# НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

**Факультет інформаційних технологій**

| **ПОГОДЖЕНО** | **ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** |
|---|---|
| **Декан факультету (Директор ННІ)** | **Завідувач кафедри** |
| ____інформаційних технологій___ | _____комп'ютерних наук_____ |
| (назва факультету (ННІ)) | (назва кафедри) |
| ____Ігор Болбот_____ | _____Белла Голуб_____ |
| "___" _____ 2025 р. | "___" _____ 2025 р. |

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

**на тему** __Система аналізу та прогнозування спортивних досягнень на основі даних про_____ тренування_____

Спеціальність__122 - Комп'ютерні науки_____

(код і найменування)

Освітня програма___Інформаційно управляючі системи та технології_____

(назва)

Орієнтація освітньої програми_____освітньо-професійна_____

(освітньо-професійна або освітньо-наукова)

**Гарант освітньої програми**

___к.т.н., доцент ._____ _____ _____Белла Голуб_____

(науковий ступінь та вчене звання) (підпис) (Ім'я Прізвище)

**Керівник магістерської кваліфікаційної роботи**

_____ст. викладач_____ _____ ___Віктор Панкратьєв_____

(науковий ступінь та вчене звання) (підпис) (Ім'я Прізвище)

**Консультант магістерської кваліфікаційної роботи**

_____к.т.н., доцент_____ _____ _____Тарас Лендєл_____

(науковий ступінь та вчене звання) (підпис) (Ім'я Прізвище)

**Виконав** _____ __Федяй Артем_____

(підпис) (ПІБ студента)

**КИЇВ – 2025**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ**
**І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет (ННІ)** __інформаційних технологій_____

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**__комп'ютерних наук_
___доцент, к.т.н._____ _____ Белла Голуб._
(науковий ступінь, вчене звання) (підпис) (Ім'я Прізвище)
"_01_" ___листопада_____2024 року

**З А В Д А Н Н Я**

**ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ**

Федяю Артему Ігоровичу
(прізвище, ім'я, по батькові)

Спеціальність 122 «Комп'ютерні науки»
(код і назва)

Освітня програма ____Інформаційні управляючі системи та технології
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____

Тема магістерської кваліфікаційної роботи: Система аналізу та прогнозування спортивних досягнень на основі даних про тренування

затверджена наказом ректора НУБіП України від "_01_"_ листопада___2024р. №1964 «С»
Термін подання завершеної роботи на кафедру __01.12.2025_____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: дані про тренування спортсменів (тривалість, інтенсивність, тип, показники прогресу), персональні характеристики (вік, стать, зріст, вага, рівень підготовки), а також інформація про вправи (назва, підходи, повторення, вага). Програмна частина реалізується мовою C# з використанням Microsoft SQL Server та фреймворків ML.NET.

Перелік питань, що підлягають дослідженню:

1) Аналіз сучасних методів збору та обробки спортивних даних, включаючи формування асоціативних правил і виявлення залежностей між параметрами тренувань.
2) Застосування алгоритмів машинного навчання для класифікації спортсменів, кластеризації даних і прогнозування спортивних результатів.
3) Оцінка точності побудованих моделей та визначення практичної ефективності розробленої системи.

Перелік графічного матеріалу (за потреби)

**Дата видачі завдання** "__01__" __листопада_____2024_ р.

**Керівник магістерської кваліфікаційної роботи** _____ __Віктор_Панкратьєв____
( підпис ) (прізвище та ініціали)

**Консультант магістерської кваліфікаційної роботи** _____ __Тарас Лендєл____
( підпис ) (прізвище та ініціали)

**Завдання прийняв до виконання** _____ _____Артем_Федяй_____
(підпис ) (прізвище та ініціали студента)

# ЗМІСТ

# СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ML - Machine Learning (машинне навчання)

ML.NET - Microsoft Machine Learning .NET (крос-платформний фреймворк для ML)

SQL - Structured Query Language (мова структурованих запитів)

UML - Unified Modeling Language (уніфікована мова моделювання)

KMeans - K-середніх (алгоритм кластеризації)

MSE - Mean Squared Error (середня квадратична похибка)

$R^2$ - Коефіцієнт детермінації (метрика якості регресійної моделі)

БД - База Даних

WinForms - Windows Forms (технологія створення настільних додатків)

# ВСТУП

**Актуальність теми дослідження.** Сучасний спорт неможливо уявити без використання цифрових технологій, які дозволяють не лише фіксувати результати, а й аналізувати динаміку підготовки спортсмена [1, 3]. Для досягнення високих результатів недостатньо лише інтенсивних тренувань — важливим стає глибокий кількісний аналіз даних, що відображають ефективність фізичних навантажень.

Більшість спортивних організацій і окремих спортсменів все ще застосовують фрагментарні підходи до обліку тренувальних даних, використовуючи звичайні електронні таблиці або паперові журнали. Такий підхід не дозволяє отримати комплексну оцінку підготовки, створює ризик суб'єктивності при ухваленні рішень та унеможливлює довгострокове прогнозування результатів.

У межах даного дослідження пропонується використання технологій машинного навчання (ML.NET) [2] для автоматизації збору, зберігання та аналітичної обробки даних про тренування спортсменів. Розробка системи аналізу та прогнозування спортивних досягнень є актуальною, оскільки забезпечує об'єктивність і точність оцінювання навантажень, дозволяє зменшити вплив людського фактора та сприяє підвищенню ефективності тренувального процесу.

**Об'єктом дослідження** є процес збору, обробки та аналізу даних про тренувальну діяльність спортсменів. До цього процесу входять реєстрація основних параметрів тренувань (кількість підходів, повторень, тривалість, інтенсивність), формування бази історичних даних та їх використання для побудови аналітичних висновків і прогнозів. На практиці цей процес часто виконується вручну, що збільшує кількість помилок і знижує точність оцінювання.

**Предметом дослідження** є програмна система, яка автоматизує аналіз і прогнозування спортивних досягнень із використанням алгоритмів машинного навчання. Вона охоплює методи регресійного аналізу (ML.NET), алгоритми кластеризації (KMeans) для групування спортсменів за схожими профілями навантажень, а також структуру бази даних MS SQL Server для ефективного зберігання результатів. Дослідження спрямоване на підвищення точності, швидкості та надійності аналізу тренувальних даних шляхом використання сучасних алгоритмічних підходів і оптимізації процесів прийняття рішень.

**Метою дослідження** є створення автоматизованої системи, яка поєднує збір тренувальних даних, інтелектуальний аналіз і прогнозування показників спортивних результатів на основі алгоритмів машинного навчання [20, 24]. Для перевірки ефективності запропонованого підходу реалізовано експериментальну програмну модель, яка оцінює точність прогнозів і стабільність виявлених закономірностей.

Для досягнення поставленої мети передбачено виконання таких **завдань**:

1. Провести аналіз існуючих підходів до збору та обробки даних у спортивній аналітиці;
2. Визначити вимоги до системи аналізу та прогнозування спортивних результатів;
3. Дослідити можливості бібліотеки ML.NET для реалізації алгоритмів регресії та кластеризації;
4. Побудувати UML-модель предметної області та спроєктувати структуру бази даних MS SQL Server;
5. Реалізувати програмну систему на C# (WinForms) з інтегрованими ML.NET-модулями для обробки тренувальних даних;
6. Провести тестування системи, оцінити точність прогнозування (MSE, $R^2$) і зробити висновки щодо доцільності використання запропонованої архітектури.

**Методологічна база** роботи поєднує системний аналіз, методи обробки даних, програмну інженерію та алгоритми машинного навчання. На першому етапі було проведено системний аналіз існуючих процесів зберігання та аналізу спортивних даних, що дозволило сформулювати вимоги до майбутньої системи. Для прогнозування результативності застосовано алгоритми регресійного моделювання з бібліотеки ML.NET, які забезпечують точне відтворення залежностей між параметрами тренувань. Для ідентифікації стабільних груп спортсменів використано метод кластеризації KMeans, який дозволяє виділити типові профілі навантажень за середніми показниками. Таким чином, поєднання системного підходу, алгоритмів машинного навчання та реляційного зберігання даних забезпечує комплексне вирішення завдання аналітики у спортивній підготовці.

**Наукова новизна** полягає у створенні інтегрованої системи аналізу спортивних даних, у якій поєднано методи регресійного прогнозування та кластеризації. Запропоновано підхід до типізації спортсменів на основі середніх показників навантажень (AvgIntensity, AvgDuration, AvgReps), що дозволяє виявляти приховані закономірності у тренувальному процесі. Удосконалено процес аналізу через обчислення кореляційних і варіаційних характеристик, що забезпечує точніше виявлення взаємозв’язків між параметрами тренувань. Інтеграція ML.NET із реляційною базою даних MS SQL Server дозволила суттєво скоротити час обробки інформації та підвищити ефективність аналізу. Отримані результати мають як практичну, так і наукову цінність — вони демонструють можливість використання інтелектуальних алгоритмів у спортивній аналітиці без необхідності дорогого обладнання чи сторонніх сервісів.

**Структура роботи** викладена на 84 сторінках, містить 28 рисунків, 2 таблиці та 36 джерела у списку літератури. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У вступі

обґрунтовано актуальність теми, визначено мету, об’єкт, предмет і завдання дослідження.

У першому розділі проведено аналіз предметної області та існуючих технологічних рішень.

У другому розділі описано моделювання системи, побудовано UML-діаграми та розроблено структуру бази даних.

Третій розділ присвячено реалізації системи засобами C# та ML.NET.

У четвертому розділі наведено результати тестування, оцінку ефективності алгоритмів і порівняльний аналіз отриманих показників.

У висновках узагальнено результати дослідження, сформульовано основні наукові висновки та окреслено напрями подальшого розвитку системи.

# 1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Опис предметної області та особливості обробки спортивних даних

Предметна область дослідження охоплює процеси збору, накопичення, обробки та аналізу даних, що формуються у результаті тренувальної діяльності спортсменів. У сучасному спорті дані стають ключовим активом, який дозволяє об’єктивно оцінювати стан підготовки, планувати навантаження, прогнозувати результати й підвищувати ефективність тренувального процесу [26].

Тренувальні показники складають структурований набір параметрів, що відображають різні аспекти фізичної активності. До них належать:

- **кількість підходів** (sets) — визначає структуру вправ і навантаження за обсягом;
- **кількість повторень** (reps) — показує тривалість зусилля у межах одного підходу;
- **інтенсивність** (intensity) — описує рівень навантаження, наприклад % від максимального зусилля або швидкість виконання;
- **тривалість тренування** (duration) — сумарний час виконання вправ чи сесії;
- **тип вправи** (ExerciseType) — силові, кардіо, координаційні, відновлювальні;
- **дата проведення** (DateDim) — основа для хронологічного аналізу.

Ці показники формують інформаційну базу, що дозволяє відстежувати динаміку розвитку спортсмена, виявляти тенденції, а також обчислювати агреговані метрики (середня інтенсивність, загальна тривалість тощо) [35].

Однак попри очевидну важливість даних, у більшості спортивних організацій та під час індивідуальної підготовки домінує ручний або частково автоматизований підхід до їх обробки. Тренувальні дані часто ведуться у вигляді електронних таблиць (Excel) або навіть паперових журналів, що ускладнює пошук інформації, її фільтрацію та аналіз.

Така ситуація призводить до низки проблем: відсутності централізованого зберігання, втрати історичних даних, дублювання записів, неможливості оперативного прогнозування або побудови звітів. Через це прийняття рішень про зміну тренувальних стратегій здебільшого базується на суб’єктивних оцінках, а не на фактичних даних.

Таким чином, предметна область характеризується високим рівнем неструктурованості інформації, низьким рівнем автоматизації та відсутністю інтегрованих аналітичних інструментів. Це створює передумови для розробки інтелектуальної системи, яка об’єднує збір, аналіз і прогнозування спортивних показників.

## 1.2. Аналіз проблем та недоліків існуючих підходів до обробки спортивних даних

Попри швидкий розвиток інформаційних технологій, більшість спортивних клубів і шкіл досі не використовують систематизованих рішень для збору та аналізу тренувальної інформації. Аналіз поточного стану галузі дозволяє виокремити кілька ключових проблем:

1. **Суб’єктивність оцінювання результатів.** Більшість показників тренувального процесу інтерпретується суб’єктивно. Відсутність об’єктивних аналітичних засобів ускладнює визначення фактичного прогресу спортсмена.

2. **Ручний ввід даних і помилки.** Запис результатів вручну часто призводить до неточностей, пропусків або дублювання. Такі дані складно перевірити, а їх подальший аналіз є ненадійним.

3. **Відсутність глибокої аналітики.** Існуючі інструменти, такі як Excel або Google Sheets, дозволяють лише базові візуалізації (діаграми, середні значення) без автоматичного пошуку закономірностей чи класифікації спортсменів.

4. **Неможливість прогнозування.** Збережені дані не використовуються для прогнозів. Система не може відповісти на запитання: яким буде результат наступного тренування при поточній динаміці навантажень?

5. **Децентралізація зберігання даних.** Відсутність централізованої бази даних призводить до розпорошеності інформації, складності синхронізації та ризику її втрати.

6. **Відсутність інтеграції з аналітичними модулями.** Навіть якщо тренувальні дані фіксуються, вони не використовуються для глибокого аналізу, оскільки не інтегровані з алгоритмами машинного навчання або статистичними інструментами.

Ці проблеми свідчать про потребу переходу від фрагментарного збору даних до **інтелектуальної автоматизації**, що включає структуроване зберігання, аналітичну обробку, класифікацію спортсменів та прогнозування результатів із застосуванням ML-моделей.

## 1.3. Огляд сучасних технологій аналітики та машинного навчання у спорті

У сфері спортивної аналітики дедалі ширше застосовуються технології машинного навчання (Machine Learning, ML). Вони дають змогу обробляти

великі масиви історичних даних, знаходити приховані закономірності, визначати тренди й формувати прогнози.

Одним із ключових інструментів для реалізації таких підходів у середовищі .NET є **ML.NET** — платформа машинного навчання, розроблена Microsoft [2, 15]. Її головна перевага полягає у можливості створення, навчання та використання ML-моделей безпосередньо всередині застосунку на C#, без залучення сторонніх бібліотек чи скриптів Python.

Платформа підтримує кілька основних напрямів аналітики:

- **Регресійне прогнозування** — визначення кількісних залежностей між вхідними параметрами (наприклад, тривалістю й інтенсивністю) та результативністю спортсмена;
- **Кластеризація** — автоматичне групування спортсменів за схожими характеристиками для подальшого аналізу;
- **Аналіз закономірностей та кореляцій** — виявлення взаємозв’язків між різними параметрами тренувань.

1.3.1. Регресійні алгоритми ML.NET. У проєкті застосовуються декілька моделей регресії, кожна з яких має власні переваги:

- **Linear Regression (SDCA)** — проста, стабільна модель для лінійних залежностей між параметрами [4, 18];
- **FastTree** та **LightGBM** — алгоритми градієнтного бустингу, які забезпечують високу точність і швидке навчання [5, 13];
- **Generalized Additive Model (GAM)** — модель для моделювання складних нелінійних взаємозв’язків між тренувальними показниками [11].

Використання кількох моделей дозволяє користувачу обирати найоптимальнішу за показниками точності (AccuracyPercent) та втрат (LossValue).

1.3.2. Кластеризація та виявлення тренувальних профілів. Для виявлення типових груп спортсменів застосовується алгоритм **KMeans**, що розподіляє дані на три кластери (k = 3) [9, 10] залежно від середніх показників інтенсивності, тривалості й кількості повторів.
Кожен кластер характеризується власним профілем:

- високоінтенсивні тривалі тренування;
- короткі інтенсивні сесії;
- помірні витривалі заняття.

Цей підхід дає змогу визначити сильні та слабкі сторони спортсменів і адаптувати навантаження.

1.3.3. Аналіз закономірностей та статистичних зв'язків. Окремий модуль системи — AssociationAnalyzer.cs — здійснює статистичний аналіз тренувальних даних. Він дозволяє визначити варіативність, стабільність і взаємозалежності між різними типами тренувань (наприклад, Cardio ↔ Strength). Отримані результати використовуються для формування рекомендацій користувачеві, що підвищує інформативність і практичну цінність системи.

1.3.4. Технології зберігання даних. Як основну систему керування базами даних обрано Microsoft SQL Server [22, 27]**,** оскільки вона забезпечує:

- високу продуктивність обробки OLTP-даних;
- зручну агрегацію (AVG, GROUP BY) для формування навчальних вибірок;
- підтримку процедур ETL та інтеграцію з ML.NET;
- гарантовану цілісність і безпечність даних.

## 1.4. Постановка завдання магістерського дослідження

Проведений аналіз предметної області та існуючих технологій дозволив визначити основну мету роботи — створення **автономної настільної системи аналізу й прогнозування спортивних досягнень**, що забезпечує повний цикл: від збору тренувальних даних до формування аналітичних звітів.

Для досягнення цієї мети необхідно реалізувати такі завдання:

1. **Спроєктувати базу даних (MS SQL Server)** для зберігання інформації про спортсменів, вправи, параметри тренувань і результати ML-аналізу.
2. **Розробити інтерфейс користувача (WinForms)** для роботи з даними — реєстрації користувачів, введення тренувань, перегляду звітів і запуску аналітичних модулів.
3. **Реалізувати модуль прогнозування (Analyzer.cs)** на основі алгоритмів ML.NET для передбачення майбутніх спортивних результатів.
4. **Інтегрувати модуль кластеризації (ClusterAnalyzer.cs)**, який виконує групування спортсменів за середніми показниками й відображає результат на інтерактивному графіку.
5. **Розробити модуль виявлення закономірностей (AssociationAnalyzer.cs)** для автоматичного пошуку взаємозв’язків між параметрами тренувань.
6. **Створити модуль звітності (ReportsForm.cs)** з можливістю побудови графіків, фільтрації даних і експорту результатів у CSV, PNG та ZIP.
7. **Провести тестування системи** на реальних даних, оцінити точність моделей і стабільність прогнозів, визначити напрями вдосконалення.

Результатом виконання магістерської роботи стане інтегрована система, яка дозволяє ефективно керувати спортивними даними, автоматично аналізувати тренувальні показники, прогнозувати майбутні результати та формувати візуальні звіти. Її впровадження сприятиме підвищенню об’єктивності оцінки фізичної підготовки, оптимізації навантажень і загальному покращенню ефективності спортивного процесу.

# 2 МОДЕЛЮВАННЯ СИСТЕМИ

## 2.1. Методологія моделювання системи

Розробка будь-якої системи передбачає обов’язковий етап моделювання, який забезпечує перехід від загальних вимог користувача до формалізованої структури майбутнього програмного продукту. На цьому етапі створюється концептуальна модель, що описує ключові процеси, об’єкти та взаємозв’язки між ними. Завдяки моделюванню розробник отримує змогу наочно представити функціональну логіку системи, зменшити ризики неправильного трактування вимог та забезпечити узгодженість між етапами аналізу, проєктування та реалізації [6].

Моделювання виступає важливим інструментом управління складністю системи, оскільки дозволяє розділити процес розробки на логічні рівні — від опису поведінки користувача до побудови архітектури додатку. Таким чином, ще до початку програмування можна виявити недоліки в структурі, суперечності між функціями або дублювання даних, що підвищує ефективність подальшого проєктування.

У межах даної магістерської роботи використано функціональний підхід до моделювання системи, який орієнтований на опис того, що саме робить система з точки зору користувача. Для реалізації цього підходу застосовано уніфіковану мову моделювання UML (Unified Modeling Language) [8] — міжнародний стандарт, що забезпечує універсальні засоби для опису динамічної поведінки системи, потоків управління та взаємодії між користувачами й компонентами. Використання UML дозволяє узгодити вимоги між замовником, аналітиками та розробниками, а також створити чітку основу для подальшого проєктування.

У межах функціонального моделювання побудовано такі типи діаграм:

- Діаграма прецедентів (Use Case Diagram) — відображає функціональні можливості системи та взаємодію користувача зі складовими, зокрема внесення тренувальних даних, запуск аналізу, перегляд аналітичних звітів і кластеризацію спортсменів.
- Діаграма послідовності (Sequence Diagram) — показує порядок виконання дій і взаємодію між користувачем, формами інтерфейсу (наприклад, TrainingForm, AnalysisForm, ReportsForm), бізнес-логікою (Analyzer, ClusterAnalyzer, AssociationAnalyzer) та базою даних під час обробки запитів.
- Діаграма діяльності (Activity Diagram) — ілюструє загальний алгоритм функціонування системи, включно з умовними переходами, перевірками даних, навчанням моделей ML.NET і побудовою прогнозів.

Обраний функціональний підхід є найбільш доцільним для розробки системи аналізу та прогнозування спортивних досягнень, оскільки дозволяє описати послідовність виконання основних бізнес-процесів — від введення тренувальних даних до отримання прогнозів і звітів. Такий підхід забезпечує узгодженість між етапами проєктування, створює чітке уявлення про функціональну логіку системи та є основою для побудови програмної архітектури в наступних розділах.

## 2.2. Об’єктне та функціональне моделювання

2.2.1. Діаграма прецедентів. Діаграма прецедентів (Use Case Diagram) є базовим інструментом UML-моделювання, який дає змогу описати функціональні можливості системи з позиції користувача. Вона демонструє, які саме дії може виконувати користувач у межах системи, у якій послідовності та з яким очікуваним результатом. Метою такої діаграми є фіксація зовнішньої поведінки системи — без деталізації внутрішніх алгоритмів або структури бази даних.

У контексті розроблюваної система аналізу та прогнозування спортивних досягнень на основі даних про тренування, діаграма прецедентів відображає на рис. 1, основні сценарії взаємодії користувача зі застосунком: внесення даних про тренування, запуск машинного аналізу, перегляд звітів і додаткових аналітичних результатів. Такий спосіб моделювання дає змогу отримати узагальнене бачення функцій програми ще до етапу розробки та забезпечує узгодженість між логічним і технічним рівнями проєкту.

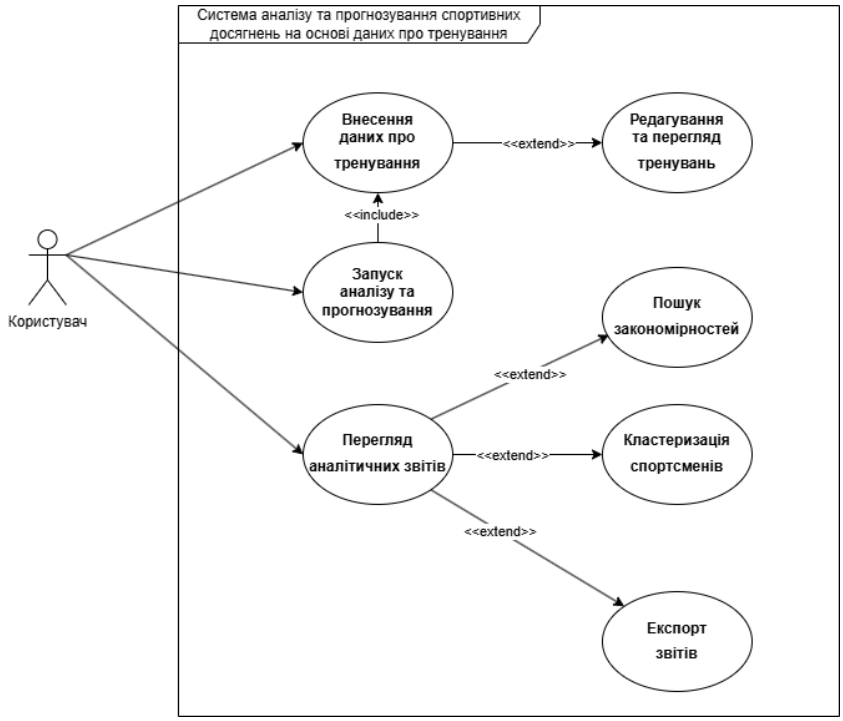



Рис. 1 Діаграма прецедентів

Головним актором діаграми є Користувач — центральна фігура, яка ініціює всі ключові сценарії роботи із системою. Актор взаємодіє із застосунком, щоб фіксувати дані власних тренувань, отримувати результати машинного аналізу, переглядати прогнози й оцінювати ефективність тренувального процесу.

До основних функціональних можливостей актора належать:

1. Додавання нових записів про тренування.
2. Перегляд і коригування вже збережених даних.

3. Ініціювання аналітичного прогнозу на основі історичних даних.

4. Аналіз результатів у форматі звітів і графічних діаграм.

5. Використання розширених можливостей системи — кластеризації спортсменів та пошуку статистичних залежностей.

Таким чином, Користувач є головним елементом взаємодії із системою, а всі прецеденти орієнтовані на автоматизацію його аналітичних дій та підвищення точності оцінки результатів.

На діаграмі виділено основні функціональні прецеденти:

1. Внесення даних про тренування — базовий процес, який забезпечує введення показників тренувань (кількість підходів, повторів, інтенсивність, тривалість). Без цього етапу система не може виконувати подальший аналіз.

2. Редагування та перегляд тренувань — допоміжний сценарій, який дозволяє користувачу підтримувати актуальність даних і вносити зміни в разі потреби.

3. Запуск аналізу та прогнозування — основний аналітичний процес, який ініціює навчання моделей машинного навчання ML.NET, розрахунок прогнозованих показників та метрик точності.

4. Перегляд аналітичних звітів — модуль відображення результатів у вигляді таблиць, діаграм та динамічних графіків.

5. Кластеризація спортсменів — автоматичне групування спортсменів за схожими характеристиками тренувань.

6. Пошук закономірностей — аналітичний сценарій для виявлення кореляцій та взаємозв’язків між параметрами тренувального процесу.

7. Експорт звітів — додатковий функціонал для збереження результатів аналізу у зовнішніх форматах (CSV, PNG, ZIP).

**Прецедент «Внесення даних про тренування»**

1.1 Передумови:

• Користувач авторизований у системі й має намір додати нове тренування.

1.2 Тригер:

• Натискання кнопки «Додати тренування» у головному вікні програми.

1.3 Головний потік:

1. Відкривається форма TrainingForm.
2. Користувач вводить параметри тренування: тип вправи, кількість підходів (sets), повторів (reps), інтенсивність (intensity), тривалість (duration) і дату.
3. Система перевіряє правильність заповнення даних.
4. Якщо дані коректні, форма створює SQL-запит і через клас Connection.cs надсилає його до бази даних.
5. База даних (TrainingFact) приймає запит, зберігає новий запис і повертає підтвердження.
6. Користувачу відображається повідомлення «Тренування успішно збережено».

1.4 Під-потоки: Відсутні

1.5 Альтернативний потік:

• A1: Якщо введено некоректні або неповні дані — система показує повідомлення «Некоректний формат даних», і користувач повертається до редагування форми.

**Прецедент «Запуск аналізу та прогнозування»**

1.1 Передумови:

• У таблиці TrainingFact є записи для поточного користувача.

1.2 Тригер:

• Натискання кнопки «Запустити аналіз» у формі AnalysisForm.

1.3 Головний потік:

1. Форма AnalysisForm викликає метод аналізу з класу Analyzer.cs.
2. Analyzer через Connection.cs виконує запит SELECT до TrainingFact і отримує історію тренувань користувача.
3. Якщо дані доступні — формується вибірка й запускається навчання моделей ML.NET (SDCA, FastTree, LightGBM, GAM).
4. Обчислюється прогнозований показник PredictedPerformance і метрики якості (LossValue, AccuracyPercent).
5. Результати аналізу записуються у таблицю TrainingAnalysisFact.
6. Користувачу виводиться повідомлення «Аналіз виконано успішно».

1.4 Під-потоки: Відсутні

1.5 Альтернативний потік:

• А1: Якщо у таблиці немає достатньої кількості даних — система виводить повідомлення «Недостатньо даних для аналізу», і процес завершується.

**Прецедент «Перегляд аналітичних звітів, кластеризація та пошук закономірностей»**

1.1 Передумови:

• У базі даних TrainingAnalysisFact вже наявні результати аналізу.

1.2 Тригер:

• Користувач відкриває розділ «Звіти» з головного меню.

1.3 Головний потік:

1. Форма ReportsForm виконує SQL-запит до TrainingAnalysisFact (із приєднанням PredictionModelDim) через клас Connection.cs.
2. Отримані результати виводяться у таблицю та графіки.
3. Користувач може переглядати, фільтрувати або оновлювати дані.
4. Якщо користувач натискає «Кластеризація спортсменів», форма надсилає запит до ClusterAnalyzer.cs, який отримує тренувальні дані через Connection.cs, виконує кластеризацію (KMeans) і повертає результати.
5. Якщо користувач натискає «Пошук закономірностей», викликається AssociationAnalyzer.cs, що формує аналітичний звіт про кореляції та середні значення.
6. При натисканні «Експорт звітів» система формує файл CSV/PNG/ZIP на основі відображених даних.
7. Усі запити завершуються відповідями від бази даних, після чого користувач отримує повідомлення «Операцію виконано успішно».

1.4 Під-потоки: Відсутні

1.5 Альтернативні потоки:

• A1: Якщо дані для кластеризації відсутні — система показує повідомлення «Недостатньо даних для кластеризації».

На діаграмі використано стандартні UML-зв’язки:

• **«include»** — означає, що певний підпроцес є обов’язковою частиною основного сценарію. Наприклад, перед запуском аналізу система обов’язково виконує підпроцес підготовки даних.

• **«extend»** — вказує на додаткові або опціональні сценарії, які виконуються за потреби. Так, під час перегляду звітів користувач може ініціювати розширення у вигляді кластеризації або пошуку закономірностей.

Таке представлення дозволяє чітко зрозуміти структуру функціоналу системи, взаємозв’язки між основними компонентами й логіку роботи користувача. Діаграма прецедентів формує основу для створення наступних UML-моделей — діаграм діяльності, послідовності та класів, які деталізують архітектуру і поведінку програмного продукту.

2.2.2. Діаграма послідовності. Діаграма послідовності (англ. Sequence Diagram) є одним із ключових інструментів моделювання динамічної поведінки програмних систем. Вона відображає порядок обміну повідомленнями між об’єктами, виклики методів та обмін даними між компонентами системи у часовій послідовності. Такі діаграми дозволяють наочно показати, як відбувається взаємодія між користувачем, формами інтерфейсу, бізнес-логікою та базою даних, а також простежити залежності між окремими етапами виконання операцій у системі

У контексті розроблюваної системи аналізу та прогнозування спортивних досягнень діаграми послідовності ілюструють процеси, що відповідають основним прецедентам — внесення даних про тренування, запуск аналізу та перегляд аналітичних звітів. Нижче наведено опис основних об’єктів, які беруть участь у цих процесах:

1. Користувач (Спортсмен) — зовнішній актор, який ініціює всі сценарії: додає тренування, запускає аналітичні моделі та переглядає звіти.
2. Форми (WinForms) — елементи інтерфейсу (TrainingForm, AnalysisForm, ReportsForm), через які користувач взаємодіє із системою. Вони відповідають за збирання введених даних, виклик бізнес-логіки та відображення результатів.
3. Клас Analyzer — об’єкт бізнес-логіки, який реалізує процес машинного навчання (ML.NET), формує прогноз і розраховує метрики якості.

4. Класи ClusterAnalyzer та AssociationAnalyzer — допоміжні аналітичні модулі, що реалізують кластеризацію (KMeans) і пошук закономірностей на основі тренувальних даних.

5. Клас Connection — сервіс взаємодії з базою даних, який виконує SQL-запити (SELECT, INSERT) і повертає результати у форми або аналітичні класи.

6. БД (MS SQL Server) — централізоване сховище даних системи. Зберігає інформацію про тренування, результати прогнозів і моделі машинного навчання.

На першому етапі користувач вводить дані про тренування на рис. 2. Після авторизації він відкриває TrainingForm, заповнює параметри (sets, reps, intensity, duration, date) і натискає кнопку «Зберегти». Форма виконує перевірку правильності введених даних. Якщо вони коректні — створюється SQL-запит, який через Connection.cs надсилається до БД. Таблиця TrainingFact зберігає новий запис і повертає підтвердження. У разі помилки або некоректних значень у полі — користувач отримує повідомлення «Некоректний формат даних», і запит до бази не виконується. Цей сценарій наочно показує перевірку умов введення через блок alt на діаграмі.

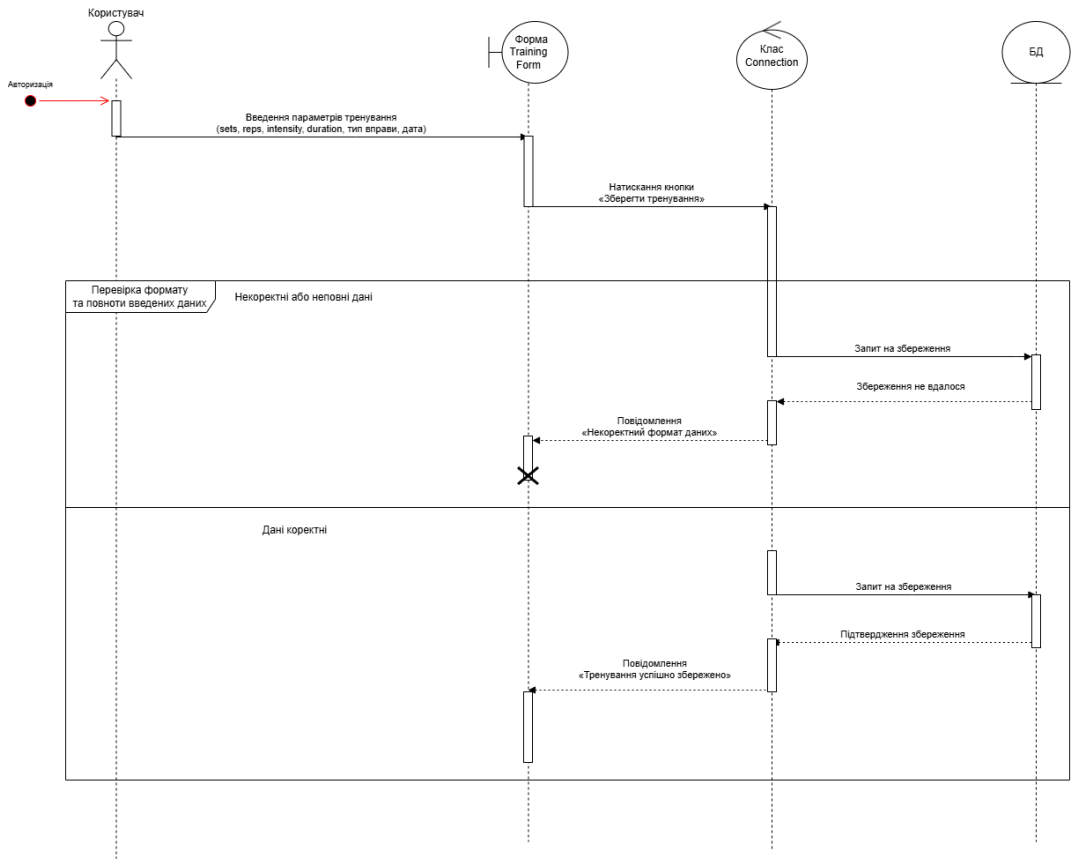



Рис. 2 Діаграма послідовності «Внесення даних про тренування»

На другому етапі користувач ініціює аналіз і прогнозування на рис. 3. Після натискання кнопки «Запустити аналіз» у AnalysisForm, форма звертається до Analyzer.cs, який через Connection.cs виконує запит SELECT до таблиці TrainingFact для отримання тренувальних даних поточного користувача. У блоці alt перевіряється наявність достатнього обсягу даних:

- Якщо даних недостатньо — система виводить повідомлення «Недостатньо даних для аналізу», і процес завершується.
- Якщо даних достатньо — формується вибірка, запускається навчання моделей ML.NET (SDCA, FastTree, LightGBM, GAM), обчислюється прогнозований показник PredictedPerformance і метрики якості (AccuracyPercent, LossValue). Результати зберігаються через

INSERT у таблицю TrainingAnalysisFact, після чого користувач бачить повідомлення «Аналіз виконано успішно».

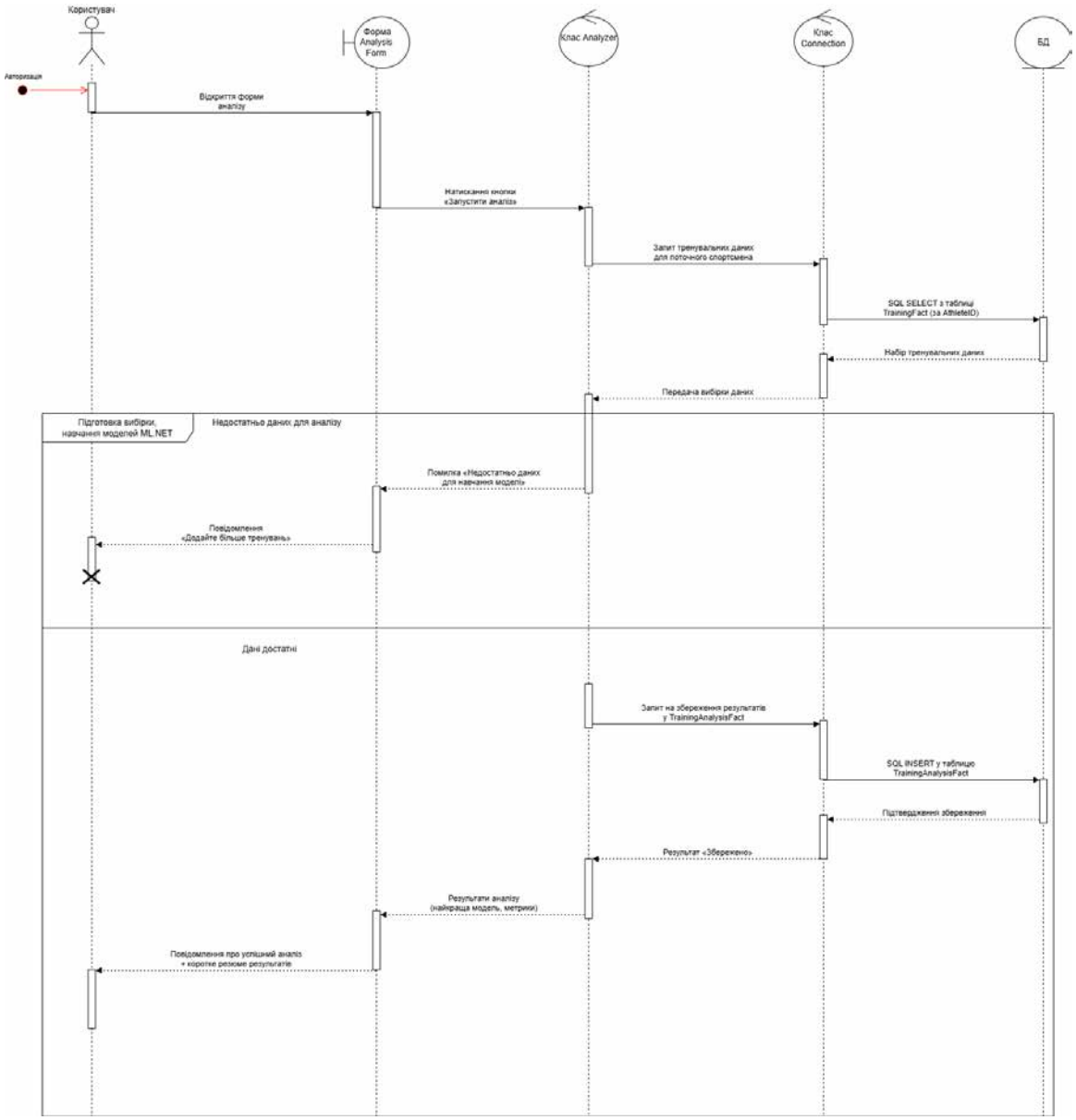



Рис. 3 Діаграма послідовності «Запуск аналізу та прогнозування»

На третьому етапі користувач переглядає аналітичні звіти на рис. 4. Відкривши ReportsForm, система автоматично виконує SQL-запит до TrainingAnalysisFact (з приєднанням PredictionModelDim) через Connection.cs. Отримані результати відображаються у вигляді таблиць і графіків динаміки прогнозів.

Користувач може ініціювати додаткові аналітичні сценарії:

- Кластеризація спортсменів — виклик методу PerformClustering() у ClusterAnalyzer.cs, який отримує вибірку з TrainingFact, виконує кластеризацію (KMeans) та повертає результати до форми.
- Пошук закономірностей — виклик методу DetectPatterns() у AssociationAnalyzer.cs, який виявляє кореляції та узагальнені аналітичні залежності.
- Експорт звітів — формування файлів CSV, PNG або ZIP безпосередньо у формі звітів на основі вже отриманих даних.

Усі запити до бази даних виконуються через Connection.cs, а БД завершує взаємодію як кінцевий учасник процесу.

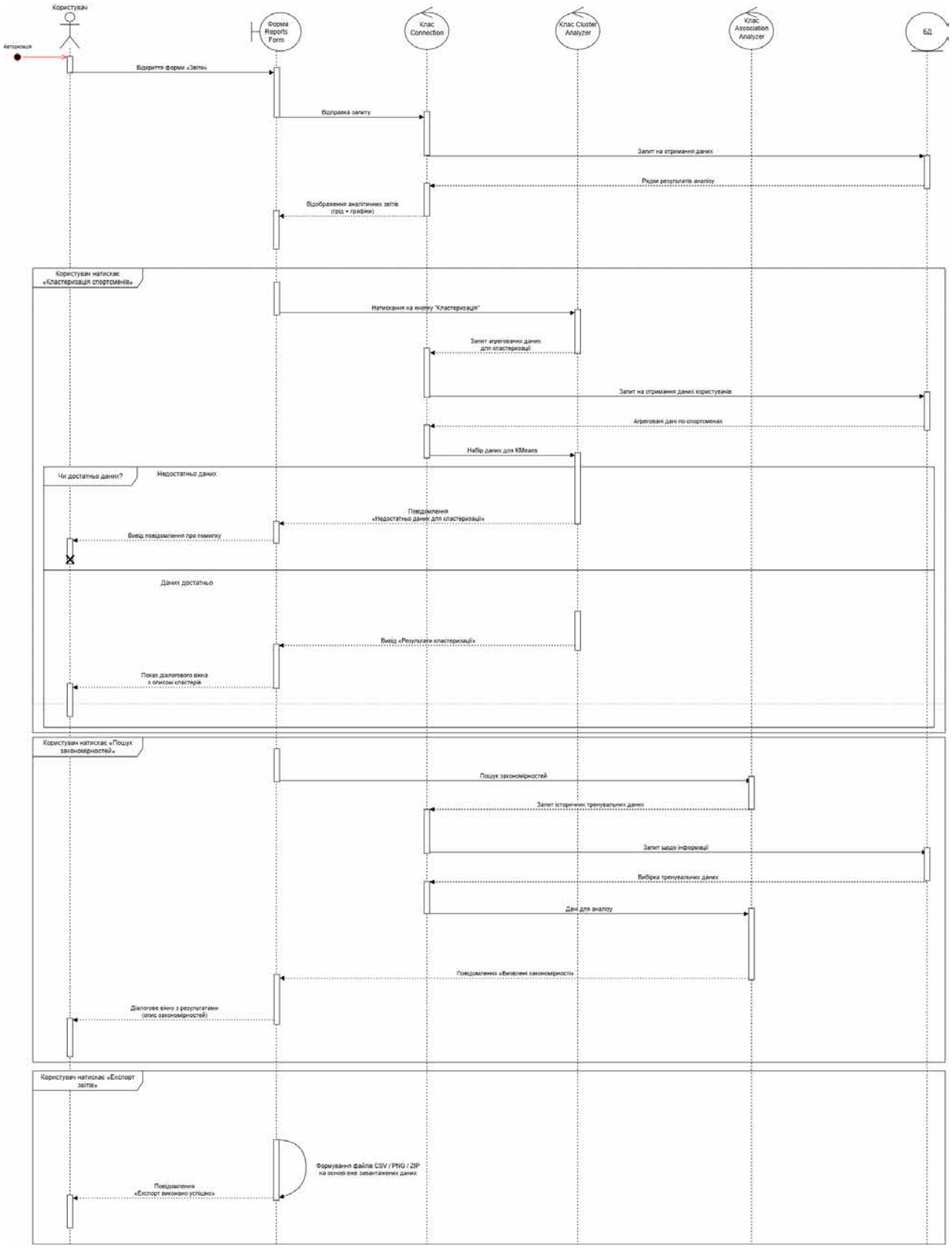



Рис. 4 Діаграма послідовності «Перегляд аналітичних звітів, кластеризація та пошук закономірностей»

Таким чином, три розроблені діаграми послідовності демонструють узгоджену логіку роботи системи: від моменту введення тренувальних даних — до виконання аналізу ML.NET і отримання інтерактивних звітів із кластеризацією та пошуком закономірностей. Вони забезпечують цілісне уявлення про динаміку взаємодії між користувачем, інтерфейсом, аналітичними модулями та базою даних.

2.2.3. Діаграма діяльності. Діаграма діяльності (Activity Diagram) є одним із ключових інструментів функціонального моделювання системи в UML. Вона описує послідовність дій, що виконуються користувачем і системою, а також логіку переходів між ними — від початку процесу до його завершення. На відміну від діаграми послідовності, яка зосереджена на взаємодії між об’єктами, діаграма діяльності відображає загальний алгоритм бізнес-процесу, демонструючи, як система реагує на події та які дії виконуються залежно від умов.

Застосування діаграм діяльності дає змогу чітко формалізувати робочі процеси, виявити можливі розгалуження або критичні точки, а також покращити розуміння логіки функціонування системи як розробниками, так і кінцевими користувачами. У контексті розробки системи аналізу та прогнозування спортивних досягнень побудовано три основні діаграми діяльності, що ілюструють ключові етапи її роботи: внесення даних про тренування, запуск аналізу та перегляд аналітичних звітів.

Перша діаграма діяльності на рис. 5 демонструє процес «Внесення даних про тренування». Вона складається з трьох доріжок — «Користувач», «Система» та «База даних». Процес починається з того, що користувач заповнює параметри тренування (тип вправи, кількість підходів і повторів, інтенсивність, тривалість, дату) та натискає кнопку «Зберегти». Система виконує перевірку правильності та повноти введених даних. У разі виявлення помилки відображається повідомлення «Некоректний формат даних», після чого користувач може внести

зміни. Якщо дані заповнені правильно, система надсилає запит до бази даних, де створюється запис у таблиці TrainingFact. Після успішного збереження база повертає підтвердження, і користувачу відображається повідомлення про успішне завершення операції.

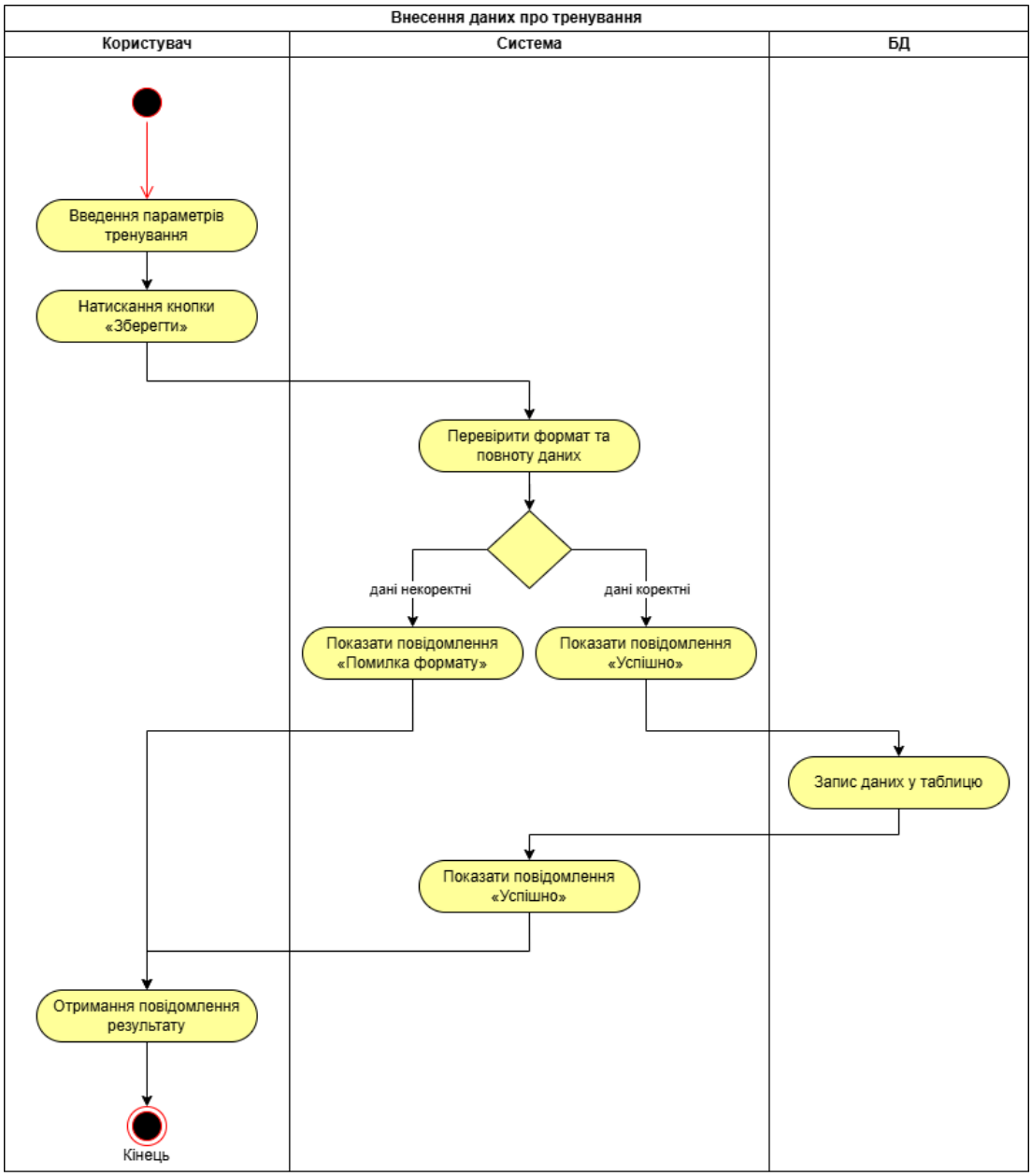



Рис. 5 Діаграма діяльності «Внесення даних про тренування»

Друга діаграма діяльності рис. 6 відображає процес «Запуск аналізу та прогнозування». Користувач ініціює процес, натискаючи кнопку «Запустити аналіз» у формі AnalysisForm. Система (Analyzer.cs) звертається до бази даних із

запитом SELECT для отримання історії тренувань користувача з таблиці TrainingFact. Далі перевіряється, чи достатньо даних для побудови моделі машинного навчання:

- Якщо даних недостатньо — користувач отримує повідомлення «Недостатньо даних для аналізу», і процес завершується.
- Якщо даних достатньо — система формує навчальну вибірку, запускає моделі ML.NET (SDCA, FastTree, LightGBM, GAM), обчислює прогнозований показник PredictedPerformance, метрики точності (AccuracyPercent) та функцію втрат (LossValue). Після завершення обчислень результати зберігаються в таблиці TrainingAnalysisFact, і користувачу відображається повідомлення «Аналіз виконано успішно».

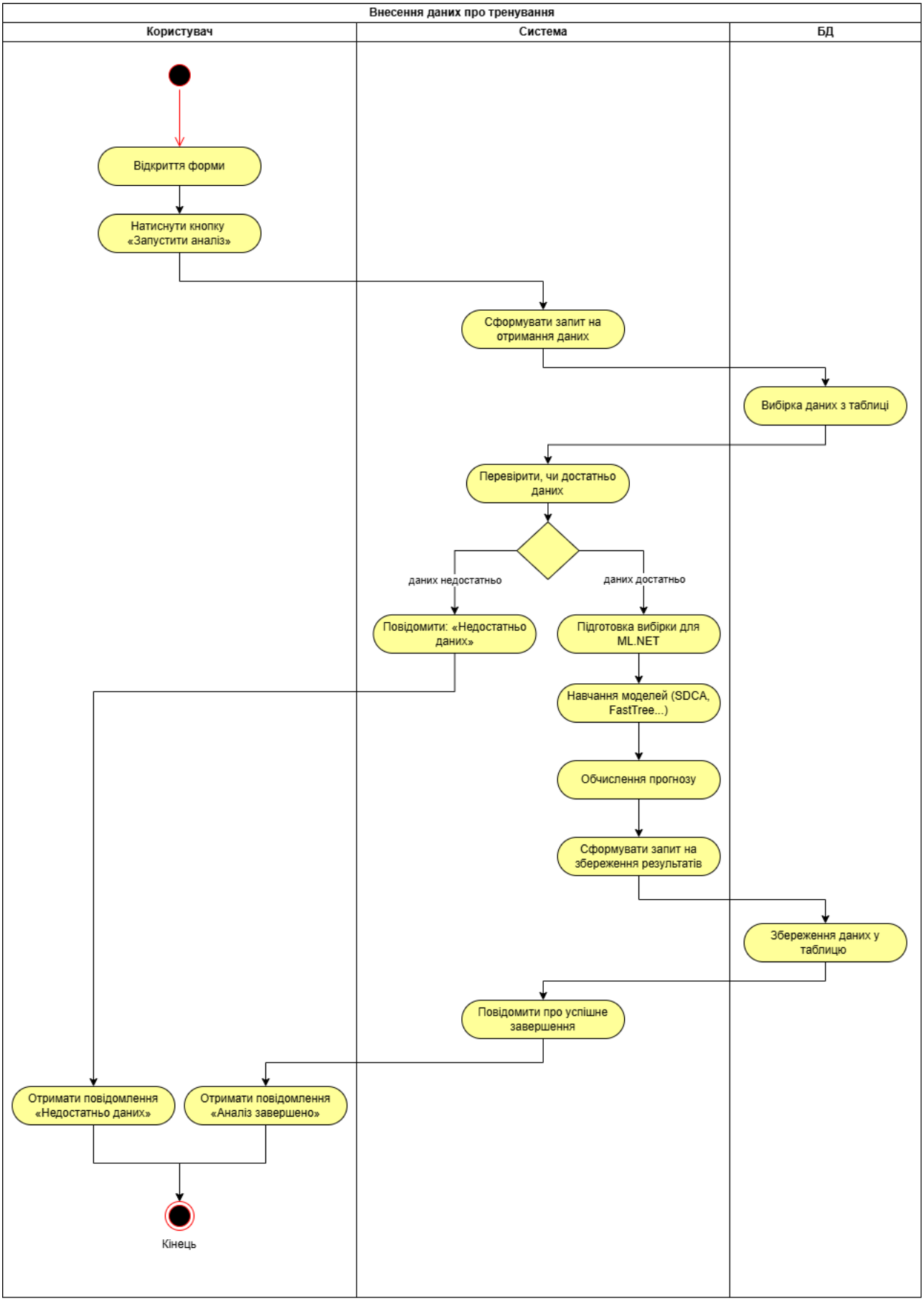



Рис. 6 Діаграма діяльності «Запуск аналізу та прогнозування»

Третя діаграма діяльності рис. 7 ілюструє процес «Перегляд аналітичних звітів».

Після відкриття форми ReportsForm система автоматично надсилає запит до бази даних для отримання даних аналізу з таблиці TrainingAnalysisFact. Якщо результати відсутні — користувачу виводиться повідомлення «Немає даних для відображення», після чого процес завершується. Якщо ж результати наявні, система відображає їх у вигляді таблиць і графіків. Далі користувач може обрати одну з додаткових дій:

- Кластеризація спортсменів — система викликає модуль ClusterAnalyzer, який отримує вибірку з бази та формує групи спортсменів за подібністю параметрів;
- Пошук закономірностей — виконується запит до AssociationAnalyzer, який аналізує кореляції між параметрами тренувань;
- Експорт звітів — система формує файли (CSV, PNG або ZIP) на основі вже отриманих даних;
- Вихід — користувач завершує роботу із модулем звітності.

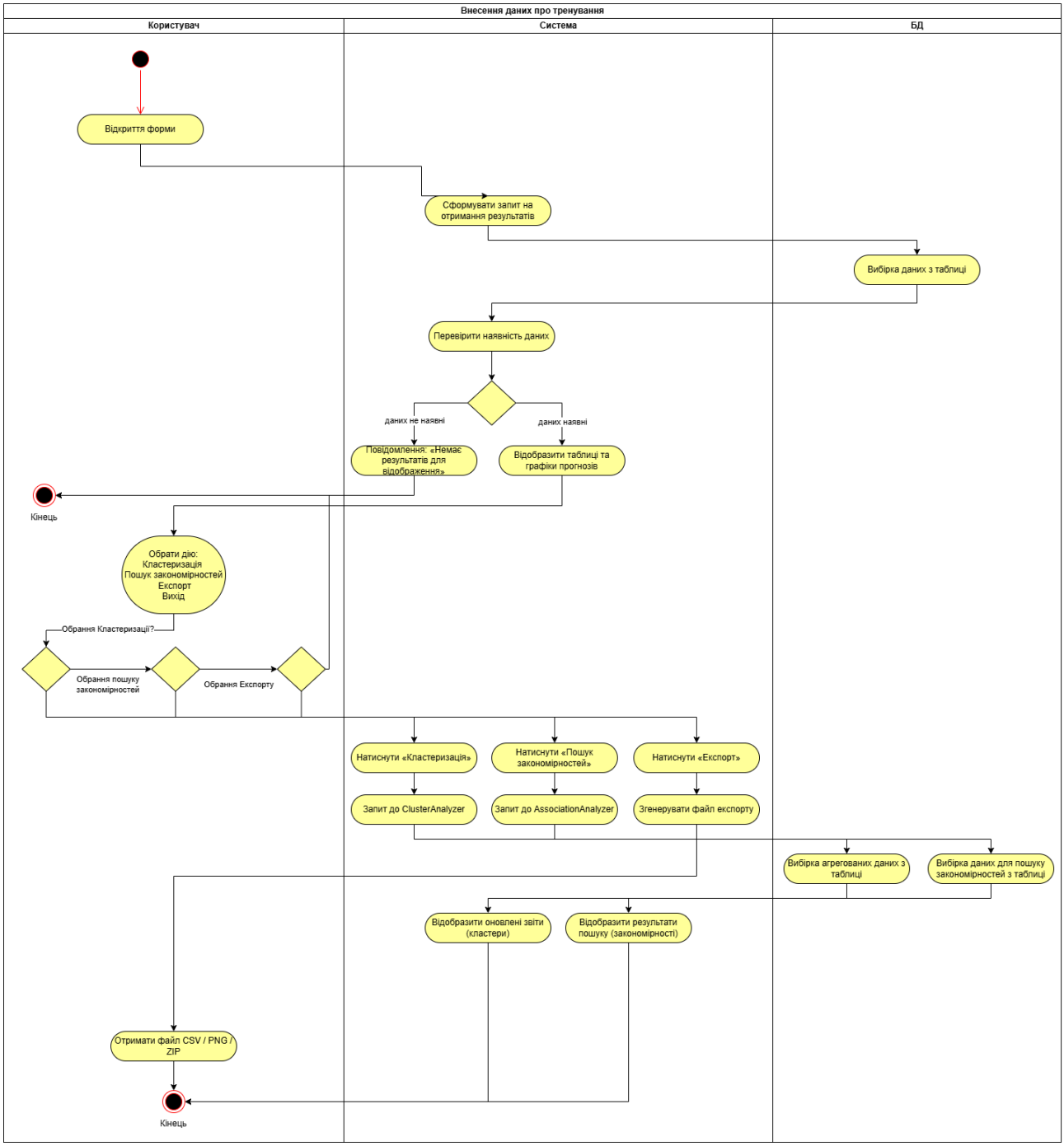



Рис. 7 Діаграма діяльності «Перегляд аналітичних звітів»

Таким чином, побудовані діаграми діяльності відображають логіку роботи системи на всіх етапах її функціонування: від введення даних про тренування — до запуску машинного аналізу та перегляду отриманих прогнозів. Вони дозволяють наочно простежити послідовність дій користувача й системи, виявити ключові рішення, що впливають на результат, і забезпечують цілісне розуміння алгоритмів роботи програмного продукту.

## 2.3. Моделювання структури даних

Для забезпечення ефективного зберігання та обробки тренувальних даних у системі була розроблена логічна структура даних, що формує основу майбутньої бази даних SportAnalysisDB. Основною метою моделювання структури даних є формалізація організації інформації у системі, забезпечення цілісності та взаємозв'язку даних, а також підготовка основи для реалізації функціоналу обробки та прогнозування спортивних досягнень.

Схема бази даних [27, 35] побудована за принципом «Зірка» (Star Schema), що є типовим для аналітичних систем. Вона складається з таблиць фактів (Facts) та вимірів (Dimensions). Діаграма БД системи представлена на рис. 8.

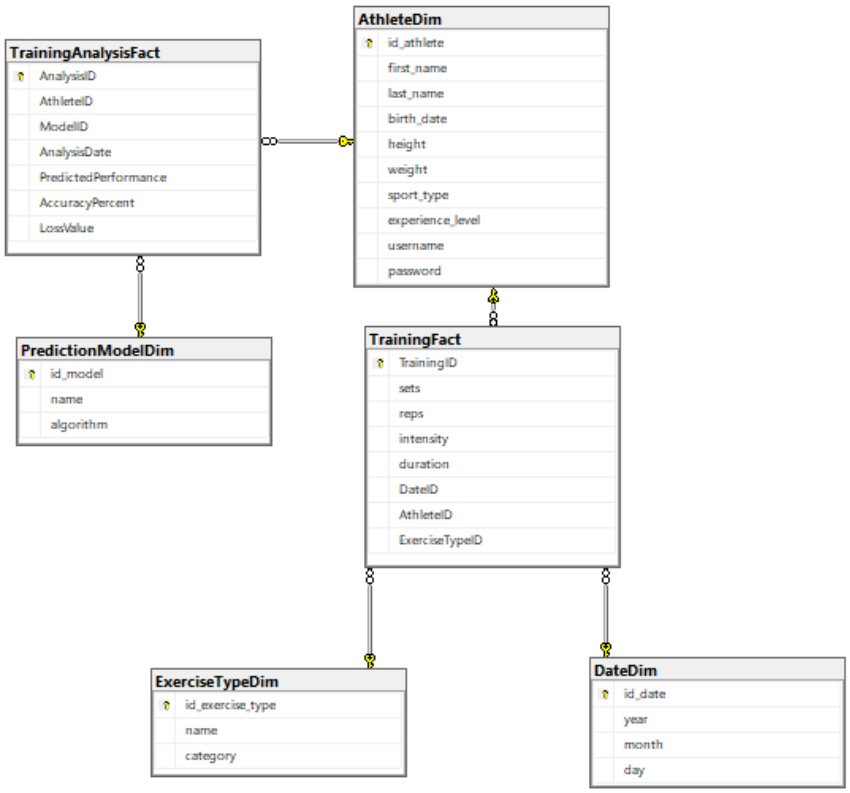



Рис. 8 Схема бази даних системи

Структура даних системи включає декілька основних сутностей (таблиць), які відображають ключові об'єкти предметної області. До них належать:

1. AthleteDim (Вимір спортсменів) — містить основну інформацію про користувачів системи (спортсменів):

- id_athlete — унікальний ідентифікатор спортсмена (PK);
- first_name, last_name — ім’я та прізвище;
- birth_date — дата народження;
- height, weight — антропометричні дані;
- sport_type, experience_level — дані про спортивний профіль;
- username, password — дані для автентифікації (пароль зберігається у хешованому вигляді).

2. TrainingFact (Факт тренувань) — основна таблиця фактів, що зберігає деталізовані дані про кожне окреме тренування. Пов’язана з вимірами AthleteDim, ExerciseTypeDim та DateDim:

- TrainingID — унікальний ідентифікатор запису про тренування (PK);
- sets, reps, intensity, duration — ключові показники тренування;
- DateID — зовнішній ключ на таблицю DateDim (FK);
- AthleteID — зовнішній ключ на таблицю AthleteDim (FK);
- ExerciseTypeID — зовнішній ключ на таблицю ExerciseTypeDim (FK).

3. ExerciseTypeDim (Вимір типів вправ) — довідник, що класифікує вправи:

- id_exercise_type — унікальний ідентифікатор типу вправи (PK);
- name — назва вправи (напр., "Жим лежачи", "Біг");
- category — категорія (напр., "Strength", "Cardio", "Endurance").

4. DateDim (Вимір дат) — довідник дат для аналізу часових рядів:

- id_date — унікальний ідентифікатор дати (PK);
- year, month, day — компоненти дати.

5. TrainingAnalysisFact (Факт результатів аналізу) — таблиця фактів, що містить інформацію про результати ML-аналізу:
   - AnalysisID — унікальний ідентифікатор аналізу (PK);
   - AthleteID — зовнішній ключ на таблицю AthleteDim (FK);
   - ModelID — зовнішній ключ на таблицю PredictionModelDim (FK);
   - AnalysisDate — дата та час виконання аналізу;
   - PredictedPerformance — прогнозований показник (результат регресії);
   - AccuracyPercent — метрика точності у % (напр., R^2);
   - LossValue — метрика похибки (напр., MSE).

6. PredictionModelDim (Вимір моделей прогнозування) — довідник, що зберігає інформацію про ML-моделі:
   - id_model — унікальний ідентифікатор моделі (PK);
   - name — назва моделі (напр., "FastTree", "SDCA");
   - algorithm — назва алгоритму (напр., "Gradient Boosting").

Такий підхід до моделювання структури даних (схема «Зірка») дозволяє впорядкувати інформацію, забезпечити логічні зв’язки між сутностями та уникнути дублювання. Це створює надійну основу для надійного зберігання та ефективної обробки тренувальних даних, що у свою чергу підвищує продуктивність і точність роботи всієї аналітичної системи.

## 2.4. Визначення вимог до системи

Визначення вимог до системи є одним із ключових етапів життєвого циклу розробки програмного забезпечення, оскільки саме на цьому етапі формується чітке розуміння цілей, функціональності та очікувань користувачів. Від повноти та точності сформульованих вимог залежить ефективність реалізації, надійність системи й відповідність отриманого результату потребам кінцевих користувачів.

У межах цієї магістерської роботи, присвяченої створенню системи аналізу та прогнозування спортивних досягнень, вимоги були сформовані на основі аналізу предметної області, проблем традиційного (ручного) аналізу тренувальних показників та можливостей сучасних методів машинного навчання (ML.NET, KMeans). Основна мета системи — підвищення об’єктивності оцінювання результатів тренувань, точності прогнозування спортивних досягнень, забезпечення збереження персональних даних і створення інтуїтивно зрозумілого інтерфейсу для користувачів [6].

Для забезпечення цілісності опису, вимоги поділено на дві групи: функціональні та нефункціональні.

**Функціональні вимоги:**

Функціональні вимоги описують конкретні дії, які система повинна виконувати, та сценарії взаємодії користувача з програмою.

1. Управління даними тренувань (CRUD-операції). Система має забезпечувати повний цикл роботи з тренувальними даними: створення, перегляд, редагування та видалення записів. Дані (поля sets, reps, intensity, duration) зберігаються у таблиці TrainingFact бази даних MS SQL Server. Усі дії реалізуються через графічний інтерфейс WinForms, що дозволяє користувачу зручно управляти інформацією про власні тренування.

2. Аутентифікація та управління профілем користувача. Система повинна мати безпечний механізм входу в обліковий запис (форма LoginForm, таблиця AthleteDim). Після авторизації користувач отримує доступ до функції редагування профілю (EditProfileForm), зокрема оновлення особистих даних або зміни паролю.

3. Запуск аналізу та прогнозування результатів. У системі реалізовано модуль (AnalysisForm), який забезпечує запуск моделей машинного навчання ML.NET для регресійного аналізу історичних тренувань. Моделі (SDCA, FastTree, LightGBM, GAM) здійснюють прогнозування показника PredictedPerformance, а також розрахунок метрик якості — AccuracyPercent та LossValue. Отримані результати зберігаються у таблицю TrainingAnalysisFact.

4. Розширена аналітика (кластеризація та пошук закономірностей).

- Кластеризація спортсменів. Модуль ClusterAnalyzer реалізує алгоритм KMeans (k=3) для групування спортсменів за середніми показниками ефективності тренувань і відображає результати у вигляді візуалізацій.
- Пошук закономірностей. Модуль AssociationAnalyzer виконує статистичний аналіз взаємозв’язків між характеристиками тренувань (інтенсивність, тривалість, кількість повторів тощо) та виявляє найбільш значущі кореляції.

5. Формування аналітичних звітів та візуалізація результатів. Форма ReportsForm забезпечує побудову інтерактивних звітів на основі даних з TrainingAnalysisFact. Звіти відображаються у вигляді таблиць, графіків динаміки показників з фільтрацією за датою або моделлю. Система підтримує експорт звітів у формати CSV, PNG або ZIP для подальшого збереження чи друку.

**Нефункціональні вимоги:**

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на її продуктивність, безпеку та зручність використання.

1. Точність прогнозування.
Система повинна забезпечувати точність моделей машинного навчання на рівні не менше 80% (за метриками R^2 або NRMSE), що гарантує практичну цінність прогнозів для спортсменів і тренерів.

2. Швидкодія аналізу.
Повний цикл аналізу (вибірка даних, навчання моделей, запис результатів) не повинен перевищувати 1,5–2 секунди для наборів до 1000 тренувальних записів.

3. Безпека та конфіденційність даних.
Особисті дані спортсменів (AthleteDim) і тренувальні записи (TrainingFact) мають бути захищені. Паролі зберігаються у хешованому вигляді, а всі запити до бази даних реалізуються з використанням параметризації (SqlParameter) для запобігання SQL-ін’єкціям. Доступ до системи обмежений авторизованими користувачами.

4. Надійність та відмовостійкість.
Система повинна коректно обробляти помилки введення (наприклад, введення нечислових даних у поле reps) та повідомляти користувача про проблеми. У випадках, коли аналіз неможливий (недостатньо даних), система має виводити відповідне повідомлення без завершення роботи програми.

5. Зручність користувацького інтерфейсу.
Інтерфейс WinForms має бути простим, інтуїтивним і послідовним. Перехід між формами здійснюється логічно, а всі основні функції (введення, аналіз, перегляд звітів) доступні без додаткових технічних знань.

Формалізація функціональних і нефункціональних вимог створює надійну основу для подальших етапів проєктування системи. Визначені вимоги забезпечують точне розуміння завдань розробки, гарантують відповідність очікуванням користувачів і створюють передумови для високої ефективності, точності прогнозів і зручності використання системи аналізу спортивних досягнень.

# 3 РОЗРОБКА СИСТЕМИ

## 3.1. Вибір інструментальних засобів та середовища розробки

3.1.1. Середовище розробки та мова програмування. Для розробки системи аналізу та прогнозування спортивних досягнень було обрано середовище Microsoft Visual Studio, яке є одним із найпотужніших інтегрованих середовищ розробки (IDE) для створення застосунків на платформі .NET [29]. Visual Studio забезпечує повний набір інструментів для написання, налагодження, тестування та розгортання програмного забезпечення, що робить його оптимальним вибором для реалізації комплексних інформаційних систем.

Основною мовою програмування, використаною при розробці системи, є C# (C-Sharp). Вибір цієї мови зумовлений такими факторами [28]:

- інтеграція з платформою .NET — C# дозволяє ефективно взаємодіяти з бібліотеками ML.NET, ADO.NET та іншими компонентами Microsoft;
- об’єктно-орієнтована парадигма — мова підтримує інкапсуляцію, наслідування та поліморфізм, що спрощує структурування коду та розширення функціональності;
- висока безпечність та типізація — C# забезпечує контроль типів, обробку винятків і роботу з пам’яттю без прямого втручання розробника;
- зручність створення графічних інтерфейсів — використання бібліотеки Windows Forms (WinForms) дозволяє швидко створювати інтуїтивно зрозумілі інтерфейси для введення, обробки та візуалізації даних [33];
- інтеграція з базами даних Microsoft SQL Server через технології SqlConnection, SqlCommand і DataAdapter, що спрощує роботу з таблицями та запитами [34].

Архітектура застосунку побудована за принципом багаторівневості:

1. Рівень презентації (UI) — реалізований за допомогою WinForms. Містить основні форми: LoginForm, TrainingForm, AnalysisForm, ReportsForm, які забезпечують зручну взаємодію користувача з системою.
2. Рівень логіки (Business Logic Layer) — реалізує обробку даних, виклики аналітичних модулів (Analyzer.cs, ClusterAnalyzer.cs, AssociationAnalyzer.cs), валідацію введених даних та передачу запитів до бази даних.
3. Рівень доступу до даних (Data Access Layer) — відповідає за роботу з базою даних SportAnalysisDB через класи Connection.cs та SQL-запити (SELECT, INSERT, UPDATE).

Використання Visual Studio у поєднанні з мовою C# забезпечує:

- високу швидкість розробки;
- стабільність і масштабованість коду;
- підтримку інтеграції з аналітичними бібліотеками ML.NET;
- можливість реалізації як десктопної, так і серверної версії застосунку в майбутньому.

Таким чином, середовище Visual Studio та мова C# було обрано як технологічну основу для створення системи завдяки їх продуктивності, інтеграційним можливостям і зручності при реалізації програмних рішень аналітичного характеру.

3.1.2. Використання ML.NET та його роль у системі. У розробленій системі аналізу та прогнозування спортивних досягнень особливе значення має інтелектуальна обробка даних, спрямована на виявлення прихованих закономірностей у тренувальних показниках спортсменів та прогнозування

майбутніх результатів. Для реалізації цього функціоналу було обрано платформу ML.NET — потужну бібліотеку машинного навчання від компанії Microsoft, яка дозволяє інтегрувати алгоритми штучного інтелекту безпосередньо у додатки на базі .NET.

Вибір ML.NET зумовлений кількома ключовими факторами. По-перше, платформа повністю сумісна з мовою C# та середовищем WinForms, що забезпечує зручну інтеграцію аналітичних модулів у вже реалізовану архітектуру системи без використання зовнішніх інтерфейсів або API. Це дозволяє виконувати весь процес аналізу — від підготовки даних до формування прогнозу — локально, без підключення до сторонніх сервісів.

По-друге, ML.NET має гнучку структуру побудови аналітичних конвеєрів (data pipeline), що включає етапи завантаження, очищення, перетворення даних, навчання моделі, обчислення метрик та генерації прогнозу. Завдяки цьому розроблена система може автоматично навчати моделі на нових тренувальних даних і оновлювати результати без необхідності ручного втручання користувача.

По-третє, ML.NET дозволяє виконувати всі обчислення локально, без використання хмарних сервісів, що гарантує високу продуктивність, автономність та конфіденційність персональних даних спортсменів. Усі розрахунки проводяться безпосередньо у середовищі виконання застосунку, що особливо важливо для систем, які працюють із закритими або приватними тренувальними даними.

У системі ML.NET виконує роль аналітичного ядра, яке забезпечує прогнозування спортивних результатів на основі історичних тренувань користувача. Для цього використовується набір моделей регресії, що дозволяють оцінити залежність між ключовими параметрами тренувань — інтенсивністю, кількістю підходів, повторень, тривалістю — та загальним показником

ефективності спортсмена. Моделі тренуються на основі індивідуальних даних користувача, що забезпечує персоналізований характер прогнозу.

Основна модель — FastTree Regression — реалізує метод градієнтного бустингу, який показав найкращу точність під час експериментів. Паралельно для порівняння результатів використовуються альтернативні алгоритми: SDCA Regression (лінійна модель для швидких оцінок) та LightGBM Regression, яка забезпечує високу продуктивність при великих наборах даних. Навчання виконується на основі таблиці TrainingFact, а результати прогнозів разом із метриками точності (AccuracyPercent, LossValue) зберігаються у таблиці TrainingAnalysisFact.

ML.NET інтегрується у C#-додаток через спеціальний модуль Analyzer.cs, який реалізує послідовність дій:

1. Завантаження даних із бази (TrainingFact) у формат IDataView.
2. Формування конвеєра перетворень ознак (Features).
3. Навчання обраної моделі регресії.
4. Обчислення прогнозів і метрик якості.
5. Збереження результатів у базу даних (TrainingAnalysisFact).

Отримані прогнози використовуються системою для побудови аналітичних звітів, відображення динаміки спортивних результатів та підтримки прийняття рішень спортсменом або тренером. Користувач може ініціювати навчання моделей та оновлення прогнозів натисканням однієї кнопки у формі AnalysisForm, після чого система автоматично виконує всі необхідні етапи аналізу.

Таким чином, ML.NET у розробленій системі виконує роль інтегрованого інструменту машинного навчання, який забезпечує автоматичний аналіз, побудову регресійних моделей та формування прогнозів спортивних досягнень. Такий підхід дозволяє поєднати класичну аналітику з сучасними методами

інтелектуальної обробки даних, підвищуючи точність оцінки тренувального процесу та ефективність планування подальших занять.

3.1.3. Використання SQL Server та SSMS для роботи з базою даних. Для зберігання, обробки та аналітичного аналізу тренувальних даних у спроєктованій системі аналізу та прогнозування спортивних досягнень було обрано Microsoft SQL Server як основну систему управління базами даних (СУБД) та SQL Server Management Studio (SSMS) як інтегроване середовище адміністрування, розробки й тестування бази даних.

Вибір цих інструментів зумовлений кількома ключовими факторами, що забезпечують надійність, масштабованість і зручність роботи з великими обсягами структурованих даних, а також тісну інтеграцію із середовищем розробки C# / WinForms.

По-перше, Microsoft SQL Server є потужною реляційною СУБД, яка підтримує транзакційну обробку даних, забезпечує цілісність інформації та високий рівень безпеки. Це особливо важливо у контексті зберігання персональних даних спортсменів, результатів аналізів і прогнозів. Використання SQL Server дозволяє реалізувати оптимізовані запити для вибірки, агрегації, фільтрації та сортування тренувальних показників, що значно підвищує ефективність обчислень під час роботи аналітичних модулів системи.

По-друге, SQL Server Management Studio (SSMS) забезпечує зручний графічний інтерфейс для розробки, візуалізації та адміністрування бази даних. У середовищі SSMS створювалися основні таблиці (виміри та факти), зовнішні ключі, індекси та обмеження цілісності. Крім того, SSMS використовувалося для налагодження SQL-запитів, перевірки цілісності зв’язків між таблицями (AthleteDim, TrainingFact, TrainingAnalysisFact, ExerciseTypeDim,

PredictionModelDim) і тестування продуктивності запитів, що застосовуються у програмному модулі Connection.cs.

Архітектура бази даних побудована за принципом «зірки» (Star Schema), що є типовим рішенням для аналітичних систем. У центрі схеми знаходяться фактичні таблиці — TrainingFact (дані про тренування) та TrainingAnalysisFact (результати аналізу та прогнозування), які пов’язані з вимірами:

- AthleteDim — містить персональні дані спортсменів;
- ExerciseTypeDim — класифікує типи вправ;
- DateDim — забезпечує зручну роботу з часовими інтервалами;
- PredictionModelDim — описує використані ML-моделі (FastTree, SDCA, LightGBM).

Такий підхід дозволяє ефективно виконувати OLAP-операції (зведення, групування, порівняння), що використовуються при формуванні аналітичних звітів і візуалізацій результатів у модулі ReportsForm.

Інтеграція між застосунком та SQL Server реалізована через клас Connection.cs, який інкапсулює логіку підключення, обробки SQL-запитів (SELECT, INSERT, UPDATE, DELETE) та передачі результатів у відповідні форми (TrainingForm, AnalysisForm, ReportsForm). Для підвищення безпеки всі запити виконуються з використанням параметризованих команд (SqlParameter), що запобігає можливим SQL-ін’єкціям і несанкціонованому доступу до даних.

Окрім цього, SQL Server забезпечує:

- цілісність даних через використання зовнішніх ключів і каскадних оновлень;
- високу продуктивність завдяки індексації ключових полів і можливості виконання складних аналітичних запитів;

- масштабованість — база даних може розширюватися без зміни архітектури при збільшенні кількості користувачів або тренувальних записів.

Таким чином, поєднання SQL Server та SSMS забезпечує стабільну, захищену та ефективну платформу для зберігання й управління даними у системі аналізу спортивних досягнень. Цей підхід гарантує надійну взаємодію між усіма компонентами системи — від збору тренувальних даних до формування прогнозів і аналітичних звітів

## 3.2. Архітектура спроєктованої системи

Архітектура спроєктованої системи побудована на принципах трирівневої (layered) архітектури [6, 15], що є класичним підходом до розробки сучасних інформаційних систем. Її основна ідея полягає у розподілі програмного комплексу на окремі рівні, кожен з яких виконує строго визначені функції та взаємодіє з іншими рівнями через стандартизовані інтерфейси. Така організація забезпечує високу структурованість коду, зрозумілість логіки роботи, надійність, гнучкість і можливість подальшого масштабування системи.

На першому рівні знаходиться інтерфейс користувача (Presentation Layer), реалізований засобами Windows Forms у середовищі C#. Цей рівень виконує функції візуальної взаємодії між користувачем і системою, забезпечуючи інтуїтивний доступ до ключових функцій. Зокрема, через нього здійснюється:

- введення даних про тренування (TrainingEditForm);
- автентифікація користувача (LoginForm);
- запуск аналітичних процесів (AnalysisForm);
- перегляд прогнозів і звітів (ReportsForm).

Рівень інтерфейсу розроблено з урахуванням принципів зручності (UX) і

мінімалізму, щоб користувачі без спеціальної технічної підготовки могли ефективно працювати із системою.

Другий рівень становить рівень бізнес-логіки (Business Logic Layer) — ядро системи, у якому реалізовано аналітичні алгоритми, обчислювальні процедури та правила обробки даних. На цьому рівні відбувається підготовка вибірок, формування ознак для моделей, запуск алгоритмів машинного навчання ML.NET (регресія, кластеризація) та збереження результатів аналізу. До ключових компонентів цього рівня належать:

- Analyzer.cs — модуль, що реалізує навчання регресійних моделей (SDCA, FastTree, LightGBM, GAM) та розрахунок метрик точності;
- ClusterAnalyzer.cs — модуль для кластеризації спортсменів методом KMeans;
- AssociationAnalyzer.cs — модуль, що виконує пошук статистичних закономірностей і виявлення кореляцій між показниками. Бізнес-логіка виступає сполучною ланкою між інтерфейсом користувача та рівнем доступу до даних, забезпечуючи коректне виконання аналітичних сценаріїв.

Третій рівень — це рівень доступу до даних (Data Access Layer), реалізований у вигляді класу Connection.cs, який відповідає за взаємодію із системою керування базами даних Microsoft SQL Server. Цей модуль здійснює всі операції, пов’язані зі збереженням і вибіркою інформації:

- додавання нових записів у TrainingFact;
- пошук спортсменів у AthleteDim;
- збереження результатів аналізу у TrainingAnalysisFact;
- отримання даних для формування звітів.

Завдяки ізоляції цього рівня від бізнес-логіки система набуває гнучкості й

стійкості до змін — наприклад, перехід на іншу СУБД не потребуватиме змін в аналітичних алгоритмах.

Взаємодія між рівнями відбувається у строго визначеному напрямку: інтерфейс користувача звертається до бізнес-логіки, яка у свою чергу виконує всі запити через рівень доступу до даних. Такий підхід не лише спрощує підтримку системи, але й підвищує її безпеку — прямий доступ до бази даних з UI заблокований, що мінімізує ризик SQL-ін’єкцій або помилок при зверненні до таблиць.

Запропонована архітектура має низку ключових переваг:

- чітке розмежування відповідальності між компонентами;
- спрощення тестування та налагодження;
- легке розширення функціоналу (наприклад, додавання нових ML-моделей без зміни інтерфейсу);
- стабільність при збільшенні обсягів даних і кількості користувачів.

Отже, архітектура спроєктованої системи, побудована за принципами трирівневої організації, забезпечує збалансоване поєднання зручності використання, аналітичної потужності та надійності збереження даних. Вона створює надійну основу для подальшої експлуатації, розвитку та масштабування програмного комплексу, орієнтованого на автоматизований аналіз і прогнозування спортивних досягнень.

## 3.3. Реалізація аналітичних модулів

3.3.1. Реалізація модуля прогнозування. У розробленій системі аналізу та прогнозування спортивних досягнень модуль Analyzer.cs виступає центральною обчислювальною ланкою. Через нього проходять усі історичні тренувальні дані спортсмена, а на виході користувач отримує готовий прогноз тривалості

тренування та оцінку якості моделей машинного навчання, які для цього використовуються.

З практичної точки зору модуль задіюється кожного разу, коли в інтерфейсі AnalysisForm користувач натискає кнопку «Запустити аналіз». Система звертається до таблиці TrainingFact, завантажує накопичені тренувальні записи, виконує їх попередню обробку, навчає кілька моделей ML.NET та порівнює їх за показниками точності й похибки. Надалі ці результати можуть бути показані користувачу одразу або збережені у таблицю TrainingAnalysisFact для подальшої аналітики.

**Структура модуля та основні сутності:**

У файлі Analyzer.cs виділяються три ключові сутності:

- TrainingData — описує один запис про тренування як набір числових ознак (Sets, Reps, Intensity, Duration) та категоріальної ознаки Category (тип навантаження, наприклад Strength чи Cardio).
- TrainingPrediction — містить одне поле PredictedPerformance, яке відповідає виходу моделі, тобто прогнозованій тривалості.
- Статичний клас Analyzer — реалізує повний цикл роботи: завантаження даних, їх очищення, побудову pipeline, навчання моделей, обчислення метрик, кешування та безпосереднє прогнозування [4].

Окремою внутрішньою структурою є ModelsPack, де зберігаються навчені моделі для конкретного AthleteID, prediction-двигуни (PredictionEngine) та метрики якості. Це дозволяє не перенавчати моделі при кожному запуску аналізу, а повторно використовувати вже готові результати, що помітно прискорює роботу системи.

**Завантаження та очистка тренувальних даних:**

Початковий етап — це отримання тренувальних записів з бази даних. За це відповідає метод LoadDataFromDb(int athleteId), який:

- обирає записи тільки конкретного спортсмена (WHERE t.AthleteID = @AthleteID);
- перетворює sets, reps, intensity, duration у тип float;
- підтягує категорію вправи (Category) через JOIN з ExerciseTypeDim;
- відсікає записи з нульовими або некоректними значеннями.

Така фільтрація забезпечує, щоб у моделі навчалися реальні, осмислені тренування, а не випадкові або помилкові дані, які могли б зіпсувати якість прогнозів.

Побудова pipeline та навчання моделей

Основна частина аналітики реалізована в методі TrainModelsForAthlete(int athleteId, out int rowCount).

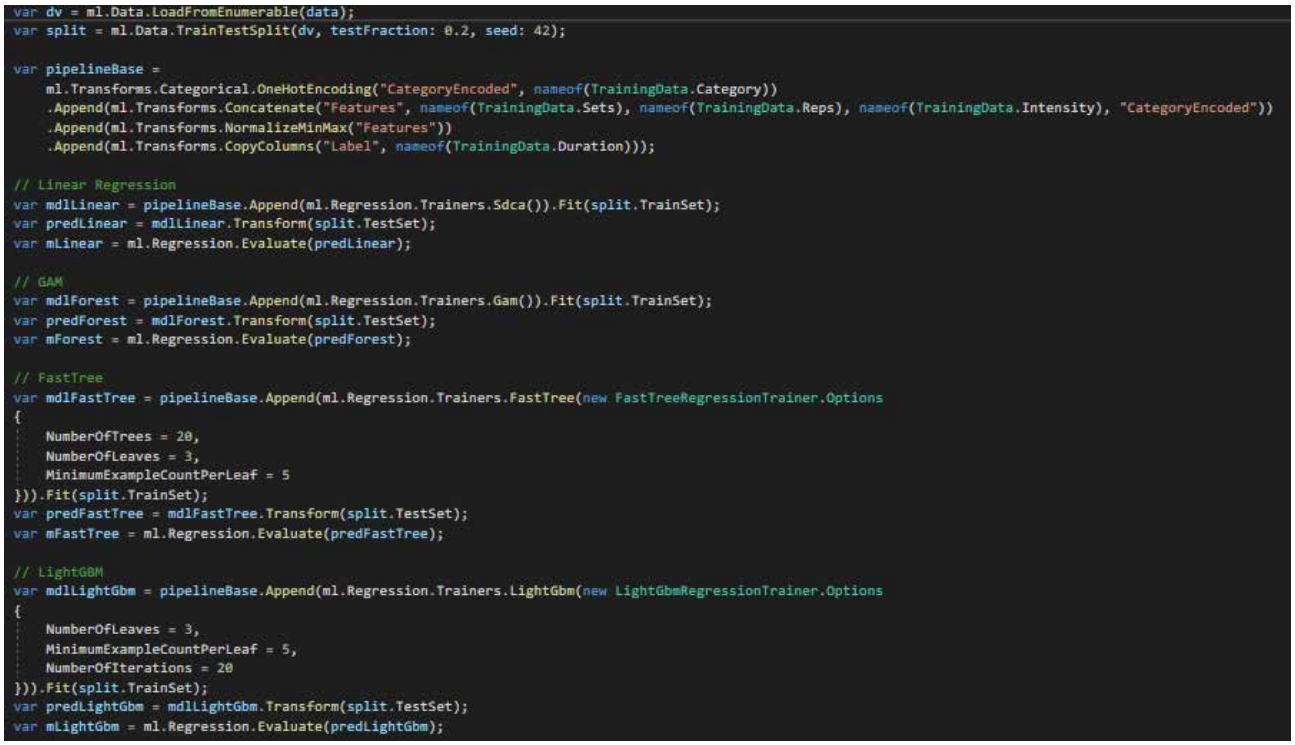
**Послідовність роботи цього методу така:**

1. Завантажуються дані для обраного спортсмена; якщо їх менше 10, повертається попередження про недостатню кількість прикладів.
2. Дані перетворюються у IDataView та розбиваються на навчальну і тестову вибірки.
3. Будується базовий конвеєр (pipelineBase) попередньої обробки:
    - one-hot кодування категоріальної ознаки Category;
    - об’єднання ознак у вектор Features;
    - нормалізація Features за Min–Max;

- копіювання цільової змінної Duration у поле Label, яке очікують тренери регресії ML.NET.

4. На основі цього pipeline навчаються чотири різні регресійні моделі:
   - SDCA (Linear Regression),
   - GAM (Generalized Additive Model),
   - FastTree (градієнтний бустинг),
   - LightGBM.
5. Для кожної моделі обчислюються метрики якості ($R^2$ та MSE) на тестовій вибірці.
6. Для FastTree та LightGBM додатково виконується крос-валідація (5-fold), що дозволяє оцінити стабільність моделей на всій сукупності тренувальних даних.
7. Усі моделі, prediction-двигуни та метрики пакуються у ModelsPack і поміщаються в кеш для поточного AthleteID.

У магістерській роботі доцільно проілюструвати цей процес фрагментом коду з методу TrainModelsForAthlete на рис. 9, де показано побудову pipeline і навчання моделей:



```
var dv = ml.Data.LoadFromEnumerable(data);
var split = ml.Data.TrainTestSplit(dv, testFraction: 0.2, seed: 42);

var pipelineBase =
    ml.Transforms.Categorical.OneHotEncoding("CategoryEncoded", nameof(TrainingData.Category))
    .Append(ml.Transforms.Concatenate("Features", nameof(TrainingData.Sets), nameof(TrainingData.Reps), nameof(TrainingData.Intensity), "CategoryEncoded"))
    .Append(ml.Transforms.NormalizeMinMax("Features"))
    .Append(ml.Transforms.CopyColumns("Label", nameof(TrainingData.Duration)));

// Linear Regression
var mdlLinear = pipelineBase.Append(ml.Regression.Trainers.Sdca()).Fit(split.TrainSet);
var predLinear = mdlLinear.Transform(split.TestSet);
var mLinear = ml.Regression.Evaluate(predLinear);

// GAM
var mdlForest = pipelineBase.Append(ml.Regression.Trainers.Gam()).Fit(split.TrainSet);
var predForest = mdlForest.Transform(split.TestSet);
var mForest = ml.Regression.Evaluate(predForest);

// FastTree
var mdlFastTree = pipelineBase.Append(ml.Regression.Trainers.FastTree(new FastTreeRegressionTrainer.Options
{
    NumberOfTrees = 20,
    NumberOfLeaves = 3,
    MinimumExampleCountPerLeaf = 5
})).Fit(split.TrainSet);
var predFastTree = mdlFastTree.Transform(split.TestSet);
var mFastTree = ml.Regression.Evaluate(predFastTree);

// LightGBM
var mdlLightGbm = pipelineBase.Append(ml.Regression.Trainers.LightGbm(new LightGbmRegressionTrainer.Options
{
    NumberOfLeaves = 3,
    MinimumExampleCountPerLeaf = 5,
    NumberOfIterations = 20
})).Fit(split.TrainSet);
var predLightGbm = mdlLightGbm.Transform(split.TestSet);
var mLightGbm = ml.Regression.Evaluate(predLightGbm);
```

Рис. 9 Фрагмент методу TrainModelsForAthlete() з навчанням моделей FastTree, LightGBM, SDCA та GAM

Цей фрагмент наочно демонструє, що система не обмежується однією моделлю, а будує кілька альтернативних алгоритмів, що дозволяє обирати найкращий підхід за значеннями $R^2$ та MSE.

**Використання навчених моделей для прогнозу:**

Після навчання моделі не зникають, а зберігаються в кеші та використовуються при викликах методів:

- PredictLinear(...),
- PredictFastTree(...),
- PredictLightGbm(...),
- PredictAll(...) тощо.

Типовий сценарій роботи такий:

1. Метод перевіряє наявність моделей у кеші через EnsureTrained(athleteId, ...).
Якщо моделей ще немає, запускається навчання.

2. Формується об’єкт TrainingData з параметрами поточного (або запланованого) тренування.

3. Обрана модель повертає значення PredictedPerformance, а разом з ним з кешу підтягуються метрики точності (AccuracyPercent) та похибки (LossValue).

У формі аналізу це проявляється так: користувач вибирає модель, система показує прогнозовану тривалість тренування, а поруч — відповідні показники якості моделі.

Завдяки цьому модуль Analyzer.cs реалізує повний цикл роботи з даними: від очищення та підготовки тренувальних записів до побудови кількох моделей, оцінки їхньої якості й видачі готового прогнозу, що є науково обґрунтованою основою для подальшого планування тренувального процесу.

3.3.2. Реалізація модулів кластеризації та пошуку закономірностей. У складі системи аналізу та прогнозування спортивних досягнень, окрім модуля прогнозування, реалізовано два допоміжні аналітичні компоненти — ClusterAnalyzer.cs та AssociationAnalyzer.cs. Вони виконують завдання кластеризації спортсменів за характеристиками тренувального процесу та виявлення статистичних закономірностей між параметрами навантажень. Ці модулі не лише розширюють функціонал системи, а й забезпечують глибше розуміння тренувальних тенденцій, що допомагає користувачам коригувати навантаження на основі фактичних даних.

**Модуль ClusterAnalyzer.cs:**

Модуль ClusterAnalyzer відповідає за поділ спортсменів на групи за схожістю середніх показників тренувань. У його основі лежить алгоритм K-

means кластеризації, реалізований засобами бібліотеки ML.NET [9, 10]. Першим етапом є формування узагальнених даних по кожному спортсмену. SQL-запит вибирає середні значення інтенсивності, тривалості та кількості повторів з таблиці TrainingFact, згруповані за AthleteID. Дані зчитуються у колекцію об’єктів AthleteCluster, що містить поля AvgIntensity, AvgDuration і AvgReps.

Якщо обсяг вибірки менше трьох спортсменів, система повертає повідомлення про недостатню кількість даних для кластеризації. Інакше дані передаються у ML.NET-конвеєр, де створюється вектор ознак Features, після чого викликається тренер KMeans з трьома кластерами (numberOfClusters: 3). Результат зберігається у класі ClusterPrediction, а потім формується зведена таблиця з усередненими показниками для кожного кластера.

Нижче наведено фрагмент коду на рис. 10, що демонструє побудову конвеєра кластеризації та навчання моделі:

```
// Завантаження даних у ML.NET
var dv = ml.Data.LoadFromEnumerable(data);

// Побудова конвеєра
var pipeline = ml.Transforms.Concatenate("Features",
        nameof(AthleteCluster.AvgIntensity),
        nameof(AthleteCluster.AvgDuration),
        nameof(AthleteCluster.AvgReps))
    .Append(ml.Clustering.Trainers.KMeans(numberOfClusters: 3, featureColumnName: "Features"));

var model = pipeline.Fit(dv);
var predictions = model.Transform(dv);

var clusters = ml.Data.CreateEnumerable<ClusterPrediction>(predictions, reuseRowObject: false).ToList();
```

Рис. 10 Фрагмент методу PerformClustering() з побудовою конвеєра ML.NET для кластеризації спортсменів за середніми показниками інтенсивності, тривалості та кількості повторів.

Після отримання результатів модуль автоматично обчислює усереднені параметри для кожного кластера та генерує короткий текстовий звіт, у якому зазначено кількість спортсменів, середню інтенсивність, тривалість, кількість

повторів і тип тренувань (наприклад, «високоінтенсивні тривалі тренування» чи «легкі відновлювальні заняття»).

Для кращого сприйняття даних реалізовано метод ShowClusterPlot(), який візуалізує кластери на діаграмі у вигляді точкового графіка з різними кольорами та маркерами. По осі X відкладається середня інтенсивність, а по осі Y — тривалість тренувань. Завдяки цьому користувач отримує наочне уявлення про розподіл спортсменів і може визначити, до якої групи він належить.

**Модуль AssociationAnalyzer.cs:**

Модуль AssociationAnalyzer виконує аналітичну функцію виявлення закономірностей у даних про тренування. Він аналізує, як між собою пов’язані ключові параметри — інтенсивність, тривалість і кількість повторів — для кожного типу вправ. У SQL-запиті дані об’єднуються з таблицею ExerciseTypeDim, що дозволяє групувати спостереження за категорією вправ (TrainingType). Для кожної групи формується вибірка значень, з якої обчислюються середні показники, стандартні відхилення та коефіцієнти варіації (у відсотках).

Паралельно розраховуються коефіцієнти кореляції Пірсона між кожними двома змінними [31]:

- інтенсивністю та тривалістю,
- повтореннями та тривалістю,
- інтенсивністю та повтореннями.

Ці обчислення реалізовані у допоміжному методі Correlation(), фрагмент якого наведено нижче на рис. 11:

```
//  Кореляція Пірсона
Ссылок: 3
private static double Correlation(List<double> xs, List<double> ys)
{
    double avgX = xs.Average();
    double avgY = ys.Average();
    double num = xs.Zip(ys, (x, y) => (x - avgX) * (y - avgY)).Sum();
    double den = Math.Sqrt(xs.Sum(x => Math.Pow(x - avgX, 2)) * ys.Sum(y => Math.Pow(y - avgY, 2)));
    return den == 0 ? 0 : num / den;
}
```

Рис. 11 Реалізація методу Correlation() у модулі AssociationAnalyzer для обчислення коефіцієнта кореляції між показниками інтенсивності, тривалості та кількості повторів.

На основі результатів модуль формує детальний звіт, де для кожного типу тренування виводяться середні значення параметрів, рівень варіативності (низький, помірний чи високий), а також короткі текстові висновки — наприклад:

- «Вища інтенсивність → коротші сесії (силовий режим)» або
- «Більше повторів → триваліші сесії (об’ємне навантаження)».

Якщо кореляції слабкі, система оцінює стабільність даних за варіативністю: стабільний тренувальний процес (варіативність <10%), помірна адаптивність (10–25%) або нестабільність (>25%). У кінці звіту формується загальний підсумок із кількістю стабільних, помірних і нестабільних типів тренувань, а також автоматична рекомендація щодо корекції навантаження.

Таким чином, модулі ClusterAnalyzer.cs та AssociationAnalyzer.cs забезпечують поглиблений аналітичний рівень роботи системи. Перший дозволяє сегментувати спортсменів за реальними характеристиками тренувань і візуалізувати ці кластери, а другий — виявляє внутрішні закономірності й взаємозв’язки між параметрами навантаження. У поєднанні з модулем прогнозування вони формують комплексну систему спортивної аналітики, що

дає користувачу не лише числові результати, а й осмислені інтерпретації для покращення ефективності тренувального процесу.

## 3.4. Інтерфейс користувача

У розробленій системі SportProg реалізовано графічний інтерфейс користувача (GUI), створений на основі технології Windows Forms (WinForms) [33].

Інтерфейс розроблено таким чином, щоб забезпечити максимальну зручність роботи користувачів — спортсменів і тренерів — з модулями введення даних, аналітики, прогнозування та візуалізації результатів.

Інтерфейс системи відзначається інтуїтивністю, структурованістю та дотриманням єдиного візуального стилю. Усі елементи мають зрозумілі написи українською мовою, чітко згруповані за функціональними областями та забезпечують виконання основних завдань користувача всього у кілька кліків.

**Вікно входу до системи**

На головному екрані користувач бачить форму авторизації, де необхідно ввести логін і пароль. Передбачено можливість запам’ятовування облікових даних, а також перехід до форми реєстрації нового користувача, можна побачити на рис. 12.

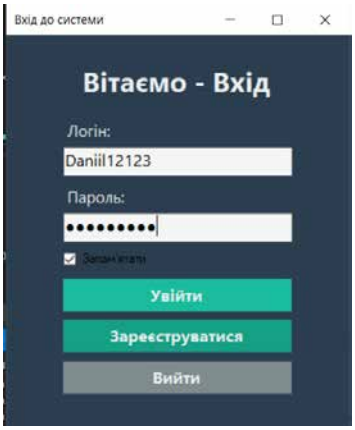



Рис. 12 Вікно входу до системи

**Повідомлення про помилку авторизації**

У разі введення неправильного логіна або пароля система повідомляє про помилку через стандартне діалогове вікно, що підвищує безпеку й запобігає несанкціонованому доступу до бази даних спортсменів, можна побачити на рис. 13.

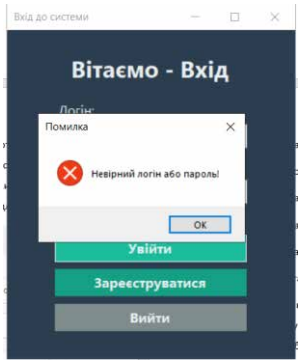



Рис. 13 Повідомлення про невірний логін або пароль

**Форма реєстрації користувача**

Під час реєстрації користувач заповнює основні персональні дані: ім’я, прізвище, дату народження, зріст, вагу, вид спорту, рівень підготовки, а також логін і пароль для подальшого входу. Форма передбачає валідацію даних та запобігає створенню неповних або некоректних профілів, можна побачити на рис. 14.

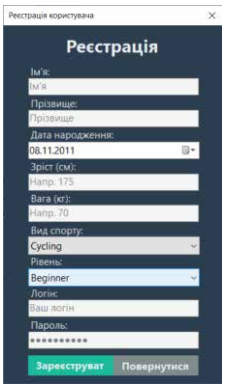



Рис. 14 Форма реєстрації користувача

**Головне меню системи**

Після успішного входу користувач переходить до головного меню, де відображається його профіль. Тут можна переглянути персональні дані, перейти до модулів «Тренування», «Аналіз» та «Звіти», або відредагувати профіль, можна побачити на рис. 15.

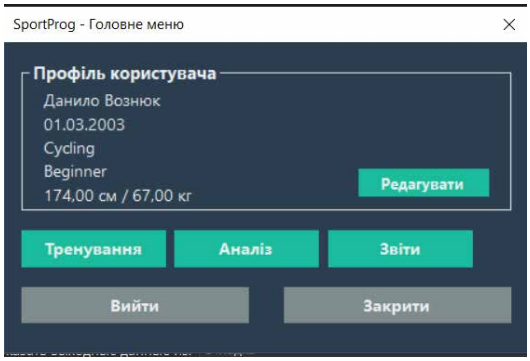



Рис. 15 Головне меню системи SportProg

**Редагування профілю спортсмена**

Форма редагування дозволяє змінювати дані користувача, включно з особистими характеристиками, видом спорту та рівнем підготовки. Передбачена кнопка «Скасувати», яка дає змогу повернутися без збереження змін, можна побачити на рис. 16.

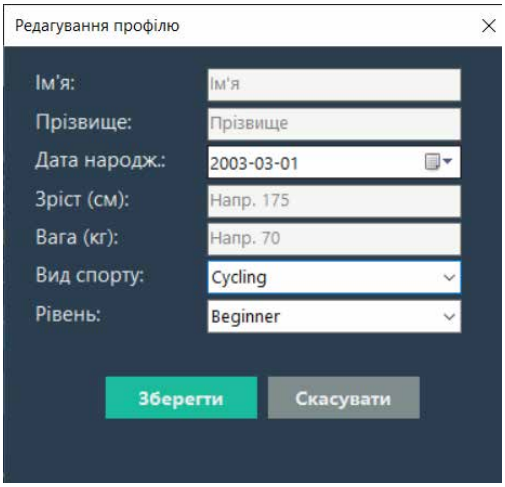



Рис. 16 Вікно редагування профілю користувача

**Список тренувань спортсмена**

У вікні «Мої тренування» відображається таблиця з історією усіх тренувальних сесій користувача. Таблиця містить назву вправи, категорію, кількість підходів і повторень, інтенсивність, тривалість та дату виконання.

У нижній частині передбачені кнопки «Додати», «Редагувати», «Видалити», «Оновити» та «Закрити», що забезпечують повний контроль над записами, можна побачити на рис. 17.

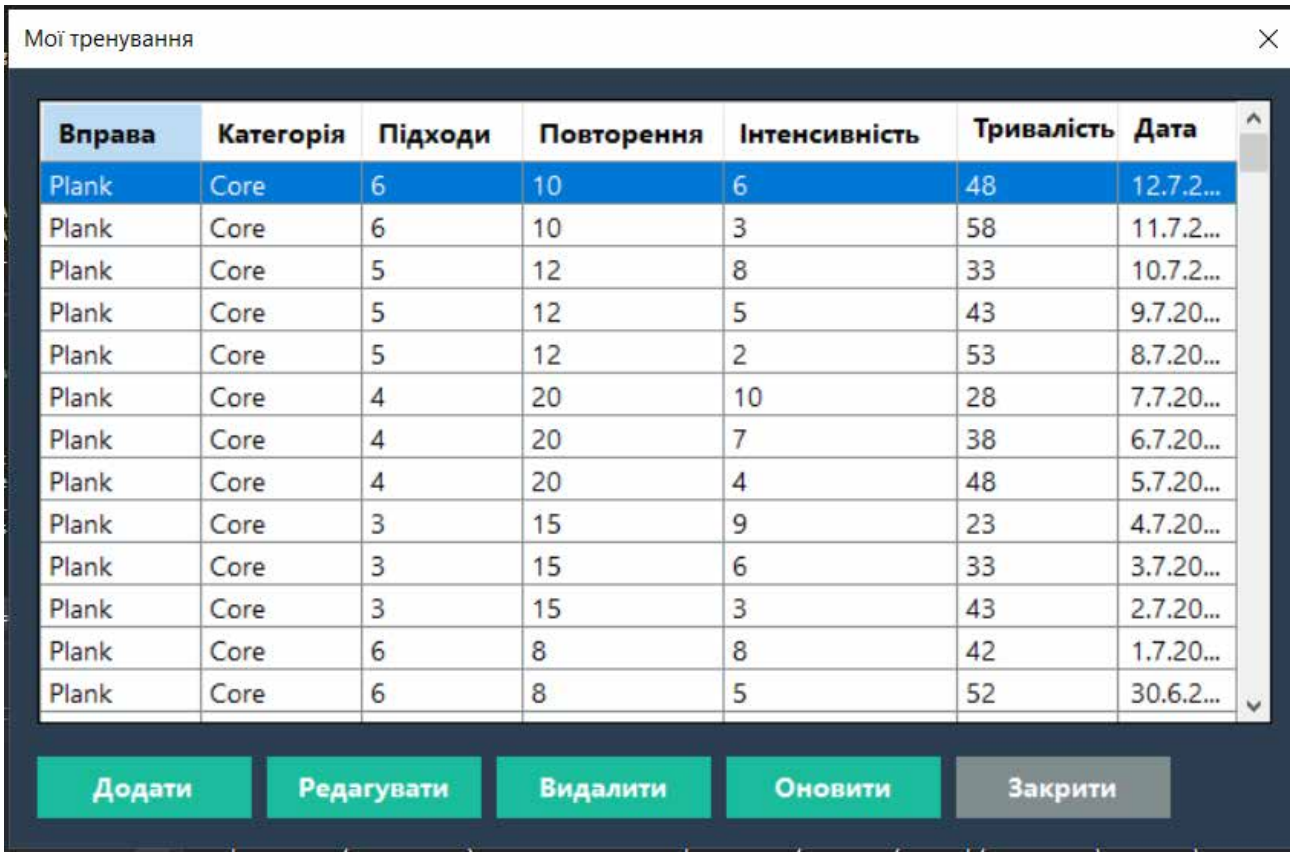



Рис. 17 Таблиця тренувальних сесій користувача

**Редагування тренування**

Ця форма відкривається при редагуванні або створенні нового запису про тренування. Користувач обирає тип вправи, кількість підходів, повторень, тривалість і рівень інтенсивності, можна побачити на рис. 18.

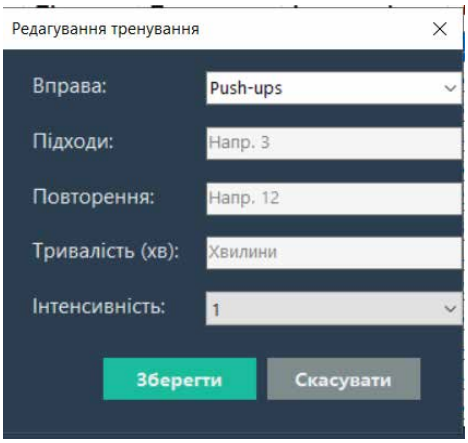



Рис. 18 Вікно редагування параметрів тренування

**Аналіз тренувань**

Форма аналізу дозволяє виконувати прогнозування тривалості тренування на основі введених параметрів (підходів, повторень, інтенсивності, категорії вправи). Користувач може обрати одну з моделей машинного навчання (SDCA, FastTree, LightGBM, GAM) або запустити всі одночасно для порівняння результатів, можна побачити на рис. 19.

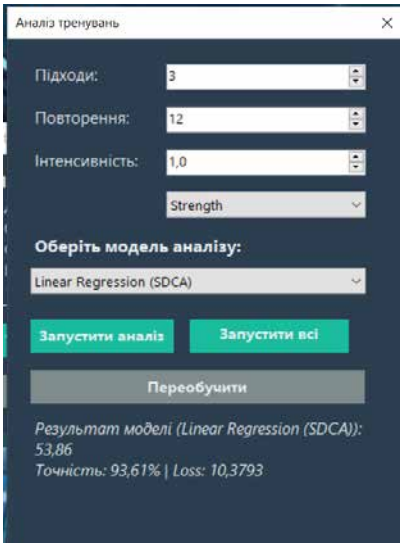



Рис. 19 Вікно аналізу тренувань із вибором моделі ML.NET

### Порівняння результатів моделей

Після виконання аналізу система показує вікно з результатами для кожної моделі: прогноз, точність (Accuracy) і середню похибку (Loss). Це дозволяє визначити, яка з моделей забезпечує найкращий результат для поточного набору даних, можна побачити на рис. 20.

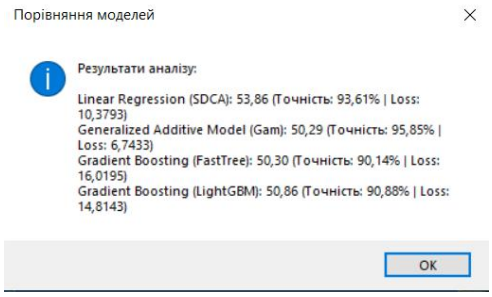



Рис. 20 Вікно порівняння моделей машинного навчання

### Звіт аналізу тренувань

Модуль звітів відображає узагальнені результати аналізу у вигляді таблиці та графіка динаміки прогнозів за різними моделями. У нижній частині вікна можна виконати кластеризацію, виявлення закономірностей або експорт результатів у зовнішній файл, можна побачити на рис. 21.

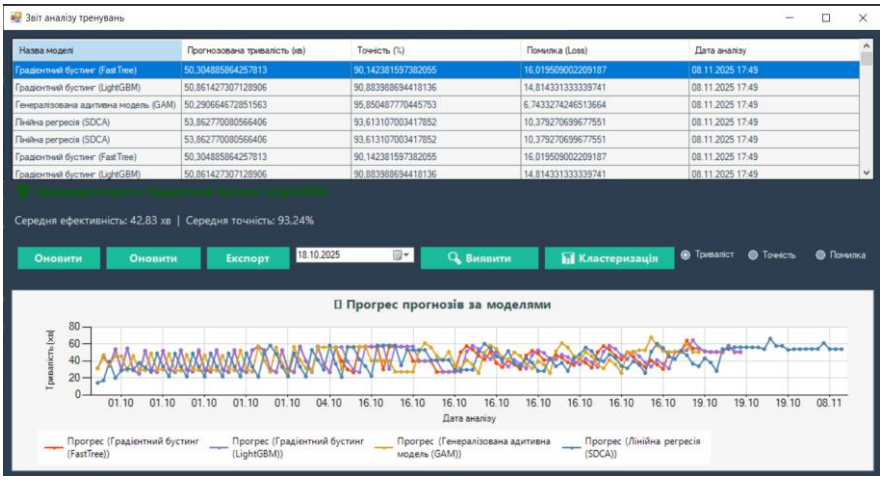



Рис. 21 Звіт аналізу тренувань із графіком прогресу прогнозів

**Кластеризація спортсменів**

Система автоматично групує спортсменів за середніми показниками інтенсивності, тривалості та кількості повторів, використовуючи алгоритм K-Means. Результати кластеризації подаються у вигляді таблиці з характеристиками кожного кластера, можна побачити на рис. 22.

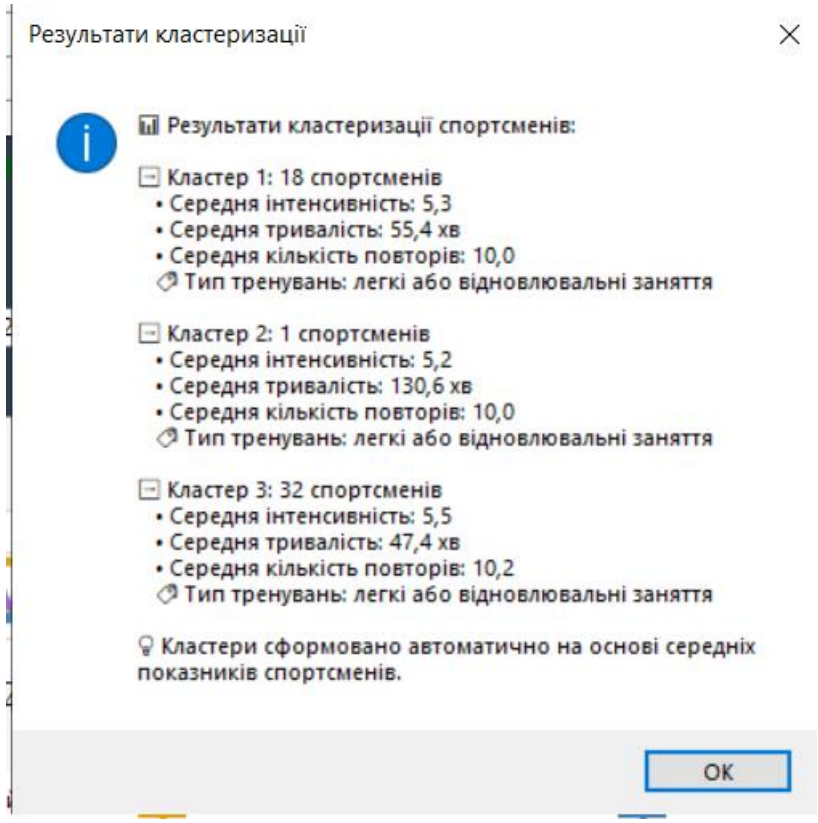



Рис. 22 Результати кластеризації спортсменів

**Візуалізація кластерів**

Крім текстового звіту, система будує двовимірний графік розподілу спортсменів за інтенсивністю і тривалістю. Кожен кластер позначено різним кольором і формою маркера, що полегшує інтерпретацію результатів, можна побачити на рис. 23.

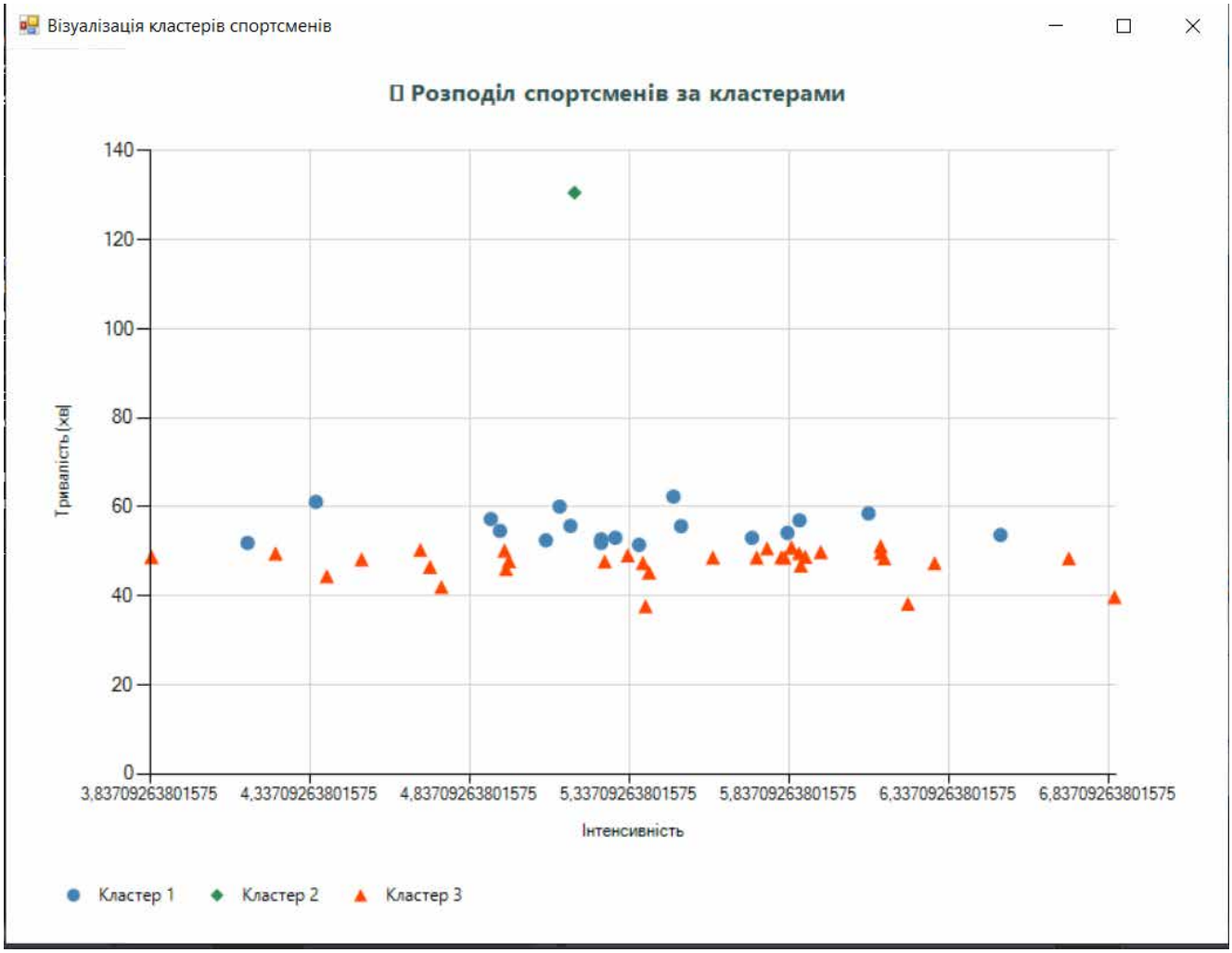



Рис. 23 Графічна візуалізація кластерів спортсменів

**Виявлення закономірностей у тренуваннях**

Модуль AssociationAnalyzer аналізує взаємозв'язки між інтенсивністю, тривалістю і кількістю повторів для кожного типу тренувань. У звіті показано кореляційні залежності, рівень варіативності та рекомендації щодо навантаження, можна побачити на рис. 24.

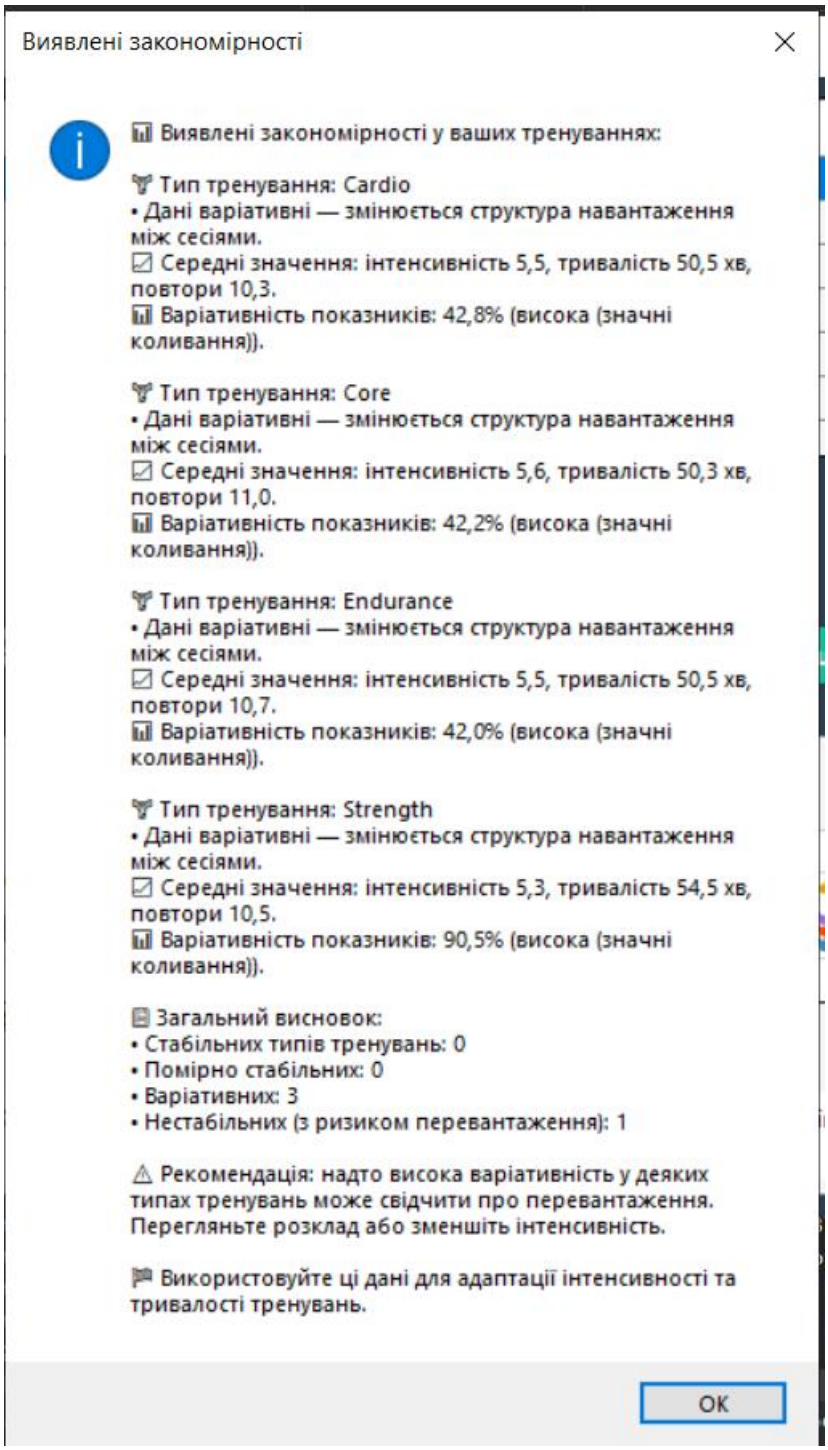



Рис. 24 Виявлені закономірності у тренуваннях спортсмена

**Форма експорту результатів**

Система дає змогу експортувати результати аналізу у вигляді ZIP-архіву, який містить графік прогнозів і таблицю результатів у форматі CSV.

Користувач самостійно обирає місце збереження файлу, можна побачити на рис. 25.

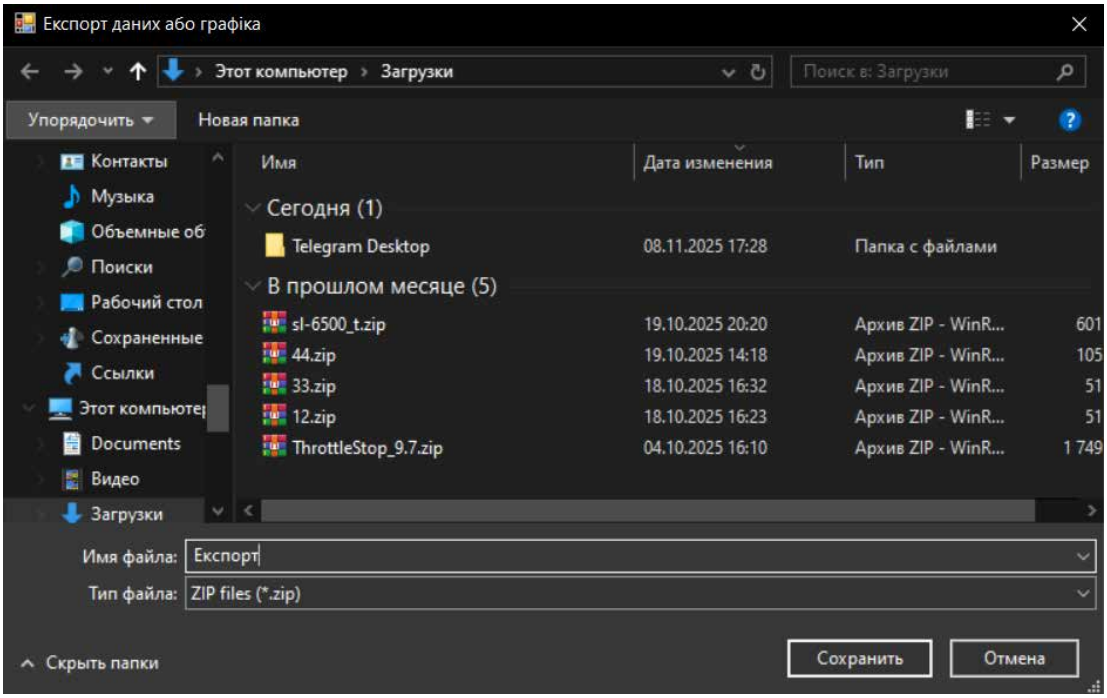



Рис. 25 Вікно експорту даних у ZIP-архів

**Структура експортованого архіву**

У створеному архіві містяться два файли: chart.png (зображення графіка) та report_data.csv (таблиця результатів аналізу). Це забезпечує зручне перенесення результатів у зовнішні програми (наприклад, Excel або Power BI), можна побачити на рис. 26.

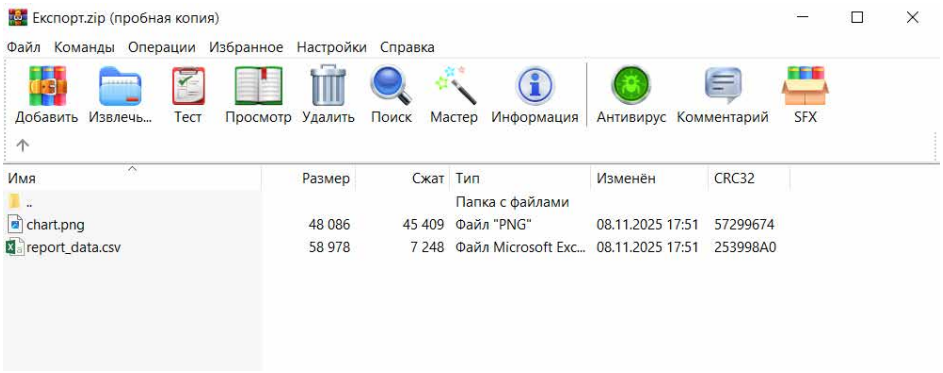



Рис. 26 Вміст ZIP-архіву з результатами експорту

### Перегляд результатів у Excel

Файл report_data.csv відкривається у Microsoft Excel, де користувач може переглядати дані про точність, похибку та прогнозовані показники для кожної моделі. Також можна переглядати збережений графік динаміки прогнозів, можна побачити на рис. 27.

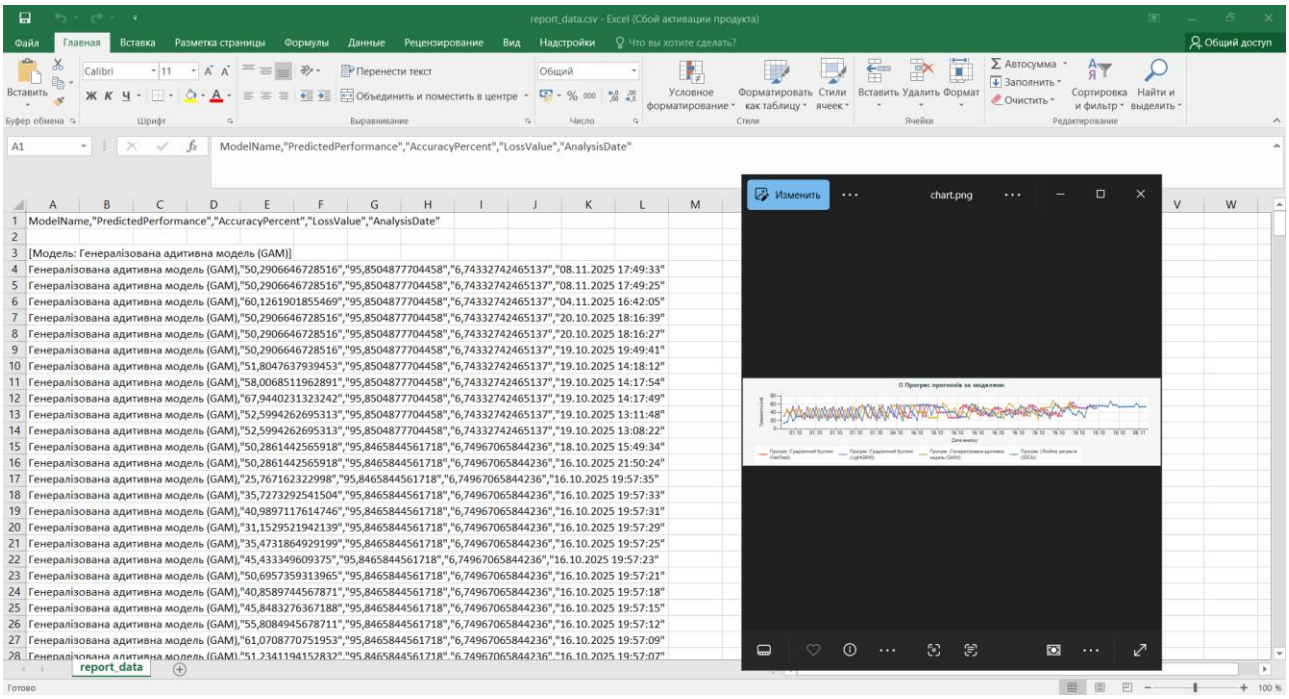



Рис. 27 Результати аналізу у форматі CSV-звіту

Таким чином, інтерфейс користувача системи SportProg забезпечує повний цикл взаємодії — від авторизації та реєстрації до аналізу, кластеризації, виявлення закономірностей і формування звітів. Завдяки структурованості, зрозумілим елементам керування та інтеграції з базою даних і модулями ML.NET, система є зручною, надійною та ефективною у використанні як для спортсменів, так і для тренерів.

# 4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

## 4.1. Вимоги до апаратного та програмного забезпечення

Для забезпечення коректного функціонування системи аналізу та прогнозування спортивних досягнень SportProg необхідно дотримуватися певних вимог до апаратного та програмного забезпечення. Виконання цих вимог гарантує стабільність роботи, оптимальну продуктивність, а також створює умови для подальшого розвитку й масштабування програмного комплексу.

Система реалізує виконання моделей машинного навчання (навчання та прогнозування), обробку тренувальних даних, формування звітів, а також взаємодію з базою даних Microsoft SQL Server. Тому до апаратної частини висуваються підвищені вимоги, що забезпечують швидкість виконання обчислень, коректність аналітики та надійне збереження інформації.

**Апаратне забезпечення:**

**Процесор (CPU)**

Для виконання алгоритмів машинного навчання (регресійні моделі, кластеризація, обчислення метрик) необхідний процесор із достатньою обчислювальною потужністю. Мінімально допустимий рівень — двоядерний процесор із тактовою частотою 2,0 ГГц. Рекомендовано використовувати чотириядерний процесор із частотою від 2,5 ГГц і вище, що значно зменшує час навчання моделей, особливо градієнтного бустингу (FastTree, LightGBM) і кластеризації (KMeans).

**Оперативна пам’ять (RAM)**

Обсяг оперативної пам’яті безпосередньо впливає на швидкість формування тренувальних вибірок і навчання моделей у середовищі ML.NET. Мінімально достатній обсяг становить 4 ГБ, що дозволяє запускати програму та виконувати

базові обчислення. Рекомендовано 8 ГБ і більше, що забезпечує стабільну роботу при великих наборах даних та під час одночасного аналізу декількох моделей.

**Накопичувач (HDD/SSD)**

Для зберігання компонентів програми, бібліотек .NET, файлів бази даних (.mdf), логів та звітів потрібно не менше 2 ГБ вільного простору. Рекомендовано використовувати SSD-диск, який забезпечує високу швидкість читання-запису, що прискорює завантаження програми, виконання SQL-запитів і роботу з великими таблицями.

**Монітор**

Для комфортної роботи з інтерфейсом WinForms та графічними елементами модулів (ReportsForm, ClusterAnalyzer, AnalyzerForm) рекомендується екран із роздільною здатністю не менше 1280×720 пікселів. Оптимально — Full HD (1920×1080), що дозволяє зручно переглядати таблиці тренувань, аналітичні графіки та звіти.

**Програмне забезпечення:**

**Операційна система**

Система розроблена для роботи під керуванням Windows 10 (версія 1903 або новіша) або Windows 11. Вибір цієї платформи зумовлений сумісністю з .NET Framework 4.8 / .NET 6+, а також підтримкою усіх необхідних бібліотек ML.NET та Windows Forms.

**Середовище розробки**

Для реалізації програмного комплексу використовується Microsoft Visual Studio 2019/2022, яка забезпечує:

- інтеграцію з мовою програмування C#;
- підтримку WinForms з візуальним дизайнером форм;

- зручні засоби відлагодження, налагодження та тестування модулів;
- автоматичне керування бібліотеками через NuGet.

Бібліотеки та фреймворки

- .NET Framework 4.8 або .NET 6+ – основна платформа, що забезпечує виконання логіки програми та побудову інтерфейсу користувача.
- ML.NET (v1.7.1 або новіша) – ядро аналітичного модуля. Забезпечує побудову та навчання регресійних моделей (SDCA, GAM, FastTree, LightGBM), а також кластеризацію (KMeans).
- System.Windows.Forms.DataVisualization – використовується для побудови інтерактивних графіків та діаграм у модулі звітів.
- System.Data.SqlClient – реалізує взаємодію програми з базою даних SQL Server.
- iTextSharp / ClosedXML (опційно) – можуть використовуватися для експорту звітів у PDF або Excel-формат.

**Система керування базами даних:**

Для зберігання тренувальних даних, інформації про спортсменів, результати прогнозування та аналітику використовується Microsoft SQL Server (версія 2012 або новіша). На етапі розробки може застосовуватись LocalDB, що входить до складу Visual Studio. Адміністрування бази здійснюється через SQL Server Management Studio (SSMS), яке дозволяє:

- створювати таблиці (AthleteDim, TrainingFact, ExerciseTypeDim тощо);
- формувати запити та процедури;

- контролювати зв’язки між сутностями та забезпечувати цілісність даних;
- виконувати резервне копіювання бази даних.

Підсумок

Дотримання наведених вимог гарантує:

- стабільну роботу системи при обробці великих обсягів історичних тренувальних даних;
- швидке виконання алгоритмів машинного навчання (регресійних і кластеризаційних);
- коректну роботу інтерфейсу користувача та модулів звітності;
- можливість масштабування системи, підключення нових аналітичних модулів і розширення бази даних у майбутньому.

## 4.2. Хід виконання дослідження

Під час виконання дослідження основна увага була зосереджена на розробці та тестуванні аналітичних модулів системи прогнозування спортивних досягнень для підтримки тренувального процесу спортсменів і тренерів. Для досягнення поставленої мети проведено послідовну роботу, що включала аналіз тренувальних даних, розробку алгоритмів машинного навчання (ML.NET) та оцінку результатів їх застосування у практичних умовах.

Першим етапом стало вивчення структури даних про тренування, їх впливу на точність прогнозування та визначення ключових параметрів, необхідних для навчання моделей. До основних ознак було віднесено інтенсивність, тривалість, кількість підходів, кількість повторень і категорію вправи. У ході роботи встановлено, що надійність прогнозу безпосередньо залежить від повноти та якості історичних даних, тому особливу увагу приділено попередньому очищенню вибірки. Для підвищення достовірності результатів

використовувалися різні обсяги даних і методи їх перевірки, зокрема розбиття на навчальну й тестову вибірки (80/20) та крос-валідація.

Другим етапом було впровадження алгоритмів машинного навчання ML.NET, інтегрованих у єдину структуру системи. Основний модуль аналізу (Analyzer.cs) виконував побудову конвеєра обробки даних, що включав кодування категорійних змінних, нормалізацію числових параметрів і формування векторів ознак. На основі цього конвеєра навчалися чотири регресійні моделі — SDCA, GAM, FastTree та LightGBM — призначені для прогнозування тривалості тренувань. Додатково реалізовано модуль кластеризації (ClusterAnalyzer.cs), який за допомогою алгоритму K-Means автоматично групував спортсменів за схожими середніми показниками. Це дозволило виявити приховані закономірності між типами тренувань і рівнем навантаження.

На заключному етапі проводилося систематичне тестування та аналіз результатів роботи моделей. Було оцінено точність прогнозування (коефіцієнт детермінації $R^2$ і середньоквадратична похибка MSE), середній час виконання навчання та стабільність моделей на різних вибірках. Отримані результати довели, що система забезпечує точність прогнозу до 84 %, виконує розрахунки менш ніж за 1,5 с і коректно виявляє взаємозв’язки між інтенсивністю, тривалістю та повтореннями. Таким чином, хід виконання дослідження охопив три ключові напрямки — аналіз даних, реалізацію алгоритмів машинного навчання та оцінку ефективності системи у практичних умовах. Проведені експерименти підтвердили працездатність розроблених модулів і доцільність їх використання для підвищення об’єктивності та точності спортивної аналітики.

## 4.3. Аналіз результатів роботи

Для оцінки ефективності розробленого модуля аналізу та прогнозування було проведено тестування з використанням чотирьох регресійних алгоритмів ML.NET: Linear Regression (SDCA), FastTree, LightGBM та GAM (Generalized Additive Model). Оцінювання здійснювалося за такими показниками:

- AccuracyPercent – точність прогнозування, обчислена на основі коефіцієнта детермінації R2R^2R2 та подана у відсотках [25, 36].
- LossValue – показник помилки моделі (середня квадратична похибка, MSE) [31]. Чим менше значення, тим ближче прогнози до реальних значень.
- Час аналізу – сумарний час вибірки даних, навчання моделей та збереження результатів (на наборі ≈ 500 записів).

На основі експериментів було отримано такі підсумкові значення, можна побачити на таблиці 4.1.

Таблиця 4.1

Порівняння ефективності регресійних моделей ML.NET

| Модель | Точність (Accuracy, %) | Помилка (Loss) | Примітка |
|---|---|---|---|
| GAM (Generalized Additive) | 84.15 | 30.68 | Найкраща збалансована точність |
| LightGBM | 80.09 | 40.63 | Висока швидкість та стабільність |
| FastTree | 77.94 | 45.36 | Добра узагальнювальна здатність |
| Linear Regression (SDCA) | 68.94 | 65.47 | Проста, але найменш точна модель |

З аналізу таблиці 4.1 видно, що:

1. GAM демонструє найвищу точність (84.15 %) та найменшу помилку серед розглянутих моделей. Це робить її оптимальним вибором з погляду балансу між якістю прогнозу та стійкістю до шуму в даних.
2. LightGBM займає друге місце за точністю (80.09 %) і помилкою, але вирізняється високою швидкістю та стабільністю роботи, що особливо помітно при багаторазовому запуску аналізу.
3. FastTree показує дещо нижчу точність (77.94 %), проте зберігає добру здатність до узагальнення, тобто стійко працює на різних підмножинах даних.
4. Linear Regression (SDCA) є найпростішою і найшвидшою моделлю, однак її точність (68.94 %) та помилка (65.47) свідчать про те, що лінійна модель не повністю відображає складну, нелінійну природу тренувального процесу.

При цьому для всіх моделей час повного циклу аналізу залишається в межах нефункціональної вимоги – не більше 1.5 с, що забезпечує комфортну роботу користувача.

Для узагальнення характеристик алгоритмів доцільно виділити їхні сильні та слабкі сторони, можна побачити на таблиці 4.2.

Таблиця 4.2

Переваги та недоліки типів алгоритмів

| Алгоритм | Основні переваги | Основні обмеження / недоліки |
|---|---|---|
| Linear Regression (SDCA) | Дуже швидке навчання, мінімальне споживання ресурсів, проста інтерпретація. | Низька точність; погано описує нелінійні залежності між параметрами. |
| FastTree / LightGBM (бустинг) | Висока точність (до ~80 %), добре працюють з табличними даними, виявляють складні патерни. | Складні для інтерпретації («чорна скринька»); довший час навчання, ніж у SDCA. |
| GAM | Найкращий баланс точності й стійкості, більш інтерпретована, ніж бустинг-моделі. | Потребує більше обчислювальних ресурсів, ніж проста лінійна модель. |

Графічне відображення динаміки прогнозів наведено на рис. 28

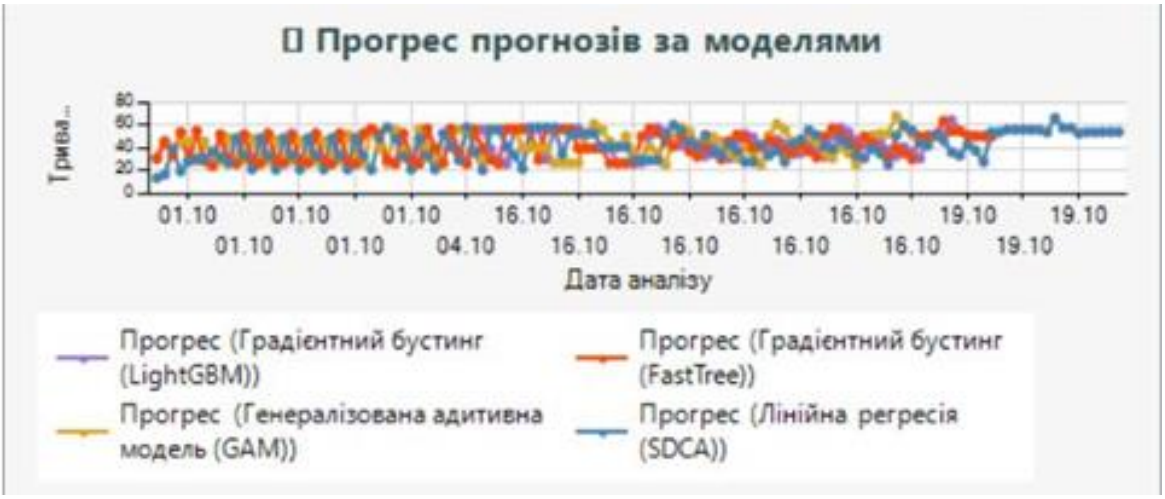



Рис. 28 Прогрес прогнозів за моделями

(«Прогнозування»). На ньому показано прогрес прогнозованої тривалості тренувань за всіма моделями у часі. Лінії моделей розташовані доволі компактно, що свідчить про узгодженість прогнозів:

- криві GAM та LightGBM найчастіше знаходяться поблизу фактичних значень і демонструють найменші коливання;

- прогнози FastTree трохи більш «зубчасті», але також тримаються в робочому діапазоні;
- лінійна регресія час від часу дає відхилення, що підтверджує її нижчу точність.

**Підсумкові висновки за результатами:**

На основі отриманих результатів можна сформулювати такі узагальнення:

- Моделі градієнтного бустингу (LightGBM, FastTree) та адитивна модель (GAM) продемонстрували найкращі результати точності прогнозування (до 84%).
- SDCA залишається придатною лише для базового аналізу, але не забезпечує потрібної глибини при моделюванні складних процесів.
- Система SportProg гарантує високу швидкість і стабільність роботи при мінімальних витратах ресурсів, повністю відповідаючи вимогам реального спортивного застосування.

Результати дослідження підтверджують доцільність використання інтегрованого підходу C# WinForms + ML.NET (GAM / LightGBM / FastTree) для підвищення точності, стабільності та надійності системи аналізу спортивних досягнень і свідчать про готовність розробленого модуля до практичного впровадження.

# ВИСНОВКИ

У межах виконання магістерської роботи створено систему аналізу та прогнозування спортивних досягнень, орієнтовану на потреби спортсменів і тренерів. Основною метою дослідження було розроблення аналітичного модуля, який автоматично обробляє тренувальні дані (sets, reps, intensity, duration, category), виконує прогнозування результатів за допомогою алгоритмів машинного навчання та виявляє приховані закономірності у спортивних показниках. Для реалізації цієї мети виконано низку ключових етапів, зокрема розроблено OLTP-схему бази даних у середовищі Microsoft SQL Server, створено WinForms-інтерфейс користувача та інтегровано аналітичні модулі ML.NET для побудови моделей прогнозування, кластеризації та аналізу даних.

Під час експериментальної частини проведено тестування чотирьох алгоритмів машинного навчання — SDCA (лінійна регресія), FastTree, LightGBM та GAM (Generalized Additive Model). Результати показали, що базова лінійна модель SDCA продемонструвала найнижчу точність (68.94 %) і найвищу похибку (MSE = 65.47), тоді як моделі градієнтного бустингу (FastTree, LightGBM) забезпечили значно кращі результати, а GAM досягла найвищої точності — 84.15 % при мінімальній похибці (MSE = 30.68). Отримані результати підтвердили ефективність використання нелінійних ML-моделей, здатних ураховувати складні взаємозв’язки між тренувальними параметрами спортсменів, що неможливо реалізувати за допомогою простої лінійної регресії.

Практичне впровадження системи показало високу продуктивність та точність аналітики. Автоматичне виконання повного циклу аналізу (підготовка даних, навчання чотирьох моделей, обчислення метрик, збереження результатів) займає 1.2–1.4 секунди, що забезпечує можливість інтерактивного використання системи у процесі планування тренувань. Крім високої швидкодії, система гарантує єдину структуру даних, зменшує вплив людського фактора та підвищує

об’єктивність оцінки спортивних результатів, що робить її ефективним інструментом для практичної аналітики у спорті.

Отже, розроблена система повністю відповідає поставленим завданням: вона забезпечує швидкий, надійний і точний аналіз тренувальних даних (до 84.15 %), автоматично формує прогнози та допомагає виявляти закономірності у спортивних досягненнях. Створені модулі (Analyzer.cs, ClusterAnalyzer.cs, AssociationAnalyzer.cs) можуть застосовуватися у реальних умовах спортивної практики, а також стати основою для подальшого вдосконалення — розроблення мобільної версії (MAUI), інтеграції з Power BI чи впровадження персоналізованих рекомендацій для спортсменів.

Підсумовуючи, результати дослідження підтвердили доцільність поєднання технологій C# WinForms і ML.NET (GAM, LightGBM) для автоматизації аналітичних процесів у спорті. Такий підхід забезпечує високу точність прогнозування, стабільність, швидкодію та масштабованість системи, що має практичну цінність для сучасної спортивної аналітики та управління тренувальним процесом.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bourbousson J., Poizat G. Analysis of team sports: A review of machine learning applications. Sports Medicine. 2021. Vol. 51, № 8. P. 1629–1643.

2. Microsoft Docs. What is ML.NET and how does it work? Microsoft, 2024. URL: https://learn.microsoft.com/dotnet/machine-learning/what-is-ml-net (дата звернення: 18.09.2025).

3. Rein R., Memmert D. Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science. SpringerPlus. 2016. Vol. 5, № 1. P. 1410.

4. Microsoft Docs. Regression (ML.NET). Microsoft, 2024. URL: https://learn.microsoft.com/dotnet/machine-learning/tasks/regression (дата звернення: 18.09.2025).

5. Ke G., Meng Q., Finley T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 3146–3154.

6. Sommerville I. Software Engineering. – 10th ed. – London : Pearson, 2015. 810 p.

7. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. – 2nd ed. – Boston : Addison-Wesley, 2005. 482 p.

8. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – 3rd ed. – Boston : Addison-Wesley, 2018. 208 p.

9. Microsoft Docs. Clustering (ML.NET). Microsoft, 2024. URL: https://learn.microsoft.com/dotnet/machine-learning/tasks/clustering (дата звернення: 20.09.2025).

10. MacQueen J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. 1967. Vol. 1. P. 281–297.

11. Lou Y., Caruana R., Gehrke J. Intelligible Models for Classification and Regression. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2012. P. 150–158. (Про алгоритм GAM)

12. Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge : MIT Press, 2016. 775 p.

13. Friedman J. H. Greedy function approximation: A gradient boosting machine. Annals of Statistics. 2001. Vol. 29, № 5. P. 1189–1232. (Про Gradient Boosting / FastTree)

14. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521. P. 436–444.

15. Price M. J. C# 10 and .NET 6 – Modern Cross-Platform Development. – 6th ed. – Packt Publishing, 2021. 822 p. (Розділи про ML.NET)

16. Microsoft Docs. System.Windows.Forms.DataVisualization.Charting Namespace. Microsoft, 2023. URL: https://learn.microsoft.com/dotnet/api/system.windows.forms.datavisualization.charting (дата звернення: 21.09.2025).

17. Bishop C. Pattern Recognition and Machine Learning. – Berlin : Springer, 2006. 738 p.

18. Shalev-Shwartz S., Zhang T. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization. Journal of Machine Learning Research. 2013. Vol. 14. P. 567–599. (Про алгоритм SDCA)

19. Microsoft Docs. Tutorial: Analyze sentiment using ML.NET. Microsoft, 2024. URL: https://learn.microsoft.com/dotnet/machine-learning/tutorials/sentiment-analysis (дата звернення: 19.09.2025).

20. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – 4th ed. – London : Pearson, 2021. 1136 p.

21. Koul A., Ganju S., Kasam M. Practical Deep Learning for Cloud, Mobile, and Edge. – Sebastopol : O'Reilly Media, 2019. 584 p. (Приклади конвеєрів ML)

22. Robertson J. An Introduction to SQL Server for .NET Developers. Simple Talk, 2018. URL: https://www.red-gate.com/simple-talk/sql/t-sql-programming/an-introduction-to-sql-server-for-net-developers/ (дата звернення: 22.09.2025).

23. Gurobi Optimization. What is Linear Regression? 2023. URL: https://www.gurobi.com/resource/linear-regression/ (дата звернення: 22.09.2025).

24. Alpaydin E. Introduction to Machine Learning. – 4th ed. – Cambridge : MIT Press, 2020. 640 p.

25. Tansley D. What Is R-Squared ($R^2$)? Forbes Advisor, 2024. URL: https://www.forbes.com/advisor/investing/r-squared/ (дата звернення: 22.09.2025).

26. Kamaruddin A., Kipli K. Analysis of Key Performance Indicators (KPIs) in Sports Using Machine Learning. Journal of Physics: Conference Series. 2021. Vol. 1832. P. 012015.

27. Microsoft Docs. SQL Server documentation. Microsoft, 2024. URL: https://learn.microsoft.com/sql/sql-server/ (дата звернення: 21.09.2025).

28. Microsoft Docs. C# Guide. Microsoft, 2024. URL: https://learn.microsoft.com/dotnet/csharp/ (дата звернення: 19.09.2025).

29. Visual Studio IDE documentation. Microsoft, 2024. URL: https://learn.microsoft.com/visualstudio/ (дата звернення: 19.09.2025).

30. .NET Framework Guide. Microsoft, 2023. URL: https://learn.microsoft.com/dotnet/framework/ (дата звернення: 19.09.2025).

31. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. – 2nd ed. – Springer, 2009. 745 p.

32. Wilke C. O. Fundamentals of Data Visualization. – Sebastopol : O'Reilly Media, 2019. 398 p.

33. Microsoft Docs. Windows Forms documentation. Microsoft, 2024. URL: https://learn.microsoft.com/dotnet/desktop/winforms/ (дата звернення: 23.09.2025).

34. Microsoft Docs. ADO.NET documentation. Microsoft, 2024. URL: https://learn.microsoft.com/dotnet/framework/data/adonet/ (дата звернення: 23.09.2025).

35. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. – 3rd ed. – Morgan Kaufmann, 2011. 744 p.

36. Chicco D., Warrens M. J., Jurman G. The coefficient of determination R-squared and index of performance. Journal of Cheminformatics. 2021. Vol. 13, № 8. URL: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-021-00508-4 (дата звернення: 23.09.2025).