НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

004.4:796.015

«ПОГОДЖЕНО»
Декан факультету
інформаційних технологій
Болбот І. В., д. т. н., проф.
_____
____ _____ 2024 р.

«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»
Завідувач кафедри комп'ютерних наук
Голуб Б.Л., к.т.н., доцент
_____
____ _____ 2024 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Програмне забезпечення інтелектуальної системи моніторингу параметрів людини при фітнес-тренуванні»

Спеціальність 121 – Інженерія програмного забезпечення
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
д.т.н., проф. _____ Семко В. В
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____ _____ Бородкіна І.Л.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Ольчедаєвський Д.Ю.
(підпис) (ПІБ студента)

КИЇВ-2024

Зміст

ВСТУП

На даний час фітнес індустрія займає важливу частину повсякденного життя ледь не кожної людини. Відповідно стає все більше систем моніторингу параметрів людини які прогнозують та радять кращі варіанти фітнес занять. **Актуальність** даної роботи полягає в тому, що програмна розробка спираючись на попередні результати тренувань та медичні показники пропонує максимально інтенсивне тренування для максимального спалення калорій яке не повинне зашкодити здоров'ю.

Об'єктом дослідження виступають показники життєдіяльності людини при фітнес тренуванні.

Предметом дослідження виступає система прогнозування та рекомендацій тренувань.

Мета досліження – створення ПЗ для рекомендацій тренувань та порівняння різних моделей передбачення.

Завдання:

— провести системний аналіз,

— сформулювати вимоги,

— побудувати моделі предметної області,

— побудувати архітектуру системи дослідження,

— сформулювати отримані результати і висновки

Методи дослідження. Для досліджень використані 4 моделі прогнозування, а саме модель лінійної регресії, метод поліноміальної регресії, метод найближчого сусіда та метод кубічних сплайнів.

Наукова новизна. В даній роботі вперше було досліджено придатність вищезгаданих чотирьох методів для рекомендацій тренувань.

Апробація результатів дослідження. Було прийнято участь у двох конференціях: XIV та XV МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА

КОНФЕРЕНЦІЯ МОЛОДИХ ВЧЕНИХ «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА»

Структура роботи. Робота складається з 4 розділів, висновків та додатків. В першому розділі проаналізована предметна область та актуальні мови та інструменти реалізації задачі. В другому розділі описано модель системи, її структуру класів та проектування графічного інтерфейсу. В третьому розділі описано особливості програмної реалізації системи. В четвертому розділі описано результати досліджень та висновки на основі цих результатів.

Обсяг роботи 62 стр., 15 рис., 13 лістингів, 21 посилань на літературні джерела.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА АКТУАЛЬНИХ ІНСТРУМЕНТІВ РОЗРОБКИ ПЗ

1.1 Аналіз предметної області

Аналіз предметної області фітнес-тренувань зосереджується на управлінні даними про фізичну активність користувачів, їхні тренування та результати. Фітнес-система, як правило, має на меті відстежувати прогрес користувачів, допомагати їм досягати своїх спортивних цілей і надавати зворотній зв’язок щодо їхніх досягнень.

Кожен користувач має базові персональні характеристики, такі як вік, стать і рівень фізичної підготовки. Ці дані важливі для правильного аналізу тренувань і результатів, адже вони впливають на те, як організм реагує на фізичне навантаження. Наприклад, для обчислення безпечної частоти серцевих скорочень важливо враховувати вік, оскільки максимальний пульс знижується з роками.

Кожна людина може мати свої власні цілі, які система повинна враховувати. Одні користувачі можуть прагнути досягти високих результатів у силових вправах, інші – покращити витривалість або схуднути. Система має адаптуватися до цих різних цілей, відстежуючи відповідні показники: вагу для силових тренувань, швидкість та дистанцію для кардіо-навантажень, кількість спалених калорій для загального контролю.

Система дозволяє кожному користувачу відстежувати свій прогрес у різних видах тренувань. Це можуть бути рекорди в бігу, велотренуваннях, підйомі ваги або інших вправах. Регулярне оновлення даних про тренування

допомагає користувачам бачити, як вони покращують свої результати з часом, і мотивує їх до подальших досягнень.

Для забезпечення безпеки система має враховувати фізіологічні особливості кожного користувача, зокрема, частоту серцевих скорочень. Наприклад, під час тренувань система може попереджати користувача, якщо його пульс перевищує безпечний рівень, що може бути небезпечним для здоров'я. Ці попередження важливі особливо для людей старшого віку або тих, хто має серцево-судинні проблеми.

Фітнес-система також може адаптувати тренувальні програми відповідно до індивідуальних потреб користувача. Якщо користувач ставить за мету підвищити силу, система може рекомендувати більше силових вправ. Якщо метою є покращення витривалості, акцент робиться на кардіо-тренуваннях. Це персоналізований підхід, який допомагає користувачам досягати своїх цілей ефективніше.

1.2. Аналіз актуальних мов програмування

Оскільки буде реалізовуватися прикладний додаток для віндовс розглянемо основні мови програмування які використовуються для написання таких додатків.

Найпопулярнішими на даний час є

— C++

— C#

— Java

— Python

C++ — це потужна, універсальна мова програмування, розроблена Б'ярном Страуструпом у 1980-х роках як розширення мови C. Вона поєднує низькорівневі можливості C з об'єктно-орієнтованими підходами, що дозволяє створювати програми різної складності, від операційних систем до ігор та наукових

симуляцій. Одна з головних переваг C++ — це контроль над апаратними ресурсами, що робить її ідеальною для задач, де продуктивність та ефективність є критично важливими [1].

C++ підтримує різні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване та навіть функціональне програмування. Це дозволяє програмістам вибирати стиль, який найкраще підходить для конкретної задачі. Наприклад, об'єктно-орієнтоване програмування дозволяє структурувати програму на класи та об'єкти, що сприяє кращій організації коду, його повторному використанню та легшому супроводу [2].

Мова C++ також відома своєю системою управління пам'яттю. Вона дозволяє програмістам працювати з динамічним виділенням пам'яті за допомогою операторів `new` і `delete`. Це забезпечує гнучкість, але вимагає від програміста ретельного контролю за використанням ресурсів, щоб уникнути таких проблем, як витік пам'яті. Разом із тим, в новіших версіях C++ з’явилися засоби для автоматизації цього процесу, такі як розумні вказівники (smart pointers), що значно спрощують роботу з пам'яттю [3].

C# — це сучасна об'єктно-орієнтована мова програмування, розроблена компанією Microsoft у 2000 році як частина платформи .NET. Вона була створена з метою поєднати простоту мов високого рівня, таких як Java, з потужністю та ефективністю, властивими C++ і C. C# спроектована для розробки широкого спектра додатків, включаючи веб-програми, настільні додатки, мобільні застосунки та ігри.

Однією з ключових особливостей C# є його сильна типізація і підтримка об'єктно-орієнтованого програмування [4]. Це означає, що все в C# — це об'єкти, і код легко організувати в класи та об'єкти, що підвищує його читабельність і спрощує підтримку. Мова також підтримує важливі концепції ООП, такі як наслідування, поліморфізм і інкапсуляція, що дозволяє створювати масштабовані та багаторазово використовувані рішення.

C# інтегрована з платформою .NET, що забезпечує доступ до великої бібліотеки готових класів і функцій, які значно спрощують розробку. Наприклад, можна легко працювати з базами даних, маніпулювати файлами, відправляти HTTP-запити або створювати графічний інтерфейс користувача без необхідності писати багато коду з нуля. Завдяки цьому C# часто вибирають для корпоративних рішень і розробки великих програмних продуктів [5].

Окрім того, C# постійно розвивається і підтримує сучасні концепції програмування, такі як асинхронне програмування, вирази Lambda, властивості автоматичного збирання сміття (garbage collection) для управління пам'яттю [6].

Java — це одна з найпопулярніших і найширше використовуваних мов програмування, розроблена компанією Sun Microsystems (тепер належить Oracle) у 1995 році. Основною метою створення Java було забезпечення незалежності від платформи, що досягається завдяки концепції "Write Once, Run Anywhere" (WORA). Це означає, що код, написаний на Java, може виконуватися на будь-якій системі, що підтримує Java Virtual Machine (JVM), без необхідності його переписувати чи компілювати для конкретної платформи [7].

Java — це об'єктно-орієнтована мова програмування, що сприяє модульному підходу до розробки програм. Вона дозволяє програмістам створювати додатки, організовані в класи і об'єкти, що робить код більш структурованим і зручним для підтримки та повторного використання. Java також підтримує такі важливі концепції ООП, як наслідування, поліморфізм і інкапсуляція [8].

Однією з ключових переваг Java є її безпека та стабільність, що робить її придатною для розробки критичних корпоративних застосунків, фінансових систем, мобільних додатків (через Android SDK), веб-додатків і навіть систем вбудованого програмного забезпечення. Крім того, Java має велику екосистему бібліотек і фреймворків, таких як Spring, Hibernate, і JavaFX, що дозволяє розробникам швидко створювати функціональні, масштабовані та продуктивні додатки [9].

Також Java має вбудовані механізми управління пам'яттю, зокрема автоматичне збирання сміття (garbage collection), що звільняє програмістів від необхідності вручну очищати пам'ять. Це допомагає уникати витоків пам'яті, полегшуючи розробку великих і складних програм.

Python — це високорівнева мова програмування, відома своєю простотою, читабельністю та гнучкістю. Вона була створена Гвідо ван Россумом і випущена у 1991 році. Основною ідеєю Python є забезпечення максимальної зрозумілості та зменшення складності коду, завдяки чому вона стала однією з найпопулярніших мов у світі для розробників різного рівня [10].

Однією з головних переваг Python є її легкість у засвоєнні, що робить мову ідеальною для початківців. Завдяки простому синтаксису, який нагадує англійську мову, код на Python легко читати та писати. Python є інтерпретованою мовою, що означає, що код виконується рядок за рядком, без необхідності компіляції, що прискорює розробку та тестування [11].

Python є універсальною мовою, яка підтримує різні парадигми програмування: процедурне, об'єктно-орієнтоване та функціональне програмування. Це дозволяє використовувати її для широкого спектра завдань — від веб-розробки (за допомогою фреймворків, таких як Django або Flask) до наукових розрахунків, аналізу даних, штучного інтелекту та машинного навчання (бібліотеки NumPy, pandas, TensorFlow тощо) [12].

Ще однією значною перевагою Python є велика кількість сторонніх бібліотек і модулів, що дозволяють швидко додавати нові можливості до проєктів. Наприклад, для роботи з великими даними використовуються бібліотеки pandas і NumPy, для машинного навчання — TensorFlow і scikit-learn, для автоматизації — Selenium, а для побудови веб-додатків — Django або Flask.

Python також має активну спільноту розробників, що постійно оновлює і розширює екосистему мови, роблячи її ще потужнішою та придатною для вирішення практично будь-яких завдань.

Python — це універсальна мова програмування, яка широко використовується завдяки своїй простоті та читабельності. Вона підходить як для початківців, так і для досвідчених програмістів. Однією з основних переваг Python є його кросплатформеність, що дозволяє запускати програми на різних операційних системах, таких як Windows, macOS і Linux.

Python має велику бібліотеку готових модулів і пакетів, що дозволяє швидко виконувати різні завдання, від веб-розробки до аналізу даних і машинного навчання. Завдяки активній спільноті розробників постійно з'являються нові бібліотеки та інструменти, що розширюють можливості мови. Python також популярний у наукових колах, де його використовують для проведення досліджень, моделювання та обробки великих обсягів даних.

Окрім того, Python підтримує різні парадигми програмування, що дозволяє писати код у різних стилях. Це робить Python гнучким інструментом, який підходить для різноманітних проєктів — від простих скриптів до великих програмних систем.

Для реалізації нашої системи обрано мову програмування C# через її гнучкість та швидкодію, велике комьюніті яке допоможе розібратися з нетиповими проблемами та велику кількість бібліотек.

1.3. Аналіз середовищ розробки додатків

Існує кілька середовищ розробки для C#

— MS Visual Studio

— MS Visual Studio Code

— JetBrains Rider

— SharpDevelop

SharpDevelop — це безкоштовне середовище розробки програмного забезпечення для Windows, створене для мови програмування C# і платформ .NET [13]. Воно була розроблене для забезпечення альтернативи більш

відомим інструментам, таким як Microsoft Visual Studio, зосереджуючись на простоті використання та швидкості. SharpDevelop дозволяє розробникам створювати як консольні, так і графічні програми, а також веб-додатки за допомогою ASP.NET.

Однією з головних переваг SharpDevelop є його легкість у встановленні та використанні. Середа пропонує інтуїтивно зрозумілий інтерфейс, що дозволяє швидко почати розробку, навіть для новачків. SharpDevelop підтримує функції автоматичного завершення коду, що значно підвищує продуктивність програмістів, а також має вбудовані засоби для налагодження коду, що дозволяє ефективно відстежувати помилки.

SharpDevelop також має розширені можливості для роботи з проектами, зокрема підтримує формати проектів Visual Studio. Це означає, що розробники можуть легко імпортувати існуючі проекти та працювати над ними в SharpDevelop. Крім того, середовище має підтримку плагінів, що дозволяє додавати нові функції та адаптувати інструмент до потреб конкретного проекту.

Незважаючи на те, що SharpDevelop не є таким популярним, як Visual Studio, він залишається цінним інструментом для розробників, які шукають безкоштовну, легку у використанні альтернативу для розробки .NET-додатків. Оскільки SharpDevelop підтримує основні функції C# та .NET, він може бути корисним для навчання та швидкої розробки невеликих проектів.

JetBrains Rider — це потужне інтегроване середовище розробки (IDE) для програмування на C# та інших мовах, розроблене компанією JetBrains. Rider об’єднує в собі найкращі функції, характерні для інших продуктів JetBrains, таких як ReSharper і IntelliJ IDEA, що робить його ідеальним вибором для розробників, які працюють з технологіями .NET і .NET Core [14].

Однією з ключових переваг Rider є його інтуїтивно зрозумілий інтерфейс та підтримка багатьох сучасних технологій, включаючи ASP.NET, Unity, Xamarin та інші. IDE надає потужні інструменти для налагодження, тестування та профілювання додатків, що дозволяє розробникам ефективно знаходити й

усувати помилки у своїх програмах. Крім того, Rider забезпечує безперебійну інтеграцію з системами контролю версій, такими як Git, що спрощує командну роботу над проектами.

Rider також має вдосконалені функції автодоповнення коду та рефакторингу, які допомагають підвищити продуктивність розробників. Завдяки підтримці широкого спектра мов і технологій, таких як JavaScript, TypeScript, HTML, CSS, а також популярних фреймворків, таких як Angular і React, Rider стає універсальним інструментом для веб-розробки.

Крім того, JetBrains Rider доступний на всіх основних платформах, включаючи Windows, macOS і Linux, що дозволяє розробникам працювати в знайомому середовищі незалежно від операційної системи. Rider активно розвивається, що гарантує користувачам доступ до нових функцій і вдосконалень, забезпечуючи підтримку останніх версій .NET і інших технологій.

Visual Studio — це універсальне середовище розробки, яке було створене компанією Microsoft для підтримки розробки програмного забезпечення. Воно підходить для роботи з різними мовами програмування, такими як C#, F#, C++, VB.NET та іншими. Ця IDE дозволяє створювати широкий спектр додатків, включаючи веб-сайти, мобільні додатки та програмне забезпечення для настільних ПК [15].

Однією з основних особливостей Visual Studio є її потужний інструментарій для налагодження, який дозволяє розробникам легко виявляти та усувати помилки у коді. Середовище надає детальні звіти про помилки, а також можливість відстежувати виконання програми в реальному часі. Завдяки цьому, програмісти можуть швидко адаптувати свої рішення до нових вимог або умов.

Visual Studio також включає в себе функції, що допомагають автоматизувати рутинні завдання. Наприклад, автозаповнення коду та рекомендації щодо рефакторингу роблять процес написання коду більш ефективним. Крім того, IDE має зручну інтеграцію з системами контролю версій,

такими як Git, що дозволяє командам спільно працювати над проектами, зберігаючи при цьому історію змін.

Це середовище постійно розвивається, з регулярними оновленнями, які вносять нові функції та поліпшення. Visual Studio підтримує також різноманітні розширення та плагіни, які дозволяють користувачам адаптувати середовище відповідно до специфічних потреб проектів. Завдяки всім цим можливостям, Visual Studio залишається одним із найпопулярніших і найвідоміших інструментів серед розробників у світі.

Visual Studio Code — це безкоштовний редактор коду, розроблений компанією Microsoft, який став популярним серед розробників завдяки своїй легкості та гнучкості. Цей редактор підтримує безліч мов програмування, таких як JavaScript, Python, C#, Java, PHP та багато інших. Він доступний на різних платформах, включаючи Windows, macOS і Linux [16].

Однією з ключових переваг Visual Studio Code є його модульність. Користувачі можуть встановлювати різноманітні розширення, які розширюють функціональність редактора, додаючи підтримку нових мов, інструментів, тем оформлення та інших корисних функцій. Це дозволяє кожному налаштувати редактор під свої потреби та вподобання.

Visual Studio Code має вбудовані інструменти для налагодження, а також підтримує інтеграцію з системами контролю версій, такими як Git. Це спрощує командну роботу та дозволяє легко відстежувати зміни в коді. Редактор також включає функції автозаповнення, підсвічування синтаксису та підказки, що допомагають розробникам писати код швидше та ефективніше.

Крім того, Visual Studio Code підтримує функцію роботи з терміналом, що дозволяє виконувати команди прямо з середовища редактора, що підвищує зручність роботи. Завдяки активній спільноті розробників і регулярним оновленням, Visual Studio Code продовжує вдосконалюватися, ставлячи нові можливості та поліпшення в руки користувачів. Це робить його одним з найпопулярніших редакторів коду серед програмістів у всьому світі.

В нашому випадку вибір падає на MS Vidual Studio як на найпрогресивніше середовище розробки яке включає в себе nuget та інтеграцію з github.

Тут важливо згадати про Nuget та github.

NuGet [17] — це пакетний менеджер для платформ .NET, який спрощує управління бібліотеками та залежностями у проєктах. Він дозволяє розробникам легко завантажувати, встановлювати, оновлювати та видаляти бібліотеки, необхідні для їхніх програм. NuGet має велику колекцію пакетів, які містять готові до використання бібліотеки, що допомагає зекономити час на розробку, оскільки можна швидко інтегрувати популярні рішення.

Пакети NuGet зазвичай містять не лише сам код, а й метадані, такі як версія, залежності та ліцензія. Це дозволяє NuGet автоматично управляти версіями пакетів та їхніми залежностями, забезпечуючи сумісність під час оновлення. Розробники можуть створювати свої власні пакети та публікувати їх у приватних або публічних репозиторіях, що дозволяє легко ділитися кодом з командою або спільнотою.

NuGet інтегрується з такими середовищами розробки, як Visual Studio та Visual Studio Code, що дозволяє використовувати пакетний менеджер безпосередньо в IDE. Користувачі можуть шукати пакети, встановлювати їх за кілька кліків, а також використовувати консоль для управління пакетами. Це значно спрощує процес розробки, оскільки не потрібно вручну завантажувати файли бібліотек і налаштовувати їх у проєкті.

Завдяки своїй популярності та зручності використання, NuGet став стандартом для управління бібліотеками в екосистемі .NET, що робить його незамінним інструментом для розробників, які прагнуть швидко і ефективно створювати якісні програми.

GitHub — це веб-сервіс для хостингу проектів, які використовують систему контролю версій Git. Він надає розробникам можливість зберігати, відстежувати зміни у коді, а також спільно працювати над проектами. GitHub є

надзвичайно популярним серед програмістів завдяки своїм зручним функціям, які спрощують співпрацю та управління проектами [18].

Однією з ключових особливостей GitHub є можливість створення репозиторіїв, де користувачі можуть зберігати свої проекти. Репозиторії можуть бути публічними, що дозволяє всім бачити і завантажувати код, або приватними, що обмежує доступ лише для певних користувачів. GitHub також підтримує систему гілок, яка дозволяє розробникам працювати над новими функціями або виправленнями без впливу на основну версію коду.

GitHub пропонує безліч інструментів для управління проектами, включаючи трекери проблем, інструменти для рев'ю коду та можливості для автоматизації процесів за допомогою GitHub Actions. Ці функції роблять співпрацю в команді більш організованою та ефективною. Користувачі можуть також залишати коментарі до конкретних рядків коду, що спрощує обговорення та узгодження змін.

Додатково GitHub дозволяє інтегруватися з іншими сервісами та інструментами, такими як CI/CD системи, що сприяє автоматизації розгортання і тестування додатків. Завдяки великій спільноті користувачів і відкритим проектам, GitHub став центром для обміну знаннями та ресурсами в розробницькій спільноті, що робить його незамінним інструментом для сучасних програмістів.

1.4 Постановка завдання магістерської роботи

Завдання на магістерську роботу полягає в розробці програмного засобу для обліку даних при занятті фітнесом.

Для цього необхідно дослідити поточний стан та предметну область, виділити що зараз реалізовано і чого бракує.

Розробити програмний засіб з наступними функціональними та нефункціональними вимогами.

Програма повинна мати можливість вести облік як людей включаючи їх ім’я, вік, стать та інші дані. Повинна бути можливість створення, видалення та редагування даних про кожного користувача.

Також повинна бути можливість ведення обліку тренувань таких як біг, велотренування, йога та заняття в залі. Повинна бути можливість видалення та оновлення даних про тренування. Програма повинна враховувати вік та пульс при тренуванні та повідомляти про небезпеку якщо пульс надто високий.

У програми повинна бути можливість створення звітів про користувачів у форматі ексель.

Також дослідна частина. Повинна бути можливість рекомендацій велотренувань та бігу на основі попередніх даних з врахуванням віку та максимального пульсу, а також кількості спалених калорій. Повинні бути використані моделі прогнозування на основі лінійної регресії, поліноміальної регресії, методу найближчого сусіда та кубічних сплайнів.

За деякими твердженнями чим інтенсивніше тренування тим більше калорій спалює людина і тим більше худне. Тому при прогнозуванні функцією максимізації була кількість спалених калорій. А обмеженнями був максимальний пульс.

Формули 1.1 a та b показують максимальний допустимий пульс при тренувнні

$$hr = 226-a \quad (1.1a)$$

$$hr = 220-a \quad (1.1b)$$

де hr – максимальний допустимий пульс, a – вік атлета.

Формула 1.1.a стосується жінок, а 1.1b – чоловіків.

Таким чином нам потрібно спершу знайти максимальну інтенсивність при максимально допустимому пульсі, а далі при цій 3ті спрогнозувати параметри вправ.

Варто зазначити, що максимально допустимий пульс береться на 10% нижчий за той, що розрахований формулою, оскільки існують погрішності та й в цілому жодні результати тренувань не коштують здоров’я.

Після того як ми спрогнозували інтенсивність тренування можемо прогнозувати параметри тренування так, аби воно було максимально допустимо інтенсивним.

Для бігу інтенсивність залежить виключно від дистанції.

Для велотренувань інтенсивність залежить від дистанції та середньої швидкості.

Для тренувань в залі – від ваги та кількості жимів.

Для йоги інтенсивність не обчислюється, оскільки йога не сприяє схудненню.

До нефункціональних вимог варто віднести зручний інтерфейс користувача який повинен бути “резиновим” та не втомлювати очі при тривалій роботі з додатком.

Дані про користувачів та тренування повинні шифруватися з допомогою алгоритмів та методів шифрування які є достатньо стійкими чи вважаються такими на даний час. Дані повинні зберігатися окремо від програми та бути транспортабельними, тобто без прив’язки до конкретного пристрою чи апаратної частини.

1. Провести дослідження прогнозування на основі розробленого додатку.
2. Зробити висновки про методи прогнозування та їх адекватність в рамках рекомендацій фітнес вправ

Таким чином основна частина роботи полягає в розробці додатку та аналізу прогнозування які цей додаток буде робити.

РОЗДІЛ 2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Список варіантів використання

Визначення варіантів використання це один з початкових етапів проектування будь-якої системи. Нижче описано варіанти використання для дипломного проекту:

- додати інформацію про персону;
- редагувати інформацію про персону;
- видалити інформацію про персону;
- перегляд інформації про всіх персон;
- додати інформацію про тренування;

— редагувати інформацію про тренування;

— видалити інформацію про тренування;

— перегляд інформації про всі тренування конкретної персони;

— перевірка пульсу при додаванні тренування та сповіщення про небезпеку якщо він зависокий

— перевірка чи тренування не оновило особистий рекорд та відповідний запис

— перегляд особистих рекордів

— створення плану бігу для максимального спалення калорій

— створення плану велотренування для максимального спалення калорій.

На рис. 2.1 показана діаграма прецедентів(варіантів використання).

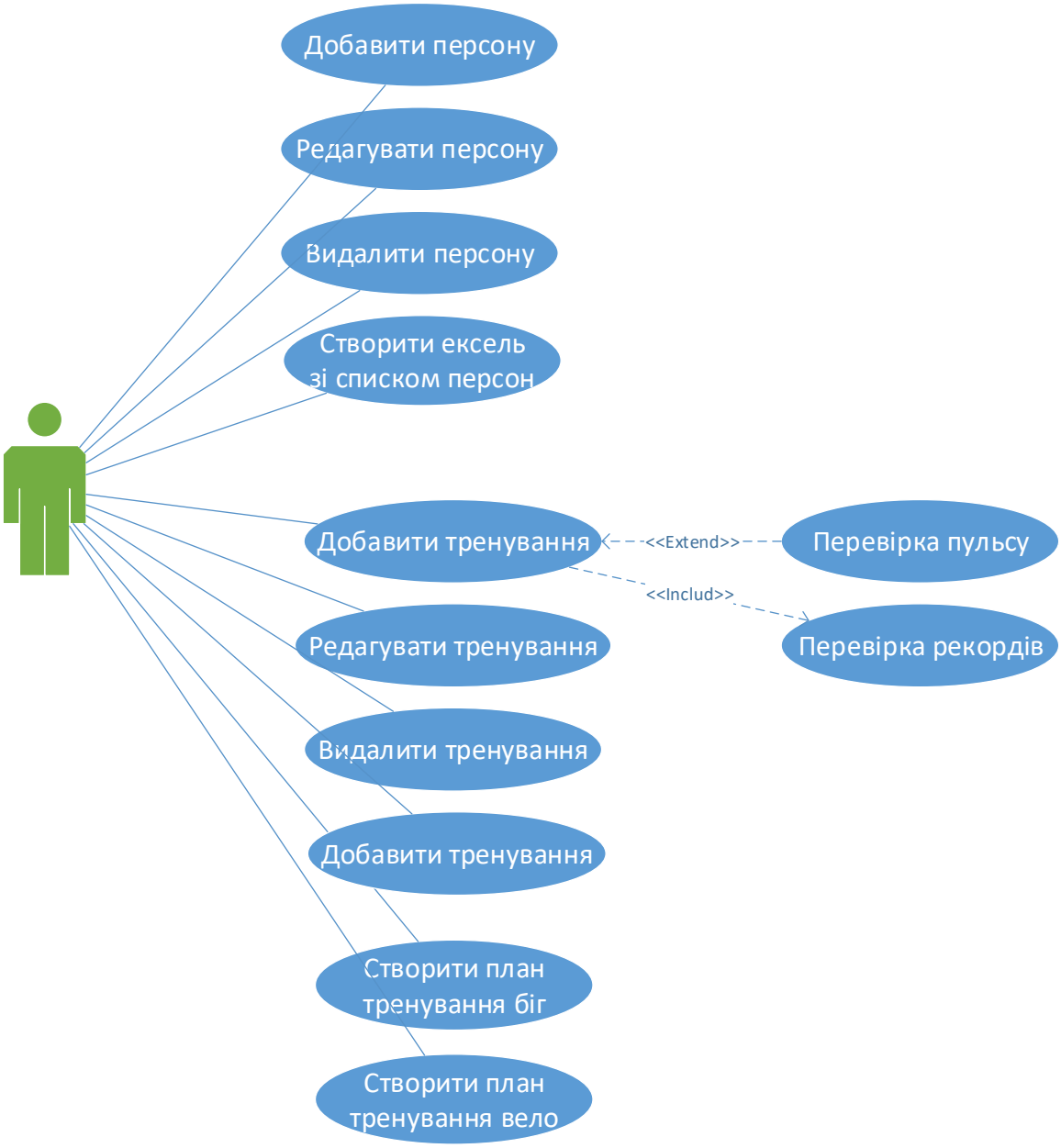


Рис. 2.1. Діаграма варіантів використання

Також потрібне збереження та завантаження даних між сесіями роботи з програмою. І не варто забувати про безпеку, необхідне шифрування збережених даних. Проте користувач не має доступу до цього функціоналу, адже він виконується автоматично при відкритті чи закритті додатку і відповідно це не є прецедентами нашої програми.

2.2 Огляд додаткового функціоналу

Для забезпечення високого рівня безпеки було реалізовано шифрування даних з допомогою алгоритму AES з 256 бітним ключем.

AES (Advanced Encryption Standard) — це алгоритм симетричного шифрування даних, який став світовим стандартом для захисту конфіденційної інформації. Створений у 2001 році як заміна для застарілого алгоритму DES, AES став важливим елементом в області кібербезпеки завдяки своїй високій надійності та ефективності. Алгоритм побудований на принципах симетричного шифрування, де для шифрування і дешифрування використовується один і той самий ключ, що робить його значно швидшим порівняно з асиметричними методами шифрування.

Основний процес шифрування в AES складається з кількох раундів, кожен з яких включає операції підстановки байтів, перемішування рядків, перестановки стовпців та застосування ключа, що постійно змінюється для кожного раунду. Кількість таких раундів залежить від довжини ключа: 128-бітний ключ вимагає 10 раундів, 192-бітний — 12 раундів, а 256-бітний — 14 раундів. Завдяки таким складним перетворенням даних, AES є надзвичайно стійким до різноманітних типів атак, включаючи відомі методи криптоаналізу.

Використання AES є надзвичайно широким і охоплює різноманітні галузі: від банківської справи і державного управління до інтернет-комерції та медичних даних. Більше того, завдяки своїй універсальності та швидкодії, AES успішно інтегрується в різні платформи, включаючи мобільні пристрої, інтернет речей та хмарні сервіси. Алгоритм є ефективним як для апаратної, так і для програмної реалізації, тому він часто використовується в мікрочіпах, що забезпечує захищеність даних на рівні пристрою.

Процес шифрування за допомогою AES розпочинається з розширення початкового ключа, що генерує серію підключів для кожного раунду шифрування. Після цього виконується перший етап змішування даних із ключем: до блоку даних застосовується операція XOR з початковим підключем. Це додає

первинну захищеність перед основними раундами, що складаються з кількох складних операцій [19].

На кожному раунді відбувається підстановка байтів, де кожен байт блоку замінюється на інший за спеціальною таблицею підстановок (S-box), що робить структуру даних менш передбачуваною. Далі, байти в рядках зсуваються вліво на певну кількість позицій залежно від номера рядка, що розсіює початкову залежність між вхідними і вихідними даними. Ще один етап — перестановка стовпців, коли застосовуються спеціальні математичні перетворення для змішування байтів всередині кожного стовпця, що додає рівень дифузії та робить блок ще складнішим для зламування. На завершення кожного раунду до блоку знову додається поточний підключ, що захищає дані від відновлення без ключа.

Фінальний раунд AES завершується так само, але без перестановки стовпців, утворюючи фінальний зашифрований блок. Всі ці етапи перетворюють дані таким чином, що навіть мінімальні зміни у вхідному блоці або ключі повністю змінюють результат шифрування, надаючи високий рівень захисту.

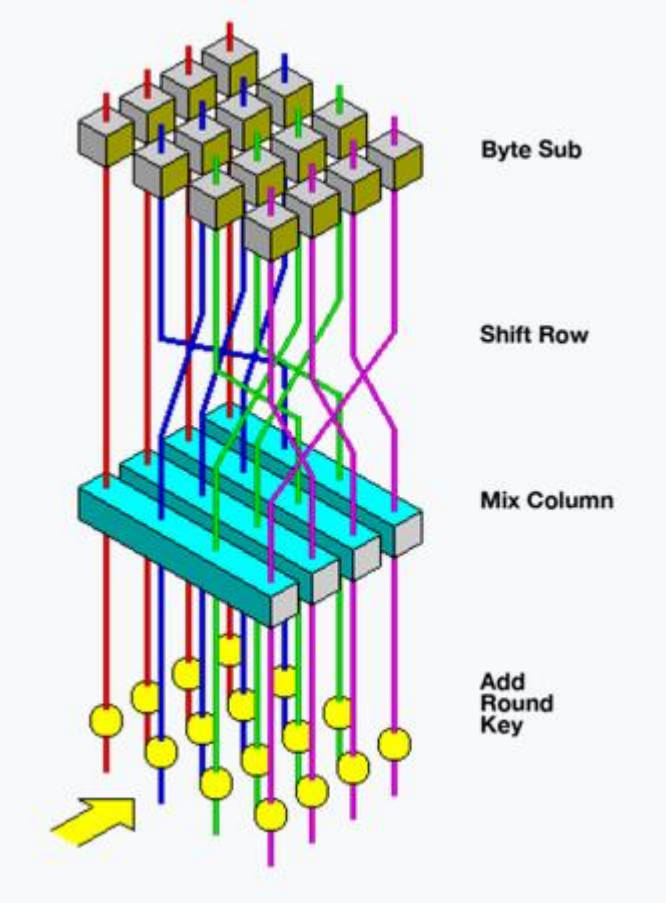
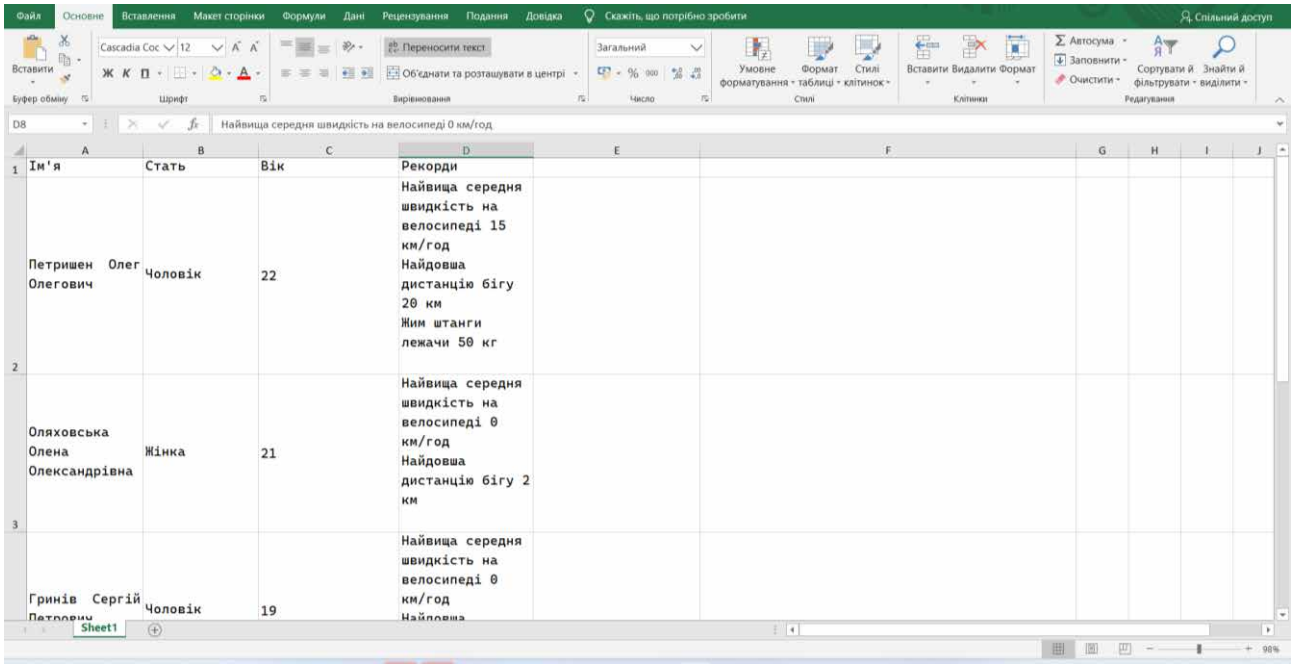


Рис. 2.2. Схема шифрування AES

Також було передбачено створення звітів у форматі ексель для збереження загальних даних про всіх персон див. рис 2.3.

Як бачимо на рисунку 2.3. в звітів міститься інформація про ПІБ, вік, стать та список особистих рекордів.



	A	B	C	D
1	Ім'я	Стать	Вік	Рекорди
2	Петришен Олег Олегович	Чоловік	22	Найвища середня швидкість на велосипеді 15 км/год Найдовша дистанцію бігу 20 км Жим штанги лежачи 50 кг
3	Оляховська Олена Олександрівна	Жінка	21	Найвища середня швидкість на велосипеді 0 км/год Найдовша дистанцію бігу 2 км
	Гринів Сергій Петрович	Чоловік	19	Найвища середня швидкість на велосипеді 0 км/год Найдовша

Рис. 2.3. Зразок звіту по персонах

Шаблон звіту міститься поряд з програмою та може бути змінений користувачем без навиків програмування та потреби перекомпіляції програми. Змінити можна все, окрім порядку стовпців з даними та самого формату даних. В подальшому цю інформацію можна конвертувати у *.pdf чи *.csv файл за потреби.

2.3 Моделювання інтерфейсу програми

При розробці інтерфейсу програми було враховано тривалу роботу застосунку та всі функціональні вимоги. Для цього було розроблено темний стиль та реалізовано резиновий інтерфейс у всіх вікнах де це було необхідно. Спершу розглянемо сам розроблений стиль.

Стиль складається з двох файлів з даними про кольори та даними про елементи управління. На рис 2.4. показано частину файлу що описує кольорову схему стилю. Вона містить більше 300 рядків де описано кольори та градієнти.

```xaml
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:system="clr-namespace:System;assembly=mscorlib"
                    xmlns:controls="clr-namespace:System.Windows;assembly=PresentationFramework">

    <SolidColorBrush x:Key="ControlForeground" Color="#FFFD9804" />
    <!--#FFFD9804 -->
    <SolidColorBrush x:Key="ControlForegroundWhite" Color="#FFEEEFFF" />
    <SolidColorBrush x:Key="BorderBrushVisual" Color="Transparent"/>
    <SolidColorBrush x:Key="ControlPressedVisualBrush" Color="Transparent"/>

    <RadialGradientBrush x:Key="ControlBackgroundNormal" GradientOrigin="0.5,0">
        <GradientStop Color="#FF585555" Offset="1"/>
        <GradientStop Color="#FE8F8D8D" Offset="0.082"/>
    </RadialGradientBrush>

    <LinearGradientBrush x:Key="ControlBackgroundOver" EndPoint="0,1" StartPoint="0,0">
        <GradientStop Color="#FFEFCF38" Offset="0"/>
        <GradientStop Color="#FFFD9804" Offset="1"/>
    </LinearGradientBrush>

    <LinearGradientBrush x:Key="ControlBackgroundPressed" EndPoint="0,1" StartPoint="0,0">
        <GradientStop Color="#FFEFA05D" Offset="0"/>
        <GradientStop Color="#FFE04800" Offset="1"/>
```

Рис. 2.4. Кольорова схема стилю

Другою частиною стилю є опис елементів управління. Див рис 2.5.

```xaml
<!-- Button -->
<Style TargetType="Button">

    <Setter Property="Foreground" Value="{StaticResource ControlForegroundWhite}" />
    <Setter Property="Background" Value="{StaticResource ControlBackgroundNormal}"/>
    <Setter Property="BorderThickness" Value="{StaticResource BorderThicknessButton}"/>
    <Setter Property="BorderBrush" Value="{StaticResource ControlBorderBrush}"/>
    <Setter Property="Padding" Value="{StaticResource MarginContentButton}"/>
    <Setter Property="FocusVisualStyle" Value="{StaticResource FocusLine}"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Grid x:Name="Root">
                    <VisualStateManager.VisualStateGroups>
                        <VisualStateGroup x:Name="CommonStates">
                            <VisualStateGroup.Transitions>
                                <VisualTransition GeneratedDuration="0:0:0.2" />
                                <VisualTransition To="Pressed" />
                                <VisualTransition From="Pressed" />
                            </VisualStateGroup.Transitions>
                            <VisualState x:Name="Normal" />
                            <VisualState x:Name="MouseOver">
                                <Storyboard>
                                    <DoubleAnimation Duration="0" Storyboard.TargetName="PressedElement" Storyboard.TargetProperty
                                    <DoubleAnimation Duration="0" Storyboard.TargetName="MouseOverElement" Storyboard.TargetProper
                                </Storyboard>
                            </VisualState>
                            <VisualState x:Name="Pressed">
                                <Storyboard>
                                    <DoubleAnimation Duration="0" Storyboard.TargetName="PressedElement" Storyboard.TargetProperty
                                    <DoubleAnimation Duration="0" Storyboard.TargetName="MouseOverElement" Storyboard.TargetProper
                                    <DoubleAnimation Duration="0" Storyboard.TargetName="BorderVisual" Storyboard.TargetProperty="
                                    <DoubleAnimation Duration="0" Storyboard.TargetName="BorderPressed" Storyboard.TargetProperty=

                                </Storyboard>
                            </VisualState>
```

Рис. 2.5. Стиль елементів управління

Даний файл описує всі можливі контроли такі як кнопки, таблиці, випадаючі списки та інші. В ньому описані анімації та переходи, реакції на наведення курсору та всі стани елементу управління.

Нижче показано як це все виглядає при виконанні програми.

Програма складається з кількох форм. На головній формі показана коротка інформація про атлетів див рис.2.6.

Рис. 2.6. Головна форма

При додаванні чи редагуванні даних про атлета відкривається форма атлета див рис 2.7. на ній є всі необхідні дані атлета, а також таблиця з його останніми заняттями.

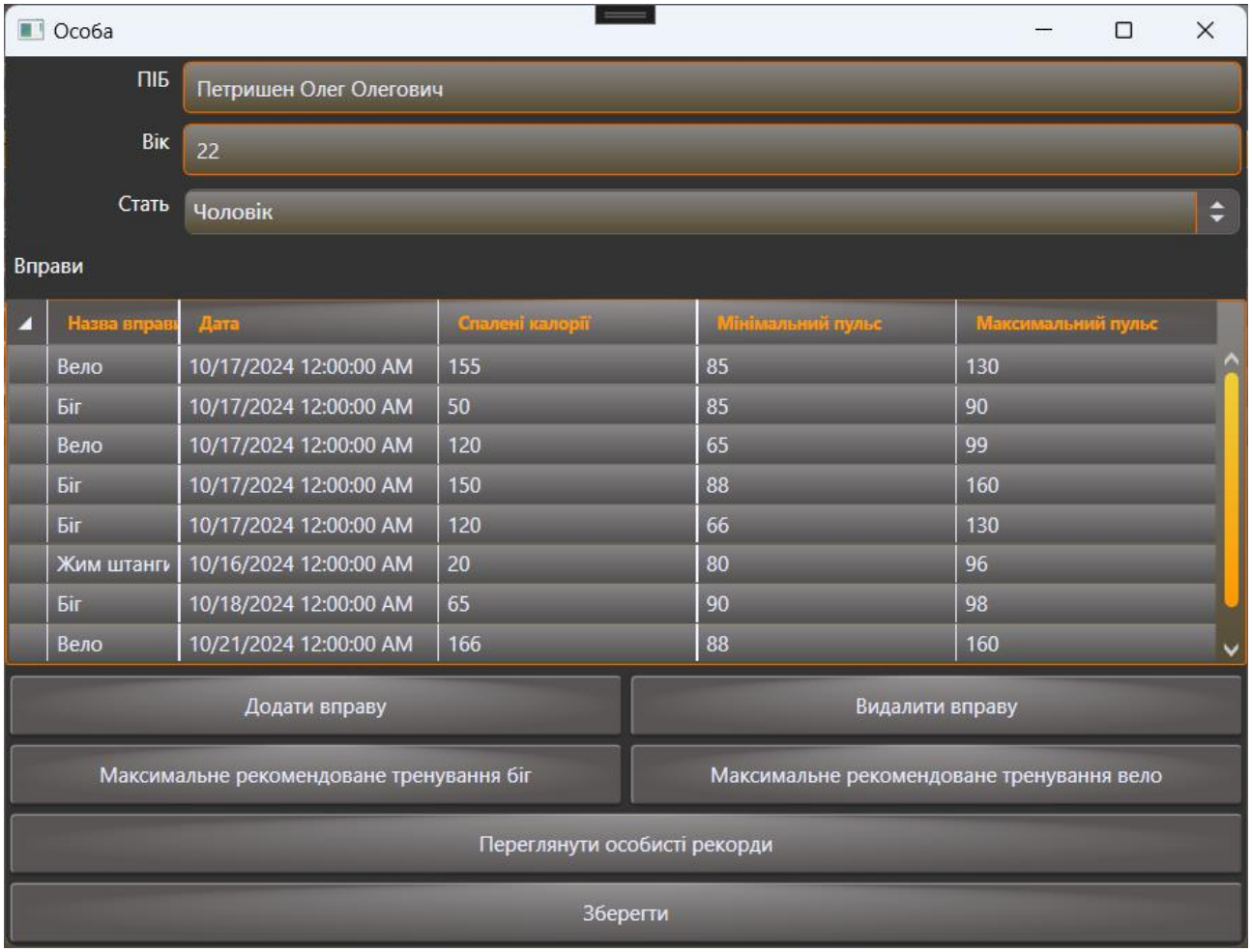


Рис. 2.7. Форма атлета

При подвійному кліку на вправу чи при натисканні кнопки додавання вправи відкривається форма вправи. Її поля змінюються в залежності від типу вправи який обрали. Важливо підкреслити, що при редагуванні вправи змінити її тип неможливо. Сама форма показана на рис 2.8.

Також варто відмітити форму прогресу. Вона відкривається при створенні звіту ексель чи інших тривалих операціях див рис 2.9.

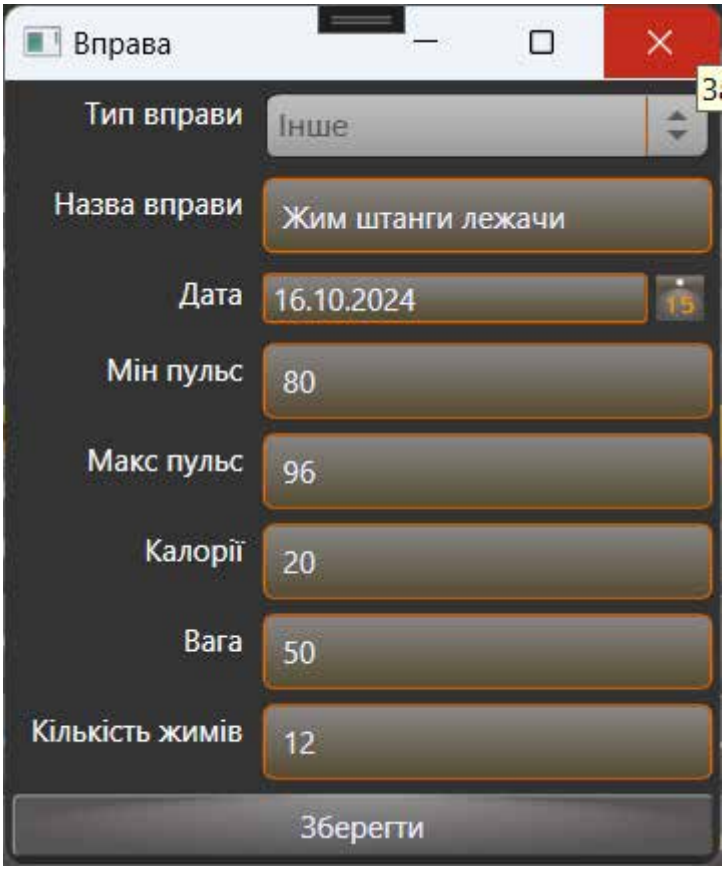


Рис. 2.8. Форма атлета

Рис. 2.9. Форма прогресу

Все це було розроблено з використанням WPF та розмітки XAML див рис 2.10.

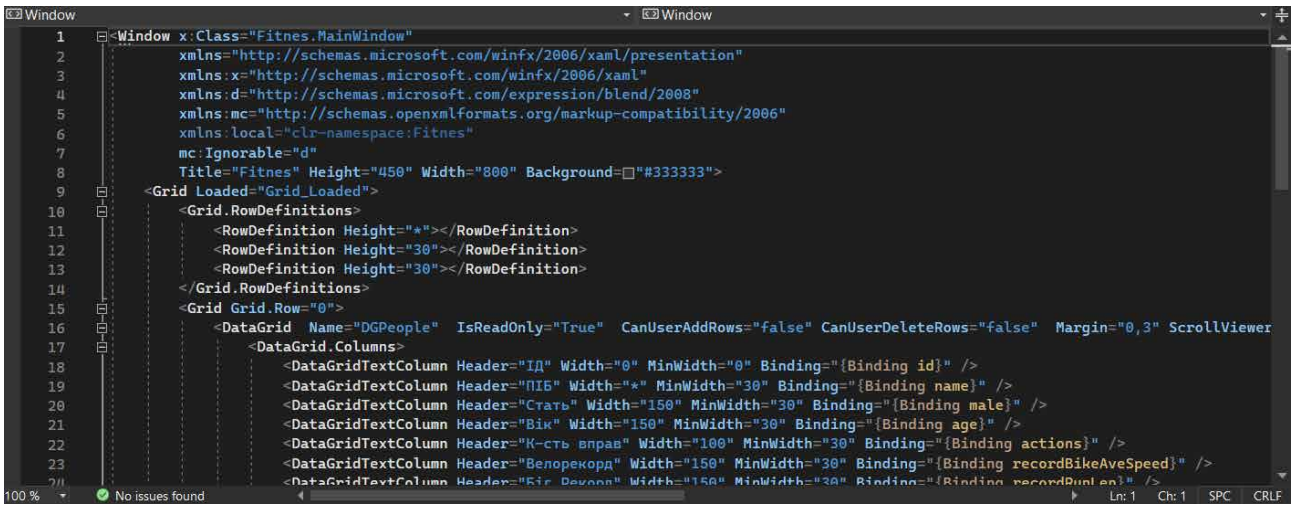


Рис. 2.10. Елементи розмітки головної форми

2.4 Моделювання класів та структур даних

Спершу було створено структуру класів що описують заняття фітнесом. Це базовий клас Action, похідний від нього клас RunAction який містить інформацію про відстань. Клас BikeAction похідний від RunAction і містить додатково поле для середньої швидкості. Також від класу Action є дочірній клас YogaAction який описує заняття йогою та клас GymAction який містить всю необхідну інформацію для опису заняття в залі.

Також є клас Person який описує атлетів. Він містить список об’єктів класу Action. Через поліморфізм кожен об’єкт класу містить інформацію про те чи інше заняття. Все це міститься в об’єкті класу DataBase який описує основну структуру даних.

Також є клас Core, який відповідає за серіалізацію та десеріалізацію з файлу.

Серіалізація та десеріалізація — це процеси, що використовуються для збереження й передачі об’єктів у зручному для зберігання та обробки форматі. В основі серіалізації лежить перетворення складного об'єкта, такого як масиви, списки чи навіть цілі класи, у послідовність байтів або текстовий формат (наприклад, JSON або XML). Це дозволяє передавати об'єкт через мережу чи

зберігати його на диску у вигляді файлу, який можна відновити пізніше. Серіалізовані об'єкти стають незалежними від конкретного середовища, у якому вони були створені, що значно спрощує передачу даних між різними системами чи платформами [20].

Десеріалізація, навпаки, є процесом відновлення початкового об'єкта з серіалізованих даних. Вона перетворює отримані байти або текст у структуровану форму, з якою можна працювати в програмі. Це дозволяє програмам обробляти дані, що надходять, як рідні об'єкти, незалежно від того, в якому форматі вони були збережені або передані.

Процеси серіалізації та десеріалізації широко використовуються в сучасних технологіях, зокрема, в обміні даними між веб-серверами й клієнтами, збереженні конфігурацій додатків, передачі даних у міжплатформових системах. Для реалізації цих процесів часто використовуються бібліотеки, що підтримують різні формати серіалізації, такі як JSON, XML, Protobuf тощо.

Також є ряд кілька класів для прогнозування це класи LinearRegression, PolynomialRegression, KNearNeighbors, CubicSplineCal та CubicSplineHR. Ці класи відповідають за передбачення на основі даних які передаються в форматі об’єктів класу ActionLR. Клас ActionLR містить три поля для відображегння калорій, максимального пульсу та інтенсивності тренувань.

Також є ряд допоміжних класів таких як DataGridHelper який використовується для зручної роботи з елементом управління DataGrid.

Окремо варто виділити клас StringCipher який використовується для шифрування та дешифрування символьних даних.

Всі ці класи представлені на діаграмі класів зображеній на рис 2.11.

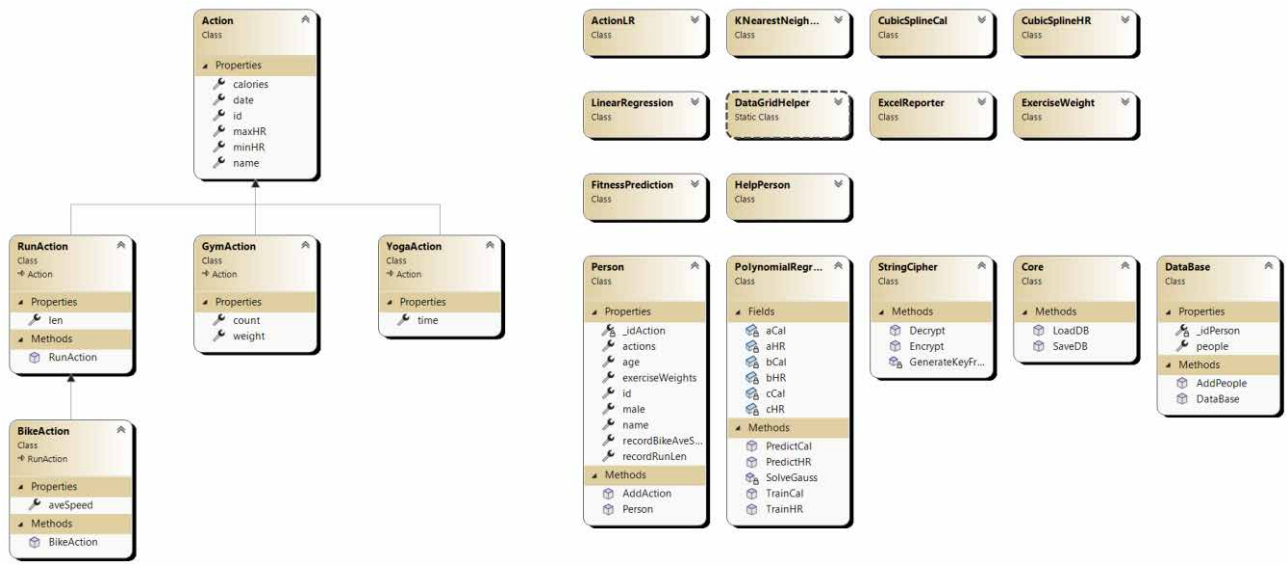


Рис. 2.11. Діаграма класів

На рис 2.11 в лівій частині показано ієрархію класів що описує типи тренувань, а в правій всі інші класи. Класів багато тож доволі важко їх помістити на один аркуш, деякі довелося згорнути та залишити лише назви, проте всі вони важливі і без них програма не існуватиме в тому вигляді в якому вона задумана.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Реалізація основних алгоритмів програми

При реалізації програми було реалізовано кілька важливих та цікавих алгоритмів. Нижче будуть описані деякі з них.

Перш за все варто виділити алгоритми серіалізації та десеріалізації у json файл.

JSON (JavaScript Object Notation) – це формат обміну даними, який використовується для зберігання та передачі даних між клієнтом і сервером або всередині застосунку. Його популярність пояснюється простотою структури та сумісністю з різними мовами програмування, такими як JavaScript, Python, Java та інші. JSON використовує пару «ключ-значення», що робить його схожим на об'єкти в мовах програмування, де кожен ключ асоціюється з певним значенням. Це дозволяє легко організовувати ієрархічні дані, які можуть бути вкладеними, що зручно для складних структур [21].

Для обміну даними JSON зазвичай використовується в API, де клієнт запитує інформацію в сервера і отримує відповідь у форматі JSON. Наприклад, клієнт може запросити список товарів, і сервер надішле JSON-дані, що містять інформацію про кожен товар: його назву, ціну, опис та інші характеристики. Клієнтський застосунок може потім розпарсити JSON і відобразити дані користувачеві у зручному вигляді. Це дозволяє передавати великі обсяги інформації в стислому та зрозумілому вигляді.

Однією з переваг JSON є його компактність та легка читабельність як для машин, так і для людей. Звичайно, JSON не завжди є найоптимальнішим вибором для зберігання даних, особливо для великих обсягів або даних, що потребують високої швидкості обробки. Проте його універсальність і простота роблять JSON одним з основних форматів для роботи з даними в сучасних веб-застосунках.

JSON має кілька важливих переваг, які зробили його одним із найпопулярніших форматів для обміну даними в сучасних застосунках. По-

перше, він є дуже простим і компактним, що дозволяє передавати дані у зручному для обробки та розуміння форматі. Завдяки цій компактності, JSON спрощує передачу великих обсягів даних між клієнтом і сервером, особливо в умовах обмеженої пропускної здатності, що може бути критично важливим для мобільних і веб-застосунків.

Другою важливою перевагою JSON є його сумісність із багатьма мовами програмування. JSON підтримує просту ієрархічну структуру ключ-значення, подібну до об’єктів у JavaScript, Python, Java та інших мовах, що дозволяє легко інтегрувати цей формат у різноманітні проєкти. Завдяки цьому розробники можуть легко серіалізувати та десеріалізувати дані з JSON у рідні об'єкти мови, що спрощує обробку інформації.

Ще однією важливою перевагою JSON є його легка читабельність як для комп'ютера, так і для людини. Це дозволяє розробникам швидко переглядати, розуміти та редагувати JSON-дані вручну, що є корисним під час тестування або відладки застосунків. Простота формату також полегшує підтримку й модернізацію коду, в якому використовується JSON, роблячи його універсальним вибором для багатьох сучасних веб- і мобільних застосунків.

На лістингу 3.1 показано код серіалізації об’єктів у файл з використанням шифрування.

Лістинг 3.1. Серіалізація

```
DataBase db = JsonConvert.DeserializeObject<DataBase>(decryptJson, new JsonSerializerSettings
{
    TypeNameHandling = TypeNameHandling.Auto
});
```

Як бачимо тут відбувається json серіалізація. Що найважливіше, то це автогенерація для типів та підтипів. Частина незашифрованого серіалізованого файлу показано на лістингу 3.2.

Лістинг 3.2. Серіалізований файл


```
"actions": [
      {
       "$type": "Fitnes.BikeAction, Fitnes",
       "aveSpeed": 14.0,
       "len": 12.0,
       "id": 12,
       "name": "Вело",
       "date": "2024-10-17T00:00:00+03:00",
       "minHR": 85,
       "maxHR": 130,
       "calories": 155.0
      },
      {
       "$type": "Fitnes.RunAction, Fitnes",
       "len": 5.0,
       "id": 4,
       "name": "Біг",
       "date": "2024-10-17T00:00:00+03:00",
       "minHR": 85,
       "maxHR": 90,
       "calories": 50.0
      },
```


Тут важливо відмітити записи на кшталт "$type": "Fitnes.BikeAction, Fitnes" та "$type": "Fitnes.RunAction, Fitnes", це автоматичні поля які добавилися оскільки серіалізоване поле має тип Action, а об’єкти в ньому завдяки поліморфізму мають типи похідних класів.

Тут варто згадати алгоритм шифрування. Для шифрування було обрпано симетричний алгоритм AES. На лістингу 3.3. показана фунція шифрування

Лістинг 3.3. Шифрування

```
public static string Encrypt(string plainText, string password)
{
   using (Aes aesAlg = Aes.Create())
   {
      // Генеруємо ключ і вектор ініціалізації (IV) на основі пароля
      var key = GenerateKeyFromPassword(password, aesAlg.KeySize / 8);
      aesAlg.Key = key;
      aesAlg.GenerateIV();
      ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);
      using (MemoryStream msEncrypt = new MemoryStream())
      {
         // Записуємо IV на початок потоку, щоб використати його під час дешифрування
         msEncrypt.Write(aesAlg.IV, 0, aesAlg.IV.Length);
         using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
         using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
         {
            swEncrypt.Write(plainText);
         }
         return Convert.ToBase64String(msEncrypt.ToArray());
      }      }      }
```

Як бачимо з лістингу вище спершу створюється об’єкт класу Aes. Даний клас розміщений в просторі імен System.Security.Cryptography. Після цього

генеруємо ключ з парою та записуємо в даний клас. Також генеруємо вектор ініціалізації.

Записуємо вектор ініціалізації і після цього добавляємо до записаного зашифрований текст поблоково.

Наступним важливим алгоритмом який варто описати є створення звітів в екселі. Для створення звітів була створена окрема форма для відображення прогресу, див рис 2.9. при кліку на кнопку спершу запускається дана форма і в залежності від змінної запускає конкретну функцію для генерації звіту ексель. В функцію генерації ексель звіту одним з параметрів передається текстовий блок на формі. Він розташований по центру прогресбару та служить для відображення прогресу.

По самій генерації звітів була створено клас ExcelReporter та функція CreatePeopleReport яка приймає як параметри об’єкт бази даних та текстовий блок для відображення прогресу користувачу.

Спершу ініціалізуються основні об’єкти необхідні для роботи з ексель файлом див лістинг 3.4.

Лістинг 3.4. Ініціалізація об’єктів ексель

```
Excel.Application xlApp;
Excel.Workbook xlWorkBook;
Excel.Worksheet xlWorkSheet;
object misValue = System.Reflection.Missing.Value;
xlApp = new Excel.Application();
```

Як бачимо тут створюються об’єкти що відповідають за сам додаток ексель, файл ексель та таблицю ексель. Також оголошується змінна misValue для зручності в подальшому.

Після ініціалізації відбувається відкриття шаблону ексель, див лістинг 3.5.

Лістинг 3.5. Відкриття шаблону ексель

```
string path = Path.Combine(Directory.GetCurrentDirectory(), "ExcelTemplates\\Peoples.xlsx");

xlWorkBook = xlApp.Workbooks.Open(path);
xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
```

Після цієї фази відбувається сам процес запису та форматування файлу ексель. Це доволі об’ємний процес тож цей код можна переглянути в додатках.

Після запису необхідних даних в файл відбувається його збереження, див лістинг 3.6.

Лістинг 3.6. Збереження звіту

```
if (File.Exists(savefile.FileName)) File.Delete(savefile.FileName);
xlWorkBook.SaveAs(savefile.FileName, Excel.XlFileFormat.xlWorkbookDefault, misValue, misValue, misValue, misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue, misValue);
xlWorkBook.Close(true, misValue, misValue);
xlApp.Quit();
```

Як бачимо з лістингу 3.6. спершу перевіряється чи файл з такою назвою існує і якщо так, то він видаляється. Потім зберігається файл, закривається книга та додаток ексель.

Проте це не все, необхідно видалити вказівники на об’єкти які були ініціалізовані на початку, інакше процес Excel висітиме в диспетчері задач.

Для цього була написана функція releaseObject лістинг якої показаний на лістингу 3.7.

Лістинг 3.7 releaseObject

```
private static void releaseObject(object obj)
{
   try
   {
      System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
      obj = null;
   }
   catch (Exception ex)
   {
      obj = null;
   }
   finally
   {
      GC.Collect();
   }
}
```

Як бачимо функція намагається видалити вказівники допоки не вийде. На жаль з розвитком віндовс та зміною його ядра та пріоритетів роботи з вказівниками в оперативній пам’яті дана функція лише розростається з новими оновленнями віндовс.

Тепер можна перейти до основного, до алгоритмів передбачення.

Алгоритм лінійної регресії показаний на лістингу 3.8.

Лістинг 3.8. Лінійна регресія

```
public void TrainCal(List<ActionLR> trainingData)
{
```

```
            int n = trainingData.Count;
            double sumX = trainingData.Sum(a => a.Duration); // Сума тривалостей тренувань (Duration)
            double sumY = trainingData.Sum(a => a.Calories); // Сума кількості спалених калорій (Calories)
            double sumXY = trainingData.Sum(a => a.Duration * a.Calories); // Сума добутків Duration на Calories
            double sumX2 = trainingData.Sum(a => a.Duration * a.Duration); // Сума квадратів тривалостей (Duration^2)

            // Обчислюємо коефіцієнти для рівняння прямої Calories = slope * Duration + intercept
            slopeCal = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX * sumX);
            interceptCal = (sumY - slopeCal * sumX) / n;
        }
```

Дана функція повертає коефіцієнти лінійного рівняння для передбачення інтенсивності тренувань на основі спалених калорій. Далі необхідно лише ввести значення у формулу і функція поверне прогнозоване значення калорій при певній інтенсивності. Саму функцію показано на лістингу 3.9

Лістинг 3.9. PredictCal

```
    public double PredictCal(double duration )
    {
        return slopeCal * duration + interceptCal;
    }
```

Для методу поліноміальної регресії було створено клас PolynomialRegression який має такі ж функції але з іншою реалізацією. На лістингу 3.10 показано

Лістинг 3.10

```
public void TrainCal(List<ActionLR> trainingData)
        {
            int n = trainingData.Count;
            double sumX = trainingData.Sum(a => a.Duration);
            double sumX2 = trainingData.Sum(a => Math.Pow(a.Duration, 2));
            double sumX3 = trainingData.Sum(a => Math.Pow(a.Duration, 3));
            double sumX4 = trainingData.Sum(a => Math.Pow(a.Duration, 4));

            double sumY = trainingData.Sum(a => a.Calories);
            double sumXY = trainingData.Sum(a => a.Duration * a.Calories);
            double sumX2Y = trainingData.Sum(a => Math.Pow(a.Duration, 2) *
a.Calories);

            // Матриця нормальних рівнянь для обчислення коефіцієнтів a, b, c
            double[,] matrix = new double[3, 4]
            {
                { n, sumX, sumX2, sumY },
                { sumX, sumX2, sumX3, sumXY },
                { sumX2, sumX3, sumX4, sumX2Y }
            };

            // Виконуємо метод Гаусса для розв'язання системи лінійних
рівнянь
            SolveGauss(matrix, true);
```

```
}
```

Дана функція формує матрицю та розв’язує її. Саме обчислення можна глянути на лістингу 3.11

Лістинг 3.11. PredictCal

```
public double PredictCal(double duration)
{
   return cCal * Math.Pow(duration, 2) + bCal * duration + aCal;
}
```

Як бачимо в даному випадку у нас квадратне рівняння

Для методу найближчих сусідів реалізовано клас KNearestNeighbors

Даний клас також має функції для передбачення калорій по інтенсивності, його можна побачити на лістингу 3.10

Лістинг 3.10 PredictDurationCal

```
public double PredictDurationCal(double duration)
{
   // Обчислюємо відстань до кожної точки даних на основі значення MaxHR
   var distances = _data
      .Select(action => new
      {
         Calories = action.Calories,
         Distance = Math.Abs(action.Duration - duration)
      })
      .OrderBy(result => result.Distance)
      .Take(_k) // Беремо k найближчих сусідів
```

```
                .ToList();

            // Обчислюємо середнє значення тривалості для найближчих сусідів
            double predictedDuration = distances.Average(result => result.Calories);
            return predictedDuration;
        }
```

Тут варто уточнити, що дана функція обчислює дані на основі 2 найближчих сусідів, цей параметр передається у функцію програмно. Дане число було обране аби можна було працювати маючи лише 3 тренування, а з ростом кількості тренувань ми будемо брати дві найближчі точки.

Останнім та найскладнішим методом прогнозування який використано в даній роботі є метод кубічних сплайнів.

Метод кубічних сплайнів – це чисельний метод для апроксимації кривих за допомогою кубічних поліномів, які забезпечують гладке з'єднання між точками даних. Основна ідея методу полягає в побудові кількох поліномів третього степеня, кожен з яких відповідає окремому інтервалу між точками, при цьому на кожній точці криві плавно з'єднуються. Це досягається шляхом встановлення обмежень на значення поліномів та їхніх похідних на кінцях інтервалів. Така аппроксимація дозволяє отримати гладку та природну криву, яка точно відповідає заданим точкам або проходить поруч із ними з мінімальними відхиленнями.

Для побудови кубічних сплайнів використовуються умови неперервності кривої, її першої та другої похідних на межах кожного інтервалу. Таким чином, для nn точок формується система з 4(n−1)4(n−1) рівнянь, де кожен поліном третього степеня задається своїми коефіцієнтами. Рішення цієї системи дає значення коефіцієнтів поліномів, які можна використати для обчислення значень функції у будь-якій точці інтервалу. Ця техніка добре підходить для випадків,

коли потрібна гладка інтерполяція з невеликим числом коливань між вузловими точками.

Кубічні сплайни широко використовуються в комп'ютерній графіці, робототехніці, обробці зображень та багатьох інших областях, де важлива плавність кривих. Завдяки своїй здатності зберігати природну форму і мінімізувати коливання, вони мають переваги над лінійною або квадратичною інтерполяцією. Наприклад, у моделюванні траєкторій або при згладжуванні даних кубічні сплайни забезпечують високий рівень точності й стабільності, особливо в умовах, де кількість вузлових точок значно перевищує кількість інтервалів між ними.

В даній реалізації було створено два класи CubicSplineHR та CubicSplineCal які роблять одне й те саме, але на основі різних даних.

Процес складається з ініціалізації та самого передбачення.

Ініціалізацію даних показано на лістингу 3.12.

Лістинг 3.12 CubicSplineCal

```
int n = data.Count - 1;
        a = new double[n + 1];
        b = new double[n];
        c = new double[n + 1];
        d = new double[n];
        x = new double[n + 1];
        for (int i = 0; i <= n; i++)
        {
           x[i] = data[i].Duration;
           a[i] = data[i].Calories;
        }
        double[] h = new double[n];
        for (int i = 0; i < n; i++)
```

```
{
    h[i] = x[i + 1] - x[i];
}
double[] alpha = new double[n];
for (int i = 1; i < n; i++)
{
    alpha[i] = (3 / h[i] * (a[i + 1] - a[i])) - (3 / h[i - 1] * (a[i] - a[i - 1]));
}

double[] l = new double[n + 1];
double[] mu = new double[n];
double[] z = new double[n + 1];
l[0] = 1;
mu[0] = 0;
z[0] = 0;
for (int i = 1; i < n; i++)
{
    l[i] = 2 * (x[i + 1] - x[i - 1]) - h[i - 1] * mu[i - 1];
    mu[i] = h[i] / l[i];
    z[i] = (alpha[i] - h[i - 1] * z[i - 1]) / l[i];
}
l[n] = 1;
z[n] = 0;
c[n] = 0;
for (int j = n - 1; j >= 0; j--)
{
    c[j] = z[j] - mu[j] * c[j + 1];
    b[j] = (a[j + 1] - a[j]) / h[j] - h[j] * (c[j + 1] + 2 * c[j]) / 3;
    d[j] = (c[j + 1] - c[j]) / (3 * h[j]);
```

```
    }
```

Перейдемо до самої функції передбачення, вона показана на лістингу 3.13.

Лістинг 3.13 PredictCal

```
public double PredictCal(double targetCalories)
    {
        int n = a.Length - 1;
        int i = n - 1;
        if (targetCalories > x[n])
        {
            i = n - 1;
        }
        else if (targetCalories < x[0])
        {
            i = 0;
        }
        else
        {
            for (int j = 0; j < n; j++)
            {
                if (targetCalories >= x[j] && targetCalories <= x[j + 1])
                {
                    i = j;
                    break;
                }
            }
        }
        double dx = targetCalories - x[i];
```

```
        return a[i] + b[i] * dx + c[i] * dx * dx + d[i] * dx * dx * dx;
    }
```

Таким чином було реалізовано один з найскладніших методів апроксимації, однак цей метод дає один з найкращих результатів див. розділ 4.

3.2 Особливості використання зовнішніх бібліотек

В даній роботі було використано дві зовнішні бібліотеки, а саме Newtonsoft.Json та Microsoft.Interop.Office.Excel.

Microsoft.Interop.Office.Excel – це бібліотека в рамках технології .NET, яка дозволяє програмам взаємодіяти з Microsoft Excel за допомогою об’єктної моделі. Ця бібліотека є частиною набору бібліотек Interop, що забезпечують інтерфейс для автоматизації та управління програмами Microsoft Office, такими як Excel, Word та PowerPoint, з коду C#. Використовуючи Microsoft.Interop.Office.Excel, розробники можуть виконувати різноманітні операції з Excel-документами, зокрема створювати й редагувати файли, змінювати вміст комірок, форматувати таблиці, запускати макроси, а також імпортувати та експортувати дані.

Основною перевагою цієї бібліотеки є її доступ до глибоких функцій Excel, що дозволяє працювати з Excel на рівні, аналогічному ручній роботі користувача. Наприклад, за допомогою C# можна відкрити існуючий файл Excel, внести зміни, зберегти його під іншим ім’ям або надіслати на друк. Microsoft.Interop.Office.Excel також надає доступ до великої кількості налаштувань Excel, таких як форматування комірок, налаштування графіків та діаграм, а також роботи з формулами, що є корисним у разі необхідності автоматизованої обробки складних даних.

Варто зазначити, що використання цієї бібліотеки має свої вимоги та обмеження. Наприклад, для її роботи на сервері необхідно встановити Microsoft Office, оскільки Microsoft.Interop.Office.Excel взаємодіє з самою програмою

Excel. Це також може створити певні обмеження для сценаріїв багатокористувацького середовища, оскільки Excel не призначений для одночасної обробки кількома потоками. Крім того, через складність налаштувань Excel API та високий рівень взаємодії з COM-об’єктами бібліотека потребує ретельного управління ресурсами для запобігання потенційним витокам пам’яті.

Дана бібліотека використовувалася для створення звітів у форматі ексель. Попри складність роботи з нею вона забезпечую високу швидкодію оскільки працює напряму з екселем.

Newtonsoft.Json — це популярна бібліотека для роботи з JSON у .NET середовищі, яка забезпечує прості та ефективні способи обробки даних формату JSON. Завдяки цій бібліотеці розробники можуть легко серіалізувати об'єкти, тобто перетворювати об'єкти в JSON-рядок, і десеріалізувати JSON назад у об'єкти. Це особливо корисно для роботи з веб-API, де JSON часто використовується як основний формат обміну даними між сервером і клієнтом. Бібліотека дозволяє зберігати дані у форматі JSON для подальшого використання, наприклад, для запису стану програми, обміну складними структурованими даними або передачі конфігурацій.

Однією з головних переваг Newtonsoft.Json є її гнучкість та розширюваність. Бібліотека надає широкий спектр налаштувань, які дозволяють кастомізувати процес серіалізації. Наприклад, можна легко змінити форматування ключів або ігнорувати певні властивості під час перетворення об'єктів. Також вона дозволяє працювати з JSON як з динамічними об’єктами, що зручно для обробки змінних форматів даних, які можуть надходити від зовнішніх API або зберігатися у файлах. Це означає, що розробникам не обов'язково створювати статичні класи для кожної можливої структури даних, а можна працювати з ними безпосередньо через бібліотеку, що економить час і спрощує процес обробки.

Ще одним значущим аспектом є висока продуктивність Newtonsoft.Json. Бібліотека оптимізована для роботи з великими обсягами даних та складними

JSON-структурами, що забезпечує швидку серіалізацію та десеріалізацію без значних витрат ресурсів. Це робить її ідеальним вибором для застосунків, які працюють з великими обсягами даних або обробляють складні об'єкти. Саме тому вона стала стандартом де-факто для роботи з JSON у середовищі .NET, забезпечуючи як потужні можливості налаштування, так і високу швидкість обробки даних, що робить її незамінним інструментом у сучасних додатках .NET.

Дана бібліотека використовувалася для серіалізації та десеріалізації даних у/з файлу. Ця бібліотека забезпечує як зручність та налаштовуваність так і швидкість виконання. Враховуючи особливість даної розробки це був критичний показник при виборі бібліотеки для серіалізації даних.

3.3 Тестування програмної системи

Ручне тестування — це метод перевірки якості програмного забезпечення, за якого тестувальник вручну виконує різні операції, щоб оцінити коректність роботи системи. Він детально досліджує поведінку програми, тестуючи окремі її функції, взаємодію інтерфейсу та реакцію на введення даних. У процесі тестувальник аналізує результати кожної дії та порівнює їх із запланованими, щоб виявити можливі невідповідності або помилки. Відсутність автоматизованих інструментів означає, що фокус роботи тестувальника повністю на інтерактивності з програмою, що дає можливість гнучко реагувати на будь-які несподівані аспекти.

Важливою перевагою ручного тестування є його здатність забезпечити якісну оцінку зручності користування програмою, оскільки тестувальник виконує ті ж дії, що й кінцевий користувач. Це дозволяє не лише перевірити технічну правильність, але й виявити можливі проблеми з дизайном, які ускладнюють роботу з програмою. Наприклад, тестувальник може помітити, що

певні функції знаходяться в незручному місці або не очевидні для користувача, що може призвести до труднощів у користуванні.

Ручне тестування охоплює широкий спектр тестових сценаріїв і зазвичай проводиться на різних етапах розробки: від базових перевірок функціональності до більш комплексних сценаріїв, які включають тестування інтеграції між модулями програми. Завдяки безпосередньому контакту з програмою, тестувальник може легко змінювати підхід до перевірок у реальному часі, якщо виявлені нові можливі точки відмови або незаплановані сценарії використання.

Попри значні переваги, ручне тестування має й обмеження, зокрема пов’язане з витратами часу та людським фактором. Воно займає більше часу, ніж автоматизовані тести, оскільки всі сценарії виконуються вручну і потребують значної уваги до деталей, що може призвести до помилок. Проте ручне тестування залишається необхідною частиною перевірки якості програмного забезпечення, оскільки дозволяє глибше зрозуміти поведінку програми з точки зору кінцевого користувача.

В рамках даного проекту було проведено 7 ітерацій тестування кожне містило близько 15-20 тестових випадків. По результатах шести ітерацій було виправлено всі знайдені баги та оптимізовано код. При останній сьомій ітерації не було знайдено багів. Таким чином програма вважається відтестованою, баги виправленими. Якщо у програмі і залишились баги то вони не критичні та для їх відтворення необхідне нетипове використання програми.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ПЕРЕДБАЧЕНЬ ІНТЕНСИВНОСТІ ТРЕНУВАНЬ

4.1 Дослідження передбачень методом лінійної регресії

Метод лінійної регресії — це статистичний метод, який використовується для моделювання залежності між двома або більше змінними та виявлення лінійного зв’язку між ними. Основна мета лінійної регресії — знайти пряму лінію, яка найкраще описує взаємозв'язок між незалежною змінною (або змінними) та залежною змінною. У випадку простої лінійної регресії, коли є лише одна незалежна змінна x і одна залежна y, модель передбачає лінійну функцію виду:

$$y = b_0 + b_1 x \tag{4.1}$$

де y — прогнозоване значення залежної змінної, x — незалежна змінна, b_0 — вільний член (перетин з віссю y), а b_1 — коефіцієнт нахилу, який показує, на скільки зміниться y при зміні x на одну одиницю.

Коефіцієнти b_0 і b_1 обчислюються методом найменших квадратів, який мінімізує суму квадратів відстаней між фактичними значеннями y і прогнозованими значеннями $\hat{y}$. Для цього використовується функція помилки, або функція втрат, яка записується так:

$$SSE = \Sigma(y_i - \hat{y}_i)^2 \quad (4.2)$$

де y_i — фактичне значення залежної змінної, $\hat{y}_i$ — значення, прогнозоване за допомогою лінійної моделі, а n — кількість спостережень. Метод найменших квадратів шукає такі значення b_0 та b_1, при яких значення функції SSE буде мінімальним.

Формули для обчислення коефіцієнтів у простій лінійній регресії такі:

$$b_1 = \Sigma((x_i - \bar{x})(y_i - \bar{y})) / \Sigma((x_i - \bar{x})^2) \quad (4.3)$$

$$b_0 = \bar{y} - b_1 \bar{x}, \quad (4.4)$$

де $\bar{x}$ і $\bar{y}$ — середні значення x та y відповідно. Після обчислення коефіцієнтів b_0 та b_1 модель можна використовувати для прогнозування значень y на основі нових значень x.

Лінійна регресія широко застосовується в економіці, соціології, біології та багатьох інших галузях для побудови прогнозів, виявлення трендів і оцінки впливу різних факторів.

4.2 Дослідження передбачень методом поліноміальної регресії

Метод найменших квадратів для поліноміальної регресії другого ступеня використовується для побудови квадратичної моделі, яка наближає дані за допомогою параболи. У цьому випадку функція регресії має вигляд:

$$y = b_0 + b_1 x + b_2 x^2, \tag{4.5}$$

де y — прогнозоване значення залежної змінної, x — незалежна змінна, а b_0, b_1 та b_2 — коефіцієнти, які потрібно знайти. Поліноміальна регресія другого ступеня дозволяє моделювати нелінійні залежності, що є більш гнучким підходом порівняно з лінійною регресією.

Коефіцієнти b_0, b_1 та b_2 визначаються мінімізацією суми квадратів відхилень між фактичними значеннями y_i та прогнозованими значеннями $\bar{y}$, що записується так:

$$SSE = \Sigma(y_i - \hat{y}_i)^2 \tag{4.6}$$

де y_i — фактичне значення залежної змінної, $\hat{y}_i$ — значення, прогнозоване за допомогою лінійної моделі, а n — кількість спостережень. Метод найменших квадратів шукає такі значення b_0 та b_1, при яких значення функції SSE буде мінімальним.

Поліноміальна регресія другого ступеня корисна для моделювання даних, де залежність між змінними має форму параболи.

4.3 Дослідження передбачень методом К-сусідів

Метод K-сусідів (K-Nearest Neighbors, KNN) є простим і потужним алгоритмом для класифікації та регресії, який базується на принципі, що схожі об'єкти знаходяться близько один до одного у просторі ознак. Цей метод працює

на основі відстані між даними, зокрема використовуючи евклідову відстань, хоча можуть бути використані й інші метрики.

Основна ідея методу полягає в тому, що для класифікації нового спостереження алгоритм шукає K найближчих сусідів у навчальному наборі даних, які вже мають відомі мітки класів. Клас нового спостереження визначається за більшістю класів його сусідів: якщо більшість сусідів належить до одного класу, нове спостереження також буде віднесено до цього класу.

Для регресії метод K-сусідів працює аналогічно, але замість того, щоб визначати клас, він обчислює середнє або зважене середнє значення цільової змінної сусідів, щоб зробити прогноз.

Вибір значення K є критично важливим. Занадто мале значення K може призвести до впливу шуму на класифікацію, тоді як занадто велике значення може розмивати кордони класів, що може знизити точність моделі. Зазвичай для вибору оптимального K проводиться валідація з використанням крос-валідації, щоб забезпечити збалансований підхід.

Метод K-сусідів є нелінійним, не параметричним методом, який легко реалізувати і зрозуміти. Проте його недоліком є те, що він може бути обчислювально витратним, особливо для великих наборів даних, оскільки вимагає обчислення відстані до всіх точок навчання для кожного нового спостереження. Для покращення швидкості часто використовують різні методи оптимізації, такі як KD-дерева або LSH (Locality-Sensitive Hashing)

4.4 Дослідження передбачень методом кубічних сплайнів

Кубічні сплайни є потужним інструментом для інтерполяції і апроксимації функцій, що використовують шматкові кубічні поліноми для побудови гладкої кривої, яка проходить через задані точки даних. Цей метод дозволяє зберегти гладкість першої та другої похідних у точках, де відбувається зміна поліномів, що робить сплайни особливо корисними в чисельному моделюванні, комп'ютерній графіці та обробці сигналів.

Спочатку необхідно мати набір контрольних точок (вузлів), через які повинна проходити крива, наприклад, точки $(x_0,y_0)(x_0,y_0)$, $(x_1,y_1)(x_1,y_1)$ тощо. Для кожної пари сусідніх вузлів (xi,yi) і (xi+1,yi+1) будується кубічний поліном, який має вигляд

$$S_i(x)=a_i+b_i(x-x_i)+c_i(x-x_i)^2+d_i(x-x_i)^3 \quad (4.7)$$

Сплайн $S_i(x)$ визначає інтервал $[x_i,x_i+1]$.

Для забезпечення плавності сплайна необхідно задати умови з'єднання, згідно з якими значення сплайнів у сусідніх точках повинні бути рівними. Також перші та другі похідні сплайнів у сусідніх точках повинні бути рівними. Для визначення коефіцієнтів a_i, b_i, c_i та d_i необхідно задати граничні умови, наприклад, фіксовані значення для крайніх точок або природні умови, коли друга похідна на краях дорівнює нулю.

В результаті з'являється система лінійних рівнянь для знаходження невідомих коефіцієнтів, яку можна розв'язати для отримання значень a_i, b_i, c_i та d_i. Кубічні сплайни забезпечують високу точність інтерполяції, зберігаючи гладкість і контролюючи поведінку кривої в межах інтервалів. Вони широко використовуються в комп'ютерній графіці, інженерії та обробці зображень, завдяки своїй здатності створювати плавні та естетично привабливі криві.

4.5 Результати дослідження.

Прогнозування відбувається в рамках бігу та велотренувань і хоч кожен атлет може і бігати і їздити на велосипеді, однак ці дані не впливають одні на одні тому для прогнозування велотренувань не використовуються дані з інших типів тренувань окрім вело. Це ж стосується з прогнозування бігу.

У нас є атлет який хоче максимально схуднути і при цьому не померти від виснаження. Тобто нам потрібно запропонувати йому такий план тренування

який спалить йому найбільшу кількість калорій і при цьому його пульс не підніметься вище певного значення.

Значення пульсу яке ми беремо до уваги визначається за формулою 4.8.

$$maxHR = (220\text{-}age)\text{-}10\% \quad (4.8a)$$

$$maxHR = (226\text{-}age)\text{-}10\% \quad (4.8b)$$

де maxHR – це максимальний пульс який ми можемо дозволити атлету, а age – це вік атлета. Формули 4.8a та 4.8b стосуються чоловіків та жінок відповідно.

Таким чином для атлета 22х років чоловічої статі максимальний допустимий пульс під час тренування рівний 198 ударів за хвилину. Однак оскільки ніхто не хоче аби атлет перестарався і перейшов свою межу будемо враховувати пульс на 10% нижчий, тобто 178 ударів за хвилину.

Далі нам потрібно спрогнозувати інтенсивність тренування яка буде при значенні пульсу в 178 ударів за хвилину. А вже після цього на основі отриманого значення інтенсивності передбачити прогнозовану кількість калорій яку спалить атлет.

Вхідні дані тренувань обробляються і залишається лише три поля це інтенсивність тренування, максимальний пульс та калорії. Інші дані нам не потрібні, хіба що для обчислення інтенсивності тренування.

На вхід у нас будуть такі дані див рис 4.1.

ActionLR.Duration	ActionLR.Calories	ActionLR.maxHR
5	50	90
15	150	160
12	120	130
8	65	98

Рис 4.1. Дані для прогнозування бігу

На рис 4.2. показано плани тренувань для бігу які були спрогнозовані різними методами.

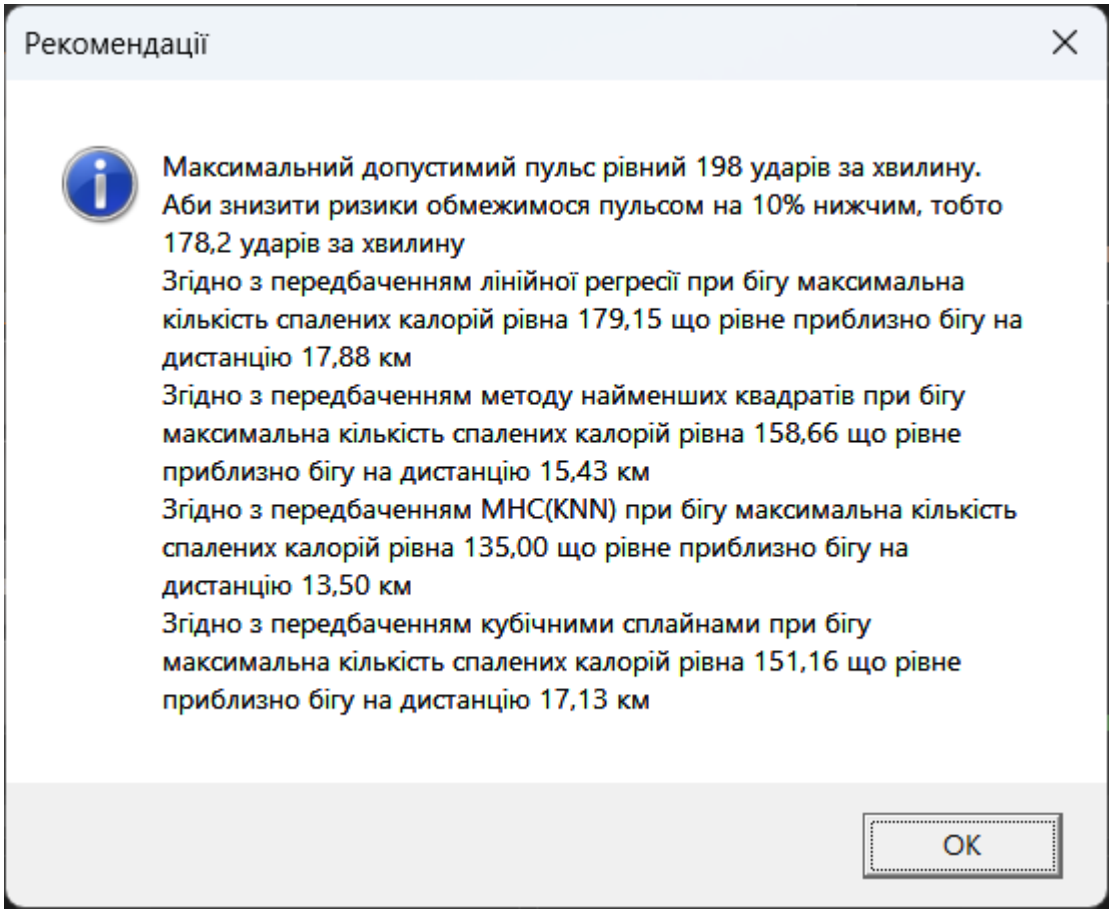


Рис 4.2. Рекомендації для бігу

Як бачимо на конкретно цих даних всі чотири методи передбачають що можна спалити від 135 до 179 калорій і для цього необхідно пробігти від 13,5 до 17.88 км.

Це доволі великий розкид, він пояснюється тим, що лінійна регресія дуже неточний метод, а метод К сусідів погано розкриває себе на невеликих вибірках. Інші два методи мають несуттєву розбіжність по спалених калоріях (5%) та по пройденій дистанції (10%)

Для велотренувань використовуються наступні дані показані на рис 4.3.

ActionLR.Duration	ActionLR.Calories	ActionLR.maxHR
168	155	130
120	120	99
182	166	160
225	200	180

Рис 4.3. Дані для прогнозування велотренувань

На рис 4.4. показані плани велотренувань.

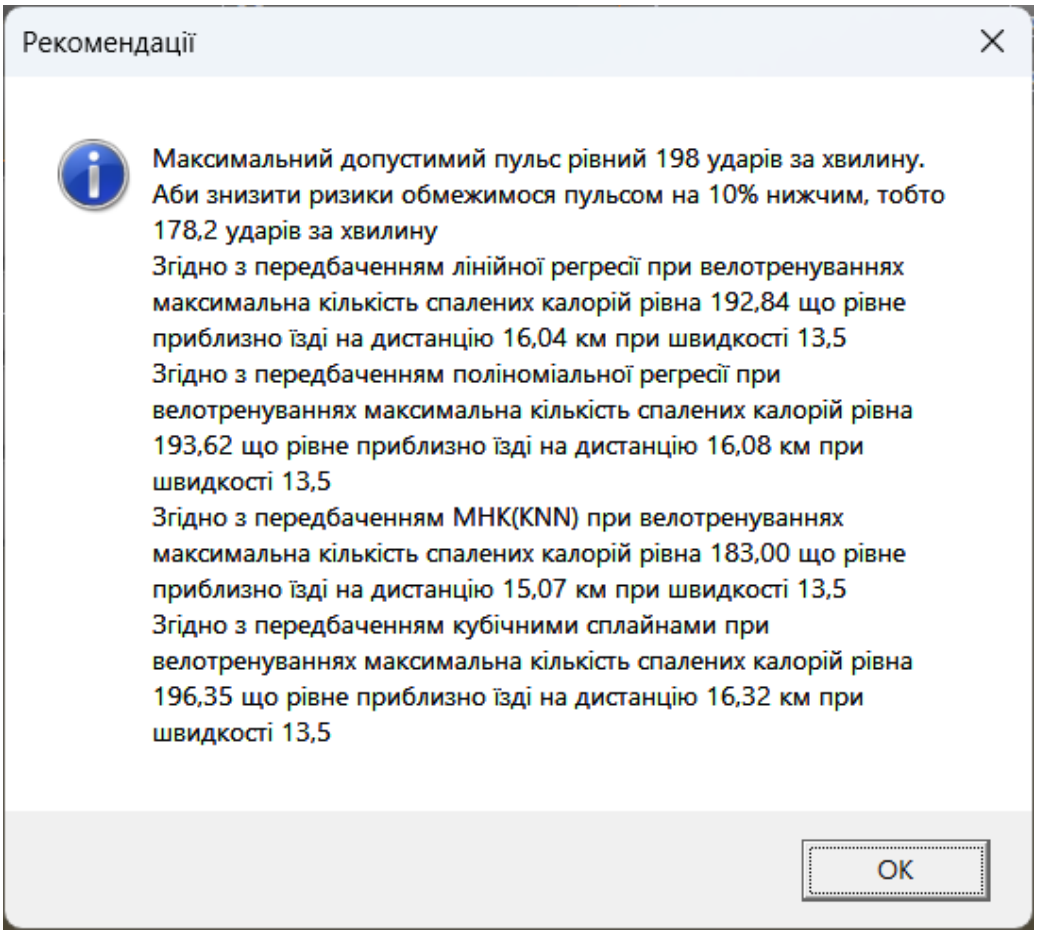


Рис 4.4. Рекомендації для велотренувань

Як бачимо на конкретно цих даних всі чотири методи передбачають що можна спалити від 183 до 196,35 калорій і для цього необхідно проїхати від 15,07 до 16,32 км з середньою швидкістю 13,5 км/год.

Значення 13,5 км/год обчислене як середня швидкість всіх тренувань. Воно допоміжне для визначення відстані, оскільки дана модель прогнозує

інтенсивність тренування, а у випадку з велотренуванням інтенсивність обчислюється як добуток середньої швидкості та відстані.

Тут варто зазначити що всі тестові значення бралися з уявної людини і можуть суттєво відрізнятися від результатів тренувань інших людей, проте для аналізу прогнозування вони цілком годяться.

Таким чином метод ітерацій показав непогані результати якщо всі дані знаходяться в суттєвому розкиді і їх багато.

Метод поліноміальної регресії показав хороші результати при прогнозуванні даних які розподіляються за нормальним законом розподілу, проте знаходяться близько один від одного.

Метод К-сусідів показав мабуть найгірші результати, при використанні 2 сусідів, однак він годиться при використанні прогнозування вже третього тренування.

Одним з найкращим методом прогнозування виявився метод кубічних сплайнів, однак він дуже неточний якщо є результати тренувань показують різні результати при схожих вхідних параметрах. Тобто якщо вказати умовно інтенсивність 100 при бігу на дистанцію 5 км і 150 при бігу на дистанцію 5,1 км, то метод кубічних сплайнів почне прогнозувати з суттєвою похибкою яка буде зменшуватися чим далі дані віддаляються від точки в 5 км.

Для цього необхідно проводити так звану чистку даних. Тобто знаходити дані які суттєво відрізняються від інших та не брати їх в розрахунок. Для цього може згодитися метод лінійної регресії чи метод поліноміальної регресії. Необхідно ввести певне значення похибки X та видалити всі дані які відрізняються більше ніж на X від прогнозованих в своїй точці.

ВИСНОВКИ

У магістерській роботі було проведено аналіз предметної області, актуальних мов програмування та середовищ розробки програмного забезпечення, що дозволило визначити найефективніші інструменти для

реалізації поставленого завдання, а саме мову програмування C# та середовище розробки MS Visual Studio. На основі аналізу було сформульовано вимоги до програмної системи, описано можливі варіанти її використання, а також спроектовано інтерфейс, структури даних та класи для реалізації основної функціональності.

У процесі моделювання системи було розроблено ключові сценарії використання та розглянуто додатковий функціонал, що покращує зручність користування програмою. Окрему увагу приділено моделюванню інтерфейсу користувача, який забезпечує інтуїтивність і зручність у роботі з системою. Завдяки ретельному моделюванню класів і структур даних було створено оптимальні рішення для обробки й зберігання інформації в межах системи, що є основою для надійної роботи програми.

На етапі реалізації програмної системи було запропоновано та впроваджено основні алгоритми, що забезпечують ефективність виконання основних операцій програми. Зокрема, були застосовані зовнішні бібліотеки такі як Newtonsoft.json та Microsoft.Interop.Office.Excel, які значно прискорили процес розробки та підвищили функціональні можливості системи. Після реалізації проведено тестування системи для виявлення та виправлення помилок, що забезпечило стабільність і надійність роботи програми.

У заключному етапі роботи проведено дослідження методів передбачення інтенсивності тренувань за допомогою методів лінійної регресії, поліноміальної регресії, К-сусідів та кубічних сплайнів. Аналіз результатів продемонстрував, що кожен із розглянутих методів має свої переваги і недоліки залежно від специфіки даних та мети аналізу. Підсумовуючи, обрані методи та інструменти дозволили розробити програмну систему, яка відповідає вимогам та забезпечує точність у прогнозуванні інтенсивності тренувань. Це робить її ефективним інструментом для подальшого використання та вдосконалення в межах предметної області.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++. — 2016. — 1328 p. — ISBN 978-5-8459-1949-6. Siddhartha Rao. Sams Teach Yourself

C++ in One Hour a Day, 7th Edition. — Moscow: Williams, 2013. — 688 p. — ISBN 978-5-8459-1825-3.

2. Stephens D. R. C++. A Collection of Recipes. — KUDITS-PRESS, 2007. — 624 p. — ISBN 5-91136-030-6.

3. Stephen Prata. The C++ Programming Language (C++11). Lectures and Exercises = C++ Primer Plus, 6th Edition (Developer's Library). — 6th ed. — M.: Williams, 2012. — 1248 p. — ISBN 978-5-8459-1778-2.

4. John Skeet. C# in Depth, 3rd ed., New Translation = C# in Depth, 3rd ed. — Moscow: Williams, 2014. — 608 p. — ISBN 978-5-8459-1909-0.

5. Christian Nagel et al. C# 5.0 and the .NET 4.5 Platform for Professionals = Professional C# 5.0 and .NET 4.5. — Moscow: Dialectika, 2013. — 1440 p. — ISBN 978-5-8459-1850-5.

6. A. Hejlsberg, M. Torgersen, S. Wiltamuth, P. Gold. The C# Programming Language (Covering C# 4.0), 4th Ed. — SPb.: "Piter", 2012. — 784 p. — ISBN 978-5-459-00283-6. Archived October 10, 2011 at the Wayback Machine

7. Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland, David Svoboda. Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs. 2013. — 256 c. — ISBN 9780133439526.

8. Cay S. Horstmann, Core Java. Volume I - Fundamentals (Tenth Edition). 2016. — 864 c. — ISBN 978-0134177304.

9. Cay S. Horstmann. Core Java. Volume II - Advanced Feature (Tenth Edition), 2016. — 976 c. — ISBN 978-0134177298.

10. Jan Palach. Parallel Programming with Python. — Packt Publishing Ltd, 2014.

11. Yaworski Michal, Ziade Tarek. Expert Python Programming (Fourth Edition). 2021. — 630 c. — ISBN 978-1801071109

12. Eric Matthes. Python Crash Course, 3rd Edition: A Hands-On, Project-Based Introduction to Programming. 2023. — 552 p. — ISBN 978-1718502703.

13. SharpDevelop: веб-сайт. URL: https://sourceforge.net/projects/sharpdevelop/ (дата звернення: 23.10.2024).

14. Rider: веб-сайт. URL: https://www.jetbrains.com/rider/ (дата звернення: 23.10.2024).

15. Visual studio: веб-сайт. URL: https://visualstudio.microsoft.com/ru/ (дата звернення: 23.10.2024).

16. VS code: веб-сайт. URL: https://code.visualstudio.com/updates/v1_60 (дата звернення: 23.10.2024).

17. Nuget: веб-сайт. URL: https://www.nuget.org/ (дата звернення: 23.10.2024).

18. Github: веб-сайт. URL: https://github.com/ (дата звернення: 23.10.2024).

19. Advanced Encryption Standard: веб-сайт. URL: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard (дата звернення: 24.10.2024).

20. Серіалізація. URL: https://uk.wikipedia.org/wiki/Серіалізація (дата звернення: 27.10.2024).

21. JSON. URL: https://en.wikipedia.org/wiki/JSON (дата звернення: 28.10.2024).