplace=True)


stats = session_df.agg({
    'duration': ['median', 'min', 'max', 'std'],
    'intensity': ['mean', 'min', 'max', 'std']
})
print(stats)


# Видалення дуже великих значень
# session_df = session_df[session_df['duration'] < certain_threshold]
# session_df = session_df[session_df['intensity'] < certain_threshold]

session_df["program"] = session_df["program"].astype('category')
#session_df.to_hdf("data/session_data_prep.h5", key="data", format='table',
complib='blosc', complevel=9)
session_df.to_pickle("data/session_data_prep.pkl")
```

prep_data_user.py - Очищення та підготовка даних user

```
from ml_tools import *

user_df = pd.read_pickle("data/user_data.pkl")

# Заповнення пропущених значень в 'firstname' та 'language' модою
user_df['firstname'].fillna(user_df['firstname'].mode()[0], inplace=True)
user_df['language'].fillna(user_df['language'].mode()[0], inplace=True)

# Обробка пропущених значень в 'activated'
# Ми можемо залишити ці значення як є, або замінити їх на спеціальне значення
# Наприклад, заміна на дату далеку в минулому
user_df['activated'].fillna(pd.Timestamp('1900-01-01'), inplace=True)

print(user_df.isnull().sum())
print(user_df.describe())
user_df.to_pickle("data/user_data_prep.pkl")
```

prep_data_user_profile.py - Очищення та підготовка даних user_profile

```
from ml_tools import *

user_df = pd.read_pickle("data/user_data_prep.pkl")
patient_profile_df = pd.read_pickle("data/patient_profile_data.pkl")

print(patient_profile_df.info())

# Fill sex by first name from user
profiles_without_sex = patient_profile_df[patient_profile_df['sex'].isnull()]
tqdm_iterator = tqdm(total=len(profiles_without_sex), desc="Processing", unit="row")
for index, row in profiles_without_sex.iterrows():
    user = user_df[user_df['user_id'] == row['user_id']]
    if not user.empty:
        firstname = user['firstname'].iloc[0]
        g_g = guess_gender(firstname)
        if g_g is not None:
            patient_profile_df.at[index, 'sex'] = g_g
    tqdm_iterator.update(1)
tqdm_iterator.close()

# Calculating the gender distribution
sex_distribution = patient_profile_df['sex'].dropna().value_counts(normalize=True)
sexes = sex_distribution.index.tolist()
probabilities = sex_distribution.values.tolist()

# Setting the floor for the remaining entries
for index, row in patient_profile_df[patient_profile_df['sex'].isnull()].iterrows():
    patient_profile_df.at[index, 'sex'] = np.random.choice(sexes, p=probabilities)

# Прибираємо викиди для полей birth, weight, height
patient_profile_df = remove_outliers(patient_profile_df, 'birth')
patient_profile_df = remove_outliers(patient_profile_df, 'weight')
```

```
patient_profile_df = remove_outliers(patient_profile_df, 'height')

# Calculate the mean or median values of the date of birth for each sex
birth_stats = patient_profile_df.groupby('sex')['birth'].agg(['mean', 'median'])

# Filling in missing values in birth by gender
for sex, stats in birth_stats.iterrows():
    fill_value = stats['median']
    # Fill in the values for the respective gender
    patient_profile_df.loc[(patient_profile_df['sex'] == sex) &
(patient_profile_df['birth'].isnull()), 'birth'] = fill_value


# Fields for filling
fields = ['weight', 'height', 'physical_level', 'practice_level', 'frequency_level',
'problem_arises']

# Calculation of median values for each field by gender
medians = patient_profile_df.groupby('sex')[fields].median()

# Filling in missing values
for field in fields:
    for sex in medians.index:
        median_value = medians.at[sex, field]
        patient_profile_df.loc[(patient_profile_df['sex'] == sex) &
(patient_profile_df[field].isnull()), field] = median_value


# Create missing patient profiles

# Calculate sex distributions and mean values for other fields
sex_distribution = patient_profile_df['sex'].dropna().value_counts(normalize=True)
means = patient_profile_df.groupby('sex').mean()

tqdm_iterator = tqdm(total=len(user_df), desc="Processing", unit="row")
new_records = []
for index, row in user_df.iterrows():
    if row['user_id'] not in patient_profile_df['user_id'].values:
        guessed_sex = guess_gender(row['firstname'])
        if guessed_sex is None:
            sexes = sex_distribution.index.tolist()
            probabilities = sex_distribution.values.tolist()
            guessed_sex = np.random.choice(sexes, p=probabilities)

        new_record = {'user_id': row['user_id'], 'sex': guessed_sex}
        for field in ['weight', 'height', 'physical_level', 'practice_level',
'frequency_level', 'problem_arises',
                      'birth']:
            new_record[field] = means.at[guessed_sex, field]

        new_records.append(new_record)
```

```
    tqdm_iterator.update(1)
tqdm_iterator.close()

patient_profile_df = pd.concat([patient_profile_df, pd.DataFrame(new_records)],
ignore_index=True)

print(patient_profile_df.info())
patient_profile_df.to_pickle("data/patient_profile_data_prep.pkl")
```

analysis_session_anomaly.py - наліз Викидів session

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


session_df = pd.read_pickle("data/session_data_prep.pkl")


# Аналіз Викидів
# Викиди часто вказують на помилки в даних або незвичайні умови.
# Існує кілька методів для виявлення викидів, одним з найпоширеніших є використання
IQR.

# Визначення квартилів
Q1 = session_df['duration'].quantile(0.25)
Q3 = session_df['duration'].quantile(0.75)
IQR = Q3 - Q1

# Визначення меж викидів
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Фільтрація даних, що лежать за межами викидів
filtered_duration = session_df[(session_df['duration'] >= lower_bound) &
(session_df['duration'] <= upper_bound)]

# Повторення процесу для інтенсивності
Q1_int = session_df['intensity'].quantile(0.25)
Q3_int = session_df['intensity'].quantile(0.75)
IQR_int = Q3_int - Q1_int
lower_bound_int = Q1_int - 1.5 * IQR_int
upper_bound_int = Q3_int + 1.5 * IQR_int
filtered_intensity = session_df[(session_df['intensity'] >= lower_bound_int) &
(session_df['intensity'] <= upper_bound_int)]

# Кореляційний Аналіз
# Кореляційний аналіз допомагає виявити статистично значущі зв'язки між змінними.
```

```
# Використовуємо фільтровані дані для кореляційного аналізу
correlation_matrix = filtered_duration[['duration', 'intensity']].corr()

# Виводимо кореляційну матрицю
print(correlation_matrix)

# Візуалізація кореляційної матриці
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Кореляційна матриця між duration та intensity')
plt.show()
```

analysis_session_bodypart.py - Аналіз bodypart

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


session_df = pd.read_pickle("data/session_data_prep.pkl")


# Аналіз bodypart
# Групування даних за 'bodypart' і підрахунок кількості сесій для кожної частини тіла
bodypart_counts = session_df['bodypart'].value_counts()

# Візуалізація розподілу сесій за частинами тіла
plt.figure(figsize=(10, 6))
sns.barplot(x=bodypart_counts.values, y=bodypart_counts.index)
plt.title('Розподіл Сесій за Частинами Тіла')
plt.xlabel('Кількість Сесій')
plt.ylabel('Частина Тіла')
plt.show()
```

analysis_session_clusters.py - Кластеризація Користувачів за Тривалістю і Інтенсивністю Сесій

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from ml_tools import *

session_df = pd.read_pickle("data/session_data_prep.pkl")
# session_df = session_for_current_year(session_df)

# Кластеризація Користувачів за Тривалістю і Інтенсивністю Сесій
```

```
# Сегментація Користувачів:

# Підготовка даних для кластеризації
X = session_df[['duration', 'intensity']].copy()

# Нормалізація даних

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Вибір кількості кластерів
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X_scaled)

# Призначення кластерних міток
session_df['cluster'] = kmeans.labels_

# Візуалізація кластерів
plt.figure(figsize=(10, 6))
sns.scatterplot(x='duration', y='intensity', hue='cluster', data=session_df,
palette='viridis')
plt.title('Кластеризація Користувачів за Тривалістю і Інтенсивністю Сесій')
plt.show()

session_df.to_pickle("data/session_data_clusters.pkl")
```

analysis_session_clusters_bodypart.py - Візуалізація кількості сесій за вибраною частиною тіла у кожному кластері

```
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns
from ml_tools import *

session_df = pd.read_pickle("data/session_data_clusters.pkl")

# Візуалізація кількості сесій за вибраною частиною тіла у кожному кластері
plt.figure(figsize=(14, 7))
sns.countplot(x='bodypart', hue='cluster', data=session_df, palette='viridis')
plt.title('Розподіл Вибраних Частин Тіла для Тренувань по Кластерах')
plt.xlabel('Частина Тіла')
plt.ylabel('Кількість Сесій')
plt.xticks(rotation=90)  # Повертаємо підписи осі X для кращої видимості
plt.legend(title='Кластер')
plt.show()
```

analysis_session_duration.py - Розподіл тривалості сесій

```
import pandas as pd
```

```
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


session_df = pd.read_pickle("data/session_data_prep.pkl")


# Розподіл тривалості сесій
plt.figure(figsize=(10, 6))
sns.histplot(session_df['duration'], bins=50, kde=True)
plt.title('Розподіл Тривалості Сесій')
plt.xlabel('Тривалість')
plt.ylabel('Кількість')
plt.show()
```

analysis_session_hours.py - Часовий Аналіз

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


session_df = pd.read_pickle("data/session_data_prep.pkl")


# Часовий Аналіз
# Часовий аналіз може виявити закономірності в поведінці користувачів, які залежать
від часу.

# Створення нових часових змінних
session_df['hour'] = session_df['date'].dt.hour
session_df['day_of_week'] = session_df['date'].dt.dayofweek
session_df['month'] = session_df['date'].dt.month

# Аналіз розподілу сесій по годинах
hourly_distribution = session_df['hour'].value_counts().sort_index()

# Візуалізація
plt.figure(figsize=(14, 7))
sns.barplot(x=hourly_distribution.index, y=hourly_distribution.values)
plt.title('Розподіл Сесій по Годинах')
plt.xlabel('Година')
plt.ylabel('Кількість Сесій')
plt.show()
```

analysis_session_intencity.py - Розподіл інтенсивності

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


session_df = pd.read_pickle("data/session_data_prep.pkl")

# Розподіл інтенсивності
plt.figure(figsize=(10, 6))
sns.histplot(session_df['intensity'], bins=50, kde=True)
plt.title('Розподіл Інтенсивності')
plt.xlabel('Інтенсивність')
plt.ylabel('Кількість')
plt.show()
```

analysis_session_month.py - Аналіз по Місяцях

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


session_df = pd.read_pickle("data/session_data_prep.pkl")

# Детальний Аналіз Часових Змінних
# Аналіз по Місяцях
# Створення стовпця для місяця
session_df['month'] = session_df['date'].dt.month

# Групування даних за місяцем і підрахунок кількості сесій
month_counts = session_df['month'].value_counts().sort_index()

# Візуалізація розподілу сесій по місяцях
plt.figure(figsize=(12, 6))
sns.barplot(x=month_counts.index, y=month_counts.values)
plt.title('Розподіл Сесій по Місяцях')
plt.xlabel('Місяць')
plt.ylabel('Кількість Сесій')
plt.show()
```

analysis_session_user_correlations.py - Обчислення кореляцій між фізичними характеристиками та тренувальними показниками

```
import pandas as pd

import matplotlib
```

```
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns
from ml_tools import *

session_df = pd.read_pickle("data/session_data_prep.pkl")
user_df = pd.read_pickle("data/user_data_prep.pkl")
patient_profile_df = pd.read_pickle("data/patient_profile_data_prep.pkl")

# З'єднання session_df з user_df
session_user_df = session_df.merge(user_df, on='user_id', how='left')

# З'єднання отриманого датафрейму з patient_profile_df
full_df = session_user_df.merge(patient_profile_df, on='user_id', how='left')

# Обчислення кореляцій між фізичними характеристиками та тренувальними показниками
correlation_matrix = full_df[['duration', 'intensity', 'weight', 'height',
'physical_level', 'practice_level', 'frequency_level', 'problem_arises']].corr()

# Візуалізація кореляційної матриці
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Кореляційна Матриця Фізичних Характеристик та Тренувальних Показників')
plt.show()
```

analysis_session_week.py - Аналіз по Днях Тижня

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


session_df = pd.read_pickle("data/session_data_prep.pkl")

# Детальний Аналіз Часових Змінних
# Аналіз по Днях Тижня:
# Створення стовпця для дня тижня
session_df['day_of_week'] = session_df['date'].dt.dayofweek

# Групування даних за днем тижня і підрахунок кількості сесій
weekday_counts = session_df['day_of_week'].value_counts().sort_index()

# Візуалізація розподілу сесій по днях тижня
plt.figure(figsize=(10, 6))
sns.barplot(x=weekday_counts.index, y=weekday_counts.values)
plt.title('Розподіл Сесій по Днях Тижня')
plt.xlabel('День Тижня')
plt.ylabel('Кількість Сесій')
plt.xticks(range(7), ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Нд'])
plt.show()
```

analysis_user_lifecycle_duration.py - Розрахунок тривалості життєвого циклу користувача в додатку

```
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns

# Розрахунок тривалості життєвого циклу користувача в додатку з використанням даних
про реєстрацію та останню активність.


session_df = pd.read_pickle("data/session_data_prep.pkl")
user_df = pd.read_pickle("data/user_data_prep.pkl")
patient_profile_df = pd.read_pickle("data/patient_profile_data_prep.pkl")

# Об'єднання user_df з patient_profile_df
combined_df = user_df.merge(patient_profile_df, on='user_id', how='left')

# Знаходження останньої дати активності для кожного користувача
last_activity = session_df.groupby('user_id')['date'].max().reset_index()
last_activity.rename(columns={'date': 'last_activity_date'}, inplace=True)

# Об'єднання цієї інформації з основним датафреймом
combined_df = combined_df.merge(last_activity, on='user_id', how='left')

# Розрахунок тривалості життєвого циклу
combined_df['lifecycle_duration'] = (combined_df['last_activity_date'] -
combined_df['registered']).dt.days

# Видалення користувачів без відомої дати останньої активності
cleaned_df = combined_df.dropna(subset=['last_activity_date'])

# Перевірка результату видалення
print(cleaned_df.info())
print(cleaned_df.head())

cleaned_df.to_pickle("data/user_data_combined.pkl")


combined_df = pd.read_pickle("data/user_data_combined.pkl")

user_activity_df = pd.read_pickle("data/user_activity.pkl")
user_regularity_df = pd.read_pickle("data/user_regularity.pkl")

# Перетворення булевої ознаки regularity у числовий формат для кореляційного аналізу
user_regularity_df['regularity'] = user_regularity_df['regularity'].astype(int)

# Об'єднання user_activity_df та user_regularity_df з основним датафреймом combined_df
```

```
combined_df = combined_df.merge(user_activity_df, on='user_id', how='left')
combined_df = combined_df.merge(user_regularity_df, on='user_id', how='left')

# Видалення користувачів без відомої regularity
combined_df = combined_df.dropna(subset=['regularity'])

# Видалення користувачів без відомої activity_cluster
combined_df = combined_df.dropna(subset=['activity_cluster'])

# Обмеження датафрейму тільки числовими ознаками для кореляційної матриці
numerical_and_boolean_features = combined_df.select_dtypes(include=['float64',
'int64', 'int32', 'bool'])
numerical_and_boolean_features['lifecycle_duration'] =
combined_df['lifecycle_duration'] # додавання цільової змінної

# Розрахунок кореляційної матриці
correlation_matrix = numerical_and_boolean_features.corr()

# Візуалізація кореляційної матриці
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Розширена кореляційна матриця ознак')
plt.show()

combined_df.to_pickle("data/user_data_combined_lifecycle_duration.pkl")
```

model_user_activity.py - Класифікації користувачів за рівнем активності

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
import seaborn as sns


# Класифікації користувачів за рівнем активності (активні, помірно активні, пасивні)
# на основі даних про кількість та тривалість їх тренувань

session_df = pd.read_pickle("data/session_data_prep.pkl")
user_df = pd.read_pickle("data/user_data_prep.pkl")
patient_profile_df = pd.read_pickle("data/patient_profile_data_prep.pkl")

# Об'єднання session_df з user_df
combined_df = session_df.merge(user_df, on='user_id', how='left')

# Об'єднання combined_df з patient_profile_df
combined_df = combined_df.merge(patient_profile_df, on='user_id', how='left')
```

```
# Обрахунок загальної тривалості тренувань та кількості сесій для кожного користувача
activity_df =
combined_df.groupby('user_id').agg(total_duration=pd.NamedAgg(column='duration',
aggfunc='sum'),

session_count=pd.NamedAgg(column='duration', aggfunc='count')).reset_index()

# Нормалізація даних
scaler = StandardScaler()
X_scaled = scaler.fit_transform(activity_df[['total_duration', 'session_count']])

# Використання KMeans для кластеризації
kmeans = KMeans(n_clusters=3, random_state=42)
activity_df['activity_cluster'] = kmeans.fit_predict(X_scaled)

# Визначення міток активності на основі кластерів
activity_df['activity_level'] = activity_df['activity_cluster'].map({0: 'пасивні', 1:
'помірно активні', 2: 'активні'})

# Перевірка результатів класифікації
print(activity_df.head())

print(len(activity_df))
activity_df.to_pickle("data/user_activity.pkl")


# Побудова графіка розкиду для total_duration
plt.figure(figsize=(10, 6))
sns.boxplot(x='activity_level', y='total_duration', data=activity_df)
plt.title('Розподіл Загальної Тривалості Тренувань по Рівнях Активності')
plt.xlabel('Рівень Активності')
plt.ylabel('Загальна Тривалість Тренувань')
plt.show()

# Побудова графіка розкиду для session_count
plt.figure(figsize=(10, 6))
sns.boxplot(x='activity_level', y='session_count', data=activity_df)
plt.title('Розподіл Кількості Сесій по Рівнях Активності')
plt.xlabel('Рівень Активності')
plt.ylabel('Кількість Сесій')
plt.show()
```

model_user_regularity.py - Визначення регулярності використання додатку

```
import pandas as pd

import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# Визначення регулярності використання додатку
combined_df = pd.read_pickle("data/session_data_prep.pkl")

# Сортування даних за 'user_id' та 'date'
combined_df_sorted = combined_df.sort_values(by=['user_id', 'date'])

# Розрахунок часових інтервалів між сесіями
combined_df_sorted['previous_session'] =
combined_df_sorted.groupby('user_id')['date'].shift(1)
combined_df_sorted['interval'] = (combined_df_sorted['date'] -
combined_df_sorted['previous_session']).dt.days

# Видалення рядків, де інтервали відсутні (перші сесії кожного користувача)
combined_df_sorted.dropna(subset=['interval'], inplace=True)

# Класифікація користувачів на основі середнього інтервалу
threshold = 7  # Користувачі, які займаються раз на 7 днів
combined_df_sorted['regularity'] = combined_df_sorted['interval'].apply(lambda x:
'регулярно' if x <= threshold else 'нерегулярно')

# Групування даних для отримання загальної класифікації по кожному користувачу
regularity_classification =
combined_df_sorted.groupby('user_id')['regularity'].agg(lambda x: (x ==
'регулярно').all()).reset_index()

# Перевірка результатів класифікації
print(regularity_classification.head())

# Перевірка балансу класів
regularity_counts = regularity_classification['regularity'].value_counts()

print(len(regularity_classification))
regularity_classification.to_pickle("data/user_regularity.pkl")

# Візуалізація балансу класів
plt.figure(figsize=(8, 6))
sns.barplot(x=regularity_counts.index, y=regularity_counts.values)
plt.title('Розподіл Користувачів за Регулярністю Тренувань')
plt.xlabel('Регулярність Тренувань')
plt.ylabel('Кількість Користувачів')
plt.show()
```

model_user_lifecycle_duration_dtr.py - Прогнозування тривалості життєвого циклу користувача в додатку (Decision Tree Regression)

```
from ml_tools import *
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# Decision Tree Regression
model = DecisionTreeRegressor()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('MAE:', mean_absolute_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))

script_duration()
```

model_user_lifecycle_duration_gbr.py - Прогнозування тривалості життєвого циклу користувача в додатку (Gradient Boosting Regressor)

```
from ml_tools import *
from scipy.stats import uniform, randint
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)
```

```
# Gradient Boosting Regressor
# Створення моделі градієнтного бустінгу
gbr = GradientBoostingRegressor(random_state=42)

# Визначення простору гіперпараметрів для RandomizedSearchCV
param_dist = {
    'n_estimators': randint(50, 150),
    'max_depth': randint(3, 10),
    'min_samples_split': randint(2, 10),
    'min_samples_leaf': randint(1, 10),
    'learning_rate': uniform(0.01, 0.19)
}

# Проведення випадкового пошуку
random_search = RandomizedSearchCV(
    estimator=gbr, param_distributions=param_dist,
    n_iter=10, cv=3, random_state=18, n_jobs=-1
)

# Тренування моделі з випадковим пошуком
random_search.fit(X_train, y_train)

# Виведення кращих параметрів
best_params = random_search.best_params_
print("Кращі параметри:", best_params)

# Оцінка моделі з кращими параметрами
best_model = random_search.best_estimator_
predictions = best_model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
rmse = mse ** 0.5
print(f'Середньоквадратична помилка (RMSE): {rmse}')

model_gdbr = GradientBoostingRegressor(
    learning_rate=0.044549642820591547,
    max_depth=5,
    min_samples_leaf=9,
    min_samples_split=2,
    n_estimators=148,
    random_state=42
)

model_gdbr.fit(X_train, y_train)
save_model('user_lifecycle_duration_gdbr.joblib', model_gdbr)
```

model_user_lifecycle_duration_lasorr.py - Прогнозування тривалості життєвого циклу користувача в додатку (Lasso/Ridge Regression)

```
from ml_tools import *
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
import numpy as np
from sklearn.linear_model import Lasso, Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# Lasso/Ridge Regression
lasso = Lasso()
lasso.fit(X_train, y_train)
ridge = Ridge()
ridge.fit(X_train, y_train)

y_pred_lasso = lasso.predict(X_test)
y_pred_ridge = ridge.predict(X_test)

print('Lasso MAE:', mean_absolute_error(y_test, y_pred_lasso))
print('Ridge MAE:', mean_absolute_error(y_test, y_pred_ridge))

print('Lasso RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_lasso)))
print('Ridge RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_ridge)))

print('Lasso R^2:', r2_score(y_test, y_pred_lasso))
print('Ridge R^2:', r2_score(y_test, y_pred_ridge))

alphas = [0.001, 0.01, 0.1, 1, 10, 100]
grid = {'alpha': alphas}
grid_search = GridSearchCV(Ridge(), grid, cv=5, scoring='neg_mean_squared_error')

grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
print(np.sqrt(-grid_search.best_score_))

script_duration()
```

model_user_lifecycle_duration_lgb.py - Прогнозування тривалості життєвого циклу користувача в додатку (LightGBM)

```
from ml_tools import *
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.model_selection import GridSearchCV
import lightgbm as lgb
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# LightGBM
model = lgb.LGBMRegressor()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('MAE:', mean_absolute_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))

params = {'num_leaves': [10, 20, 31],
          'max_depth': [5, 8, 15],
          'n_estimators': [100, 250, 500],
          'learning_rate': [0.01, 0.05, 0.1]}

gs = GridSearchCV(lgb.LGBMRegressor(), params, cv=3)
gs.fit(X_train, y_train)

print(gs.best_params_)
print(gs.best_score_)

model = lgb.LGBMRegressor(learning_rate=0.01, max_depth=8, n_estimators=500,
num_leaves=31)

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print('MAE:', mean_absolute_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))

script_duration()
```

model_user_lifecycle_duration_lr.py - Прогнозування тривалості життєвого циклу користувача в додатку (Linear Regression)

```
from ml_tools import *
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('MAE:', mean_absolute_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))

script_duration()
```

model_user_lifecycle_duration_rfr.py - Прогнозування тривалості життєвого циклу користувача в додатку (Random Forest Regressor)

```
from ml_tools import *
from scipy.stats import uniform, randint
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
```

```
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# Random Forest Regressor
# Створення моделі випадкового лісу
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Тренування моделі
model.fit(X_train, y_train)

# Прогнозування на тестовому наборі
predictions = model.predict(X_test)

# Оцінка моделі
mse = mean_squared_error(y_test, predictions)
rmse = mse ** 0.5

print(f'Середньоквадратична помилка (RMSE): {rmse}')


param_dist = {
    'n_estimators': randint(100, 200),  # Зменшений діапазон для n_estimators
    'max_depth': [3, 5, 10, None],  # Конкретний список можливих значень для max_depth
    'min_samples_split': randint(2, 10),  # Зменшений діапазон для min_samples_split
    'min_samples_leaf': randint(1, 10),  # Зменшений діапазон для min_samples_leaf
    'bootstrap': [True, False]
}

# Ініціалізація моделі
rf = RandomForestRegressor(random_state=42)

# Ініціалізація RandomizedSearchCV
random_search = RandomizedSearchCV(rf, param_distributions=param_dist, n_iter=10,
cv=5, random_state=42)

# Проведення оптимізації
random_search.fit(X_train, y_train)

# Виведення кращих параметрів
print("Кращі параметри:", random_search.best_params_)

# Оцінка моделі з кращими параметрами
best_rf = random_search.best_estimator_
predictions = best_rf.predict(X_test)
rmse = mean_squared_error(y_test, predictions, squared=False) # squared=False повертає
RMSE
print(f"Оптимізована RMSE: {rmse}")

script_duration()
```

model_user_lifecycle_duration_stacking.py - Прогнозування тривалості життєвого циклу користувача в додатку (Stacking XGBoost and LightGBM)

```
from ml_tools import *

import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np

from sklearn.linear_model import LinearRegression
import xgboost as xgb
import lightgbm as lgb


# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# Stacking XGBoost and LightGBM
from sklearn.model_selection import KFold

# Разбиваем данные на фолды
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Обучаем базовые модели с использованием кросс-валидации
xgb_preds = np.zeros((X_train.shape[0]))
lgb_preds = np.zeros((X_train.shape[0]))

for fold, (train_idx, val_idx) in enumerate(kf.split(X_train)):
    X_tr, X_val = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_tr, y_val = y_train.iloc[train_idx], y_train.iloc[val_idx]

    xgb_model = xgb.XGBRegressor()
    xgb_model.fit(X_tr, y_tr)
    xgb_preds[val_idx] = xgb_model.predict(X_val)

    lgb_model = lgb.LGBMRegressor()
    lgb_model.fit(X_tr, y_tr)
    lgb_preds[val_idx] = lgb_model.predict(X_val)

# Обучаем финальную модель
stack_model = LinearRegression()
```

```
stack_model.fit(np.c_[xgb_preds, lgb_preds], y_train)

# Делаем предсказания на тесте
xgb_test_preds = xgb_model.predict(X_test)
lgb_test_preds = lgb_model.predict(X_test)

y_pred = stack_model.predict(np.c_[xgb_test_preds, lgb_test_preds])

model_metrics(y_test, y_pred)

script_duration()
```

model_user_lifecycle_duration_svr.py - Прогнозування тривалості життєвого циклу користувача в додатку (Support Vector Regression)

```
from ml_tools import *
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# Support Vector Regression (SVR)
regressor = SVR(kernel='linear')
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)

print('MAE:', mean_absolute_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))
script_duration()
```

model_user_lifecycle_duration_xgb.py - Прогнозування тривалості життєвого циклу користувача в додатку (XGBoost)

```
from ml_tools import *
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
```

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Прогнозування тривалості життєвого циклу користувача в додатку з
# використанням даних про реєстрацію та останню активність.
combined_df = pd.read_pickle("data/user_data_combined_lifecycle_duration.pkl")

# Вибір ознак та цільової змінної
features = combined_df[['regularity', 'session_count', 'total_duration']]
target = combined_df['lifecycle_duration']

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=13)

# XGBoost
model = xgb.XGBRegressor()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('MAE:', mean_absolute_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))

params = {'max_depth': [3, 5, 7],
          'n_estimators': [100, 200, 300],
          'learning_rate': [0.1, 0.05, 0.01]}

gs = GridSearchCV(xgb.XGBRegressor(), params, cv=5, n_jobs=-1,
scoring='neg_root_mean_squared_error')

gs.fit(X_train, y_train)

print(gs.best_params_)
print(np.sqrt(-gs.best_score_))

model = xgb.XGBRegressor(max_depth=5, n_estimators=200, learning_rate=0.05)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('MAE:', mean_absolute_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))


script_duration()
```

model_user_program_recommend_item_based.py - Надання рекомендацій щодо оптимальних електростимуляційних програм (Item-based)

```
from ml_tools import *
from scipy import spatial

sessions_df = load_df('session_data_prep')
sessions_df = session_for_current_year(sessions_df)

# Створити нову ознаку - комбінація програми і частини тіла
sessions_df['program_bodypart'] = sessions_df['program'].astype(str) + '_' +
sessions_df['bodypart'].astype(str)


# Для формування матриці "користувач-програма/частина_тіла" нам знадобляться дані про
те, які програми і
# скільки разів запускав кожен користувач. Ця інформація є в датасеті sessions_df у
стовпцях user_id,
# program та n_uses.
usage_data = sessions_df.groupby(['user_id',
'program_bodypart'])['program_bodypart'].count().reset_index(name='n_uses')
print(usage_data.head())

# перетворимо це у матрицю
user_item_matrix = usage_data.pivot(index='user_id',
                                columns='program_bodypart',
                                values='n_uses').fillna(0)

# Надання рекомендацій щодо оптимальних електростимуляційних програм (Item-based)

# Функція обчислення косинусної схожості
def cosine_similarity(vec1, vec2):
    return 1 - spatial.distance.cosine(vec1, vec2)


# Отримати топ схожих програм
def get_similar_items(item_id, n=5):
    user_vectors = user_item_matrix[user_item_matrix[item_id] > 0]
    similarities = []

    for item in user_item_matrix.columns:
        if item != item_id:
            vec1 = user_vectors[item_id]
            vec2 = user_vectors[item]
            sim = cosine_similarity(vec1, vec2)

            similarities.append((item, sim))

    return sorted(similarities, key=lambda x: x[1], reverse=True)[:n]


# Рекомендації
def get_item_based_recs(user_id, n=5):
```

```
    user_items = user_item_matrix.loc[user_id][user_item_matrix.loc[user_id] >
0].index

    recs = {}
    for item in user_items:
        similar_items = get_similar_items(item, 10)
        for si, sim in similar_items:
            if si not in user_items:
                recs[si] = recs.get(si, 0) + sim

    recs_sorted = sorted(recs.items(), key=lambda x: x[1], reverse=True)
    rec_programs_sorted_list = list(map(lambda x: x[0], recs_sorted))[:n]
    print(f"Recommended programs sorted: {rec_programs_sorted_list}")
    return rec_programs_sorted_list


# Функція для обрізки даних користувача
def slice_user_data(user_id, slice_ratio=0.8):
    user_data = user_item_matrix.loc[user_id]
    non_zero = user_data[user_data != 0]

    # Випадково вибираємо частину даних для видалення
    num_to_remove = int(len(non_zero) * slice_ratio)
    idx_to_remove = np.random.choice(non_zero.index, size=num_to_remove,
replace=False)

    user_data.loc[idx_to_remove] = 0

    return user_data


# Вибір тестового користувача
test_user_id = np.random.choice(user_item_matrix.index)
print(f'Test user: {test_user_id}')
used_programs = get_used_programs(test_user_id, user_item_matrix)
recommend_programs = get_item_based_recs(test_user_id, n=5)
if len(recommend_programs):
    calculate_metrics_PRF(recommend_programs, used_programs)
else:
    print(f"Sorry, I can't give you recommendations for the user {test_user_id}")



'''

# Вибір двох тестових
test_users = np.random.choice(user_item_matrix.index, size=10)

# Зберігаємо повні дані для тестових юзерів
full_data = user_item_matrix.loc[test_users].copy()

precision_values = []
```

```
recall_values = []
f1_values = []

# Обрізка їх даних на 80%
for user_id in test_users:
    print(f"user_id => {user_id}")
    print(get_non_zero(user_item_matrix.loc[user_id]))
    print("----")
    print(get_non_zero(slice_user_data(user_id)))
    user_item_matrix.loc[user_id] = slice_user_data(user_id)

    user_recs = get_item_based_recs(user_id, n=5)
    print(user_recs)

    # Відновлюємо повні дані для розрахунку метрик
    user_item_matrix.loc[user_id] = full_data.loc[user_id]

    user_profile = user_item_matrix.loc[user_id][user_item_matrix.loc[user_id] !=
0].index

    # Розрахунок метрик
    precision = len(set(user_profile) & set(user_recs)) / len(user_recs)
    recall = len(set(user_profile) & set(user_recs)) / len(user_profile)
    if precision + recall == 0:
        f1 = 0
    else:
        f1 = 2 * precision * recall / (precision + recall)

    precision_values.append(precision)
    recall_values.append(recall)
    f1_values.append(f1)


print(f'Average precision: {np.mean(precision_values)}')
print(f'Average recall: {np.mean(recall_values)}')
print(f'Average F1: {np.mean(f1_values)}')
```

model_user_program_recommend_user_based.py - Надання рекомендацій щодо оптимальних електростимуляційних програм (User-based)

```
from ml_tools import *
from scipy import spatial

sessions_df = load_df('session_data_prep')
sessions_df = session_for_current_year(sessions_df)

# Створити нову ознаку - комбінація програми і частини тіла
sessions_df['program_bodypart'] = sessions_df['program'].astype(str) + '_' +
sessions_df['bodypart'].astype(str)
```

```
# Для формування матриці "користувач-програма/частина_тіла" нам знадобляться дані про
те, які програми і
# скільки разів запускав кожен користувач. Ця інформація є в датасеті sessions_df у
стовпцях user_id,
# program та duration.
usage_data = sessions_df.groupby(['user_id',
'program_bodypart'])['program_bodypart'].count().reset_index(name='n_uses')
user_item_matrix = usage_data.pivot(index='user_id',
                                columns='program_bodypart',
                                values='n_uses').fillna(0)



# Надання рекомендацій щодо оптимальних електростимуляційних програм (User-based)

def get_user_similarity(user1, user2):
    '''
    Функція розрахунку схожості між користувачами за дпомогою косинусної міри
    :param user1:
    :param user2:
    :return:
    '''
    vector1 = user_item_matrix.loc[user1].values
    vector2 = user_item_matrix.loc[user2].values
    similarity = 1 - spatial.distance.cosine(vector1, vector2)
    return similarity


def get_similar_users(user_id, num_users=10):
    '''
    функцію для отримання списку найбільш схожих користувачів на основі розрахованої
міри схожості
    :param user_id: ідентифікатор користувача
    :param num_users: кількість потрібних схожих користувачів
    :return: повертає num_users найбільш схожих.
    '''
    similar_users = []
    for other_user in user_item_matrix.index:
        if other_user != user_id:
            similarity = get_user_similarity(user_id, other_user)
            similar_users.append((other_user, similarity))

    similar_users = sorted(similar_users, key=lambda x: x[1],
reverse=True)[:num_users]
    return similar_users


def evaluate_recs(user_id, recs):
    '''
    Розрахунок метрік для оцінки якості алгоритму user-based рекомендацій
    :param user_id:
    :param recs:
```

```
    :return:
    '''
    user_profile = user_item_matrix.loc[user_id].replace(0, np.nan).dropna()

    relevant_programs = set(user_profile.index.tolist())
    recommended_programs = set(recs)
    print('relevant_programs => ', relevant_programs)
    print('recommended_programs => ', recommended_programs)

    precision = len(relevant_programs & recommended_programs) /
len(recommended_programs)
    recall = len(relevant_programs & recommended_programs) / len(relevant_programs)
    if precision + recall == 0:
        f1 = 0
    else:
        f1 = 2 * precision * recall / (precision + recall)

    print(f'Precision: {precision:.3f}')
    print(f'Recall: {recall:.3f}')
    print(f'F1 score: {f1:.3f}')


# Алгоритм user-based рекомендацій:
# 1) Для конкретного користувача знайти список найбільш схожих (використовуємо функцію
get_similar_users)
# 2) Для кожного схожого користувача взяти програми, які він використовував, але яких
НЕмає в профілі цільового користувача
# 3) Зважити ці програми на значення схожості даного користувача з цільовим
# 4) Об'єднати та відсортувати програми-кандидати по зваженій сумі і повернути топ-N


K = 20  # кількість схожих користувачів
N = 10  # кількість рекомендацій


def get_user_based_recs(user_id):
    '''
    Функція отримання рекомендацій
    :param user_id:
    :return:
    '''
    # Знаходимо K подібних користувачів
    similar_users = get_similar_users(user_id, K)
    # print(f'Similar users: {similar_users}')

    # Словник для рейтингів програм
    rec_programs = {}

    target_user = user_item_matrix.loc[user_id]
    # target_user_no_zero = target_user.dropna()
    target_user_no_zero = target_user[target_user != 0]
    target_user_programs = target_user_no_zero.index.tolist()
```

```
    # Перегляд схожих користувачів
    for user, similarity in similar_users:
        # Отримуємо програми цього користувача
        current_user = user_item_matrix.loc[user]
        current_user_non_zero = current_user[current_user != 0]
        # current_user_non_zero = current_user.dropna()

        user_programs = current_user_non_zero.index.tolist()

        # Враховуємо тільки нові для цільового
        new_programs = set(user_programs).difference(set(target_user_programs))

        for program in new_programs:
            rating = similarity * user_item_matrix.at[user, program]
            rec_programs[program] = rec_programs.get(program, 0) + rating

    # Сортуємо і повертаємо ТОП N
    # print(f"Recommended programs: {rec_programs}")
    rec_programs_sorted = sorted(rec_programs.items(),
                                 key=lambda x: x[1],
                                 reverse=True)
    #print(f"Recommended programs sorted: {rec_programs_sorted}")
    rec_programs_sorted_list = list(map(lambda x: x[0], rec_programs_sorted))
    print(f"Recommended programs sorted: {rec_programs_sorted_list}")
    return rec_programs_sorted_list[:N]


# Вибір тестового користувача
test_user_id = np.random.choice(user_item_matrix.index)
print(f'Test user: {test_user_id}')
used_programs = get_used_programs(test_user_id, user_item_matrix)
recommend_programs = get_user_based_recs(test_user_id)
if len(recommend_programs):
    calculate_metrics_PRF(recommend_programs, used_programs)
else:
    print(f"Sorry, I can't give you recommendations for the user {test_user_id}")
```

model_user_program_recommend_with_cluster.py - Надання рекомендацій щодо оптимальних електростимуляційних програм (Hybrid algorithm with cluster)

```
from ml_tools import *
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from collections import defaultdict
from sklearn.metrics.pairwise import pairwise_distances

sessions_df = load_df('session_data_prep')
sessions_df = session_for_current_year(sessions_df)

# Створити нову ознаку - комбінація програми і частини тіла
```

```
sessions_df['program_bodypart'] = sessions_df['program'].astype(str) + '_' +
sessions_df['bodypart'].astype(str)

# Для формування матриці "користувач-програма/частина_тіла" нам знадобляться дані про
те, які програми і
# скільки разів запускав кожен користувач. Ця інформація є в датасеті sessions_df у
стовпцях user_id,
# program та n_uses.
usage_data = sessions_df.groupby(['user_id',
'program_bodypart'])['program_bodypart'].count().reset_index(name='n_uses')
print(usage_data.head())

# перетворимо це у матрицю
user_item_matrix = usage_data.pivot(index='user_id',
                                    columns='program_bodypart',
                                    values='n_uses').fillna(0)

# Надання рекомендацій щодо оптимальних електростимуляційних програм (Hybrid algorithm
with cluster)

# Масштабування даних
scaler = StandardScaler()
scaled_vectors = scaler.fit_transform(user_item_matrix)


# Визначаємо функцію для оцінки розподілу користувачів по кластерах
def evaluate_cluster_balance(random_state, data, n_clusters=2):
    kmeans = KMeans(n_clusters=n_clusters, random_state=random_state)
    labels = kmeans.fit_predict(data)
    _, counts = np.unique(labels, return_counts=True)
    return np.std(counts)  # Використовуємо стандартне відхилення як міру
рівномірності розподілу


def find_the_best_random_state(data, n_clusters=2, max_random_state=100):
    # Цикл для пошуку оптимального random_state
    best_state = None
    best_balance = float('inf')
    tqdm_iterator = tqdm(total=max_random_state, desc="Processing", unit="row")
    for state in range(max_random_state):  # діапазон перебору
        balance = evaluate_cluster_balance(state, data, n_clusters)
        if balance < best_balance:
            best_balance = balance
            best_state = state
        tqdm_iterator.update(state)
    tqdm_iterator.close()
    print(f"Best random_state: {best_state} with balance score: {best_balance}")
    return best_state

#find_the_best_random_state(scaled_vectors)

# Кластеризатор
```

```
kmeans = KMeans(n_clusters=2, random_state=31)

# Навчання та предикція кластерів
clusters = kmeans.fit_predict(scaled_vectors)

clusters2, counts = np.unique(clusters, return_counts=True)
print(dict(zip(clusters2, counts)))

# Аналіз центроїдів
top_programs = defaultdict(list)
for i, row in enumerate(kmeans.cluster_centers_):
    top_prog = user_item_matrix.columns[row.argsort()[-5:]]
    top_programs[i].extend(top_prog)

print(f"Top programs: {top_programs}")


def get_user_cluster(user_id):
    # Отримання індексу та відповідного кластеру для user_id
    user_index = user_item_matrix.index.get_loc(user_id)
    user_cluster = clusters[user_index]
    return user_cluster


def find_similar_users(user_id, n, user_item_matrix, clusters):
    # Перевірка наявності user_id у DataFrame
    if user_id in user_item_matrix.index:
        # Використання індексу DataFrame для отримання відповідного кластеру
        user_cluster = get_user_cluster(user_id)

        # Обчислення косинусних відстаней
        distances = pairwise_distances(user_item_matrix,
user_item_matrix.loc[user_id].values.reshape(1, -1), metric='cosine')

        # Вибір користувачів з того ж кластера
        same_cluster_indices = np.where(clusters == user_cluster)[0]

        # Фільтрація відстаней для користувачів у тому ж кластері
        same_cluster_distances = distances[same_cluster_indices]

        # Вибір топ-N найближчих користувачів (виключаючи самого користувача)
        top_indices = np.argsort(same_cluster_distances.flatten())[1:n+1]

        # Повернення ID схожих користувачів
        return user_item_matrix.index[same_cluster_indices[top_indices]]
    else:
        print(f"Not found user with user_id: {user_id}")
        return []


def get_recommend_programs(user_id, n=5):
    similar_users = find_similar_users(user_id, n, user_item_matrix, clusters)
```

```
    # print(similar_users)
    # Рекомендація програм
    recommended_programs = defaultdict(int)
    for user in similar_users:
        for program in user_item_matrix.columns[user_item_matrix.loc[user] > 0]:
            recommended_programs[program] += 1

    recommended_programs = sorted(recommended_programs.items(), key=lambda x: x[1],
reverse=True)
    recommended_program_names = [program[0] for program in recommended_programs[:n]]

    #print("Recommended Programs:", recommended_program_names[:n])
    return recommended_program_names


random_users = user_item_matrix.sample(n=1000).index.tolist()

precision_values = []
recall_values = []
f1_values = []
for user_id in random_users:
    # print(f"user_id: {user_id}")
    used_programs = get_used_programs(user_id, user_item_matrix)
    recommend_programs = get_recommend_programs(user_id, n=10)
    precision, recall, f1 = calculate_metrics_PRF(recommend_programs, used_programs)
    precision_values.append(precision)
    recall_values.append(recall)
    f1_values.append(f1)
    # print("--------------------------------------------------")

print(f'Average precision: {np.mean(precision_values)}')
print(f'Average recall: {np.mean(recall_values)}')
print(f'Average F1: {np.mean(f1_values)}')
```

model_user_program_recommend_with_knn.py - Надання рекомендацій щодо оптимальних електростимуляційних програм (Aadopted algorithm with KNN)

```
from ml_tools import *
from collections import defaultdict
from sklearn.neighbors import NearestNeighbors

sessions_df = load_df('session_data_prep')
sessions_df = session_for_current_year(sessions_df)

# Створити нову ознаку - комбінація програми і частини тіла
sessions_df['program_bodypart'] = sessions_df['program'].astype(str) + '_' +
sessions_df['bodypart'].astype(str)

# Для формування матриці "користувач-програма/частина_тіла" нам знадобляться дані про
те, які програми і
```

```
# скільки разів запускав кожен користувач. Ця інформація є в датасеті sessions_df у
стовпцях user_id,
# program та n_uses.
usage_data = sessions_df.groupby(['user_id',
'program_bodypart'])['program_bodypart'].count().reset_index(name='n_uses')

# перетворимо це у матрицю
user_item_matrix = usage_data.pivot(index='user_id',
                                    columns='program_bodypart',
                                    values='n_uses').fillna(0)

# Надання рекомендацій щодо оптимальних електростимуляційних програм (Aadopted
algorithm with KNN)

def kNN_recommended_programs(user_id, user_item_matrix, n_neighbors=5,
top_n_programs=5):
    # Ініціалізація моделі k-NN
    model_knn = NearestNeighbors(metric='cosine', algorithm='brute',
n_neighbors=n_neighbors, n_jobs=-1)
    model_knn.fit(user_item_matrix)

    # Знаходження k найближчих сусідів
    distances, indices =
model_knn.kneighbors(user_item_matrix.loc[user_id].values.reshape(1, -1),
n_neighbors=n_neighbors)

    # Агрегування популярності програм
    program_popularity = defaultdict(int)
    for idx in indices.flatten():
        for program in user_item_matrix.columns[(user_item_matrix.iloc[idx] > 0)]:
            program_popularity[program] += 1

    # Вибір топ-n найбільш популярних програм
    recommended_programs = sorted(program_popularity.items(), key=lambda x: x[1],
reverse=True)[:top_n_programs]
    recommended_programs = [program[0] for program in recommended_programs]

    return recommended_programs


random_users = user_item_matrix.sample(n=1000).index.tolist()
precision_values = []
recall_values = []
f1_values = []
for user_id in random_users:
    used_programs = get_used_programs(user_id, user_item_matrix)
    recommend_programs = kNN_recommended_programs(user_id, user_item_matrix,
top_n_programs=10)
    precision, recall, f1 = calculate_metrics_PRF(recommend_programs, used_programs)
    precision_values.append(precision)
    recall_values.append(recall)
    f1_values.append(f1)
```

```
print(f'Average precision: {np.mean(precision_values)}')
print(f'Average recall: {np.mean(recall_values)}')
print(f'Average F1: {np.mean(f1_values)}')
```