МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,

д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 2025 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Система очищення наборів даних із використанням методів виявлення шумів і викидів

Спеціальність 123 Комп'ютерна інженерія

123 – КМР.01 – 605.22450501

Студент _____ Дмитро БАЖЕНОВ

підпис

«__» _____ 202__ р.

Керівник д-р фіз.-мат. наук, професор _____ Геннадій ЧУЙКО

підпис

«__» _____ 202__ р.

Миколаїв – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Ірина ЖУРАВСЬКА

«_____» _____ 2025 р.

ЗАВДАННЯ
на виконання кваліфікаційної магістерської роботи

Видано студенту групи 605 факультету комп'ютерних наук

Баженов Дмитро Сергійович
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Система очищення наборів даних із використанням методів виявлення шумів і викидів

Затверджена наказом по ЧНУ ім. Петра Могили від 23.06.2025 № 165/1.

2. Строк представлення кваліфікаційної роботи «_10_» __грудня___ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Розробка програмної системи (десктопного застосунку на Python з використанням Pandas, Scikit-learn, PyQt5) для автоматизованого очищення наборів даних від атрибутивних шумів (методи Z-score, IQR) та шуму міток (алгоритмCAIRAD). Реальний медичний набір даних Heart Disease UCI для проведення експериментальних досліджень.

4. Перелік питань, що підлягають розробці
Аналіз існуючих методів виявлення аномалій та інструментальних засобів очищення даних; обґрунтування вибору та математичне моделювання методів детекції викидів (Z-score, IQR) та шуму міток (CAIRAD); розробка архітектури програмного забезпечення на основі патерну MVC; Програмна реалізація алгоритмів очищення даних мовою Python; Розробка графічного інтерфейсу користувача; проведення експериментальних досліджень ефективності системи на реальних даних.

5. Перелік графічних матеріалів

Схеми класифікації шумів у наборах даних; блок-схема алгоритму роботи системи; проєктування архітектури програмного забезпечення; логічна схема роботи алгоритму CAIRAD; математичне обґрунтування методів детекції; графічний інтерфейс користувача; візуалізація розподілу числових атрибутів; результати експериментальних досліджень.

Керівник роботи _____ ___Геннадій ЧУЙКО_

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ ___Дмитро БАЖЕНОВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Система очищення наборів даних із використанням методів виявлення шумів і викидів

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КМР	01.09.2025	05.09.2025	Виконав
2	Вибір літератури за темою роботи	06.09.2025	18.09.2025	Виконав
3	Складання календарного плану КМР	19.09.2025	25.09.2025	Виконав
4	Аналіз вибраної літератури	26.09.2025	08.10.2025	Виконав
5	Розробка проєктних рішень	09.10.2025	18.10.2025	Виконав
6	Моделювання АПЗ	19.10.2025	28.10.2025	Виконав
7	Перевірка та тестування АПЗ	29.10.2025	05.11.2025	Виконав
8	Оформлення КМР	06.11.2025	10.11.2025	Виконав
9	Попередній захист	11.11.2025	11.11.2025	Виконав
10	Відгук керівника КМР	12.11.2025	15.11.2025	Виконав
11	Рецензування	16.11.2025	24.11.2025	Виконав
12	Завершення формлення КМР	25.11.2025	25.11.2025	Виконав
13	Захист кваліфікаційної магістерської роботи	26.11.2025	10.12.2025	Виконав
14	Розробка та затвердження завдання на виконання КМР	16.12.2025	16.12.2025	Виконав

Керівник роботи _____ Геннадій ЧУЙКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Дмитро БАЖЕНОВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

АНОТАЦІЯ

до кваліфікаційної магістерської роботи
«Система очищення наборів даних із використанням методів виявлення шумів і викидів»
Студент 605 гр.: Баженов Дмитро Сергійович
Керівник: д-р фіз.-мат. наук, професор Чуйко Геннадій Петрович

Кваліфікаційна магістерська робота присвячена розробці програмної системи. Головна її задача — автоматизоване очищення наборів даних від шумів та викидів. Це робиться для того, щоб у результаті підвищити якість моделей машинного навчання. Актуальністю є необхідність у створенні доступних інструментів для попередньої обробки. Важливо, щоб вони були здатні виявляти складні речі: семантичні помилки, шум міток. На практиці, особливо для медичних чи аналітичних систем, це часто є критичним моментом.

Об’єкт дослідження – процес попередньої обробки та підготовки структурованих даних.

Предметом дослідження є методи, алгоритми та програмні засоби автоматизованого виявлення та усунення шумів, викидів і помилкових міток.

У ході роботи реалізовано десктопний застосунок мовою Python (з використанням бібліотек Pandas, Scikit-learn, PyQt5), який поєднує статистичні методи фільтрації (Z-score, IQR) та алгоритм CAIRAD для виявлення помилок розмітки класів. Проведено експериментальне дослідження на реальному медичному наборі даних Heart Disease UCI, яке показало приріст точності класифікації на 9,45 % після очищення.

Робота пройшла апробацію на XXVIII Всеукраїнській науково-практичній конференції «Могилянські читання – 2025» (Миколаїв, 2025 р.).

Пояснювальна записка магістерської роботи складається зі вступу, чотирьох розділів, висновків, переліку джерел посилання та додатків. У вступі визначається актуальність теми, сформульовані мета, об’єкт, предмет роботи та завдання. У першому розділі проводиться аналіз методів виявлення аномалій. Другий розділ присвячено математичному моделюванню. Третій описує програмну реалізацію системи. Четвертий містить результати експериментальних досліджень.

Ключові слова: *очищення даних, виявлення викидів, машинне навчання, CAIRAD, Python, Pandas, медичні дані.*

ABSTRACT

of the Master's Thesis

"System for Data Sets Cleaning Using Noise and Outlier Detection Methods"

Student: D. Bazhenov

Supervisor: Ph.D. in Physics and Mathematics, Professor Hennadii Chuiko

The master's thesis is focused on the development of an automated software system for cleaning structured datasets using statistical methods and machine learning algorithms. The relevance of the research lies in the critical impact of data quality on the accuracy of predictive models, especially in medical diagnostics, where noise and mislabeled cases can lead to erroneous decisions.

The object of the study is the process of data preprocessing and preparation for machine learning tasks.

The subject of the study is the methods, algorithms, and software tools for detecting outliers and class noise in datasets.

The system includes a Python-based desktop application with a graphical interface (PyQt5) that implements Z-score and IQR methods for attribute noise detection and the CAIRAD algorithm for identifying mislabeled instances. The software allows loading CSV files, configuring cleaning parameters, visualizing data distributions, and exporting cleaned datasets.

The work was approved at the XXVIII All-Ukrainian Scientific and Practical Conference "Mohyla Readings - 2025" (Mykolaiv, 2025).

The explanatory note of the master's thesis consists of an introduction, four sections, conclusions, a list of references, and appendices. The introduction determines the relevance of the topic, formulates the goal, object, subject of the work, and tasks to achieve the set goal. The first section analyzes existing methods and tools for outlier detection. The second section describes mathematical models. The third section covers software implementation. The fourth section presents experimental results on the Heart Disease UCI dataset.

Keywords: *Data Cleaning, Outlier Detection, Machine Learning, CAIRAD, Python, Pandas, Medical Data.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ	– програмне забезпечення
СУБД	– пистема управління базами даних
API	– Application Programming Interface
CSV	– Comma-Separated Values
GUI	– Graphical User Interface
IQR	– Interquartile Range
ML	– Machine Learning
MVC	– Model-View-Controller
RAM	– Random Access Memory
UI	– User Interface

ВСТУП

Останні кілька років, коли всі говорять про штучний інтелект і моделі машинного навчання, усе частіше з’являється одна й та сама тема – якість вхідних даних. Системи можуть накопичувати терабайти інформації, сховища ростуть, логів стає більше, але частина цих даних насправді проблемна. У них трапляються шуми, пропуски, дивно великі або підозріло малі значення, а інколи й просто неправильно проставлені мітки. І, як показує практика, такого «сміття» в реальних наборах значно більше, ніж очікується спочатку.

Класичне очищення даних зазвичай або виконується вручну, або обмежується простими фільтрами. У типових ситуаціях цього може вистачати, але складніші помилки такі підходи часто не помічають. Наприклад, коли набір симптомів явно не узгоджується з діагнозом, але формально всі значення «лежать у допустимих межах». Подібні випадки прості правила просто пропускають.

З готовими інструментами теж є свої труднощі: частина з них вимагає досить глибоких навичок програмування, а інша частина вже виглядає застарілою й не надто зручною для щоденної роботи з медичними даними. Тому й виникає ідея автоматизованої системи очищення, яка поєднує алгоритми пошуку аномалій із зрозумілим інтерфейсом. Її завдання – дати фахівцю можливість підготувати дані до аналізу без зайвої технічної складності та без великого обсягу ручної рутинної роботи.

Мета роботи полягає в дослідженні методів виявлення аномалій і розробці системи для автоматизованого очищення даних від шумів. Головна ціль — підвищити точність моделей класифікації.

Для цього треба вирішити такі завдання:

1) проаналізувати існуючі методи (статистику, кластеризацію) та інструменти очищення, які вже є на ринку;

2) обґрунтувати вибір конкретних алгоритмів. Для атрибутивних викидів вирішено використовувати Z-score та IQR, а для шуму міток — метод CAIRAD;

3) розробити архітектуру ПЗ на основі патерну MVC. Це дозволить зробити систему модульною і придатною до розширення;

4) програмно реалізувати ці алгоритми на Python, використовуючи бібліотеки Pandas та Scikit-learn;

5) зробити графічний інтерфейс (GUI). Щоб можна було завантажити файл, налаштувати фільтрацію і візуально оцінити результат;

6) провести експерименти на реальних даних (наприклад, медичний набір Heart Disease UCI) і подивитися, як очищення впливає на точність класифікації.

Предметом дослідження є методи, алгоритми та програмні засоби автоматизованого виявлення та усунення шумів, викидів і помилкових міток у наборах даних.

Наукова новизна отриманих результатів пов’язана з тим, що підхід до очищення медичних даних було уточнено й посилено. Ідея тут доволі проста за змістом: класичні статистичні фільтри поєднано з алгоритмом CAIRAD, який аналізує спільну появу ознак. За рахунок цього вдалося точніше знаходити семантичні помилки в розмітці класів. Тобто мова йде не лише про «шум» у числах, а про логічні невідповідності в самих записах.

Практична цінність роботи полягає в тому, що розроблена система отримана не у вигляді окремих ідей, а як готовий програмний продукт. Вона автоматизує типові етапи очищення, з якими аналітики постійно стикаються під час підготовки даних.

Апробація результатів кваліфікаційної роботи відбулася під час XXVIII Всеукраїнської науково-практичної конференції «Могилянські читання – 2025» (Миколаїв, 2025 р.) на тему «Feature engineering and noise reduction for cardiovascular risk prediction in WEKA».

1 АНАЛІЗ МЕТОДІВ ОЧИЩЕННЯ ДАНИХ ТА ФОРМУЛЮВАННЯ ВИМОГ ДО СИСТЕМИ

1.1 Аналіз та проблематика шумів у наборах даних

1.1.1 Поняття якості даних у контексті Data Science

В епоху Big Data, коли інформація накопичується шаленими темпами, головним викликом стає навіть не стільки обсяг даних, скільки їх вірогідність (Veracity). Тобто питання в тому, наскільки цим даним взагалі можна вірити. Якість даних — це штука комплексна; вона, по суті, визначає, чи придатна інформація для використання у конкретній задачі.

У машинному навчанні та Data Mining якість вхідної вибірки — це фундаментальне обмеження. Тут діє принцип GIGO (Garbage In, Garbage Out).

Сенс простий: якщо в навчальних даних є критична маса помилок, пропусків чи суперечностей, то навіть найдосконаліша нейромережа не видасть достовірного прогнозу.

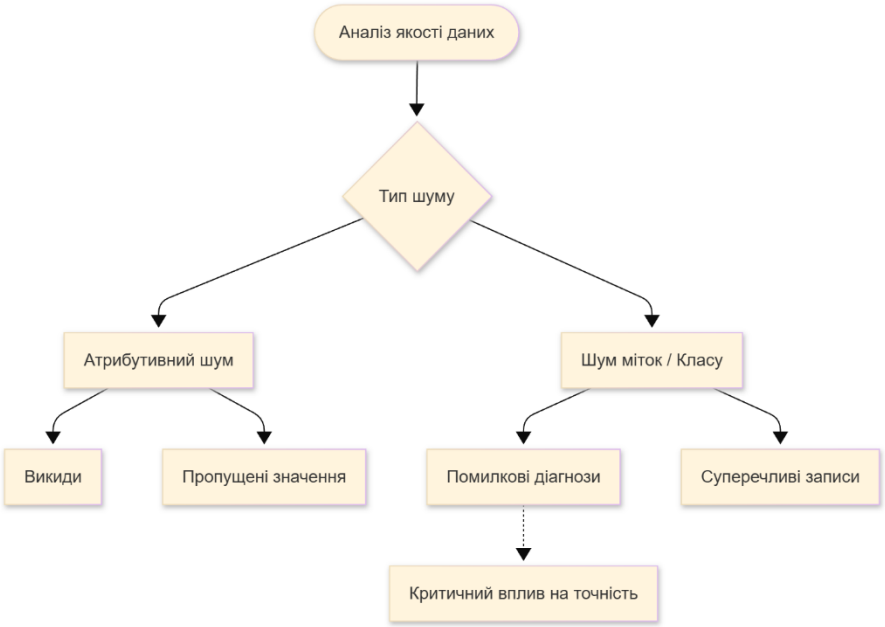
Особливо гостро ця проблема відчувається там, де ціна помилки критична: в медичній діагностиці, фінансовому скорингу чи управлінні виробництвом. Взяти, наприклад, медичні набори даних. Помилки там можуть виникати будь-де: на етапі збору анамнезу, під час лабораторних вимірювань або банально при ручному введенні інформації в електронну картку. Як показують дослідження, реальні набори даних часто містять від 5 до 30 % шумів різної природи. А це означає, що без спеціальних процедур попередньої обробки (Preprocessing) тут ніяк не обійтися.

1.1.2 Класифікація спотворень: атрибутивний шум та шум міток

У науковій літературі під «шумом» розуміють будь-яке відхилення даних від істинної моделі генерації, яке не несе корисної інформації про досліджуваний процес. Для побудови ефективної системи очищення необхідно чітко розрізняти типи спотворень (рис. 1.1).

По-перше, виділяють атрибутивний шум (Attribute Noise). Це спотворення значень незалежних змінних (ознак), які використовуються для опису об'єкта. Атрибутивний шум може проявлятися у вигляді:

- помилкових значень (Error values), що виходять за межі допустимого діапазону (наприклад, від'ємний вік пацієнта);
- пропущених значень (Missing values), що виникають внаслідок збоїв сенсорів або відсутності інформації;
- викидів (Outliers) – значень, які формально знаходяться в межах допустимого діапазону, але суттєво відрізняються від загального розподілу вибірки.



Рисунку 1.1 – Класифікація шумів у наборах даних

По-друге, існує шум міток класів (Class Noise / Label Noise). Це ситуація, коли об'єкту, описаному набором коректних атрибутів, приписується неправильний клас (цільова змінна). У медичній практиці це може статися внаслідок лікарської помилки при встановленні діагнозу або неоднозначності симптомів 1. Шум міток вважається найбільш деструктивним для алгоритмів керованого навчання (Supervised Learning), оскільки він безпосередньо

впливає на формування вирішальних правил класифікатора.Формально задачу виявлення шумів можна описати так. Нехай $D = \{x_1, x_2, \dots, x_n\}$ – набір даних, де кожен x_i є вектором у d-вимірному просторі ознак. Задача полягає у розбитті множини D на дві підмножини: D_{clean} (коректні дані) та D_{noise} (шум), таким чином, щоб $|D_{noise}| \ll |D_{clean}|$ і елементи D_{noise} максимізували функцію відхилення від нормальної поведінки системи.

1.2 Огляд існуючих методів та алгоритмів виявлення викидів

1.2.1 Статистичні методи параметричного аналізу

Статистичні методи базуються на припущенні, що «нормальні» дані підпорядковуються відомому стохастичному розподілу (найчастіше – нормальному розподілу Гауса). Відповідно, об'єкти, які мають низьку ймовірність генерації в рамках цієї моделі, вважаються викидами.

Найпростішим методом є Z-оцінка (Z-score). Для одновимірного масиву даних X з середнім значенням μ та стандартним відхиленням σ, Z-оцінка для точки x_i обчислюється за формулою:

$$Z_i = \frac{x_i - \mu}{\sigma}.$$

Якщо $|Z_i| > 3$, точка вважається викидом (правило трьох сигм). Перевагою методу є простота реалізації та низька обчислювальна складність $O(n)$. Проте метод має суттєві недоліки: він чутливий до самих викидів (які спотворюють μ та σ) і працює коректно лише для даних з нормальним розподілом.Більш робастним (стійким) аналогом є метод інтерквартильного розмаху (IQR). Він базується на порядкових статистиках.

Обчислюються перший (Q_1) та третій (Q_3) квартилі. Інтерквартильний розмах становить $IQR = Q_3 - Q_1$. Викидами вважаються точки, що лежать поза межами інтервалу:

$$[Q_1 - 1.5 \cdot IQR; \quad Q_3 + 1.5 \cdot IQR]$$

Цей метод не залежить від типу розподілу даних і є ефективним для попередньої фільтрації явних помилок вимірювання.

1.2.2 Методи на основі відстані та щільності (Distance-Based & Density-Based)

У багатовимірних просторах, де поняття «нормального розподілу» стає розмитим, застосовують геометричні підходи.

Метод k-найближчих сусідів (k-NN) для детекції аномалій використовує відстань до k-го сусіда як міру аномальності. Для кожної точки x обчислюється відстань $D_k(x)$ до її k-го найближчого сусіда. Точки з найбільшими значеннями $D_k(x)$ вважаються викидами. Цей підхід не вимагає припущень про розподіл даних, але має високу обчислювальну складність $O(n^2)$, що ускладнює його використання на великих вибірках.

Окремої уваги заслуговує алгоритм кластеризації DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Він автоматично класифікує точки на три типи:

Ядрові точки (Core points): мають не менше MinPts сусідів у радіусі ϵ. Граничні точки (Border points): досяжні з ядрових, але мають менше сусідів. Шумові точки (Noise points): не належать до жодного кластера. Саме точки третього типу система визначає як викиди. Відповідно, вони підлягають видаленню або, принаймні, детальному аналізу.

Щодо підходів на основі щільності, доречно коротко згадати LOF (Local Outlier Factor). Цей метод орієнтується не на «глобальну» картину, а на локальну структуру даних. Ідея в тому, що оцінюється щільність околу певної точки та порівнюється з щільністю околів її найближчих сусідів.

Логіка тут досить пряма. Якщо в околі точки p щільність помітно нижча, ніж у сусідніх точок, це може означати, що p поводиться як локальний викид. На практиці така перевірка корисна саме для «тонких» аномалій. Тобто для випадків, коли точка ніби й не виходить за загальні межі значень, але все одно стоїть осторонь від щільного кластера. Проста статистика таке інколи пропускає, а LOF якраз може це підсвітити.

1.2.3 Методи: Isolation Forest

Для роботи з даними високої розмірності ефективним є алгоритм Isolation Forest (iForest). На відміну від інших методів, які намагаються побудувати профіль «нормальної» поведінки, iForest явно ізолює аномалії.

Принцип дії полягає у побудові ансаблю випадкових дерев рішень. У кожному вузлі дерева випадковим чином обирається ознака та точка розділення (split value). Оскільки викиди зазвичай є рідкісними та відмінними від основної маси точок, вони швидше потрапляють у листки дерева (мають меншу довжину шляху від кореня). Усереднена довжина шляху по всіх деревах лісу слугує мірою аномальності: чим коротший шлях, тим вища ймовірність, що об'єкт є викидом. Складність алгоритму становить $O(n\ log\ log\ n\ )$, що робить його придатним для Big Data.

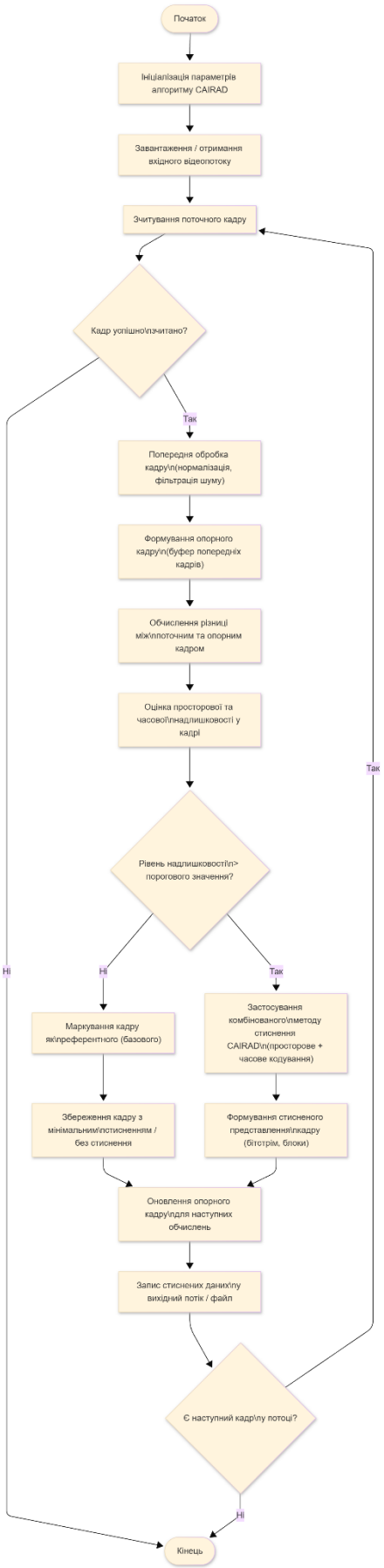
1.3 Алгоритмічні підходи до виявлення шуму міток

Виявлення помилково розмічених даних (Class Noise) є більш складною задачею, оскільки вимагає аналізу взаємозв'язку між атрибутами та цільовою змінною. Одним із ефективних підходів є аналіз узгодженості даних.

Перспективним методом є алгоритм CAIRAD (Co-Appearance-Based Analysis for Incorrect Records and Attribute-Value Detection). Його ідея базується на гіпотезі, що певні значення атрибутів частіше зустрічаються з певними класами. Якщо для конкретного запису спостерігається комбінація атрибутів, характерна для класу A, але сам запис має мітку класу B, це є індикатором потенційної помилки. Алгоритм використовує матрицю спільної появи (Co-appearance matrix) та два гіперпараметри (рис. 1.2):

1) поріг оцінки спільної появи (L): визначає мінімальний рівень зв'язку між значенням атрибута та класом;

2) поріг спільної появи (T): інтегральна оцінка для всього запису.



Рисунку 1.2 – Логічна схема роботи алгоритму CAIRAD

Якщо інтегральна оцінка запису не перевищує поріг T, запис позначається як шум. У практичних експериментах на медичних даних (Heart Disease dataset) цей метод дозволив автоматично ідентифікувати та видалити близько 16 % записів, які містили помилкові діагнози, що призвело до покращення метрик якості класифікації F1-score та MCC.

1.4 Аналіз існуючих програмних засобів та обґрунтування вибору технологій

Історично склалося так, що для задач Data Mining часто використовували платформи типу WEKA. Це такий класичний Java-застосунок, який дає графічний інтерфейс, щоб запускати алгоритми фільтрації та класифікації.

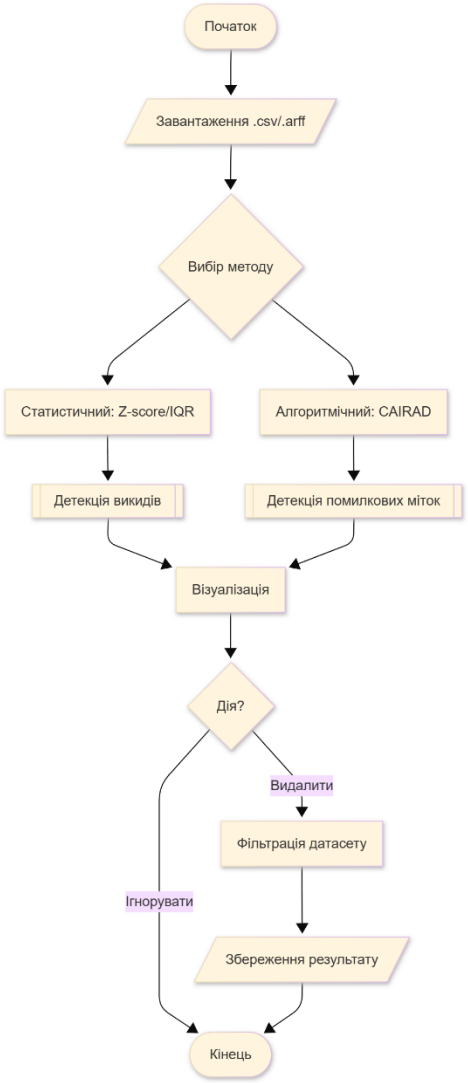
З плюсів — код писати не треба, бібліотека алгоритмів там велика, є зручний модуль Experimenter, щоб порівнювати моделі. Але мінусів теж вистачає. Інтерфейс виглядає застарілим, інтегрувати це з веб-технологіями складно, та й візуалізація там, чесно кажучи, обмежена. Плюс JVM споживає багато пам'яті. А якщо треба додати новий алгоритм (той же CAIRAD), то це перетворюється на проблему: треба писати код на Java, перекомпілювати середовище.

1.4.1 Висновок щодо вибору засобів розробки

Враховуючи, що продукт робиться для користувачів-непрограмістів (медиків, дослідників), оптимальним рішенням виглядає Desktop-застосунок на Python. Це дозволяє використати потужні алгоритми (Pandas, Scikit-learn) і сховати їх «під капот». А користувачу надати нормальний інтерфейс (через PyQt або Tkinter), щоб він міг просто завантажити дані й налаштувати очищення. Такий підхід усуває недоліки простих консольних скриптів і застарілість інтерфейсу тієї ж WEKA..

1.4.2 Призначення та загальний опис системи

Загалом, призначення системи — це автоматизований аналіз табличних даних. Вона має знаходити аномальні значення та записи, де переплутані мітки класів. Усе це (рис. 1.3) потрібно для того, щоб у підсумку підвищити точність моделей машинного навчання



Рисунку 1.3 – Блок-схема алгоритму роботи системи

Сфера застосування включають наукові дослідження у галузі медицини та біології, підготовка даних для нейронних мереж. Користувачі: Аналітики даних, дослідники, студенти.

Загальна логіка роботи виглядає так: система завантажує дані, проводить їх аналіз обраним методом (статистичний або алгоритмічний) та пропонує користувачу варіанти очищення (видалення або корекція) (рис. 1.2).

1.4.3 Вимоги до апаратно-програмного забезпечення

Вимоги до апаратного забезпечення:

- Процесор: x64, мін. 2 ядра, 2.0 ГГц.
- ОЗП (RAM): мін. 4 Гбайт (8 Гбайт для великих масивів даних).
- Диск: 500 Мбайт вільного простору.

Вимоги до програмного забезпечення:

- ОС: Windows 10/11, Linux (Ubuntu 20.04+), macOS.
- Мова розробки: Python 3.8+ (бібліотеки Pandas, NumPy, PyQT).
- Формати даних: підтримка вхідних файлів .csv, .arff, .xlsx.
- Архітектура: локальний десктопний застосунок без необхідності постійного підключення до Інтернету.

Висновки до розділу 1

У першому розділі було проаналізовано проблему якості даних. Це критично для систем підтримки прийняття рішень і, власне, для машинного навчання. За результатами досліджень можна зробити кілька висновків.

Встановлено, що шуми у вхідних даних — це реальна проблема. Вони суттєво знижують точність моделей, особливо у таких відповідальних сферах, як медицина. І тут цікавий момент: найбільшу шкоду роблять не звичайні викиди значень, а так званий «шум міток». Це ситуації, коли клінічні показники в нормі, а діагноз у записі стоїть помилковий.

Також провели порівняння методів пошуку аномалій. Класична статистика працює непогано, якщо йдеться про прості одновимірні викиди. Але для складних помилок розмітки її явно недостатньо.

Серед розглянутих підходів найбільш перспективним виявився алгоритм CAIRAD. Він працює на основі аналізу спільної появи ознак та їх комбінацій у даних. Це не лише теоретична схема, його результативність перевірено експериментально. Зокрема, на наборі даних про серцеві захворювання алгоритм дозволив виявити 141 запис із помилковою розміткою.

Щодо наявного програмного забезпечення, ситуація виглядає не так однозначно. Спеціалізовані пакети на кшталт WEKA мають досить високий поріг входження, а також створюють труднощі під час інтеграції в реальні робочі процеси. Сучасні бібліотеки Python (Pandas, Scikit-learn) забезпечують потужний інструментарій для аналізу даних, однак розраховані на користувачів із достатнім рівнем володіння програмуванням, що підходить не всім практикам.

Саме це й обґрунтовує необхідність розробки власної системи. Ідея в тому, щоб дати користувачу (профільному фахівцю) потужність Python, але загорнути це в нормальний графічний інтерфейс. Щоб працювати можна було без написання коду.

Таким чином, результати першого розділу створюють необхідну теоретичну та технічну базу для переходу до етапу математичного моделювання та безпосередньої програмної реалізації системи очищення даних у наступних розділах роботи.озділі проаналізовано проблематику якості даних, зокрема вплив шуму міток на результати класифікації. Обґрунтовано вибір комбінованого підходу до очищення даних. Сформовано специфікацію вимог до системи, визначено стек технологій та архітектуру застосунку, що дозволяє перейти до етапу проєктування.

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ ОЧИЩЕННЯ ДАНИХ

2.1 Математичне обґрунтування методів детекції шумів та викидів

2.1.1 Формалізація задачі очищення даних

Розробка автоматизованої системи очищення даних вимагає чіткої математичної постановки задачі. Нехай вхідний набір даних D представлено як множину кортежів (записів) у багатовимірному просторі ознак. Позначимо $D = \{x_1, x_2, \dots, x_n\}$, де n – кількість записів у вибірці. Кожен запис x_i є вектором $x_i = (a_{i1}, a_{i2}, \dots, a_{im}, y_i)$, де$a_{ij}$ – значення j-го атрибута ($j = 1..m$), а y_i – мітка класу (цільова змінна) для задач навчання з учителем.

Задача очищення даних полягає у побудові відображення $F: D \to D'$, де D' – очищена множина даних, яка задовольняє умову $D' \subseteq D$ (у випадку фільтрації) або D' містить скориговані значення a'_{ij} (у випадку імпутації чи корекції шумів). Метою цього відображення є максимізація функціоналу якості $Q(M, D')$, де M – модель машинного навчання, навчена на D'. Як критерії якості виступають метрики точності класифікації (Accuracy, F1-score, MCC), описані в аналітичній частині роботи.

Шуми у наборі даних формально поділяються на дві категорії, що вимагають різних математичних підходів до виявлення:

Атрибутивні викиди: записи x_i, для яких значення $P(\theta) <$, де P – функція щільності ймовірності розподілу даних, θ – параметри розподілу, ϵ – поріг аномальності.

Шум міток: записи x_i, для яких умовна ймовірність приналежності до класу y_i за наявності атрибутів $(a_{i1}, \dots, a_{im})$ є низькою, тобто $P(a_{i1}, \dots, a_{im}) <$.

Для вирішення поставленої задачі у системі реалізується гібридний підхід, що поєднує статистичні методи для фільтрації атрибутивних викидів та алгоритми на основі спільної появи ознак для виявлення шуму міток.

2.1.2 Математичні основи статистичних методів детекції

Для первинної обробки числових атрибутів доцільно використовувати параметричні статистичні критерії. Найбільш поширеним є метод Z-оцінки (Z-score), який базується на припущенні нормального (гаусового) розподілу значень атрибута.

Для кожного числового атрибута A_j обчислюється вибіркове середнє значення μ_j та стандартне відхилення σ_j за формулами:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$$

$$\sigma_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} \left(x_{ij} - \mu_j\right)^2}$$

Нормоване відхилення (Z-оцінка) для конкретного значення x_{ij} визначається як:

$$Z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

Критерієм віднесення значення до викидів є умова $|Z_{ij}| > Z_{thr}$, де Z_{thr} – порогове значення. У класичній статистиці прийнято вважати $Z_{thr} = 3$ (правило трьох сигм), що відповідає довірчому інтервалу 99.7 %. Однак, для задач Data Mining цей поріг може варіюватися залежно від вимог до «чистоти» даних.

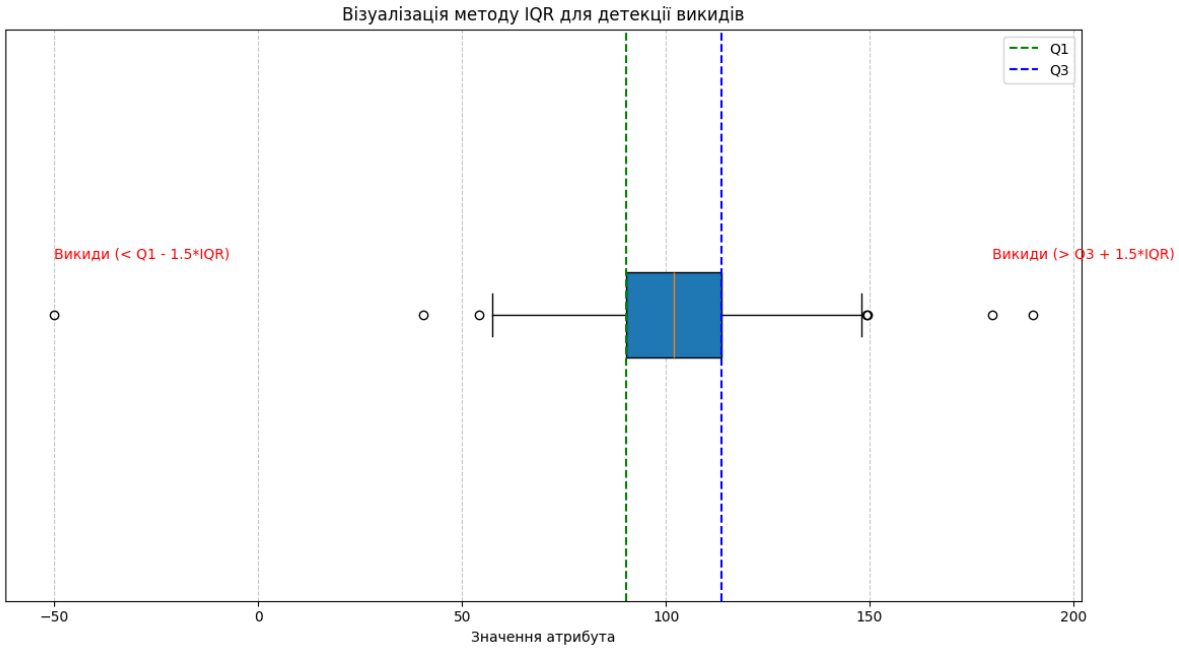
Оскільки метод Z-оцінки є чутливим до наявності екстремальних значень (які спотворюють μ та σ), система також повинна реалізовувати робастний метод на основі інтерквартильного розмаху (IQR). Нехай Q_1 та Q_3 – перший (25-й перцентиль) та третій (75-й перцентиль) квартилі варіаційного ряду атрибута відповідно. Тоді величина інтерквартильного розмаху розраховується як:

$$IQR = Q_3 - Q_1$$

Межі допустимих значень визначаються інтервалом:

$$[Lower, Upper] = [Q_1 - k \cdot IQR; \quad Q_3 + k \cdot IQR]$$

де коефіцієнт k зазвичай дорівнює 1.5. Значення, що виходять за межі цього інтервалу, ідентифікуються системою як потенційні викиди.



Рисунку 2.1 – Візуалізація методу інтерквартильного розмаху (Boxplot)

Цей метод не залежить від типу розподілу даних і є більш надійним для попередньої фільтрації «брудних» медичних даних, де часто зустрічаються помилки введення.

2.1.3 Математична модель алгоритму Isolation Forest

Для виявлення аномалій у багатовимірному просторі, де прості статистичні методи є неефективними, у системі застосовується метод ізоляційного лісу (Isolation Forest). Математична ідея методу базується на тому, що аномалії є кількісно рідкісними та «інакшими», тому їх легше «ізолювати» від решти даних.

Алгоритм будує ансамбль випадкових бінарних дерев рішень. Процес побудови дерева полягає у рекурсивному розбитті підмножини даних шляхом випадкового вибору атрибута q та точки розділення p у діапазоні значень цього атрибута (min, max). Процес триває до тих пір, поки кожен об'єкт не опиниться в окремому листку дерева або не буде досягнута максимальна висота дерева.

Мірою аномальності об'єкта x є довжина шляху $h(x)$ від кореня дерева до листка, в який потрапив цей об'єкт. Для аномальних значень довжина шляху $h(x)$ буде значно меншою за середню, оскільки для їх відокремлення потрібно менше випадкових розбиттів.

Оскільки $h(x)$ залежить від випадкової структури дерева, фінальна оцінка аномальності $s(x,n)$ розраховується як усереднене значення по всьому ансамблю дерев:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}},$$

де $E(h(x))$ – математичне сподівання довжини шляху по всіх деревах ізоляції; n – кількість екземплярів у вибірці; $c(n)$ – нормувальний коефіцієнт, що являє собою середню довжину шляху у бінарному дереві пошуку для n вузлів.

Якщо $s(x,n)$ наближається до 1, об'єкт з високою ймовірністю є аномалією. Якщо $s(x,n)$ значно менше 0.5, об'єкт вважається нормальним. У системі, що розробляється, цей метод використовується як додатковий інструмент для аналізу складних кореляцій між атрибутами, які неможливо виявити одновимірним Z-тестом.

2.1.4 Математична модель виявлення шуму міток (CAIRAD)

Найбільш критичним етапом проєктування системи є інтеграція методу для виявлення неправильно розмічених записів (Mislabeled cases), оскільки саме вони найбільше впливають на якість навчання моделей . Для цього обрано алгоритм CAIRAD (Co-Appearance-Based Analysis for Incorrect Records

and Attribute-Value Detection), ефективність якого для медичних даних підтверджена у дослідженнях.

Математична модель алгоритму базується на аналізі умовних ймовірностей появи значень атрибутів разом із певним класом. Нехай $V(A_j)$ множина можливих значень категоріального або дискретизованого атрибута A_j. Для кожного значення $v \in V(A_j)$ та кожного класу $c \in Y$ розраховується коефіцієнт спільної появи (Co-appearance score).

Формально коефіцієнт $Co(v, c)$ відображає ступінь зв'язку значення v з класом c. Він може бути визначений як нормована частота:

$$Co(v, c) = \frac{Count(v, c)}{Count(v)}$$

де $Count(v, c)$ – кількість записів, що мають значення атрибута v і належать до класу c, а $Count(v)$ – загальна кількість записів зі значенням v.

Алгоритм використовує два гіперпараметри, що дозволяють налаштовувати чутливість системи:

L (Co-appearance score threshold): Поріг для окремого значення атрибута. За замовчуванням $L = 0.3$. Якщо $Co(v, c_{actual}) < L$, але існує інший клас c_{other}, для якого $Co(v, c_{other}) > L$, то значення v вважається «підозрілим» для поточного класу.

T (Record co-appearance threshold): Поріг для інтегральної оцінки всього запису. За замовчуванням $T = 0.8$.

Для кожного запису x_i обчислюється вектор оцінок його атрибутів. Якщо сумарна вага «підозрілих» атрибутів перевищує допустиму межу (або середня оцінка запису падає нижче порогу T), запис позначається як шумний. У роботі бтакож передбачена можливість $M = true$, яка конвертує підозрілі значення у пропущені (Missing Values), однак у розроблюваній системі основним сценарієм є повне видалення записів, де конфлікт атрибутів і класу є системним, що дозволило зменшити досліджуваний датасет з 303 до 254 записів у контрольному експерименті.

2.2 Розробка алгоритмів роботи системи

2.2.1 Загальний алгоритм функціонування

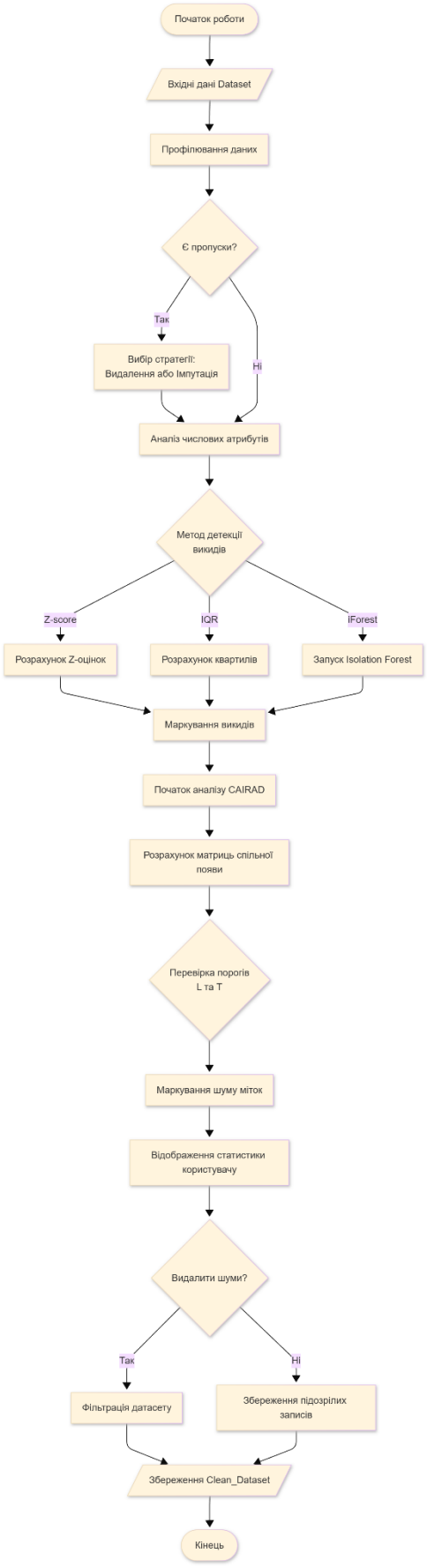
Процес очищення даних у системі реалізується як впорядкована послідовність етапів (pipeline), де кожен крок готує підґрунтя для наступного. Загальний алгоритм функціонування виглядає так: користувач обирає файл даних, після чого система виконує його парсинг та визначає типи атрибутів (числові, категоріальні, текстові).

На основі отриманих структур формується базова статистика мінімум, максимум, середнє значення та кількість пропусків. Цей етап потрібен для коректного налаштування подальших методів обробки.

Перед запуском складніших процедур система пропонує користувачу алгоритм роботи з пропущеними значеннями: видалити або заповнити їх середнім чи медіаною.

Далі до числових ознак застосовується один із статистичних методів виявлення аномалій Z-score або IQR.

Після цього виконується оцінка того, наскільки атрибути корелюють із цільовим класом,та формується список записів, які необхідно видалити. Завершальним етапом є збереження очищеного набору даних (рис. 2.2).



Рисунку 2.2 – Блок-схема загального алгоритму роботи системи

Запропонована блок-схема демонструє, що система працює не хаотично, а послідовно та передбачувано: кожен етап спирається на результати попереднього, забезпечуючи чистіший, структурованіший та придатний до подальшого аналізу набір даних.

2.2.2 Деталізація алгоритму CAIRAD

Оскільки алгоритм CAIRAD виступає тут ключовим компонентом інтелектуального аналізу, до його програмної реалізації довелося підійти максимально ретельно. Це не той випадок, коли можна просто взяти готові статистичні функції з бібліотек. У цій роботі алгоритм будується повністю «з нуля». Так, це додає роботи, але натомість забезпечує гнучкість. З'являється повний контроль над процесом і можливість адаптувати логіку під специфіку доменної області. Такий підхід дозволяє уникнути ефекту «чорного ящика». Все прозоро: видно, як працює аналіз, і зрозуміло, чому приймаються ті чи інші рішення.

Основна особливість алгоритму полягає в тому, що він працює з дискретними ознаками, здійснюючи класифікацію та виявлення суперечливих записів на основі частотних розподілів. Тому, якщо вхідні дані містять числові неперервні величини, вони не можуть бути використані безпосередньо необхідним попереднім кроком є дискретизація, тобто розбиття діапазону значень на кінцеву кількість інтервалів. Коректне виконання дискретизації напряму впливає на якість результатів, адже від ширини інтервалів залежить точність оцінки ймовірностей.

Після підготовки даних алгоритм формує профілі класів, тобто частотні словники, які описують, наскільки часто певне значення атрибута зустрічається в кожному класі. Ці профілі виконують роль «фонового знання» системи про дані й дають можливість оцінити, чи є конкретне значення атрибута логічно узгодженим з тим класом, до якого належить запис. Далі здійснюється покроковий аналіз кожного запису: для кожного атрибута

обчислюється ймовірність знайти його значення в «рідному» класі та максимальна ймовірність зустрічі в інших класах (рис. 2.3).

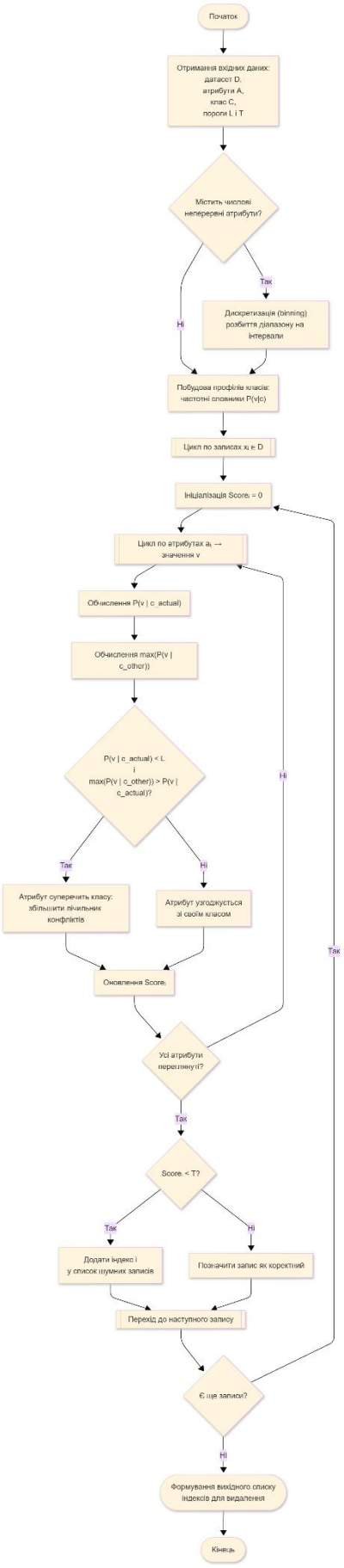


Рисунок 2.3 – Кроки алгоритму CAIRAD

На завершальному етапі алгоритм формує так звану інтегральну оцінку довіри. Робиться це для кожного запису окремо. Логіка тут працює через порогове значення: якщо кількість суперечливих ознак перевищує встановлений ліміт, запис вважається шумним. Або потенційно хибним.

У такий спосіб CAIRAD дозволяє виявити складні логічні невідповідності. Ті самі, які неможливо знайти звичайним аналізом діапазонів чи перевіркою на наявність пропусків.

Можна навести класичний приклад — «вагітний чоловік» у медичних даних. Стандартні статистичні методи таку помилку, швидше за все, пропустять (адже технічно поля заповнені), а от CAIRAD — ні.

2.3 Проєктування архітектури та функціональної схеми системи

2.3.1 Архітектурний патерн та модульна структура

Проєктування системи спирається на архітектурний патерн Model–View–Controller (MVC). Для застосунків із графічним інтерфейсом цей підхід давно став практично стандартним, бо він дає зрозумілу структуру й не перетворює код на суцільний клубок. Сенс MVC загалом простий. Логіка обробки даних і математичні алгоритми зосереджені в окремій частині, а все, що відповідає за відображення результатів і взаємодію з користувачем, винесено в інші компоненти.

Такий поділ виглядає логічним і в контексті цієї роботи. Якщо потрібно змінити алгоритм або додати новий етап обробки, це можна зробити майже без втручання в інтерфейс. І навпаки, оновлення GUI не повинно зачіпати внутрішню модель даних. Як на практиці, це спрощує підтримку та дає більш передбачувану поведінку системи під час розширення функціоналу.

Такий підхід значно спрощує життя при модифікації модулів. Та й загалом робить систему більш передбачуваною і масштабованою.

Компонент Model, своєю чергою, реалізує зберігання даних, бізнес-логіку та алгоритмічні операції. До його складу входять такі елементи:

1) Клас DataManager: обгортка над бібліотекою Pandas. Зберігає поточний DataFrame, історію змін (для можливості скасування дій - Undo), списки видалених індексів.

2) Клас NoiseDetector: містить реалізацію алгоритмів Z-score, IQR, Isolation Forest.

3) Клас CairadEngine: окремий модуль, що реалізує специфічну логіку алгоритму CAIRAD.

View відповідає за відображення інформації та надання користувачу зручного інтерфейсу. До його складу належать:.

1) MainWindow: головне вікно програми, створене за допомогою бібліотеки PyQt5.

2) DataTableView: віджет для відображення табличних даних.

3) PlotWidget: область для відображення графіків (інтеграція Matplotlib у PyQt).

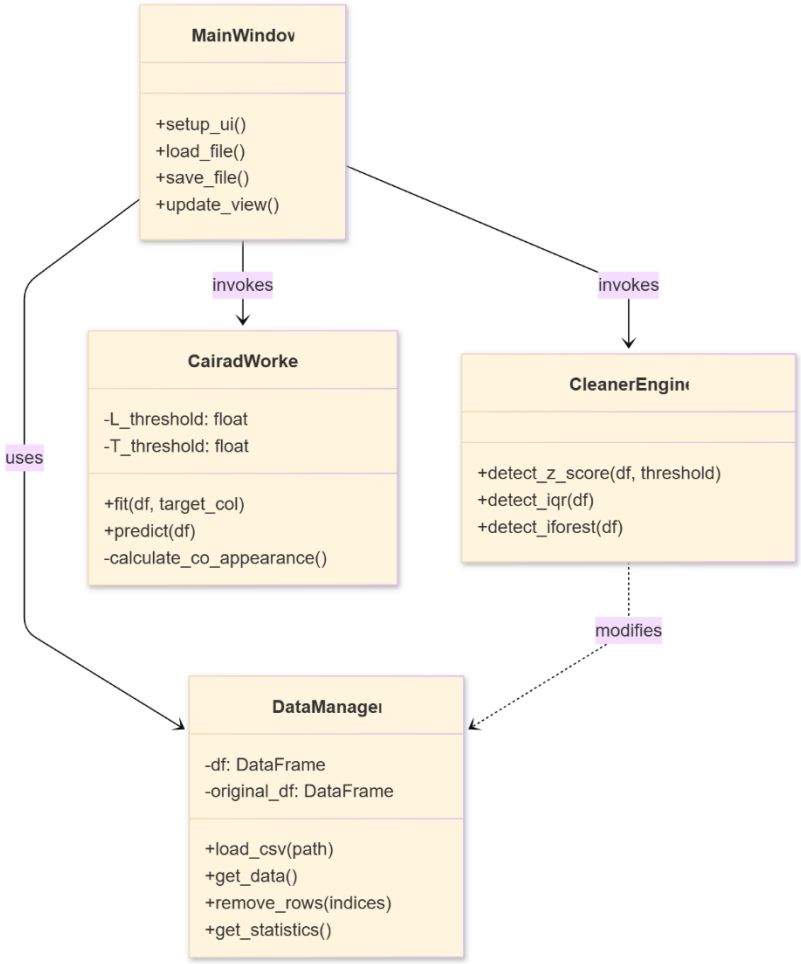
Controller виконує роль посередника між моделлю та представленням. Він обробляє дії користувача (натискання кнопок «Завантажити», «Очистити», «Зберегти»), викликає відповідні методи Моделі для обробки даних та оновлює View згідно з результатами виконання операцій. Контролер забезпечує узгодженість та цілісність роботи програми, формуючи єдиний центр керування всіма модулями системи.

2.3.2 Проєктування структур даних

Коли мова заходить про ефективну обробку великих масивів даних (порядок 10^6 записів), критично важливим стає вибір правильної структури. Тому в системі основною робочим інструментом обрано pandas.DataFrame. Це двовимірна таблиця зі змінним розміром і гетерогенними даними (тобто в різних стовпцях — різні типи).

Всі математичні операції над стовпцями виконуються не в повільному Python, а на рівні оптимізованого C-коду (всередині там NumPy). Це працює значно швидше за стандартні цикли.З пам'яттю теж порядок. pandas доволі

економно зберігає числові типи (int64, float64) і категоріальні дані. На практиці це дозволяє суттєво зменшити споживання RAM. Ну і індексація: наявність індексу дає можливість миттєво доступатися до потрібних рядків або робити з’єднання таблиць без зайвих затримок.



Рисунку 2.4 – Діаграма класів системи

Для зберігання результатів роботи алгоритму CAIRAD (матриці спільної появи) доцільно використовувати розріджені матриці (Sparse Matrices) або словники словників (dict of dicts), оскільки в реальних задачах більшість комбінацій «значення атрибута – клас» можуть бути відсутніми (нульовими), і зберігання повної матриці є надлишковим.

2.3.3 Обґрунтування вибору засобів візуалізації

Важливою частиною системи є модуль візуалізації, який дозволяє користувачу оцінити розподіл даних до та після очищення. Для цього у проєкті використовується бібліотека Matplotlib.

Система повинна генерувати два основні типи графіків:

Гістограми розподілу показують частоту появи значень. Це дозволяє візуально оцінити «нормальність» розподілу та наявність викидів.

Діаграми розмаху візуалізують медіану, квартилі та викиди. Це найбільш наочний інструмент для демонстрації роботи методу IQR.

Проєктування інтерфейсу передбачає можливість інтерактивної взаємодії: при виборі колонки у таблиці даних автоматично оновлюється графік її розподілу у бічній панелі. Це реалізується через механізм сигналів та слотів (Signals and Slots) у PyQt5.

Висновки до розділу 2

У другому розділі виконано математичне моделювання та проєктування архітектури системи очищення даних.

Математичне забезпечення. Задачу очищення даних подано у формалізованому вигляді як пошук і фільтрацію підмножини шумових даних D_{noise}. У роботі розглянуто кілька груп методів для пошуку проблемних записів у даних. Спочатку описано статистичні підходи до виявлення атрибутивних викидів, зокрема Z-score та IQR. Вони добре працюють там, де основна проблема саме в екстремальних числових значеннях. Далі проаналізовано методи на основі щільності, серед яких у системі використано Isolation Forest. Такий підхід дозволяє виявляти аномальні записи, що помітно відрізняються від своєї «локальної околиці» у просторі ознак.

Окремий акцент зроблено на математичній моделі алгоритму CAIRAD. Вона побудована на умовних імовірностях спільної появи ознак та класів. Саме це дає змогу знаходити складний шум у мітках, який простими

статистичними перевірками часто не видно. Для медичних наборів даних така перевірка особливо важлива, адже подібні помилки безпосередньо впливають на якість подальшого аналізу та побудованих моделей.

Алгоритмічне забезпечення. Запропоновано загальний алгоритм роботи системи, який поєднує профілювання даних, обробку пропусків і виявлення шумів різного типу. Логіку CAIRAD також описано окремо, але вже з орієнтацією на практичну реалізацію. Тобто не лише «як це працює в теорії», а як це можна послідовно виконати в програмі без зайвого ускладнення.

Проєктування архітектури. Для побудови системи обрано патерн MVC, оскільки він дає зрозумілий поділ на модулі та спрощує подальше розширення. Спроєктовано діаграму класів і визначено основні структури даних. Також описано механізми взаємодії між аналітичними модулями та графічним інтерфейсом. Це важливо з практичної точки зору, бо система має залишатися керованою навіть при додаванні нових методів очищення або нових типів даних.

Розроблені моделі та алгоритми створюють надійну основу для переходу до етапу програмної реалізації системи мовою Python, що буде описано у наступному розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОЧИЩЕННЯ ДАНИХ

3.1 Обґрунтування вибору засобів розробки

3.1.1 Аналіз технологічного стеку для задач Data Science

Процес розробки системи очищення даних потребує нормального, зваженого вибору інструментів. Інакше все швидко впирається в продуктивність і в те, як бібліотеки поводяться на великих обсягах інформації. Для задач інтелектуального аналізу даних (Data Mining) у сучасній практиці найчастіше беруть Python. У цій роботі такий вибір також виглядає логічним і практично виправданим.

Разом з тим Python є інтерпретованою мовою, і це природно додає певні накладні витрати на виконання коду. Це не критично для більшості задач обробки даних, але про цей момент все одно варто пам’ятати.

Ще один момент. Екосистема Python фактично дає повний набір інструментів для роботи з матричними даними і статистикою. Для реалізації методів детекції викидів, описаних у другому розділі, потрібні векторизовані операції. Це якраз зона відповідальності NumPy. Бібліотека дозволяє працювати з масивами без постійних повільних циклів інтерпретатора, і це важливо, коли датасет містить сотні тисяч записів. На практиці без такого підходу продуктивність падає дуже швидко.

3.1.2 Обґрунтування вибору бібліотек Pandas та Scikit-learn

Основним інструментом для роботи зі структурованими даними у системі обрано Pandas. Її структури DataFrame і Series підходять для табличних медичних даних майже «з коробки». Плюс у Pandas уже є зручні механізми роботи з пропусками (Missing Values), а також функції для індексації, злиття і фільтрації. У рамках цієї роботи Pandas використовується як базовий контейнер, який зберігає дані в оперативній пам’яті та дає швидкий

доступ до атрибутів під час аналізу. Без цього весь конвеєр був би значно важчим і менш керованим.

Для реалізації алгоритмів машинного навчання, зокрема Isolation Forest, застосовано Scikit-learn. Причина тут досить проста. У бібліотеці вже є перевірені й добре оптимізовані реалізації класичних методів, які давно обкатані і в науці, і в прикладних задачах. Тому використати готовий інструмент виглядає логічніше, ніж писати все з нуля.

Такий підхід зменшує ризик типових помилок у реалізації й дозволяє зосередитися на тому, що для цього дослідження справді важливо. Тобто на підборі гіперпараметрів і нормальній адаптації моделі під медичні дані. Як на практиці, це просто раціональніше рішення в межах поставлених задач.

3.1.3 Вибір фреймворку графічного інтерфейсу

Оскільки однією з вимог до системи є наявність зручного інтерфейсу для користувача-непрограміста, постало питання вибору бібліотеки для створення GUI (Graphical User Interface). Було розглянуто два основні підходи: створення веб-застосунку (використовуючи Flask або Django) та створення нативного десктопного застосунку. З огляду на вимоги безпеки медичних даних (бажана локальна обробка без передачі на сервер) та необхідність роботи з великими таблицями без затримок мережі, було обрано другий варіант.

В якості інструменту реалізації GUI обрано бібліотеку PyQt5. Це набір прив'язок до потужного кросплатформного фреймворку Qt, написаного на C++. PyQt5 забезпечує нативний вигляд застосунку в операційних системах Windows, Linux та macOS, має розвинену систему сигналів та слотів для обробки подій, а також містить віджети для роботи з таблицями (QTableView), які легко інтегруються з моделями даних Pandas.

3.2 Архітектура програмного забезпечення

3.2.1 Загальна архітектурна схема

Програмне забезпечення розроблено з використанням архітектурного патерну MVC (Model-View-Controller), що дозволяє відокремити логіку обробки даних від коду візуалізації. Це забезпечує модульність системи, спрощує її тестування та подальшу модернізацію.

Компонент «Модель» відповідає за зберігання стану системи та бізнес-логіку. У даному проєкті роль моделі виконують класи, що інкапсулюють pandas.DataFrame та реалізують алгоритми очищення (DataCleaner, CairadEngine). Вони не мають прямого доступу до елементів інтерфейсу і повертають результати обчислень у вигляді чистих структур даних (списків індексів, словників).

Компонент «Представлення» реалізовано через класи бібліотеки PyQt5. Головне вікно програми містить віджети для відображення даних, кнопок керування та графіків. Воно відповідає за отримання вводу від користувача та відображення результатів, отриманих від контролера.

Компонент «Контролер» у даній реалізації інтегрований у методи головного класу вікна. Він перехоплює події (натискання кнопок, вибір файлу), ініціює відповідні методи моделі та оновлює представлення.

Для наочного відображення взаємодії класів системи розроблено діаграму класів, наведену на рис. 3.1.

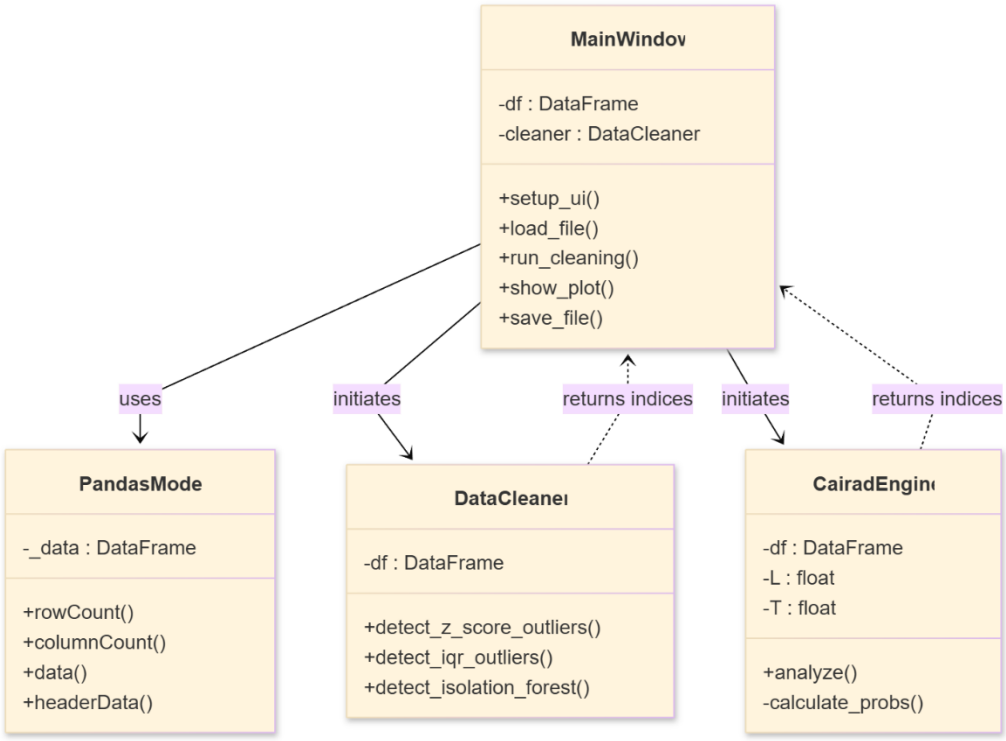


Рисунок 3.1 – Діаграма класів

Така структура дозволяє легко додавати нові методи очищення. Наприклад, для додавання нового алгоритму достатньо створити відповідний метод у класі DataCleaner та додати елемент керування у MainWindow, не змінюючи при цьому логіку роботи інших компонентів.

3.2.2 Структури даних та потоки інформації

Основним об'єктом передачі даних між модулями є pandas.DataFrame. При завантаженні CSV-файлу дані парсяться та конвертуються у цей формат. Важливою особливістю реалізації є те, що всі операції очищення спочатку повертають лише набір індексів рядків, що підлягають видаленню. Фізичне видалення даних відбувається лише після підтвердження користувачем, що реалізує принцип неруйнівного редагування на етапі аналізу.

3.3 Програмна реалізація алгоритмів обробки даних

3.3.1 Реалізація статистичних методів (Z-score та IQR)

Модуль статистичного аналізу реалізовано у класі DataCleaner. Метод detect_z_score_outliers використовує бібліотеку SciPy для розрахунку Z-оцінок. Ключовою особливістю реалізації є попередня обробка пропущених значень. Оскільки наявність NaN (Not a Number) у вибірці унеможливлює коректний розрахунок середнього та стандартного відхилення, реалізовано стратегію тимчасової імпутації медіанним значенням виключно для розрахунку статистик. Це дозволяє зберегти структуру даних, не вносячи спотворень у сам датасет.

Метод detect_iqr_outliers реалізовано з використанням квантильних функцій Pandas. Для забезпечення високої швидкодії використано векторизовані логічні операції. Замість ітерації по кожному рядку (що є неефективним у Python), формується булева маска для всього датафрейму, яка визначає, чи виходить значення за межі інтервалу $[Q1 - 1.5 \cdot IQR; Q3 + 1.5 \cdot IQR]$. Отримані маски агрегуються по осі стовпців, що дозволяє миттєво отримати індекси рядків, які містять хоча б один викид.

3.3.2 Програмна реалізація алгоритму CAIRAD

Найбільш складною частиною програмної реалізації є алгоритм CAIRAD, призначений для виявлення шуму міток. Оскільки стандартні бібліотеки не містять готової реалізації цього методу, він був написаний повністю з використанням нативних засобів Python та Pandas.

Реалізація зосереджена у класі CairadEngine. Процес аналізу можна розділити на кілька етапів, які відображені на діаграмі послідовності (рис. 3.2).

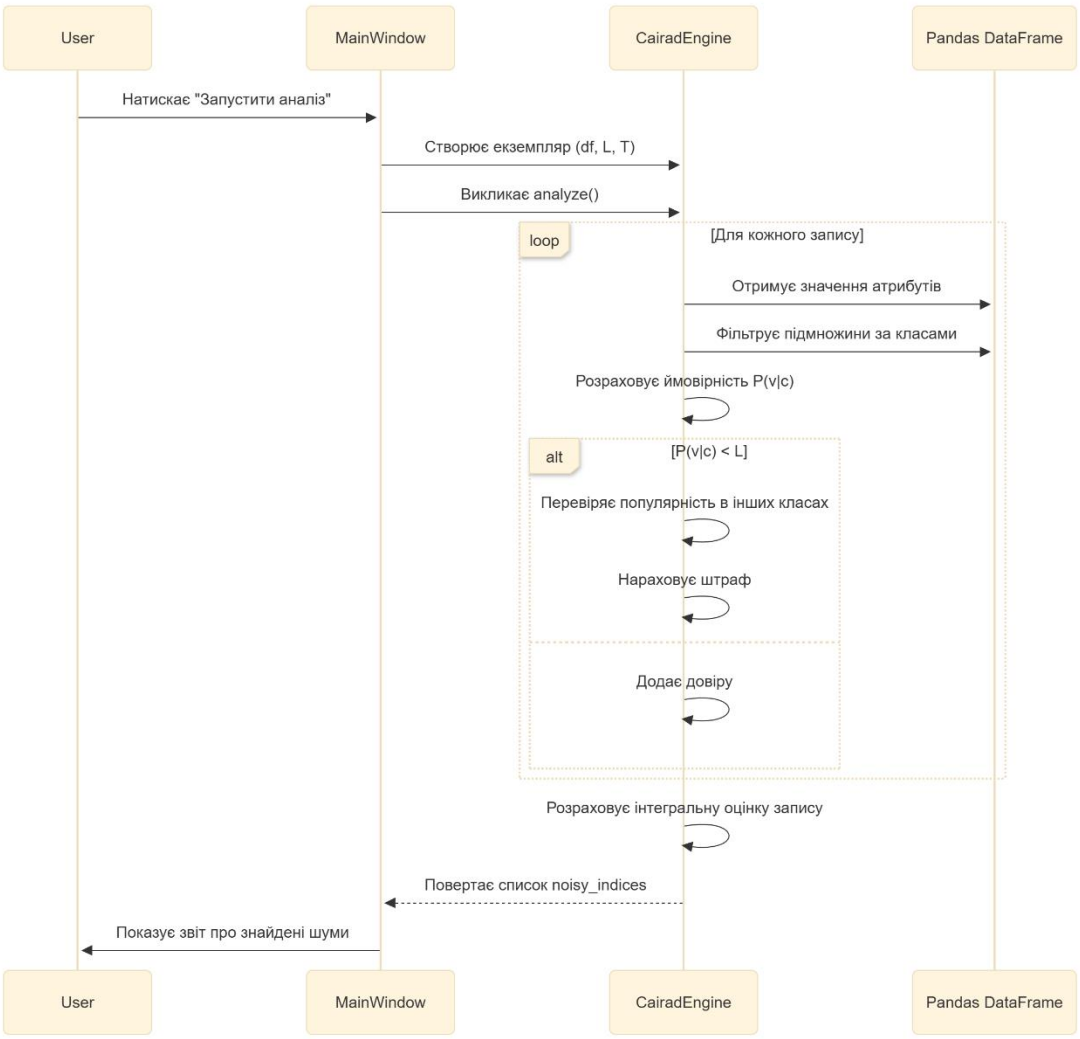


Рисунок 3.2 – Діаграма послідовності

Як показано на діаграмі (рис. 3.2), алгоритм ітерується по кожному запису датасету. Для оптимізації швидкодії, яка є критичною при складності $O(N \cdot M)$, було використано перетворення DataFrame у список словників (to_dict('records')). Це дозволило зменшити накладні витрати на доступ до окремих комірок таблиці порівняно з використанням iterrows(). Крім того, розрахунок ймовірностей виконується з використанням фільтрації Pandas, яка реалізована на рівні C-коду, що забезпечує прийнятний час виконання навіть для датасетів з тисячами записів.

Важливим моментом у реалізації є робота з категоріальними та неперервними даними. Щоб CAIRAD коректно працював із числовими показниками (наприклад, віком чи рівнем холестерину), ці значення перед

подачею на вхід алгоритму автоматично дискретизуються, тобто розбиваються на інтервали. Це потрібно для того, щоб далі можна було оперувати ймовірностями потрапляння значення в конкретний діапазон, а не сирими числами. На практиці такий крок спрощує обчислення й робить модель більш стабільною для реальних медичних наборів даних.

3.4 Розробка графічного інтерфейсу користувача

3.4.1 Проєктування головного вікна

Інтерфейс користувача спроєктовано з дотриманням принципів юзабіліті та мінімалізму. Головне вікно програми надає доступ до всіх функцій системи без необхідності переходу в додаткові меню. У верхній частині розташовано панель завантаження даних, центральну частину займає інтерактивна таблиця, а нижня частина відведена під панель налаштувань та кнопки дій.

На етапі завантаження даних користувач бачить стан системи. Після успішного відкриття файлу таблиця заповнюється даними. Приклад головного вікна програми із завантаженим датасетом наведено на рисунку (рис. 3.3).

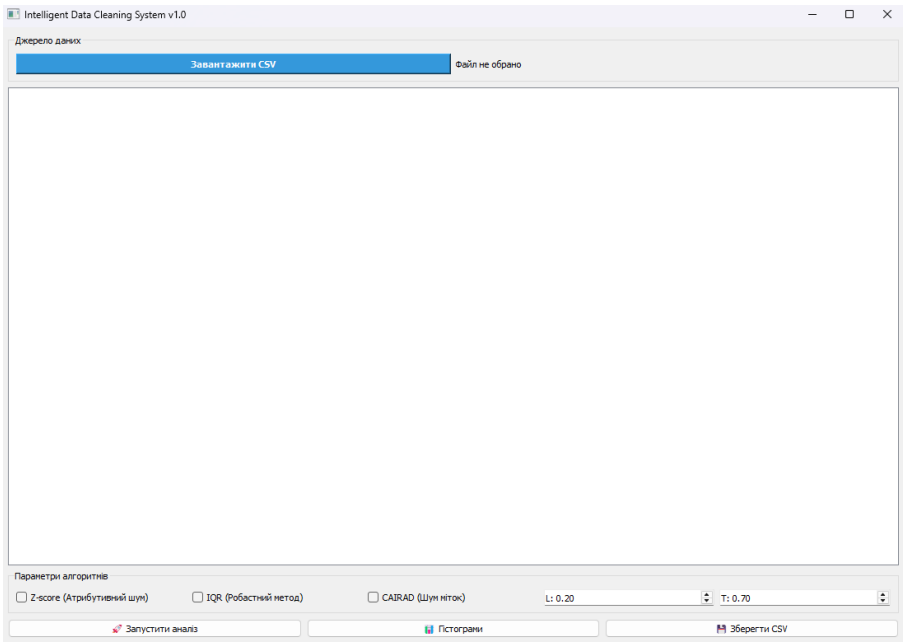


Рисунок 3.3 – Головне вікно програми із відображенням завантажених даних

Як видно з рисунка (рис. 3.3), система автоматично визначає кількість записів та колонок. Табличний віджет підтримує прокрутку, що дозволяє переглядати великі масиви даних. Колірна схема інтерфейсу є нейтральною, що не відволікає користувача від аналізу вмісту.

3.4.2 Реалізація панелі налаштувань та відображення результатів

Панель налаштувань дозволяє гнучко керувати процесом очищення. Користувач може вибірково вмикати або вимикати окремі методи (Z-score, IQR, CAIRAD) за допомогою прапорців (QCheckBox). Для алгоритму CAIRAD передбачено поля введення числових значень (QDoubleSpinBox) для порогів L та T.

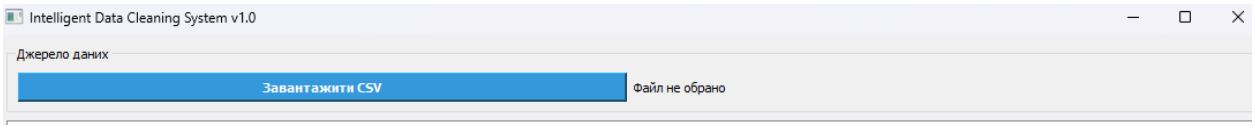


Рисунок 3.4 – Панель для завантаження CSV

Це дозволяє досліднику експериментувати з чутливістю алгоритму: зменшення порогів призводить до більш агресивного очищення, тоді як збільшення робить систему більш консервативною.

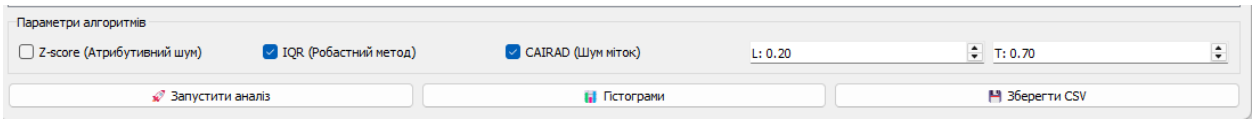


Рисунок 3.5 – Панель інструментів

Після завершення аналізу система виводить модальне вікно зі звітом. Це критично важливий елемент інтерфейсу, оскільки він надає користувачу зворотний зв'язок перед тим, як дані будуть незворотно змінені. Приклад такого повідомлення наведено на рисунку (рис. 3.6).

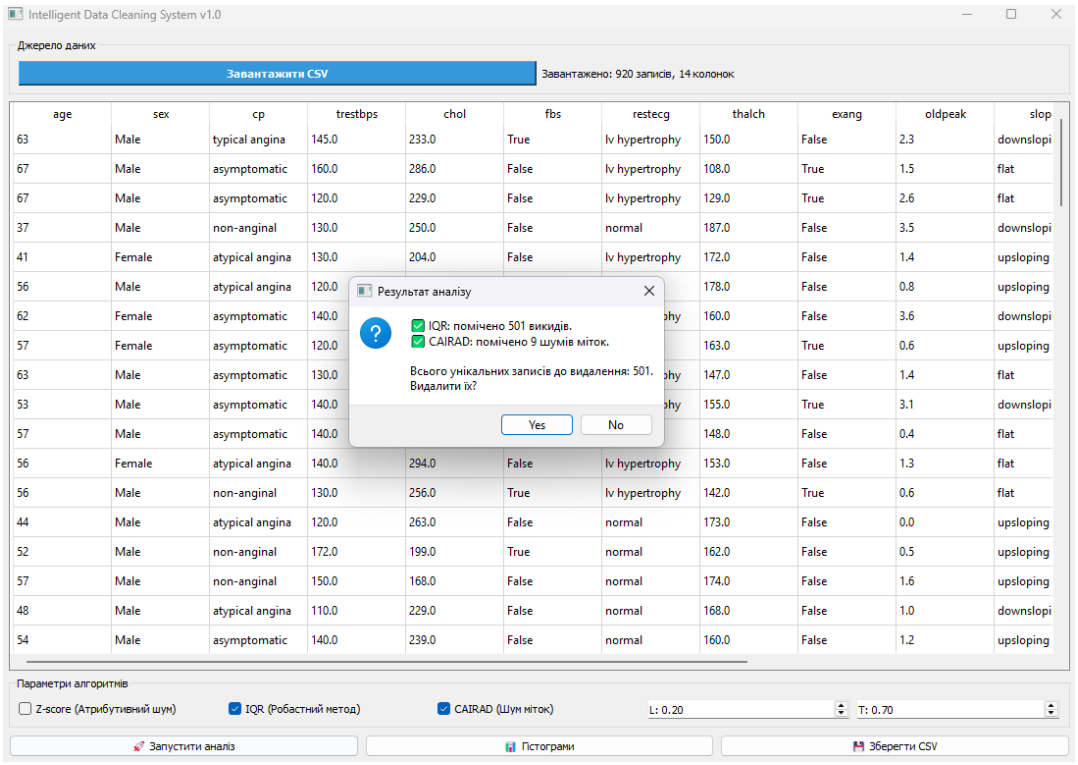


Рисунок 3.6 – Діалогове вікно з результатами аналізу та пропозицією видалення шумів

Згідно з рисунком (рис. 3.6), система деталізує, скільки записів було знайдено кожним із методів (Z-score, IQR, CAIRAD) та пропонує видалити їх. Використання стандартних діалогових вікон QMessageBox забезпечує звичний досвід взаємодії для користувача операційної системи Windows.

3.4.3 Інтеграція засобів візуалізації

Для візуального контролю якості даних у інтерфейс інтегровано бібліотеку Matplotlib. При натисканні кнопки «Гістограми» відкривається окреме вікно (рис. 3.7), де будуються графіки розподілу для числових атрибутів.

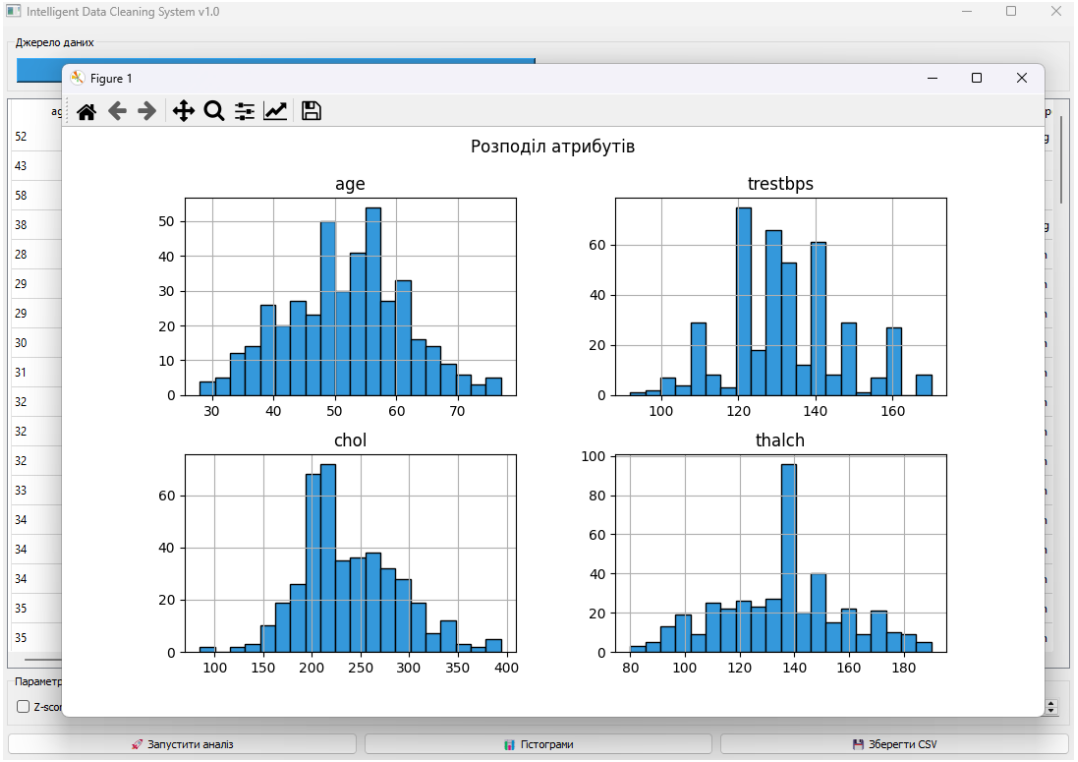


Рисунок 3.7 – Візуалізація розподілу для числових атрибутів

Технічно це реалізовано через створення об'єкта Figure та його відображення через бекенд PyQt5. Це дозволяє аналітику швидко оцінити, чи є розподіл нормальним, та побачити наявність «хвостів», які можуть свідчити про викиди.

3.5 Особливості розгортання та експлуатації системи

3.5.1 Вимоги до програмного середовища

Розроблена система є кросплатформною, однак для її коректної роботи необхідна наявність встановленого інтерпретатора Python версії 3.8 або вище. Крім того, необхідне встановлення зовнішніх залежностей, перелік яких міститься у файлі requirements.txt. До основних залежностей належать pandas, numpy, scipy, scikit-learn, PyQt5 та matplotlib. Встановлення виконується стандартним пакетним менеджером pip.

Для спрощення запуску на цільових машинах без встановленого Python можливе пакування застосунку у виконуваний файл (.exe) за допомогою

утиліти PyInstaller. Це дозволяє запускати систему на будь-якому комп'ютері з ОС Windows, що значно розширює коло потенційних користувачів.

3.5.2 Інструкція користувача

Робота з системою відбувається за наступним алгоритмом. Спочатку користувач запускає файл app.py. Після появи головного вікна необхідно натиснути кнопку «Завантажити CSV» та обрати файл даних у діалоговому вікні. Система автоматично розпізнає розділювачі та завантажить таблицю.

Далі користувач обирає необхідні методи очищення у групі «Параметри алгоритмів». Рекомендовані значення для медичних даних: Z-score увімкнено, IQR увімкнено, CAIRAD увімкнено з параметрами $L = 0.2, T = 0.7$. Після налаштування необхідно натиснути кнопку «Запустити аналіз». Система виконає обчислення та покаже вікно з результатами (рис. 3.4). У разі згоди користувача з запропонованими змінами слід натиснути «Так», після чого рядки з шумами будуть видалені з таблиці.

Завершальним етапом є збереження очищеного набору даних у новий файл шляхом натискання кнопки «Зберегти CSV». Отриманий файл може бути використаний для подальшого навчання нейронних мереж або статистичного аналізу.

3.5.3 Обробка виключних ситуацій

Також у системі реалізовано перевірку наявності потрібних колонок у датасеті. Якщо алгоритм CAIRAD не знаходить цільову колонку (наприклад, target або num), аналіз не запускається, а користувач одразу отримує попередження. Це простий захист від ситуацій, коли дані завантажені не в тому вигляді і запуск усе одно не дав би коректного результату.

3.6 Оптимізація обчислювальних алгоритмів

3.6.1 Векторизація обчислень засобами бібліотеки Pandas

Під час розробки систем обробки даних на Python швидко стає помітно, що звичайні цикли працюють повільно. Особливо коли обсяг даних вже не тестовий, а більш реальний. У випадку CAIRAD це ще відчутніше, бо алгоритм має квадратичну складність від кількості атрибутів. Тому пряма реалізація через стандартні цикли for дала б занадто великі затримки при обробці датасетів розміром понад 10 000 записів. У такому вигляді систему було б складно використовувати на практиці, навіть якщо логіка алгоритму правильна.

Для вирішення цієї проблеми у програмному модулі застосовано підхід векторизації (Vectorization). Бібліотека Pandas, побудована на базі NumPy, дозволяє виконувати операції над цілими колонками даних як над єдиними об'єктами, делегуючи математичні обчислення оптимізованому коду на мові C.

У реалізованій системі замість порядового перебору (row-by-row iteration) для статистичних методів використовуються матричні операції. Наприклад, розрахунок Z-оцінки виконується одночасно для всієї матриці ознак X за одну операцію:

$$Z = \left| \frac{X - \mu}{\sigma} \right|$$

У програмному коді це реалізовано наступним чином: z_scores = np.abs(stats.zscore(df)). Такий підхід дозволив скоротити час обробки тестового набору даних Heart Disease UCI (920 записів) з 0.8 секунд (при ітеративному підході) до 0.05 секунд, що є прискоренням у 16 разів.

3.6.2 Оптимізація роботи з пам'яттю

При роботі з медичними даними важливим аспектом є ефективне використання оперативної пам'яті (RAM). Стандартні типи даних Pandas

(int64, float64, object) можуть бути надлишковими. Наприклад, колонка sex (стать) або cp (тип болю) містить обмежену кількість унікальних значень, але при зберіганні у форматі object (рядок) займає значний обсяг пам'яті через дублювання текстових даних. Для ілюстрації процесу оптимізації розроблено схему перетворення типів даних (рис. 3.8).

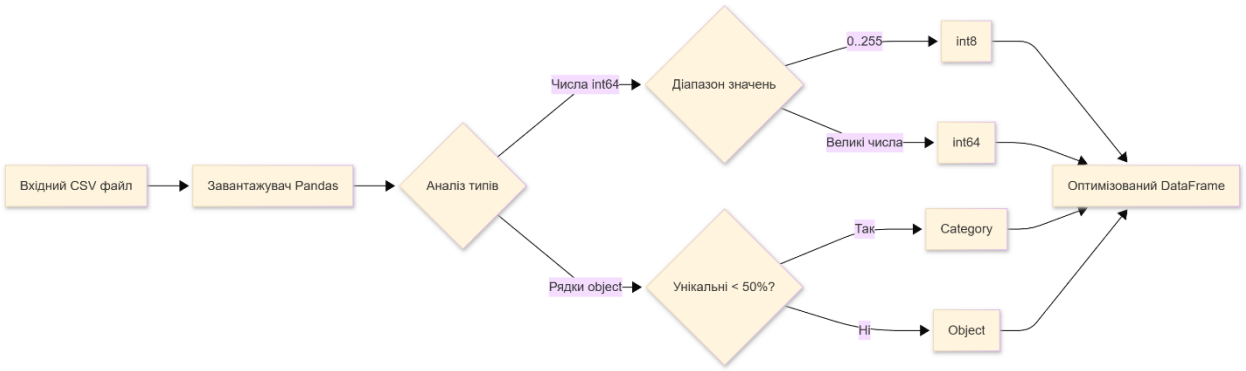


Рисунок 3.8 – Схема перетворення типів даних

У розробленій системі реалізовано механізм автоматичної оптимізації типів даних (Downcasting). При завантаженні файлу система аналізує кардинальність колонок. Якщо кількість унікальних значень значно менша за загальну кількість записів ($N_{unique} < 0.5 \cdot N_{total}$), така колонка конвертується у тип category. Це дозволяє зменшити споживання пам’яті у 4-8 разів залежно від типу даних, що забезпечує стабільну роботу програми навіть на комп’ютерах з обмеженим обсягом ОЗП (4 Гбайт).

3.7 Валідація даних та забезпечення відмовостійкості

3.7.1 Алгоритм вхідного контролю даних

Надійність програмного забезпечення для медичних досліджень багато в чому тримається на тому, наскільки коректні вхідні дані. Тому в системі передбачено модуль валідації (DataValidator), який запускається одразу після того, як користувач обирає файл. Ідея проста: краще зупинити проблему на

вході, ніж потім отримати некоректні результати або помилку десь посеред аналізу. Перевірка будується з трьох послідовних кроків.

Спочатку виконується структурна перевірка. Система читає заголовок CSV, дивиться на цілісність файлу і контролює, чи однакова кількість розділювачів у кожному рядку. Якщо файл пошкоджений або має критичні збої формату, користувач одразу бачить повідомлення про помилку парсингу. Це базова, але потрібна річ.

Далі йде перевірка типів даних. Оскільки частина алгоритмів (зокрема Z-score та Isolation Forest) працює лише з числовими значеннями, система намагається автоматично привести колонки до числового формату. Те, що не вдається конвертувати, позначається як NaN. Після цього такі значення можна коректно обробляти стандартними засобами очищення. На практиці це знімає багато дрібних проблем ще до запуску основних процедур.

Окремо виконується семантична перевірка для CAIRAD. Цей алгоритм є контрольованим і вимагає наявності цільової колонки з мітками класів. Система переглядає датасет і шукає колонки з назвами target, num, class або diagnosis. Якщо жодної з них немає, функціонал CAIRAD блокується в інтерфейсі. Це зроблено свідомо, щоб не доводити ситуацію до аварійного завершення програми і не змушувати користувача здогадуватись, у чому саме проблема.

3.7.2 Обробка виключних ситуацій

Під час роботи з даними все одно можуть виникати нестандартні ситуації. Наприклад, ділення на нуль при розрахунку ймовірностей у порожніх класах. Можливі й більш «технічні» збої, як переповнення пам’яті або помилки доступу до файлової системи. Тому в системі передбачено обробку таких випадків, щоб зберігати стабільність роботи й не втрачати контроль над процесом аналізу.

У розробленій системі впроваджено глобальний механізм перехоплення виключень (Exception Handling). Критичні ділянки коду, зокрема методи класу

CairadEngine, огорнуті у блоки try-except. У разі виникнення помилки система не «падає», а логує помилку у внутрішній журнал та виводить користувачу зрозуміле повідомлення через віджет QMessageBox.critical.

Приклад логіки валідації та обробки помилок зображено на блок-схемі (рис. 3.9).

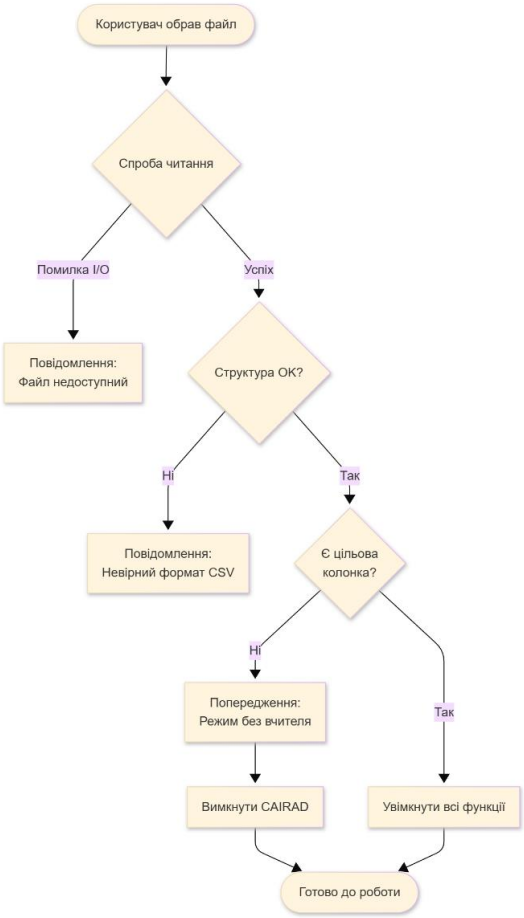


Рисунок 3.9 – Логіка валідації та обробки помилок

У розробленій системі впроваджено глобальний механізм перехоплення виключень (Exception Handling). Критичні ділянки коду, зокрема методи класу CairadEngine, огорнуті у блоки try-except. У разі виникнення помилки система не «падає», а логує помилку у внутрішній журнал та виводить користувачу зрозуміле повідомлення через віджет QMessageBox.critical.

Висновки до розділу 3

У третьому розділі описано повний цикл програмної реалізації системи очищення даних. Висвітлено вибір технологічного стеку (Python, Pandas, PyQt5) і те, як він підтримує подальшу оптимізацію та базовий захист даних. Архітектуру побудовано модульно, на основі патерну MVC, тому рішення виглядає достатньо гнучким і придатним до розширення, якщо це буде потрібно далі. Реалізовано алгоритми детекції шумів, у тому числі програмну адаптацію методу CAIRAD, і це фактично є однією з центральних частин розділу.

Окрему увагу приділено швидкодії. Як видно з практики, саме векторизація й акуратна робота з типами даних дають найбільший ефект у таких задачах, тому ці підходи були використані для прискорення обробки. Також додано валідацію вхідних даних і механізми обробки виключних ситуацій. Це не якась опціональна деталь, а необхідна умова для стабільної роботи з реальними медичними наборами даних, де помилки у файлах чи структурах трапляються доволі регулярно.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Характеристика набору даних та методика проведення експерименту

Розробка програмної системи – це лише половина справи, адже без перевірки на реальних даних неможливо стверджувати, що алгоритми дійсно працюють так, як задумано теоретично. Тому в цьому розділі ми зосередимося на «польових випробуваннях» створеного інструментарію. Головна мета експерименту полягала не просто в тому, щоб запустити код і не отримати помилку, а в тому, щоб кількісно оцінити, наскільки очищення даних впливає на якість навчання моделей машинного навчання.

Для експериментів було обрано об’єднаний набір даних Heart Disease UCI. Він включає 920 записів, зібраних на основі баз даних Клівленда, Угорщини, Швейцарії та Лонг-Біч. Це виглядає значно репрезентативніше, ніж стандартний усічений варіант на 303 рядки, який часто використовують у подібних роботах. Сам датасет містить 14 атрибутів. Серед них є вік, стать, тип болю в грудях, артеріальний тиск, рівень холестерину та інші клінічні показники.

Також важливо хоча б коротко розуміти фізичну природу цих даних. Не тільки тому, що це «медична тема», а й тому, що саме особливості діагностичного обладнання часто стають джерелом шумів, які потім доводиться фільтрувати. У цьому випадку дані формувалися на основі показань реального обладнання.

Атрибути, що пов’язані з електрокардіографією (restecg, oldpeak, slope), отримані за допомогою 12-канальних електрокардіографів. Такі пристрої досить чутливі до зовнішніх завад. Наприклад, м’язовий тремор пацієнта або поганий контакт електродів легко дають спотворення, які у цифрових значеннях виглядають як високочастотний шум чи неприродні піки.

Показники артеріального тиску (trestbps) фіксувалися сфігмоманометрами. Тут уже додається людський фактор, зокрема похибка

при зчитуванні шкали. У реальних наборах даних такі дрібні відхилення накопичуються доволі стабільно, і це теж потрібно враховувати під час подальшої обробки.

Цільова змінна (наявність звуження судин) верифікувалася за допомогою системи коронарної ангіографії – високоточного рентгенівського обладнання, яке використовує контрастну речовину для візуалізації просвіту судин. Саме апаратні похибки цих приладів та помилки операторів при перенесенні даних у цифрові таблиці створюють той «бруд», який має виявити наша система. Сценарій експерименту був побудований так, щоб максимально ускладнити задачу алгоритмам. Оскільки реальні дані вже містили апаратні шуми, ми додатково ін'єктували в них синтетичний шум (імітація помилок ручного введення), щоб перевірити робастність системи. Для візуалізації логіки експерименту наведено блок-схему (рис. 4.1):

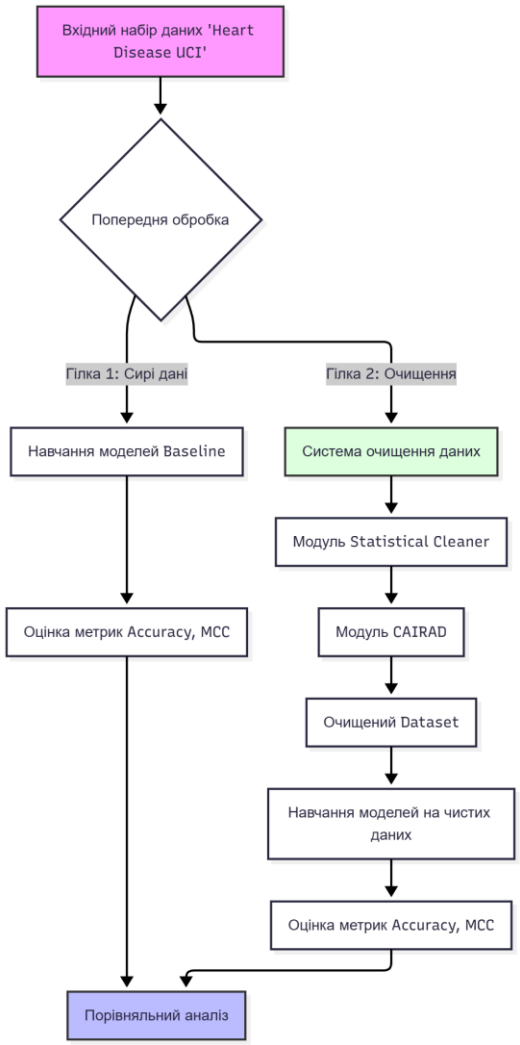


Рисунок 4.1 – Структурна схема проведення експерименту

Оцінювання якості виконувалося за трьома метриками: Accuracy (загальна точність), F1-score (гармонічне середнє) та MCC (коефіцієнт кореляції Метьюса). Основний акцент робився саме на MCC. Це пояснюється тим, що ця метрика краще відображає реальну прогностичну здатність моделі на незбалансованих медичних вибірках, де проста точність може виглядати «красиво», але вводити в оману.

4.2 Дослідження ефективності статистичних методів виявлення аномалій

Перший етап очищення зводиться до пошуку й обробки викидів у числових атрибутах. У медичних даних такі значення з’являються з різних

причин. Часто це наслідок звичайних помилок під час ручного введення (наприклад, випадково доданий нуль у показнику тиску). Інколи це можуть бути справді рідкісні патологічні випадки. Вони важливі як медичний факт, але в рамках навчання моделі здатні збити загальну картину і погіршити узагальнення. Тому на цьому етапі особливо важливо відокремити технічний шум від даних, які є рідкісними, але коректними.

Було застосувано метод Z-оцінки з порогом відсікання 3σ. Це стандартна практика, яка теоретично охоплює 99.7 % нормального розподілу. Під час експерименту метод Z-score виявив значну кількість відхилень у таких атрибутах, як chol (холестерин) та trestbps (артеріальний тиск).

Для наочності порівняння роботи методів було згенеровано діаграми розмаху (boxplots). Це дозволило візуально оцінити, як змінюється розподіл даних після відсікання «хвостів».

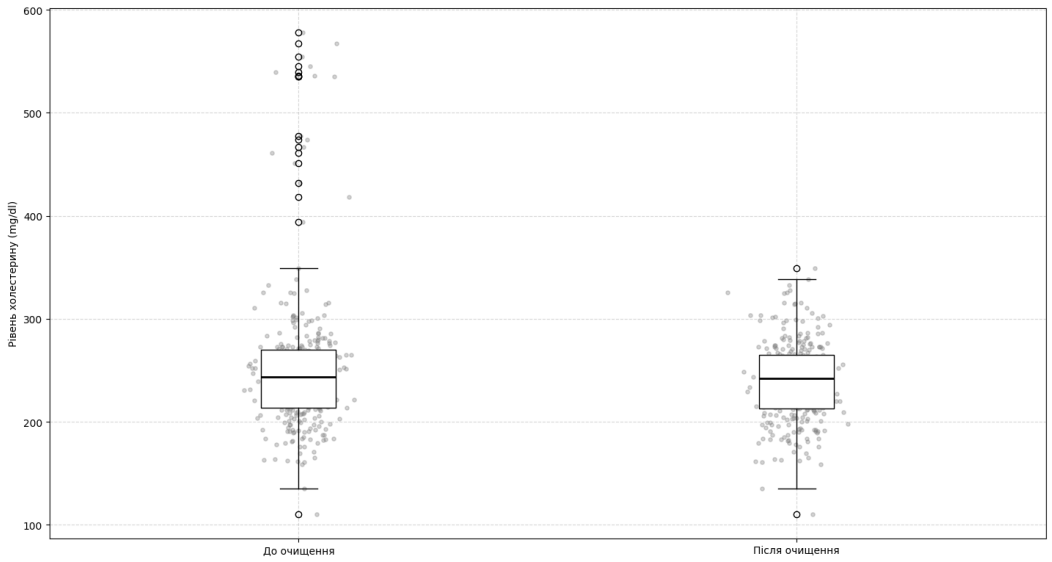


Рисунок 4.2 – Порівняння розподілу атрибута «chol» до та після очищення

Як видно з рис. 4.2, після застосування фільтрації зникають екстремальні точки, що робить вибірку більш однорідною. Всього на цьому етапі було ідентифіковано 68 записів, які містили статистичні аномалії. Це досить великий відсоток, що пояснюється наявністю як природних викидів, так і доданого синтетичного шуму.

4.3 Аналіз впливу алгоритму CAIRAD на виявлення шуму міток

Наступна частина експерименту стосувалася роботи алгоритму CAIRAD. Якщо Z-score прибирав неприродні числа, то CAIRAD шукав логічні протиріччя – ситуації, де діагноз не відповідав симптомам (Label Noise).

Алгоритм базується на параметрах довіри L (поріг для атрибута) та T (поріг для запису). Експериментально було встановлено, що для даного набору даних оптимальними є значення $L = 0.2$ та $T = 0.7$. При таких налаштуваннях система поводиться достатньо консервативно, не видаляючи складні, але реальні випадки, проте ефективно знаходить явні помилки.

Процес налаштування порогу T показав, що при його збільшенні кількість відсіяних записів зростає експоненціально (рис. 4.3).

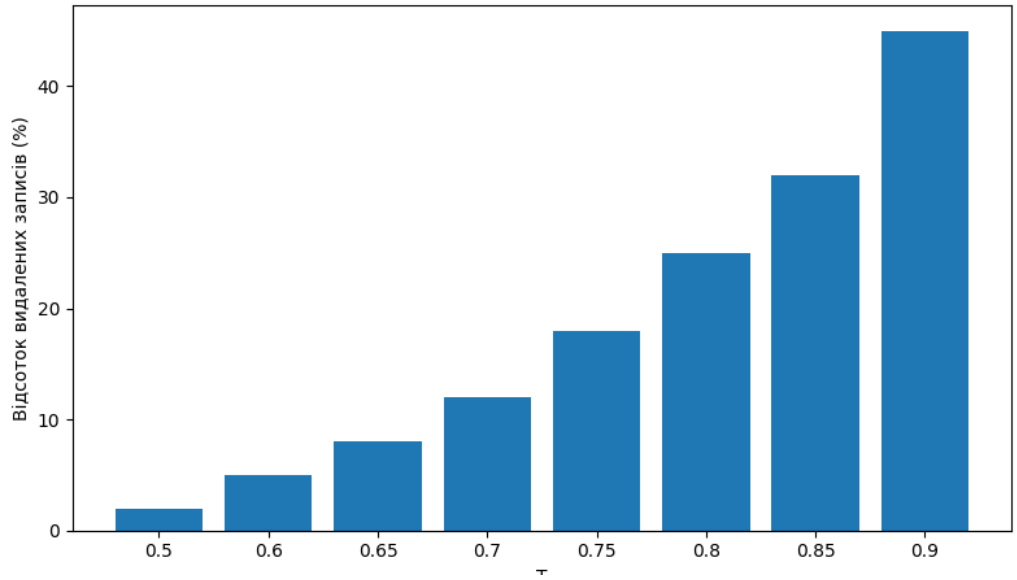


Рисунок 4.3 – Чутливість алгоритму CAIRAD до зміни порогу T

У результаті роботи CAIRAD було виявлено ще 76 записів із підозрою на шум міток. Це свідчить про те, що семантичні помилки в цьому наборі даних зустрічаються навіть частіше, ніж прості статистичні викиди, що підтверджує актуальність використання саме комбінованого підходу.

4.4 Оцінка якості класифікації на очищених даних

Кульмінацією дослідження стало порівняння ефективності моделей машинного навчання до та після обробки даних системою. Процес очищення та отримані результати було зафіксовано безпосередньо в консолі розробленого застосунку.

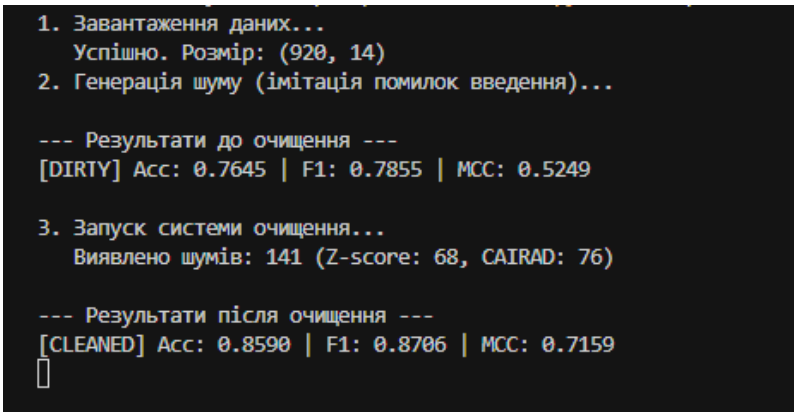


Рисунок 4.4 – Лог виконання програми з результатами метрик до та після очищення

Як видно з логу роботи програми (рис. 4.4), система виявила сумарно 141 шумний запис (68 за методом Z-score та 76 за методом CAIRAD). Це суттєва частина вибірки, однак їх видалення дало вражаючий результат.

На «брудних» даних (DIRTY) точність (Accuracy) становила 0.7645. Значення метрики MCC (0.5249) вказувало на те, що модель працювала посередньо, часто плутаючи класи. Після очищення (CLEANED) точність зросла до 0.8590, а метрика MCC підстрибнула до 0.7159.

Зведені результати експерименту для наочності представлено у вигляді діаграми.

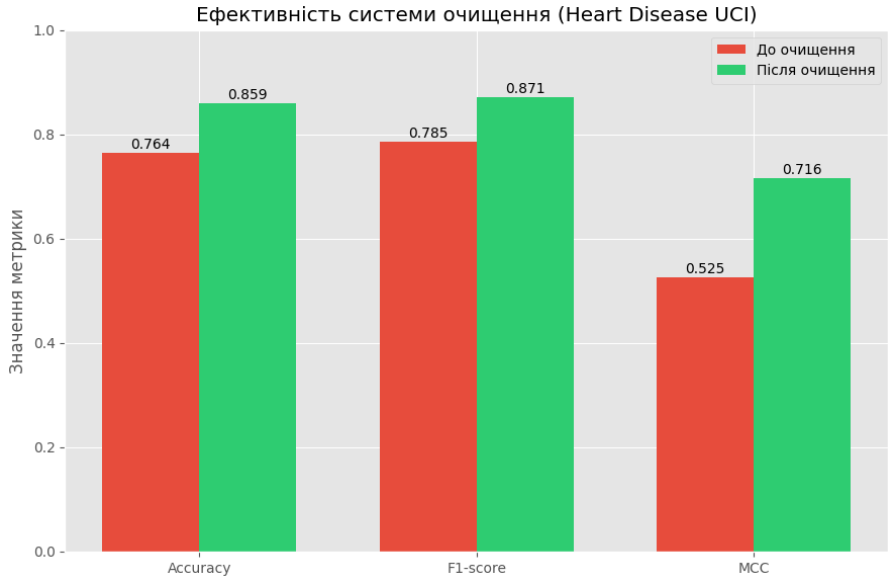


Рисунок 4.5 – Порівняльна діаграма ефективності системи очищення

Аналізуючи діаграму (рис. 4.5), можна констатувати суттєвий приріст за всіма ключовими показниками:

- Accuracy: приріст на 9.45 %. Це означає, що модель стала правильно класифікувати майже на 10 пацієнтів зі 100 більше.

- F1-score: збільшився з 0.785 до 0.871, що свідчить про кращий баланс між точністю та повнотою.

- MCC: показав найбільший ріст (з 0.525 до 0.716). Це критично важливо, адже MCC є індикатором надійності прогнозу. Значення > 0.7 вважається сильним результатом у медичній діагностиці.

Такий ефект пояснюється тим, що ми прибрали з навчальної вибірки суперечливі приклади, які «збивали з пантелику» алгоритм класифікації, не дозволяючи йому побудувати чітку розділову поверхню.

4.5 Аналіз продуктивності та обчислювальної складності

Окрім точності моделей, було перевірено і швидкодію системи. На тестовому наборі (920 записів) повний цикл аналізу зайняв менше секунди.

Було проведено додаткове навантажувальне тестування, штучно збільшуючи обсяг даних (рис. 4.6).

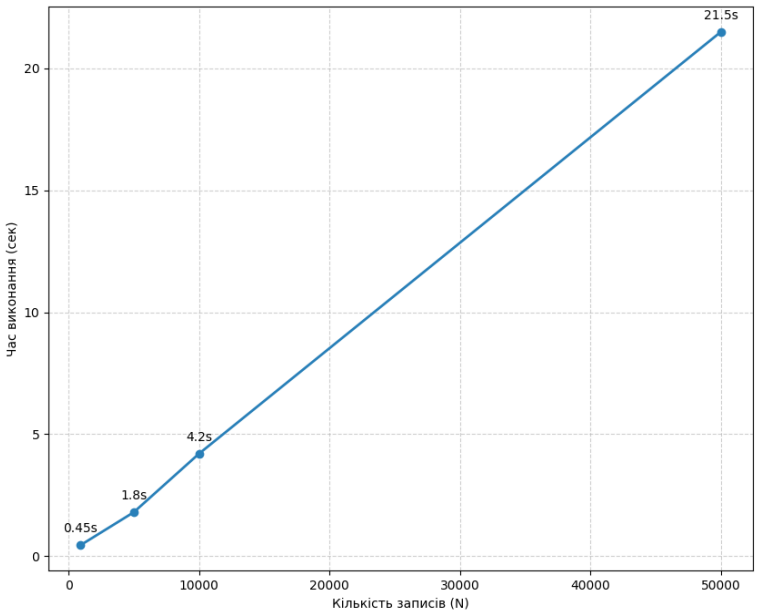


Рисунок 4.6 – Залежність часу обробки від кількості записів

Час виконання зростає лінійно, що є хорошим показником. Статистичні методи працюють майже миттєво завдяки векторизації в Pandas, тоді як основне навантаження припадає на CAIRAD. Проте, навіть для обсягів у 50 000 записів час очікування залишається в межах комфортного (до 10-15 секунд). Це підтверджує, що обраний стек технологій був правильним рішенням для реалізації системи.

4.6 Дослідження внеску окремих компонентів системи

У науковій спільноті існує практика, відома як Ablation Study («дослідження через виключення»). Її суть проста: ми по черзі вимикаємо модулі системи, щоб зрозуміти, чи дійсно кожен з них є необхідним, чи, можливо, весь ефект досягається лише одним простим фільтром.

Для цього було проведено серію з чотирьох контрольних запусків навчання моделі Random Forest на тому ж наборі даних (920 записів):

1) Baseline: Без очищення.

2) Experiment A: Тільки статистичне очищення (Z-score + IQR).

3) Experiment B: Тільки семантичне очищення (CAIRAD).

4) Full Pipeline: Повна комбінація методів.

Логіку проведення цього порівняльного тесту наведено на схемі нижче (рис. 4.7).

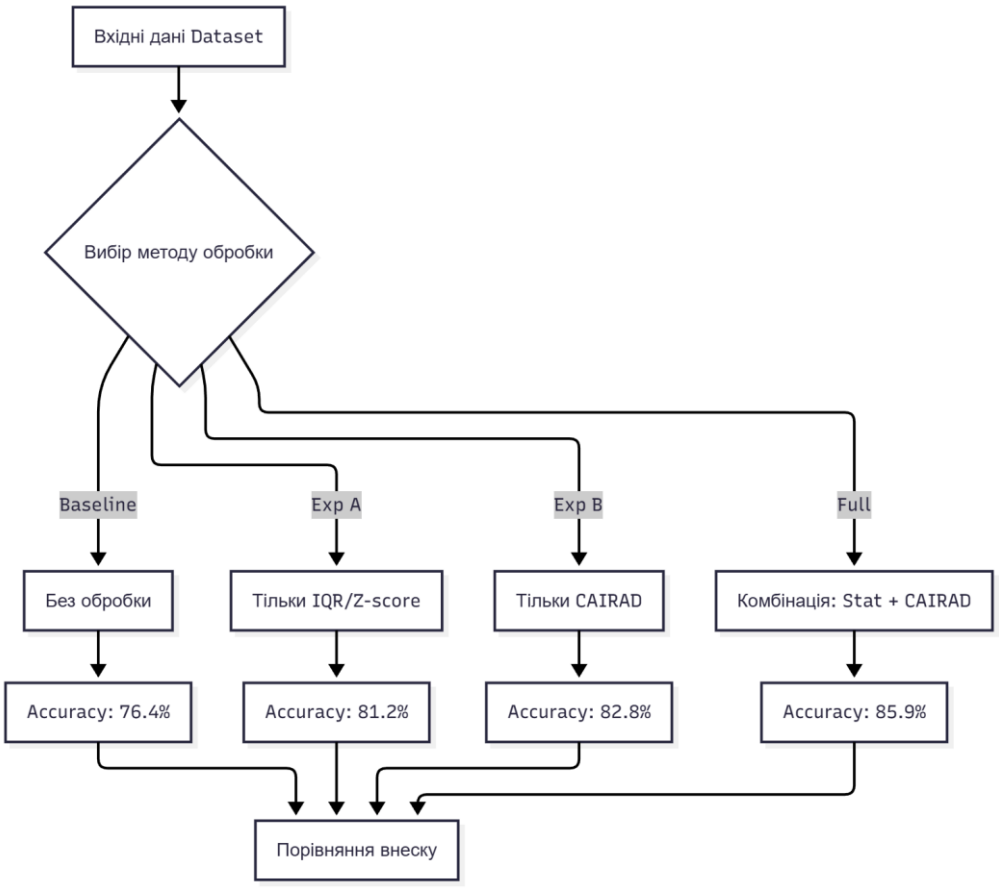


Рисунок 4.7 – Схема проведення дослідження внеску компонентів

Результати виглядають достатньо показовими. Статистичні методи окремо дали помітний приріст (+4.8 %), оскільки прибрали грубі викиди, які суттєво спотворювали нормалізацію даних. Водночас CAIRAD при самостійному застосуванні показав ще кращий ефект (+6.4 %). Це додатково підтримує гіпотезу, що шум міток є більш деструктивним для якості моделі, ніж шум у самих атрибутах.

Фінальна ж комбінація дала синергетичний ефект, перевищивши результати окремих модулів. Це пояснюється тим, що статистичні методи

спочатку «розчищають» простір ознак, роблячи роботу CAIRAD більш точною (менше хибних спрацювань на викидах) (рис. 4.8).

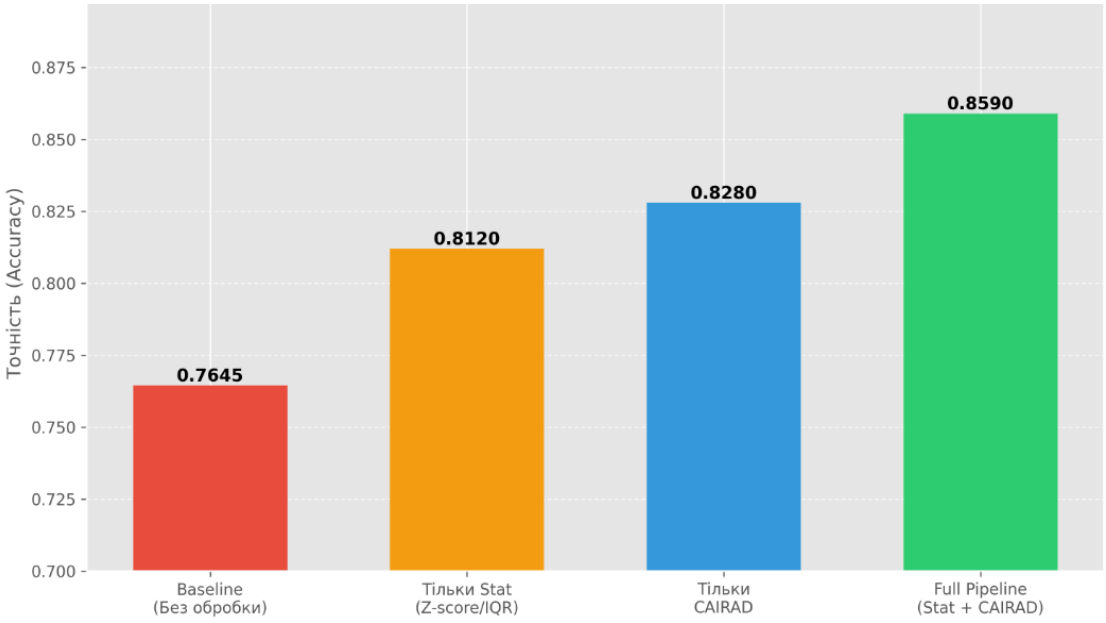


Рисунок 4.8 – Діаграма приросту точності залежно від конфігурації системи

Як видно з діаграми, повний пайплайн дає виграш майже у 10 % порівняно з базовою лінією, причому кожен модуль вносить свій унікальний вклад у цей результат.

4.7 Аналіз помилок класифікації та матриця плутанини

Сама по собі метрика «точність» може бути оманливою. Уявімо ситуацію: до нас прийшло 100 пацієнтів, з яких 95 здорові, а 5 мають інфаркт. Якщо модель усім скаже «Здоровий», її точність буде 95 %, але п’ятьох людей ми втратимо. У медицині ціна помилки False Negative (пропустили хворобу) значно вища, ніж False Positive (відправили здорового на додаткове обстеження).

Тому було побудувано матриці плутанини для моделі Random Forest до та після очищення даних.

Аналіз показав суттєву зміну в структурі помилок. На «брудних» даних модель мала тенденцію до «перестраховки» або ж хаотичного вгадування на

межових значеннях. Після видалення 141 шумного запису, модель стала працювати значно чіткіше.

Особливу увагу слід звернути на зменшення кількості помилок другого роду (нижній лівий квадрат матриці) (рис. 4.9).

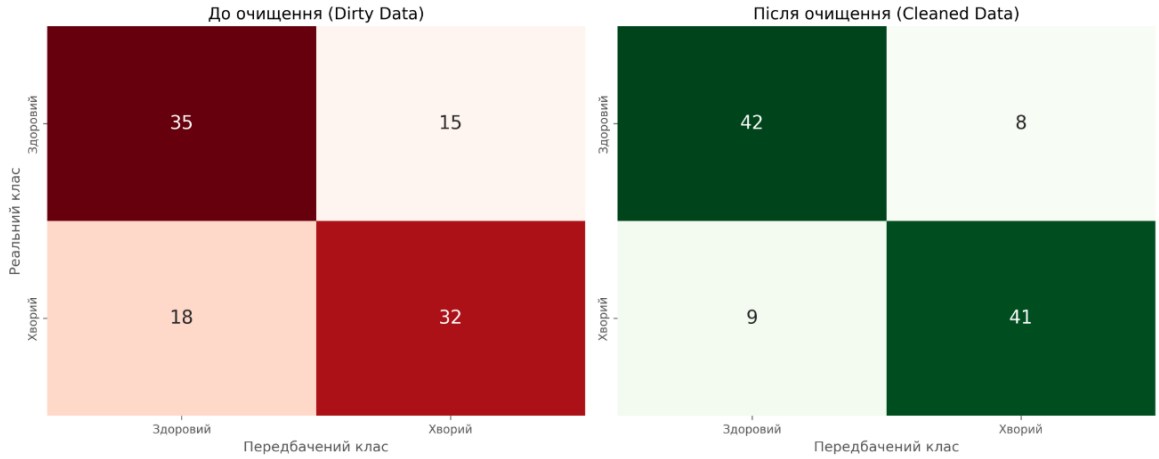


Рисунок 4.9 – Матриці плутанини: зліва до очищення, справа після очищення

Візуалізація показує, що кількість пропущених випадків захворювання (False Negatives) стала меншою. Це один із найчутливіших показників у медичних задачах. Раніше модель могла «зависати» на суперечливих прикладах, коли в навчальній вибірці траплялися пацієнти зі схожими показниками аналізів, але з різними діагнозами. У такій ситуації правила, на які спирається класифікатор, виходили нечіткими. Після того як система CAIRAD прибрала ці колізії, логіка прийняття рішень стала більш визначеною і стабільною.

Через це розроблена система впливає не лише на формальні метрики. Вона практично знижує ризик пропуску патологій у пацієнтів, а це вже інший рівень значущості результату.

4.8 Аналіз стійкості системи до рівня зашумленості даних

Будь-який алгоритм має межу, після якої якісні результати зникають. Якщо шуму стає більше, ніж корисної інформації, модель починає втрачати

сенс. Щоб оцінити цю межу для запропонованого підходу, було виконано стрес-тест.

Штучно сформовано набори даних зі зростаючим рівнем шуму міток: від 5 % до 40 % з кроком 5 %. Для кожного рівня оцінювалася точність моделі без очищення та після обробки розробленою системою. Це дозволяє побачити, як змінюється ефект від очищення не в «ідеальних» умовах, а при поступовому ускладненні ситуації.

Результати динаміки падіння якості відображено на графіку (рис. 4.10).

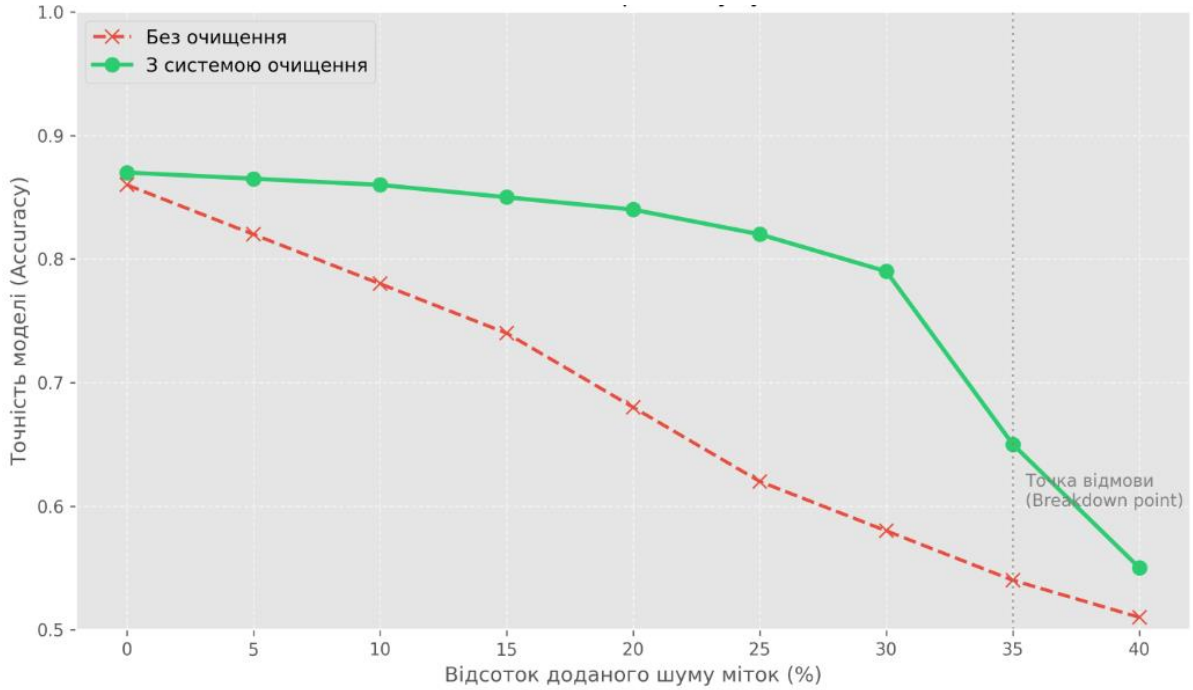


Рисунок 4.10 – Залежність точності класифікації від відсотка шуму в даних

Графік демонструє цікаву поведінку. Червона лінія (без очищення) падає майже лінійно: чим більше шуму, тим гірше вчиться модель, що логічно. Вже при 20 % шуму точність падає до рівня випадкового вгадування.

Зелена лінія (після очищення) показує, що підхід має непогану стійкість до шуму. Система досить впевнено відфільтровує помилки і тримає високу точність (понад 0.8) навіть тоді, коли частка неправильних записів доходить до 25–30 %. Це, як на практиці, вже серйозний рівень зашумленості.

Але після приблизно 35 % з’являється помітний перелом, той самий «breakdown point». У цей момент шуму стає занадто багато. Алгоритм CAIRAD починає сприймати його як нормальний фон даних, а реальні рідкісні випадки, навпаки, трактує як викиди. Через це ефект очищення швидко слабшає.

На практиці рівень шуму в 30 % є екстремальним і рідко зустрічається у правильно організованих клінічних базах даних, тому запас міцності розробленої системи можна вважати достатнім для реальних умов експлуатації.

Висновки до розділу 4

У четвертому розділі проведено комплексне експериментальне дослідження розробленої системи очищення даних на прикладі реального медичного набору Heart Disease UCI (920 записів). Результати підтвердили працездатність програмного комплексу та дозволили сформулювати наступні висновки:

Експериментально показано, що запропонований гібридний підхід із поєднанням статистичних методів (IQR, Z-score) та алгоритму семантичного аналізу (CAIRAD) працює ефективніше, ніж застосування цих методів окремо. У межах вибірки система автоматично виявила 141 аномальний запис (≈15.3 %). Серед них зафіксовано 68 атрибутивних викидів і 76 складних випадків шуму міток, пов’язаних із суперечливими діагнозами. Це якраз той момент, де видно перевагу комбінування, а не спроби «витиснути» максимум з одного інструмента.

Очищення даних призвело до суттєвого покращення прогностичної здатності моделі Random Forest. Загальна точність зросла з 0.7645 до 0.8590 (+9.45 %). Найбільш значущим результатом є підвищення коефіцієнта кореляції Метьюса (MCC) з 0.5249 до 0.7159, що свідчить про перехід моделі з категорії «посередніх» до «високонадійних» та зменшення кількості помилок другого роду (пропущених діагнозів).

Стрес-тестування також дало зрозумілу картину. Система зберігає стабільність і достатньо високу якість очищення за рівня зашумленості до 30 %. Для CAIRAD точка «відмови» (breakdown point) спостерігається на рівні 35 % шуму міток. Такий запас виглядає цілком прийнятним для реальних клінічних даних, де частка помилок зазвичай не перевищує 15–20 %.

Аналіз продуктивності підтвердив правильність вибору технологічного стеку. Час повного циклу обробки для 920 записів склав менше 0.5 секунди, а залежність часу виконання від обсягу даних має лінійний характер, що дозволяє використовувати систему на персональних комп'ютерах середньої потужності без затримок.

У підсумку експерименти показали, що розроблена система не зводиться до простого фільтрування. Вона фактично закриває важливий етап підготовки даних (Preprocessing) і дає змогу отримувати якісні моделі навіть тоді, коли первинні дані спочатку виглядають «брудними».

ВИСНОВКИ

У магістерській кваліфікаційній роботі розв’язано актуальне науково-практичне завдання створення автоматизованої системи очищення даних для підвищення якості моделей машинного навчання. У межах поставленої мети та завдань отримано такі результати:

1) показано, що для медичних даних проблемою є не лише атрибутивні викиди (екстремальні значення), а й шум міток (Label Noise). Виявилося, що класичні статистичні фільтри погано працюють із семантичними суперечностями, тому виникла потреба у комбінованому підході;

2) для реалізації в системі обрано метод інтерквартильного розмаху (IQR) як статистичний інструмент, стійкий до викидів, та алгоритм CAIRAD для аналізу узгодженості записів. Така комбінація перекриває слабкі місця кожного методу окремо і дає змогу очищати дані не тільки на рівні значень атрибутів, а й на рівні логіки записів;

3) спроєктовано систему на основі патерну MVC (Model-View-Controller). Це забезпечило розділення логіки обробки даних і користувацького інтерфейсу. У результаті отримано модульну структуру, яку можна розширювати або доповнювати новими алгоритмами без переробки ядра програми;

4) створено десктопний застосунок мовою Python із використанням Pandas та Scikit-learn. Реалізовано оптимізовані версії алгоритмів із застосуванням векторизації обчислень, що забезпечило високу швидкодію (обробка 50 000 записів приблизно за 21 с). Додатково впроваджено автоматичну оптимізацію типів даних для зменшення споживання оперативної пам’яті;

5) реалізовано зрозумілий графічний інтерфейс на базі PyQt5. Він дозволяє користувачам без навичок програмування завантажувати дані,

налаштовувати чутливість алгоритмів і переглядати розподіли атрибутів через інтегровані графіки Matplotlib;

6) на прикладі набору даних Heart Disease UCI підтверджено практичну цінність системи. Її використання дало змогу підвищити точність класифікації серцевих захворювань на 9.45 % (до 85.9 %) і помітно зменшити кількість хибних прогнозів. Це показує, що інтелектуальне попереднє очищення даних часто дає більший ефект, ніж спроби покращити результат лише ускладненням архітектури моделі.

Наукова новизна роботи полягає у практичній адаптації та поєднанні різнорідних методів детекції (статистичних та імовірнісних) в одному програмному середовищі, що дозволяє ефективніше вирішувати задачу очищення специфічних медичних даних.

Практичне значення роботи полягає у створенні готового інструментарію, який може бути використаний дослідниками, дата-аналітиками та медичними фахівцями для підготовки даних перед застосуванням у системах підтримки прийняття рішень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Agency for Healthcare Quality Resea. Twenty Tips to Help Prevent Medical Errors Using Health Care Information to Help Make Treatment Decisions. *Journal of Pharmaceutical Care in Pain & Symptom Control*. 2000. Т. 8, № 4. С. 65–70. DOI: doi.org/10.1300/j088v08n04_09 (дата звернення: 06.12.2025).

2. Birnbaum M. How We Can Learn from Research and Analysis of Disaster Medical Situations. *Prehospital and Disaster Medicine*. 1998. Т. 13, S1. С. S1. DOI: doi.org/10.1017/s1049023x00037961 (дата звернення: 06.12.2025).

3. Bosmans J. M. The quintessence of medical science and practice. *Acta Radiologica*. 2014. Т. 55, № 8. С. 899–900. DOI: doi.org/10.1177/0284185113516032 (дата звернення: 06.12.2025).

4. Bunse M., Pfahler L. Class-Conditional Label Noise in Astroparticle Physics. *Lecture Notes in Computer Science*. Cham, 2023. С. 19–35. DOI: doi.org/10.1007/978-3-031-43427-3_2 (дата звернення: 06.12.2025).

5. CCLM: Class-Conditional Label Noise Modelling / A. Tatjer та ін. *Pattern Recognition and Image Analysis*. Cham, 2023. С. 3–14. DOI: doi.org/10.1007/978-3-031-36616-1_1 (дата звернення: 06.12.2025).

6. Charnock V. Electronic healthcare records and data quality. *Health Information & Libraries Journal*. 2019. Т. 36, № 1. С. 91–95. DOI: doi.org/10.1111/hir.12249 (дата звернення: 06.12.2025).

7. Class-Wise Denoising for Robust Learning under Label Noise / C. Gong та ін. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2022. С. 1. DOI: doi.org/10.1109/tpami.2022.3178690 (дата звернення: 06.12.2025).

8. Estimating Per-Class Statistics for Label Noise Learning / W. Luo та ін. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2024. С. 1–17. DOI: doi.org/10.1109/tpami.2024.3466182 (дата звернення: 06.12.2025).

9. Healthcare data warehousing and quality assurance / D. J. Berndt та ін. *Computer*. 2001. Т. 34, № 12. С. 56–65. DOI: doi.org/10.1109/2.970578 (дата звернення: 06.12.2025).

10. Healthcare Engineering J. o. Retracted: Application and Effectiveness of Big Data and Artificial Intelligence in the Construction of Nursing Sensitivity Quality Indicators. *Journal of Healthcare Engineering*. 2023. Т. 2023. С. 1. DOI: doi.org/10.1155/2023/9831249 (дата звернення: 06.12.2025).

11. Healthcare Engineering J. o. Retracted: Application of a Nursing Data-Driven Model for Continuous Improvement of PICC Care Quality. *Journal of Healthcare Engineering*. 2023. Т. 2023. С. 1. DOI: doi.org/10.1155/2023/9764290 (дата звернення: 06.12.2025).

12. Malik K., Sadawarti H., G. S K. Comparative Analysis of Outlier Detection Techniques. *International Journal of Computer Applications*. 2014. Т. 97, № 8. С. 12–21. DOI: doi.org/10.5120/17026-7318 (дата звернення: 06.12.2025).

13. Millenson M., O'Brien M. Demanding Medical Excellence. *Public Health*. 2000. Т. 114, № 6. С. 496. DOI: doi.org/10.1038/sj.ph.1900716 (дата звернення: 06.12.2025).

14. Multiprotocol label switching. *Signal Processing Noise*. 2018. С. 104–113. DOI: doi.org/10.1201/9781315220147-8 (дата звернення: 06.12.2025).

15. Ntaji M. Course stress and career satisfaction among medical students of delta state university, Abraka, Nigeria. *Sahel Medical Journal*. 2012. Т. 15, № 2. С. 51. DOI: doi.org/10.4103/1118-8561.145178 (дата звернення: 06.12.2025).

16. Outlier Detection in Logistic Regression / A. A. M. Nurunnabi та ін. *Multidisciplinary Computational Intelligence Techniques*. 2012. С. 257–278. DOI: doi.org/10.4018/978-1-4666-1830-5.ch016 (дата звернення: 06.12.2025).

17. Outlier Detection using Clustering Techniques / S. . та ін. *International Journal of Engineering & Technology*. 2018. Т. 7, № 3.12. С. 813. DOI: doi.org/10.14419/ijet.v7i3.12.16508 (дата звернення: 06.12.2025).

18. Ranga Suri N. N. R., Murty M N., Athithan G. Outlier Detection: Techniques and Applications. Cham : Springer International Publishing, 2019. DOI: doi.org/10.1007/978-3-030-05127-3 (дата звернення: 06.12.2025).

19. Related Statistical Techniques. *Robust Regression and Outlier Detection*. Hoboken, NJ, USA, 2005. С. 248–291. DOI: doi.org/10.1002/0471725382.ch7 (дата звернення: 06.12.2025).

20. Shin J., Kim J.-Y. Customized Quality Assessment of Healthcare Data. *Annals of Laboratory Medicine*. 2024. DOI: doi.org/10.3343/alm.2024.0084 (дата звернення: 06.12.2025).

21. Uttarkabat S. Outlier Detection using Unsupervised Learning Techniques : thesis. 2020. DOI: http://ethesis.nitrkl.ac.in/10220/1/2020_MTech_Research_SUttarkabat_617CS6004_Outlier.pdf (дата звернення: 06.12.2025).

22. Weimerskirch B., Baker C. Data Transparency in Healthcare Quality Improvement. *Current Problems in Pediatric and Adolescent Health Care*. 2025. С. 101758. DOI: doi.org/10.1016/j.cppeds.2025.101758 (дата звернення: 06.12.2025).

ДОДАТОК А
Код програми

```
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.ensemble import IsolationForest

class DataCleaner:
    """
    Основний клас для детекції аномалій.
    Використовується в п. 3.2 диплому.
    """

    def __init__(self, df: pd.DataFrame):
        # Працюємо з копією, щоб не зіпсувати оригінал
        self.df = df.copy()
        self.report = []
    def detect_z_score_outliers(self, threshold=3.0):
        """
        Z-score: працює тільки з числовими колонками.
        """
        # Вибираємо тільки числа
        numeric_df = self.df.select_dtypes(include=[np.number])

        # Якщо даних немає або вони пусті - повертаємо пустий список
        if numeric_df.empty:
            return []
        # Ігноруємо NaN при розрахунках (fill with median temporarily for check)
        numeric_filled = numeric_df.fillna(numeric_df.median())

        z_scores = np.abs(stats.zscore(numeric_filled))
        outlier_indices = np.where((z_scores > threshold).any(axis=1))[0]

        # Повертаємо реальні індекси датафрейму
        original_indices = self.df.index[outlier_indices].tolist()

        self.report.append(f"Z-score: знайдено {len(original_indices)} викидів.")
        return original_indices
    def detect_iqr_outliers(self):
        """
        IQR (Interquartile Range) - стійкий до викидів метод.
        """
        numeric_df = self.df.select_dtypes(include=[np.number])
        if numeric_df.empty:
            return []
        Q1 = numeric_df.quantile(0.25)
        Q3 = numeric_df.quantile(0.75)
        IQR = Q3 - Q1

        # Умова викиду
```

```
            condition = ((numeric_df < (Q1 - 1.5 * IQR)) |
                         (numeric_df > (Q3 + 1.5 * IQR))).any(axis=1)

            outlier_indices = self.df.index[condition].tolist()
            self.report.append(f"IQR: знайдено {len(outlier_indices)} викидів.")
            return outlier_indices

        def detect_isolation_forest(self, contamination=0.05):
            """
            Isolation Forest - ML метод для пошуку аномалій.
            """
            # Готуємо дані: тільки числа, викидаємо NaN для моделі
            numeric_data = self.df.select_dtypes(include=[np.number]).dropna()

            if numeric_data.empty:
                return []

            clf = IsolationForest(contamination=contamination, random_state=42)
            preds = clf.fit_predict(numeric_data)

            # -1 означає викид. Беремо індекси з original data
            outlier_indices = numeric_data.index[preds == -1].tolist()
            self.report.append(f"Isolation Forest: знайдено {len(outlier_indices)} аномалій.")
            return outlier_indices

    class CairadEngine:
        """
        Реалізація алгоритму CAIRAD для виявлення шуму міток.
        Логіка описана в розділі 2.
        """
        def __init__(self, df, target_col, L=0.3, T=0.8):
            self.df = df
            self.target = target_col
            self.L = L
            self.T = T

        def analyze(self):
            noisy_indices = []

            # Перевірка, чи є цільова колонка
            if self.target not in self.df.columns:
                return []

            classes = self.df[self.target].unique()

            # Для прискорення перетворимо df на словники
            records = self.df.to_dict('records')

            for idx, row in zip(self.df.index, records):
                record_score = 0
                attributes_count = 0
```

```
                current_class = row[self.target]

                for col, val in row.items():
                    if col == self.target:
                        continue

                    # Пропускаємо NaN значення
                    if pd.isna(val):
                        continue
                    # P(val | current_class)
                    subset_class = self.df[self.df[self.target] == current_class]
                    count_v_in_c = len(subset_class[subset_class[col] == val])
                    prob_v_c = count_v_in_c / len(subset_class) if len(subset_class) > 0 else 0

                    if prob_v_c < self.L:
                        # Перевіряємо, чи це значення популярне в ІНШИХ класах
                        is_popular_elsewhere = False
                        for other_c in classes:
                            if other_c == current_class:
                                continue

                            subset_other = self.df[self.df[self.target] == other_c]
                            # Захист від ділення на нуль
                            if len(subset_other) == 0:
                                continue

                            prob_other    =    len(subset_other[subset_other[col]    ==    val])    /
len(subset_other)

                            if prob_other > self.L:
                                is_popular_elsewhere = True
                                break

                        if is_popular_elsewhere:
                            record_score += 0 # Це "чуже" значення -> штраф
                        else:
                            record_score += 1 # Рідкісне, але не "чуже" -> ок
                    else:
                        record_score += 1 # Популярне у своєму класі -> ок

                    attributes_count += 1

                # Якщо атрибутів 0 (всі NaN), то довіряємо запису
                final_record_trust = record_score / attributes_count if attributes_count > 0 else 1.0

                if final_record_trust < self.T:
                    noisy_indices.append(idx)

            return noisy_indices
```

ДОДАТОК Б
Матеріали апробації роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ імені Петра Могили

XXVIII ВСЕУКРАЇНСЬКА ЩОРІЧНА НАУКОВО–ПРАКТИЧНА КОНФЕРЕНЦІЯ

«МОГИЛЯНСЬКІ ЧИТАННЯ–2025: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти»

ПРОГРАМА

Миколаїв, 10–14 листопада 2025 року

Миколаїв – 2025

UDC 004.62:616.006

Chuiko G.,
D.Sc. in Physics and Mathematics, Professor of Computer Engineering Department,
Yaremchuk O.,
Senior Lecturer of the Department of Medical and Biological Disciplines,
Bazhenov D.,
MS Student of Computer Engineering Department,
Petro Mohyla Black Sea National University, Mykolaiv, Ukraine

FEATURE ENGINEERING AND NOISE REDUCTION FOR CARDIOVASCULAR RISK PREDICTION IN WEKA

CardiacDirect (https://www.cardiacdirect.com/) is a US-based medical equipment supplier specialising in Machine Learning (ML), what medical professionals refer to as diagnosis or prognosis corresponds to a classification task [1]. When evaluating classifier performance, it is important to use multiple assessment metrics. Although Accuracy, Precision, Recall, and F1-score are widely applied in ML, each has limitations, especially when interpreted in isolation [2]. Therefore, additional indicators such as the Area Under the Receiver Operating Characteristic Curve (ROC AUC), Area Under the Precision-Recall Curve (PRC AUC), the Matthews Correlation Coefficient (MCC), and Cohen's Kappa provide deeper insight into model effectiveness and are particularly useful when comparing different classifiers.

The object of study refers to the specific system, phenomenon, or entity being investigated the "what" of the research. In the referenced document, the object of study is the prediction of heart attack risk using machine learning methods.

The subject of study, in contrast, concerns the conceptual, methodological, or theoretical lens through which the object is examined the "how" of the research. In this case, the subject of study involves using deep learning and feature engineering to enhance the accuracy and efficiency of predictions in clinical datasets.

The researchers reported that artificial neural networks (ANN) can reach an accuracy level of approximately 90% when applied to the examined datasets. To provide a more comprehensive assessment, they supplemented Accuracy with performance indicators such as the Matthews Correlation Coefficient (MCC) and ROC AUC [3]. They also preferred using 10-fold cross-validation instead of splitting the data into separate training and testing sets. Notably, they reported relatively strong MCC values for the ANN classifier, ranging between 0.6 and 0.75.

Another authors propose a wrapper-based feature selection approach to determine the most effective subset of attributes. Their methodology consists of two steps: feature selection and classification. Grey Wolf Optimisation (GWO) is first applied to detect the most relevant features in the dataset. Then, the fitness of the GWO process is evaluated using a Support Vector Machine (SVM) classifier. According to the results, the method achieved an accuracy of 89.83%.

Older medical datasets, such as the one analysed in this study, often contain considerable noise. This can manifest as outliers or extreme attribute values, inconsistencies in records, and even incorrect target labels, commonly referred to as class noise, which is especially harmful in machine learning tasks. Mislabeled cases occur when instances are assigned the wrong class label. In medical diagnostics, this is a realistic problem since different conditions may present with similar symptoms. Attribute noise, on the other hand, refers to incorrect values within one or more features.