МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМЕНІ С.З. ҐЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему:

«СИСТЕМА АВТОМАТИЗОВАНОГО АНАЛІЗУ ЕЛЕКТРОКАРДІОГРАМ З ВИКОРИСТАННЯМ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ»

Виконала: ст. гр. Іт–71з
Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Шевчук Оксана Павлівна
(прізвище та ініціали)

Керівник: к.е.н., доц. Смолінский В.Б.
(прізвище та ініціали)

Рецензент: к.т.н., доц. Левонюк В.Р.
(прізвище та ініціали)

ДУБЛЯНИ 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМЕНІ С.З. ҐЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
«____» _____2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу студенту
Шевчук Оксані Павлівні
(прізвище, ім'я, по батькові)

1. Тема роботи «Система автоматизованого аналізу електрокардіограм з використанням алгоритмів машинного навчання»
   Керівник роботи: к.е.н., доцент, Смолінський В.Б
   затверджені наказом по університету №887/к–с від 30.12.2024.
2. Строк подання студентом роботи 05.02.2026 р.
3. Вихідні дані до роботи *Дослідження актуальності використання машинного навчання для читання та інтерпретації записів ЕКГ та аналіз різних алгоритмів та їх ефективності для створення відповідного програмного забезпечення*
4. Зміст розрахунково пояснювальної записки (перелік питань, які необхідно розробити)
   Вступ
   1. Цифрове представлення біоелектричних організму людини
   2. Основи та концепції машинного навчання
   3. Методологія опрацювання та набір початкових даних
   4. Результати порівняння алгоритмів машинного навчання
   5. Охорона праці та безпека в надзвичайних ситуаціях
   Висновки і пропозиції
   Список використаних джерел
   Додатки

5. Перелік ілюстраційного матеріалу *рисунки, таблиці, візуалізація алгоритмів, візуалізація датасету, моделі та матриці помилок*

6. Консультанти із розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
1,2,3,4	*Смолінський В.Б., доцент кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри інженерної механіки*		

7. Дата видачі завдання 30 грудня 2024 року

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	При–мітки
1	Написання першого розділу	30.12.2024 – 15.02.2025	
2	Написання другого розділу	16.02.2025 – 20.05.2025	
3	Виконання третього розділу та аркушів ілюстраційного матеріалу до нього	21.05.2025 – 20.08.2025	
4	Виконання четвертого розділу та аркушів ілюстраційного матеріалу до нього	21.08.2025 – 25.11.2025	
5	Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»	26.11.2025 – 30.01.2026	
6	Завершення роботи в цілому	31.01.2026 – 05.02.2026	

Студентка _____ Шевчук О.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Смолінський В.Б.
(підпис) (прізвище та ініціали)

УДК 004.93 : 631.432.3

Система автоматизованого аналізу електрокардіограм з використанням алгоритмів машинного навчання. Шевчук О.П. Кафедра інформаційних технологій. Дубляни, ЛНУВМБ ім. С. З. Ґжицького, 2026.

Кваліфікаційна робота: 73 с. текст. част., 24 рис., 3 табл., 28 джерел, 4 додатки.

У роботі проведено комплексний аналіз застосування інтелектуальних обчислювальних методів для автоматизації інтерпретації біоелектричних сигналів серця. Основна увага зосереджена на створенні алгоритмічного забезпечення, яке дозволяє ідентифікувати кардіологічні патології без прямого втручання фахівця, що суттєво пришвидшує процес діагностики.

Науковий пошук ґрунтується на парадигмах предиктивної аналітики, зокрема методах керованого навчання (supervised learning). Автором досліджено ефективність різних архітектур мультикласової класифікації, які адаптовані під специфіку розпізнавання аритмій. Для кожної моделі визначено технічні переваги та обмеження в контексті обробки медичних часових рядів.

Практична реалізація виконана за допомогою екосистеми Python. Вибір інструментарію обґрунтовано потребою у високій швидкості обробки великих масивів даних. Експериментальна частина базується на відкритому репозиторії «ECG Heartbeat Categorization Dataset». У роботі продемонстровано процес цифрової візуалізації вхідних сигналів для верифікації їхньої якості.

Окремий акцент зроблено на етапі підготовки даних (data preprocessing): реалізовано алгоритми нормалізації ознак та виділення ключових дескрипторів сигналу. Ефективність розроблених моделей підтверджена розрахунком матриць помилок (confusion matrices) та порівняльним аналізом метрик точності.

Ключові слова: аналіз часових рядів, цифрова обробка сигналів, біоелектричні сигнали, автоматизоване навчання моделей, діагностика серцевого ритму, ансамблеві методи, ЕКГ, мультикласова класифікація, метрики якості моделей.

ЗМІСТ

ВСТУП

Аналіз глобальних статистичних даних підтверджує домінантну роль серцево-судинних патологій у структурі захворюваності, що зумовлює критичну потребу у розробці високотехнологічних засобів предиктивної діагностики. В умовах цифрової трансформації охорони здоров'я пріоритетним вектором розвитку є проектування інтелектуальних систем підтримки прийняття рішень, що базуються на архітектурах для обробки великих масивів гетерогенних даних (Big Data) у режимі реального часу.

Електрокардіографічне дослідження, як основний метод моніторингу біоелектричної активності міокарда, з позицій інформаційних технологій генерує складний нестаціонарний часовий ряд. Такий сигнал характеризується специфічними амплітудно-часовими патернами (P,Q,R,S,T), автоматизоване розпізнавання яких вимагає застосування методів цифрової обробки сигналів (DSP) високої роздільної здатності. Попри високу інформативність методу, традиційна інтерпретація ЕКГ стикається з проблемою обчислювальної масштабованості, а саме: при тривалому моніторингу (наприклад, за методом Холтера) обсяг неструктурованих даних перевищує когнітивні можливості експерта щодо детального аналізу кожного циклу; висока ймовірність помилок, зумовлених людським фактором та складністю виділення ознак на фоні адитивних шумів.

Для нівелювання зазначених обмежень у даній роботі запропоновано підхід, що ґрунтується на використанні моделей глибокого машинного навчання (Deep Learning) для автоматизації процесів екстракції ознак та класифікації станів. Інтеграція згорткових нейронних мереж (CNN) для аналізу морфологічних особливостей сигналу та рекурентних структур (RNN) для моделювання часових залежностей дозволяє реалізувати наскрізну (end-to-end) обробку «сирих» даних. Такий підхід усуває необхідність ручного проектування фільтрів та дескрипторів, забезпечуючи високу робастність алгоритму при автоматизованому попередньому діагностуванні.

Мета роботи є аналіз та порівняння ефективності алгоритмів машинного навчання, реалізованих у вигляді гібридних нейронних мереж, для ідентифікації електрокардіографічних сигналів серця людини.

Об'єкт дослідження – процеси застосування алгоритмів машинного навчання до ідентифікації електрокардіографічних сигналів.

Предмет дослідження – методи цифрової обробки сигналів, алгоритми глибокого машинного навчання (CNN, LSTM) та моделі класифікації багатовимірних даних.

Завдання роботи: проаналізувати теоретичні та технічні засади цифрового представлення і комп'ютерної обробки біоелектричних сигналів організму людини; розкрити основні концепції машинного навчання та методи підготовки data-set; розробити набір програмних модулів системи комп'ютерного аналізу біомедичних сигналів та виконати порівняльне оцінювання ефективності алгоритмів машинного навчання на основі відповідного набору даних.

Такий підхід скорочує час аналізу, знижує ризик діагностичних помилок і робить діагностику зручнішою для фахівців різних профілів. Методи машинного навчання дозволяють інтегрувати численні фактори, які впливають на електрокардіографічну криву, і тим самим підвищують точність інтерпретації. Це створює надійне підґрунтя для розробки мобільних діагностичних систем та інтелектуальних сенсорів, що забезпечують безперервний контроль стану здоров'я пацієнта поза межами стаціонару.

РОЗДІЛ 1.
ЦИФРОВЕ ПРЕДСТАВЛЕННЯ БІОЕЛЕКТРИЧНИХ СИГНАЛІВ ОРГАНІЗМУ ЛЮДИНИ

1.1 Опис головних понять в прикладній сфері

Електрокардіограма, загальновідома як ЕКГ, є діагностичним інструментом, який реєструє роботу серця шляхом вимірювання його електричної активності за допомогою невеликих датчиків, розміщених на грудях, руках і ногах. Аналізуючи частоту та регулярність серцебиття, цей тест дозволяє виявити аномальні ритми, пошкодження міокарда або збільшення серцевого м’яза. Крім того, ЕКГ може вказати на дефіцит кисню (ішемію), а також продемонструвати вплив лікарських препаратів або роботу регулюючих пристроїв, таких як кардіостимулятори. Це швидке, безпечне та безболісне дослідження [1,2].

Медичні працівники використовують ЕКГ для багатьох цілей: від діагностики порушень ритму до моніторингу ефективності лікування. Тест є неінвазивним і може проводитися як у стані спокою, так і під час фізичного навантаження (стрес-тест). Хоча сучасні персональні пристрої, наприклад смарт-годинники, здатні знімати показники ЕКГ, для точної інтерпретації результатів та встановлення діагнозу критично важливо, щоб запис проводив і аналізував кваліфікований фахівець.

Стандартна ЕКГ використовує 10 кабелів для отримання 12 електричних проєкцій (відведень) серця. Ці різні «погляди» відображають кути, під якими електроди фіксують напрямок електричної деполяризації серця. Відведення від кінцівок включають три біполярні та три уніполярні (посилені) відведення, що формуються на основі електродів на лівій руці, правій руці та лівій нозі (електрод на правій нозі слугує заземленням). Біполярні відведення (I, II, III) вимірюють різницю потенціалів між двома кінцівками, тоді як посилені (aVR, aVL, aVF) порівнюють потенціал однієї кінцівки з умовним нулем [19, 21].

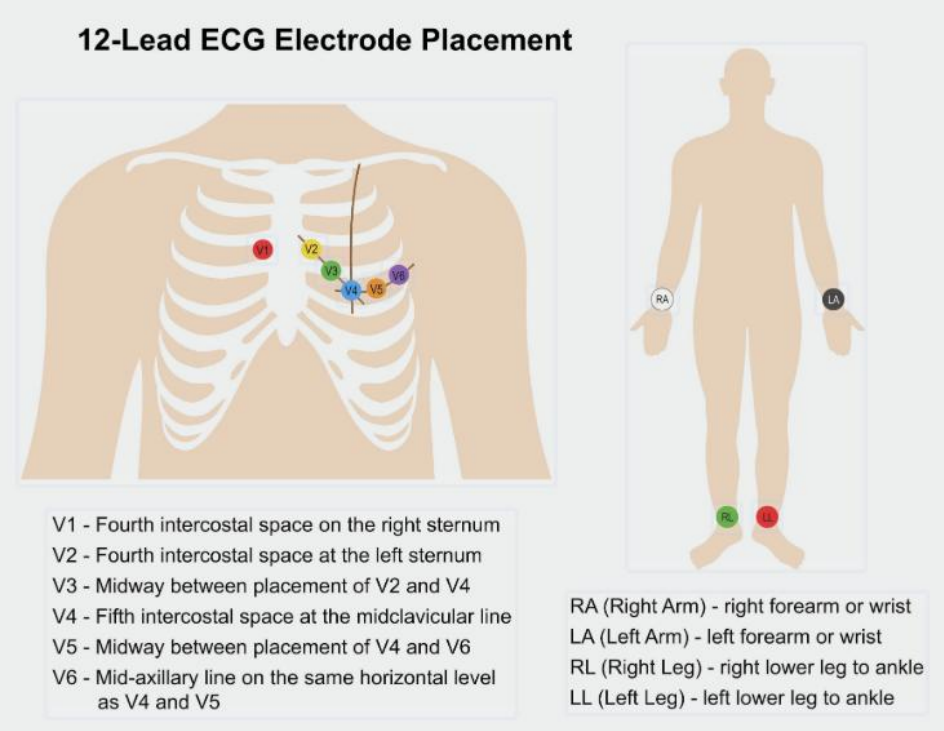

Рисунок 1.1 – Схема розміщення електродів на тілі людини

Додатково шість електродів розміщуються у стандартних позиціях на грудній клітці, формуючи грудні відведення, позначені як V1–V6, що зображено на рисунку 1.1. Ці відведення є уніполярними та реєструють різницю потенціалів між конкретним грудним електродом і середнім потенціалом, отриманим від трьох відведень з кінцівок. Така система дозволяє отримати об'ємну картину електричних процесів у серці.

1.2 Технічні аспекти обробки сигналів ЕКГ

На кардіограмі виділяють умовні елементи: зубці, інтервали, сегменти та комплекси, а також зубці R, Q, P, S, U, T (рис. 1.2).

Кожен елемент на графіку ЕКГ відповідає конкретному етапу серцевого циклу. Зубець P є першим позитивним відхиленням і відображає електричну активацію (деполяризацію) передсердь, що призводить до їхнього скорочення. Далі слідує комплекс QRS, який складається з трьох тісно пов'язаних зубців; він

представляє деполяризацію шлуночків – найпотужнішу електричну подію, яка ініціює викид крові до артерій. Нарешті, зубець Т показує реполяризацію шлуночків, тобто період їхнього електричного відновлення та підготовки до наступного циклу. Будь-яка зміна форми, амплітуди або тривалості цих зубців, а також інтервалів між ними (наприклад, інтервалу PQ або сегмента ST), дозволяє лікарю діагностувати патології: від затримок проведення імпульсу до гострого інфаркту міокарда [21].

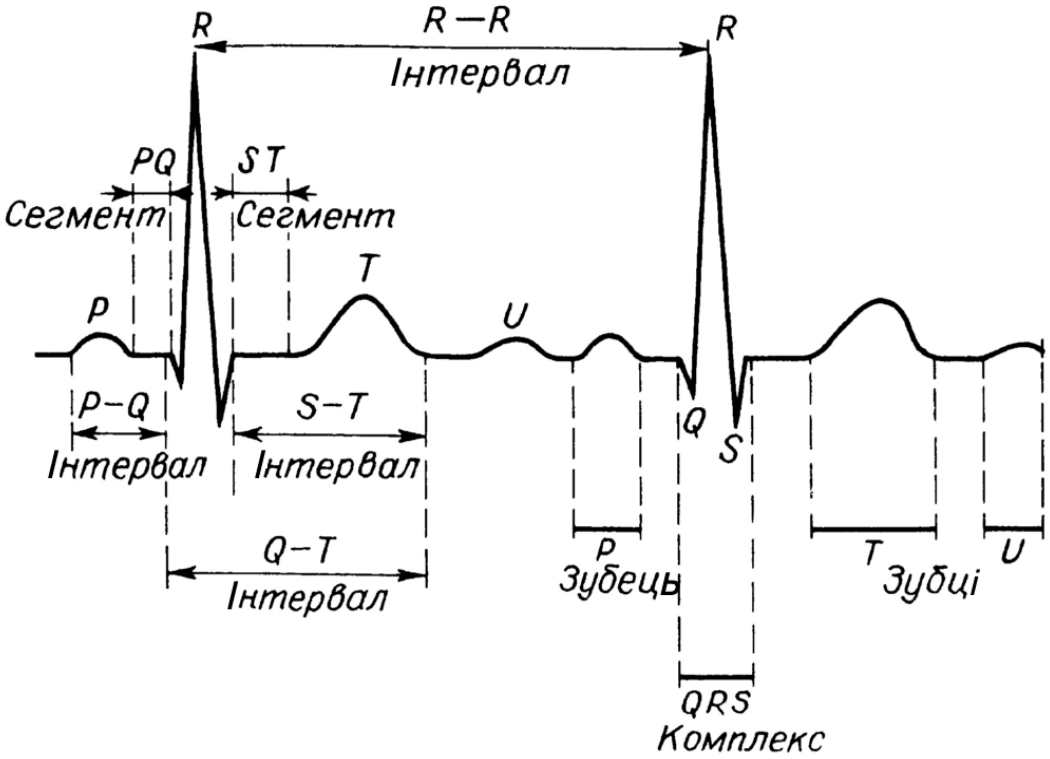

Рисунок 1.2 – Схема ЕКГ одного нормального серцевого циклу

Окрім самих зубців, критично важливим є поняття ізоелектричної лінії, а саме горизонтальної лінії на графіку, яка вказує на відсутність електричної активності в даний момент. Коли хвиля деполяризації затримується в АВ-вузлі, ми бачимо на графіку пряму лінію між зубцем Р та комплексом QRS (сегмент PQ). Це «вікно тиші» необхідне для того, щоб шлуночки встигли наповнитися кров'ю.

Для клінічного аналізу також використовуються певні часові показники:

Інтервал PQ (PR): відображає час, за який імпульс проходить від передсердь до шлуночків. Його подовження може свідчити про блокаду провідності.

Сегмент ST: ділянка між закінченням збудження шлуночків та початком їхнього відновлення. Зміщення цього сегмента вище або нижче ізолінії є головним маркером ішемії або інфаркту міокарда.

Інтервал QT: представляє загальну тривалість електричної активності шлуночків (так звану «електричну систолу»).

Шлях хвилі деполяризації через серце чітко визначений. Синоатріальний вузол (СА-вузол) виступає як природний водій ритму серця. Від нього хвиля деполяризації поширюється через передсердя до атріовентрикулярного вузла (АВ-вузол). У цьому вузлі імпульс на коротку мить затримується, що дозволяє передсердям повністю завершити скорочення перед початком роботи шлуночків.

Далі хвиля деполяризації швидко проходить через пучок Гіса, де вона розділяється на дві гілки, а саме праву та ліву ніжки пучка. Імпульс проходить уздовж міжшлуночкової перегородки до основи серця, де розгалужується на систему волокон Пуркін’є. Звідси хвиля деполяризації розподіляється по стінках шлуночків, ініціюючи їхнє синхронне скорочення.

Апарат ЕКГ обробляє сигнали, отримані від шкіри електродами, і створює графічне зображення електричної активності серця. Логіка побудови графіка проста: електрична активність, спрямована до електрода, спричиняє відхилення вгору (позитивний зубець), а активність у протилежному від нього напрямку – відхилення вниз (негативний зубець). Також важливо пам'ятати, що деполяризація та реполяризація відображаються у протилежних напрямках.

Ця базова модель електричної активності була відкрита понад сто років тому. Вона складається з трьох основних елементів, які отримали назви: зубець Р (відображає деполяризацію передсердь), комплекс QRS (деполяризація шлуночків) та зубець Т (реполяризація шлуночків). Розуміння цих елементів є фундаментом для аналізу будь-якої кардіограми.

РОЗДІЛ 2.
ОСНОВИ ТА КОНЦЕПЦІЇ МАШИННОГО НАВЧАННЯ

2.1 Фундаментальні поняття про машинне навчання

Машинне навчання (Machine Learning, ML) – це підмножина штучного інтелекту, що зосереджена на створенні алгоритмів, здатних самостійно виявляти закономірності в даних і робити точні прогнози на основі нової інформації. Головна перевага ML полягає у здатності системи приймати рішення без жорстко закодованого програмування для кожного окремого випадку. Сьогодні машинне навчання є фундаментом більшості сучасних систем: від прогнозних моделей у фінансах до автономних автомобілів та великих мовних моделей (LLM), таких як ChatGPT.

Центральною ідеєю ML є те, що продуктивність системи оптимізується через процес, який називається навчанням. Важливо розрізняти два поняття, а саме, алгоритм, як набір математичних правил і процедур, а модель, в свою чергу, як результат застосування цього алгоритму до конкретного набору даних. Простіше кажучи, до навчання у вас є алгоритм, а після навчання готова модель. Кінцевою метою є «узагальнення» (generalization), а саме здатність моделі успішно застосовувати вивчені патерни до реальних завдань, з якими вона раніше не стикалася.

Спеціалізованим напрямом машинного навчання є глибоке навчання (англ. Deep Learning), яке базується на архітектурі штучних нейронних мереж. На відміну від традиційного машинного навчання, де людині часто потрібно вручну обирати важливі ознаки даних (feature engineering), глибоке навчання здатне автоматично витягувати складні нюанси з сирих даних. Це зробило його «золотим стандартом» у розпізнаванні зображень та обробці мови, проте такий підхід потребує колосальних обчислювальних ресурсів, зокрема графічних процесорів (GPU) та величезних масивів Big Data [5,9].

Існує три основні типи машинного навчання. Навчання під наглядом (Supervised Learning) використовує розмічені дані, де для кожного входу є правильна відповідь (наприклад, спам-фільтр). Навчання без нагляду (Unsupervised Learning) працює з нерозміченими даними, самостійно шукаючи приховані структури або групи (кластери), що корисно для сегментації клієнтів або виявлення аномалій у транзакціях.

Навчання з підкріпленням (англ. Reinforcement Learning) нагадує процес дресирування: алгоритм діє шляхом спроб і помилок, отримуючи «винагороду» за правильні дії та «штрафи» за помилкові. Саме цей метод дозволяє комп'ютерам перемагати людей у складних іграх, як-от шахи або го, де стратегічна мета важливіша за миттєвий результат. Порівняльна характеристика описаних типів навчань представлена у таблиці 2.1.

Таблиця 2.1 – Порівняння типів машинного навчання

Тип навчання	Розмічені дані	Основні задачі	Приклади застосування
Supervised	Так	Класифікація, прогнозування	Медична діагностика, прогноз цін
Uncupervised	Ні	Кластеризація, пошук закономірностей	Сегментація клієнтів, виявлення аномалій
Reinforcement	Частково	Оптимізація поведінки	Робототехніка, ігрові агенти

Машинне навчання працює на основі математичної логіки. Щоб алгоритм міг «читати» дані, вони повинні бути представлені у числовому вигляді і зазвичай у формі векторів. Якщо фінансові показники перетворити на цифри легко, то тексти та зображення потребують складного процесу перетворення в «числові вбудовування» (embeddings). Якість моделі критично залежить від вибору правильних ознак (features): якщо дані будуть надто «чистими» або нерелевантними, виникне проблема перенавчання (overfitting), і модель не зможе працювати в реальних умовах.

Процес створення моделі складається з кількох етапів: від збору та очищення даних до вибору алгоритму та навчання. Піраміда навчання зображена на рис. 2.1. Після навчання настає етап валідації, де модель перевіряють на даних, яких вона раніше не бачила. Останнім кроком є розгортання та регулярний аудит. Також, варто зауважити, що моніторити не лише математичну точність, а й те, як рішення моделі впливають на бізнес чи безпеку (наприклад, чи не пропускає медична нейромережа ознаки пухлини) [9,27].

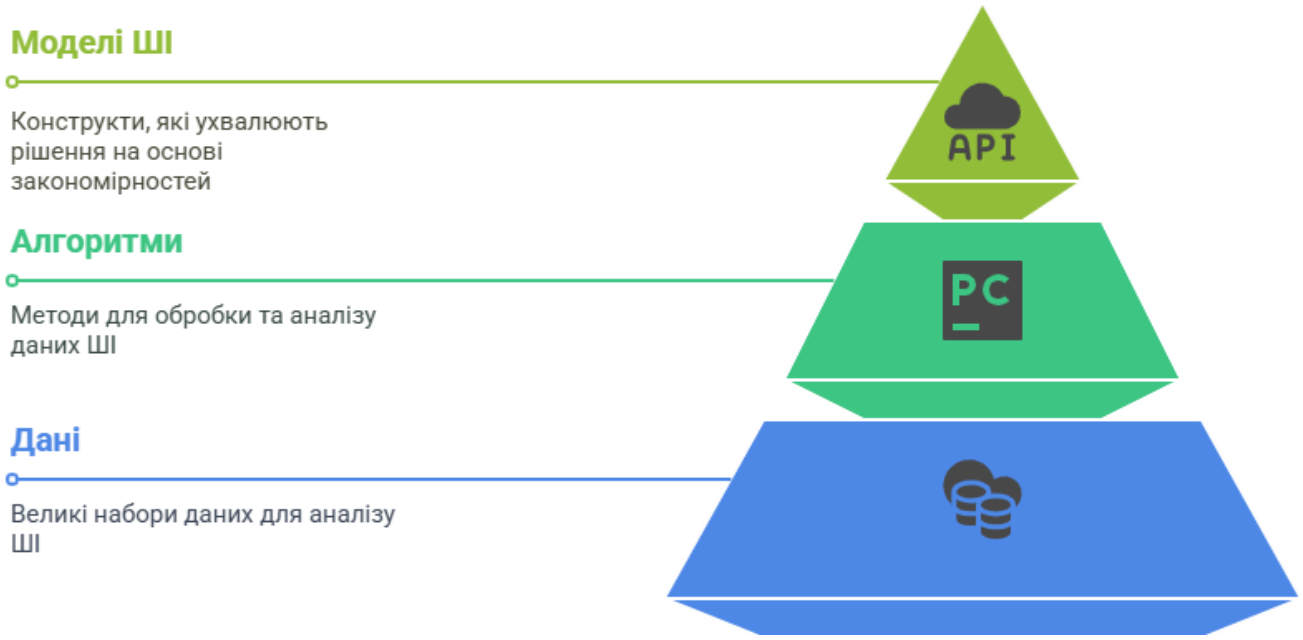

Рисунок 2.1 – Піраміда навчання ML моделі

Таким чином, машинне навчання є не просто статичний набір формул, а динамічний ітераційний процес, де успіх залежить від балансу між якістю вхідних даних, обраною архітектурою та коректністю метрик оцінки. Розуміння цих фундаментальних принципів дозволяє фахівцям переходити від простого спостереження за даними до побудови інтелектуальних систем, здатних до адаптації та самостійного розвитку в умовах постійних змін. Впровадження таких рішень у практику вимагає не лише технічних навичок, а й критичного підходу до аналізу результатів на кожному етапі життєвого циклу моделі [13,25].

Сьогодні ми живемо в еру «вузького» ШІ (Narrow AI): моделі майстерно грають у шахи або діагностують хвороби, але вони не здатні вийти за межі своєї спеціалізації. Проте машинне навчання невпинно рухається в бік створення Загального штучного інтелекту (AGI), який міг би навчатися будь-якого інтелектуального завдання на рівні з людиною.

2.2 Обробка та підготовка вхідних даних

Перш ніж алгоритм зможе зробити будь-який прогноз, реальні об’єкти мають бути перекладені на мову математики. Кожен об’єкт (наприклад, будинок) представляється у вигляді векторного вбудовування, так званого набору чисел, де кожна координата відповідає певній ознаці (площа, кількість кімнат, вік). Наприклад, будинок площею 150 м², з 3 кімнатами та віком 10 років, стає вектором [150, 3, 10]. Оптимізація моделі полягає у пошуку ідеальних «ваг» (параметрів) для цих чисел, щоб отримати максимально точний результат (ціну).

Сирі дані часто містять помилки, дублікати або порожні значення. На етапі очищення фахівці використовують імпутацію (заповнення пропусків середніми значеннями або прогнозами), щоб не втрачати цілі рядки інформації. Також важливо усунути аномалії (outliers) у вигляді критично високих або низьких показників, які не є типовими та можуть «збити з пантелику» модель, викривлюючи статистичні середні.

Для навчання під наглядом (supervised learning) даним потрібен контекст. Розмітка – це процес додавання міток до сирих даних (наприклад, позначення на рентгенівському знімку ділянки з пухлиною або ідентифікація спаму в тексті). Це «навчальний посібник» для моделі, за яким вона вчиться розрізняти об’єкти та явища в реальному світі. Алгоритми чутливі до масштабу чисел. Якщо одна ознака вимірюється в тисячах (ціна), а інша в одиницях (кількість кімнат), модель може помилково надати перевагу більшим числам. Для вирішення цього використовують нормалізацію або стандартизацію, приводячи всі значення до єдиного діапазону

(наприклад, від 0 до 1). Також на цьому етапі проводиться кодування (encoding) категоріальних даних (наприклад, назв міст), перетворюючи їх на цифрові коди.

Сучасні набори даних можуть містити сотні характеристик, багато з яких є шумом або дублюють одна одну. Зменшення розмірності дозволяє спростити модель, залишивши лише найбільш значущі ознаки. Це не лише прискорює навчання, а й допомагає уникнути перенавчання (overfitting), коли модель занадто детально запам'ятовує тренувальні дані, але стає безпорадною перед новими реальними фактами [5,8].

Фінальним кроком є поділ даних на три частини: тренувальну (для навчання моделі), валідаційну (для підбору параметрів) та тестову (для фінальної перевірки). Зазвичай використовується пропорція 80/20 або 70/30. Тестова вибірка – це «іспит» для моделі: вона містить дані, які алгоритм ніколи не бачив під час навчання, що дозволяє об’єктивно оцінити його готовність до роботи.

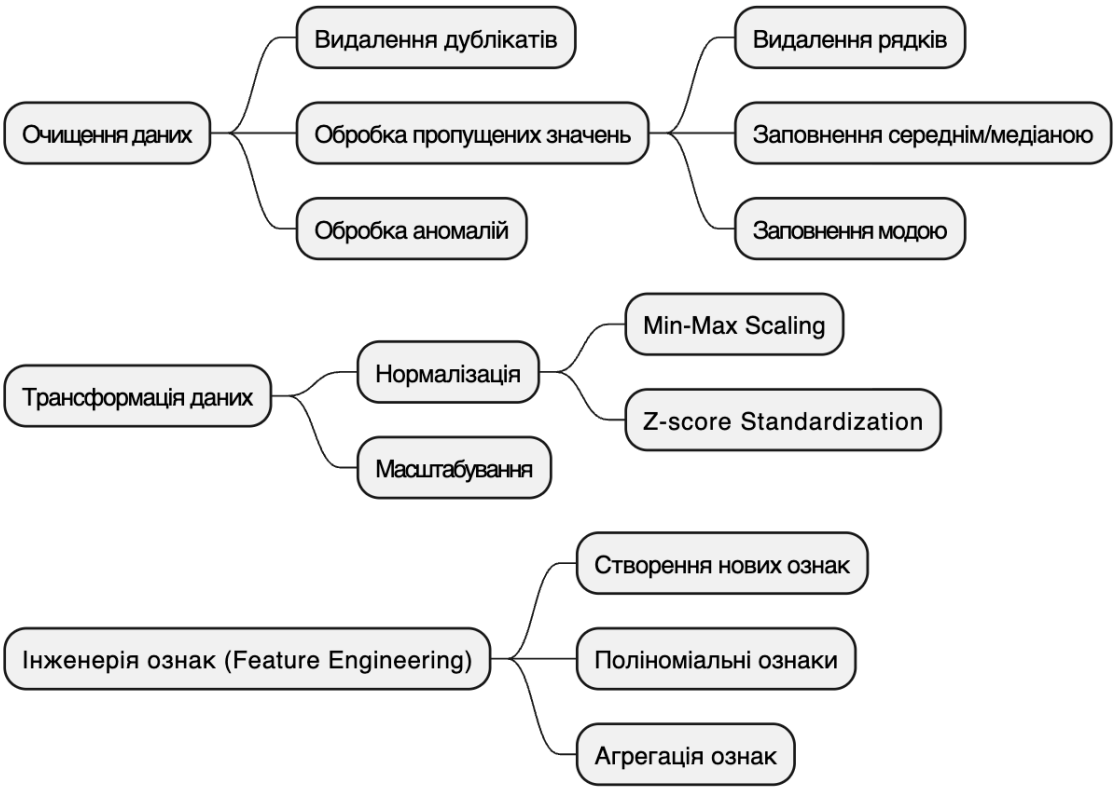

Рисунок 2.2 – Поширені техніки попередньої обробки даних

Перед запуском моделі в роботу команди ML використовують гістограми та діаграми розсіювання для «розвідувального аналізу даних» (EDA). Візуалізація допомагає побачити приховані закономірності, підтвердити гіпотези або виявити системні зміщення в даних, які могли бути пропущені під час автоматичної обробки.

2.3 Алгоритми машинного навчання

Алгоритм (метод) машинного навчання – це математична логіка та процедура, за допомогою якої система штучного інтелекту вчиться ідентифікувати закономірності в навчальних даних. Основна мета полягає в тому, щоб перенести цей досвід на нові, раніше не бачені дані для створення точних прогнозів. Хоча терміни «алгоритм» та «модель» часто вживаються як синоніми, важливо розуміти різницю: алгоритм є покроковим процесом навчання, тоді як модель кінцевим результатом (програма), що видає рішення після обробки вхідних даних.

Ключовою перевагою ML є здатність навчатися неявно. На відміну від класичного програмування, де правила задаються вручну експертом, алгоритми машинного навчання самі знаходять оптимальні шляхи вирішення задач. Головним викликом тут є генералізація – здатність моделі працювати з новими даними. Якщо модель занадто сильно підлаштовується під тренувальний набір, виникає явище перенавчання (overfitting), коли вона демонструє ідеальні результати на навчанні, але стає марною в реальних умовах.

Далі розглянемо як працюють алгоритми, розглянуті у даному дослідженні.

K-Nearest Neighbor (K-NN, метод k-найближчих сусідів) є класичним прикладом метричного алгоритму, який базується на гіпотезі подібності: якщо об'єкти мають схожі характеристики, вони, ймовірно, належать до одного класу. Кожен об'єкт у навчальній вибірці представляється як точка в багатовимірному просторі, де координати є значеннями його ознак. Відстань між точками зазвичай обчислюється за формулою Евкліда, хоча можуть використовуватися й інші метрики (Мангеттенська відстань, косинусна подібність).

Важливою особливістю K-NN є те, що він не має етапу навчання у звичному розумінні. Він просто «запам'ятовує» дані. Вся робота починається в момент класифікації якогось нового об'єкта. Через це K-NN називають «лінивим» алгоритмом. Він дуже гнучкий і легко адаптується до нових даних, просто додаючи їх у свою базу, але пошук сусідів у мільйонах записів може тривати дуже довго, що обмежує його використання в системах реального часу.

Вибір параметра k (кількості сусідів) критично впливає на результат. Якщо k занадто мале (наприклад, k=1), модель буде чутливою до кожної аномальної точки (шуму). Якщо k занадто велике, межі між класами стають розмитими. Крім того, K-NN вимагає обов'язкового масштабування ознак, оскільки ознака з великими значеннями (наприклад, зарплата в тисячах) буде домінувати над ознакою з малими значеннями (наприклад, вік), викривляючи розрахунок відстані. Принцип роботи цього алгоритму зображений на рис. 2.3.

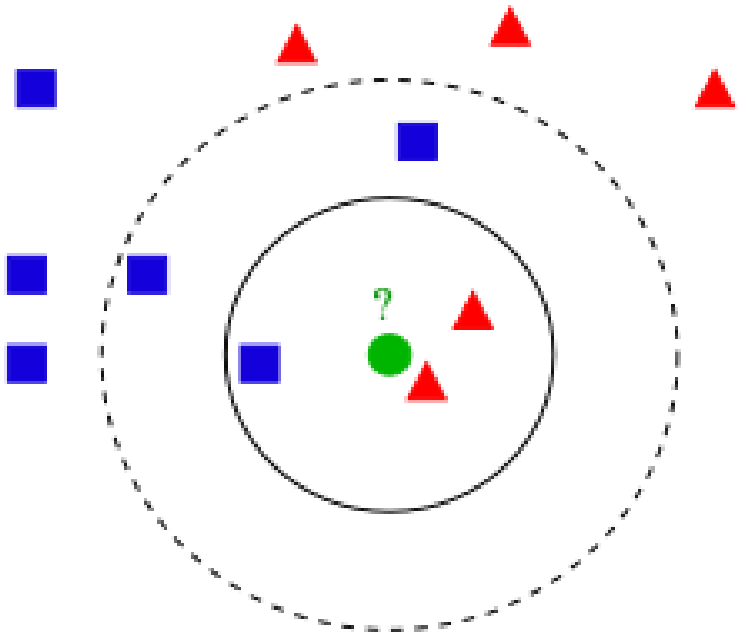

Рисунок 2.3 – Візуалізація K-NN алгоритму

Алгоритм дерева рішень працює шляхом рекурсивного розбиття набору даних на підмножини на основі значень вхідних ознак. Процес починається з кореневого вузла, який містить усю вибірку, і продовжується до тих пір, поки не

буде досягнуто «чистоти« даних (коли в групі залишаються об'єкти лише одного класу) або не спрацюють обмеження по глибині. Для вибору найкращої ознаки для розщеплення зазвичай використовують математичні критерії, такі як ентропія (міра невизначеності) або коефіцієнт Джині. Візуалізацію роботи алгоритму дерева рішень можна розглянути на рис. 2.4.

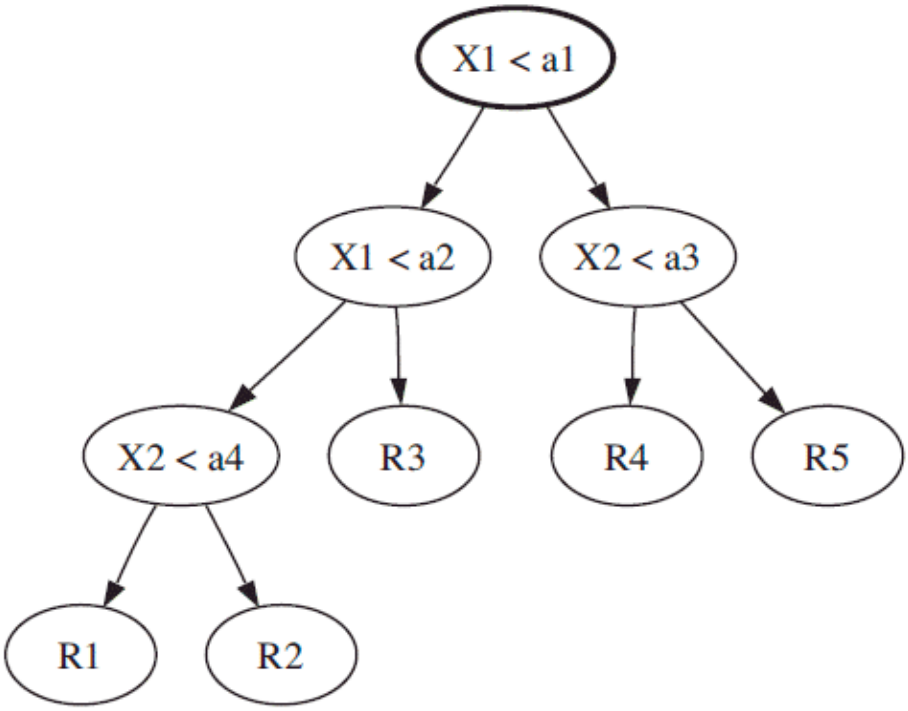

Рисунок 2.4 – Візуалізація алгоритму дерева рішень

Головною перевагою дерев є їхня здатність працювати як з числовими, так і з категоріальними даними без складної попередньої обробки. Оскільки логіка алгоритму нагадує людське мислення (серія бінарних виборів), дерева часто використовуються в сферах, де важливо пояснити причину прийнятого рішення, наприклад, у банківському скорингу або медичній діагностиці.

Однак дерева мають суттєвий мінус – вони нестабільні. Невелика зміна в навчальних даних може призвести до повної перебудови структури дерева. Крім того, без належного контролю (обрізки гілок або «pruning») дерево може стати занадто складним, запам’ятовуючи шум у даних замість загальних закономірностей, що призводить до низької якості прогнозів на нових об'єктах.

Кожен маршрут від кореня до листка дерева відображає ряд тестів і рішень, що ведуть до остаточного висновку [3, 5].

Random Forest, зображений на рис. 2.5, вирішує проблему нестабільності окремих дерев за допомогою методу, який називається беггінг (Bootstrap Aggregating). Алгоритм створює багато випадкових підмножин даних із загального набору, дозволяючи об’єктам повторюватися. На кожній такій підмножині будується окреме дерево рішень. Відмінність полягає у тому, що при побудові кожного розгалуження вибирається не найкраща ознака з усіх можливих, а найкраща з випадково обраної підмножини ознак.

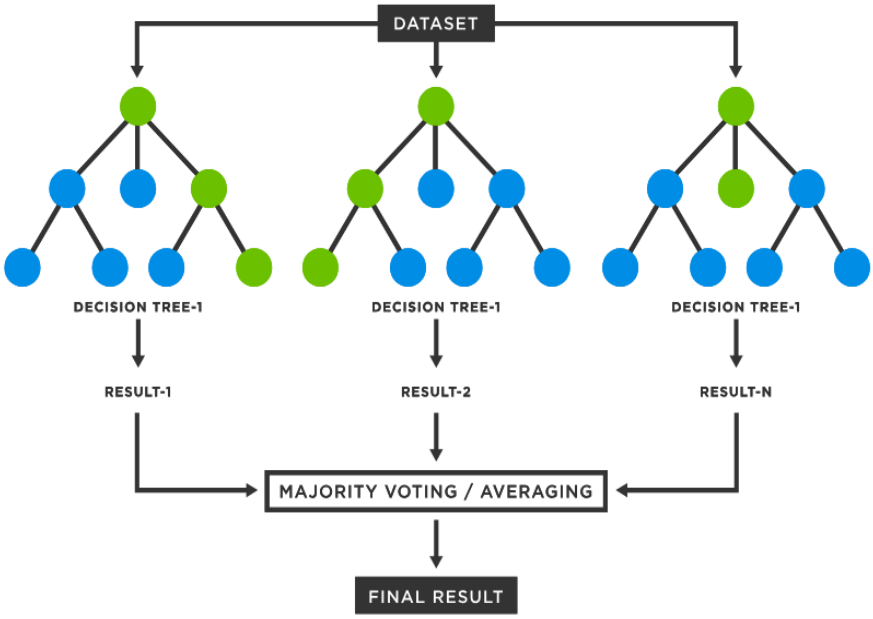

Рисунок 2.5 – Візуалізація алгоритму випадкового лісу

Такий подвійний рандом (по даних і по ознаках) призводить до того, що дерева в «лісі» стають дуже різними та незалежними один від одного. Коли приходить час робити прогноз, алгоритм збирає відповіді від усіх дерев. У задачах класифікації перемагає клас, за який проголосувало найбільше дерев, а в задачах регресії обчислюється середнє арифметичне всіх значень.

Це один із найбільш універсальних алгоритмів у сучасному машинному навчанні. Він демонструє високу точність «з коробки», майже не потребує

нормалізації даних і дуже стійкий до викидів. Проте за це доводиться платити втратою інтерпретованості: якщо логіку одного дерева легко простежити, то зрозуміти колективне рішення тисячі дерев практично неможливо.

Ансамблева природа алгоритму мінімізує ризик перенавчання, забезпечуючи стабільність прогнозів навіть на зашумлених даних без складної попередньої підготовки. Це робить «Випадковий ліс» одним із найефективніших інструментів сучасної практики Data Science завдяки балансу між точністю та обчислювальною потужністю.

Основна ідея методу опорних векторів (англ. Support Vector Machines, SVM) полягає в перетворенні вхідних даних у простір вищої розмірності, де класи можна розділити лінійно. Наприклад, якщо точки двох класів на площині змішані так, що їх не можна розділити прямою, SVM може «підняти» їх у тривимірний простір, де між ними можна провести роздільну площину. Це досягається за допомогою ядерних функцій (kernels), таких як RBF (радіально-базисна функція) або поліноміальне ядро [8,9].

Алгоритм максимізує «зазор» між класами. Ті точки, що лежать на самих краях цих зазорів і фактично тримають на собі роздільну гіперплощину, називаються опорними векторами. Якщо ми видалимо будь-яку іншу точку з навчального набору, положення межі не зміниться, але якщо змінити опорний вектор то модель перебудується. Це робить SVM дуже ефективним при роботі з малими, але якісними вибірками [12]. Розглянути принцип роботи методу опорних векторів можна на рис. 2.6.

Метод опорних векторів часто обирають для задач біоінформатики, розпізнавання рукописного тексту та класифікації зображень. Головним недоліком алгоритму є висока обчислювальна складність на дуже великих наборах даних (більше 100 000 записів) та чутливість до шуму: якщо класи сильно перекриваються, знайти чітку межу стає надто складно.

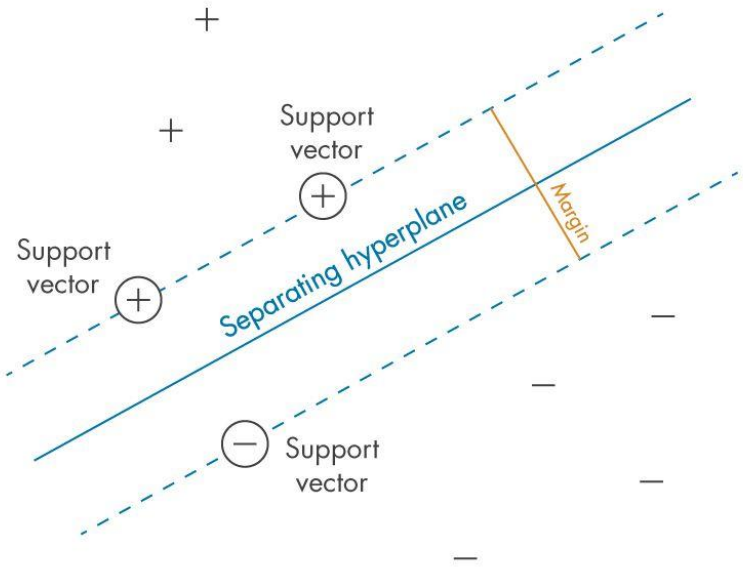

Рисунок 2.6 – Візуалізація SVM

Naïve Bayes – це ймовірнісний алгоритм, який для кожного об'єкта обчислює ймовірність його належності до кожного з класів. Він використовує формулу Байєса, щоб визначити ймовірність події за умови, що певні ознаки вже спостерігалися [9,13]. Назва «наївний» походить від припущення, що всі ознаки є абсолютно незалежними. Наприклад, для алгоритму наявність у фрукті червоного кольору ніяк не пов'язана з його круглою формою, хоча в реальності це не так [8,28]. Візуалізація цього алгоритму зображена на рис. 2.7.

Попри таку теоретичну недосконаліть, алгоритм показує дивовижні результати в аналізі текстів. Він ідеально підходить для категоризації новин або фільтрації спаму, оскільки добре працює з величезною кількістю ознак (кожне слово в тексті – це окрема ознака). Модель обчислює ймовірність того, що лист є спамом, виходячи з частоти появи певних слів (наприклад, «акція», «безкоштовно», «виграш»).

Головною перевагою Наївного Байєса є швидкість та низькі вимоги до обчислювальних ресурсів. Він може навчатися майже миттєво навіть на гігантських масивах даних. Проте він погано справляється з ситуаціями, де ознаки сильно

корелюють між собою, або коли в тестових даних з'являється ознака, якої не було під час навчання (проблема нульової ймовірності) [27].

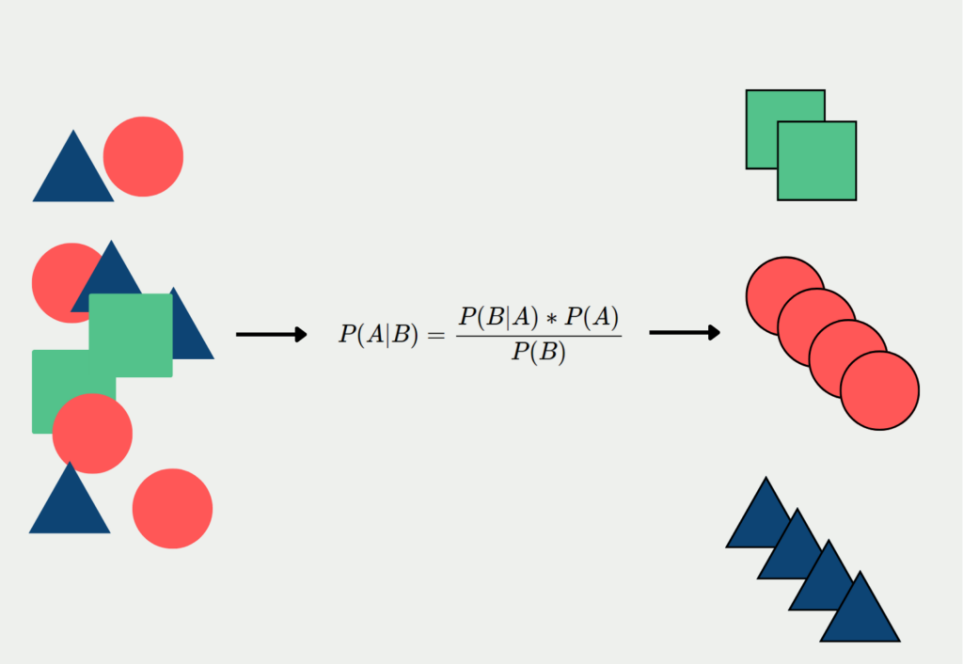

Рисунок 2.7 – Візуалізація Наївного Байєса

У контексті медичної діагностики цей алгоритм часто використовується як базовий рівень (baseline) для швидкої оцінки ризиків на основі сукупності симптомів. Його здатність ефективно працювати з пропусками в даних та висока стійкість до нерелевантних ознак роблять його корисним інструментом для первинного скринінгу великих масивів медичних карток. Хоча «наївне» припущення про незалежність може дещо спрощувати складні фізіологічні взаємозв'язки, швидкість отримання результату робить цей метод незамінним у мобільних діагностичних додатках та системах реального часу.

РОЗДІЛ 3.
МЕТОДОЛОГІЯ ОПРАЦЮВАННЯ ТА НАБІР ПОЧАТКОВИХ ДАНИХ

3.1 Використані технології та інструментарій

3.1.1 Переваги застосування мови Python

Для виконання досліджень було використано високорівневу мову програмування Python, яка стала стандартом де-факто для аналізу даних і штучного інтелекту та часто застосовується для виконання задач у різних сферах (рис. 3.1). Її головна особливість полягає у філософії «читабельність має значення»: синтаксис мови максимально наближений до звичайної англійської мови, що дозволяє розробникам зосередитися на вирішенні бізнес-задач, а не на технічних нюансах написання коду. Завдяки величезній екосистемі бібліотек, таких як Scikit-learn, TensorFlow та Pandas, Python дозволяє реалізовувати складні алгоритми машинного навчання лише кількома рядками коду, що робить її ідеальним інструментом як для швидкого прототипування, так і для побудови великих промислових систем [10,12].

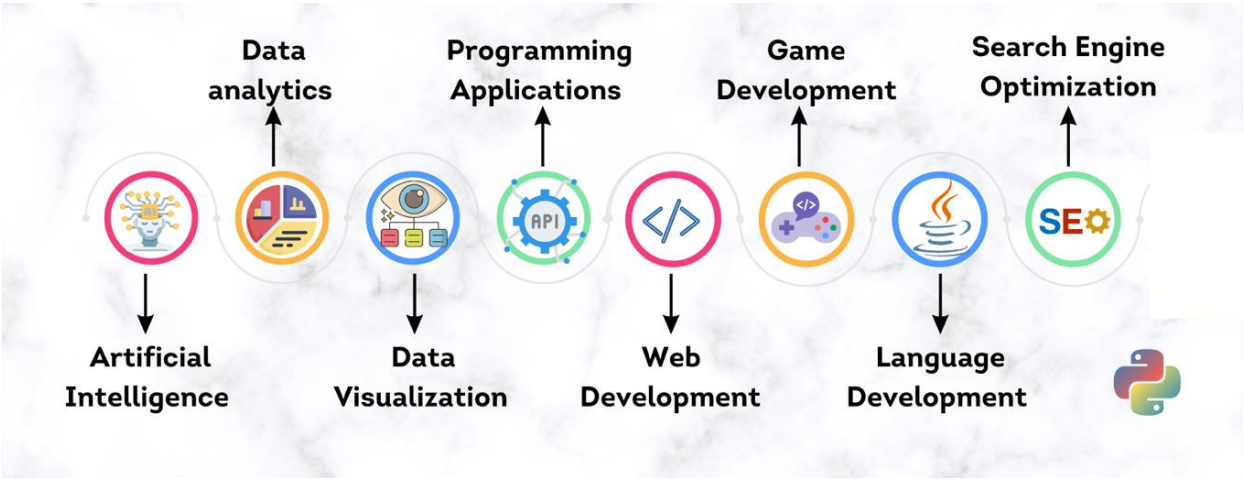

Рисунок 3.1 – Приклади застосування мови Python

У медичній галузі Python відіграє критичну роль у розвитку прецизійної медицини та автоматизованої діагностики. Алгоритми, написані на Python,

використовуються для аналізу медичних зображень (МРТ, КТ, рентген) з метою виявлення патологій на ранніх стадіях, часто перевершуючи за точністю людський око. Крім того, мова застосовується в біоінформатиці для секвенування геному та моделювання нових лікарських препаратів. Наприклад, методи опорних векторів (SVM) та нейронні мережі на базі Python допомагають прогнозувати ризики виникнення хронічних захворювань, аналізуючи тисячі параметрів з електронних медичних карток пацієнтів.

Поза межами медицини Python домінує у веб-розробці (фреймворки Django та Flask), автоматизації хмарних обчислень та фінансовому секторі. У фінтеху мову використовують для алгоритмічної торгівлі, оцінки ризиків та виявлення шахрайських транзакцій у реальному часі. Величезна спільнота розробників забезпечує постійну підтримку та створення нових інструментів, що дозволяє Python залишатися лідером у рейтингах популярності мов програмування. Завдяки своїй гнучкості, вона слугує «клеєм», що об'єднує різні технологічні стеки, від інтерфейсів користувача до складних серверних обчислень.

Однією з ключових технічних переваг Python є його здатність до безшовної інтеграції з іншими мовами програмування, такими як C та C++. Це дозволяє вирішувати проблему продуктивності: критично важливі для швидкості частини коду (наприклад, математичні обчислення в бібліотеці NumPy або обробка тензорів у PyTorch) пишуться на низькорівневих мовах, а Python виступає у ролі зручного високорівневого інтерфейсу. Таким чином, розробник отримує комбінацію швидкості розробки на скриптовій мові та потужності системного програмування, що особливо важливо при роботі з великими даними (Big Data).

3.1.2 Фреймворки Python для машинного навчання

Реалізація практичної частини дослідження базується на використанні трьох ключових бібліотек екосистеми Python, які забезпечують повний цикл роботи з

даними: Pandas (структурування та маніпуляція даними), Matplotlib (графічна інтерпретація результатів) та Scikit-learn (моделювання алгоритмів).

Бібліотека Scikit-learn є найпопулярнішим інструментом для класичного машинного навчання. Вона містить реалізації майже всіх базових алгоритмів, про які ми говорили раніше: від дерев рішень до SVM та K-NN. Головною перевагою Scikit-learn є уніфікований інтерфейс. Незалежно від того, який алгоритм ви використовуєте, процес завжди виглядає однаково: створення моделі, метод fit() для навчання та predict() для прогнозування. Крім того, вона надає зручні інструменти для підготовки та попередньої обробки даних, такі як масштабування ознак, кодування категоріальних змінних та розбиття вибірки на тренувальну й тестову.

Перш ніж застосувати алгоритм, дані потрібно підготувати, і тут на допомогу приходять NumPy та Pandas. NumPy забезпечує підтримку великих багатовимірних масивів і матриць разом із величезною колекцією високорівневих математичних функцій. Pandas, побудована на базі NumPy, пропонує зручні структури даних, як-от DataFrame (аналог таблиці в Excel чи SQL). Вона дозволяє легко маніпулювати даними: фільтрувати рядки, групувати значення, обробляти пропуски та зливати різні таблиці в одну. Без цих бібліотек підготовка даних для медичних досліджень чи фінансових звітів була б надзвичайно часомісткою.

Для графічної інтерпретації отриманих результатів та візуального аналізу розподілу ознак було використано Matplotlib, як фундаментальну бібліотеку для створення статичних, анімованих та інтерактивних графіків у Python. Вона дозволяє будувати все: від простих лінійних діаграм до складних теплових карт, як показано на рис. 3.2. Seaborn базується на Matplotlib і надає більш високорівневий інтерфейс із красивими стилями за замовчуванням. Вона спеціалізується на статистичній візуалізації, дозволяючи будувати складні графіки розподілу даних або матриці кореляцій лише одним рядком коду, що допомагає дослідникам швидко знайти закономірності у великих масивах інформації.

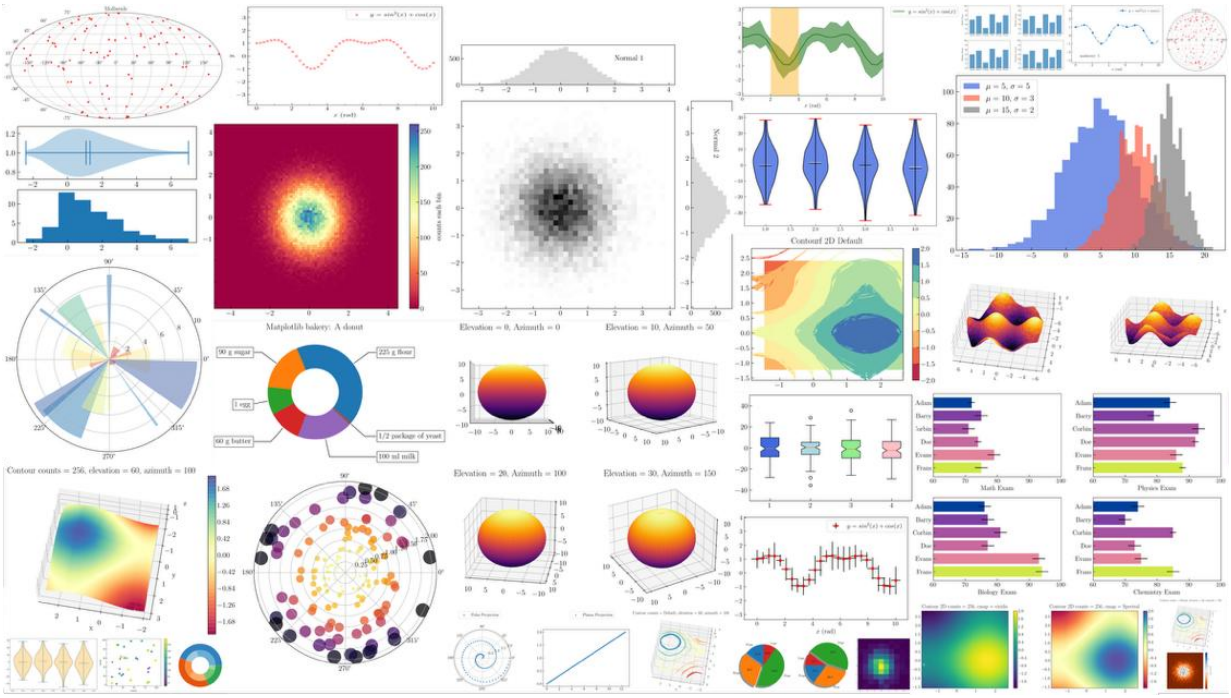

Рисунок 3.2 – Приклади використання pandas та matplotlib

Коли стандартних алгоритмів із Scikit-learn стає недостатньо для досягнення максимальної точності, дата-сайентисти переходять до бібліотек градієнтного бустингу, таких як XGBoost або LightGBM. Ці бібліотеки реалізують ансамблеві методи, схожі на Random Forest, але замість паралельної побудови дерев вони будують їх послідовно, де кожне наступне дерево виправляє помилки попередніх. Вони надзвичайно ефективні у змаганнях з аналізу даних (наприклад, на платформі Kaggle) та у промислових системах, оскільки оптимізовані для швидкості обчислень і роботи з пам'яттю, що дозволяє обробляти мільйони записів за лічені секунди [27].

3.2 Використані вхідні дані

Датасет являє собою зібрання даних, яке служить для навчання, тестування та оцінювання моделей машинного навчання. Він містить сукупність спостережень або прикладів, причому кожен приклад подано у вигляді векторів характеристик, що містять значення вхідних змінних чи ознак.

Для навчання моделей машинного навчання в рамках цього проекту був використаний безкоштовний відкритий датасет з Kaggle із назвою «ECG Heartbeat Categorization Dataset» створений Shayan Fazeli [6].

3.2.1 Опис використаного датасету

Відкритий датасет із назвою «ECG Heartbeat Categorization Dataset», розроблений SHAYAN FAZELI і доступний на Kaggle, містить 109,446 записів електрокардіограм, зібраних від сорока семи пацієнтів [6]. Кожен окремий запис складається з 187 характеристик напруги, що представляють один сигнал ЕКГ, зафіксований з одного з двох електродів. У документації зазначено, що частота дискретизації для кожного ЕКГ-сигналу в цьому датасеті становить 125 Гц, що означає, що період зчитування значень напруги дорівнює $T = 1/f = 1/125 = 8$ мс.

Записи ЕКГ позначені однією зі світових п'яти категорій для навчання моделей машинного навчання:

✓ 0 категорія: Normal beat – позначає нормальний ритм серця;

✓ 1 категорія: Supraventricular premature beat – позначає надшлуночкову екстрасистолію;

✓ 2 категорія: Premature ventricular contraction – позначає передчасне скорочення шлуночків;

✓ 3 категорія: Fusion of ventricular & normal beat – позначає злиття шлуночкового та нормального ритму;

✓ 4 категорія: Unclassifiable beat – позначає некласифікований ритм.

Отже, даний датасет пропонує широкий спектр записів ЕКГ з різноманітними порушеннями серцевого ритму, а також здоровими ритмами. Такий набір даних можуте бути застосованим для тренування і тестування моделі машинного навчання, спрямованих на класифікацію серцевих ритмів [6,7,14].

3.2.2 Візуалізація датасету

Записи в датасеті можна зручно представити у вигляді електрокардіограми.

Для наглядності, було вибрано по одному запису із кожної із чотирьох категорій: Normal beat, Premature ventricular contraction, Supraventricular premature beat, Fusion of ventricular & normal beat.

Категорія під назвою Unclassifiable beat була пропущена, оскільки не містить цінної інформації в контексті цього проекту.

Візуалізація запису із категорії 0: Normal beat. У даному датасеті, перший запис для цієї категорії має номер 0 і зображений на рис. 3.3.

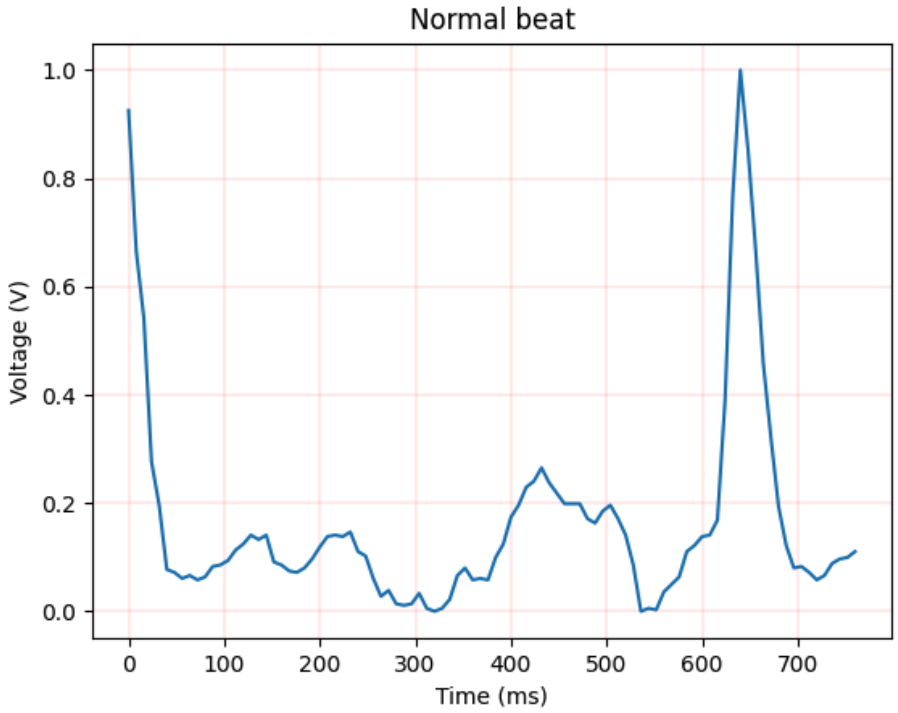

Рисунок 3.3 – Зразок ЕКГ позначений як Normal Beat

Аналізуючи цей ЕКГ запис візуально, можна помітити, що у сигналі не спостерігається відхилень від норми.

Програмний код, який використовувався для візуалізації записів ЕКГ зображений в додатку А.

Візуалізація запису із категорії 1: Supraventricular premature beat. У даному датасеті, перший запис для цієї категорії має порядковий номер 72474 і зображений на рис. 3.4.

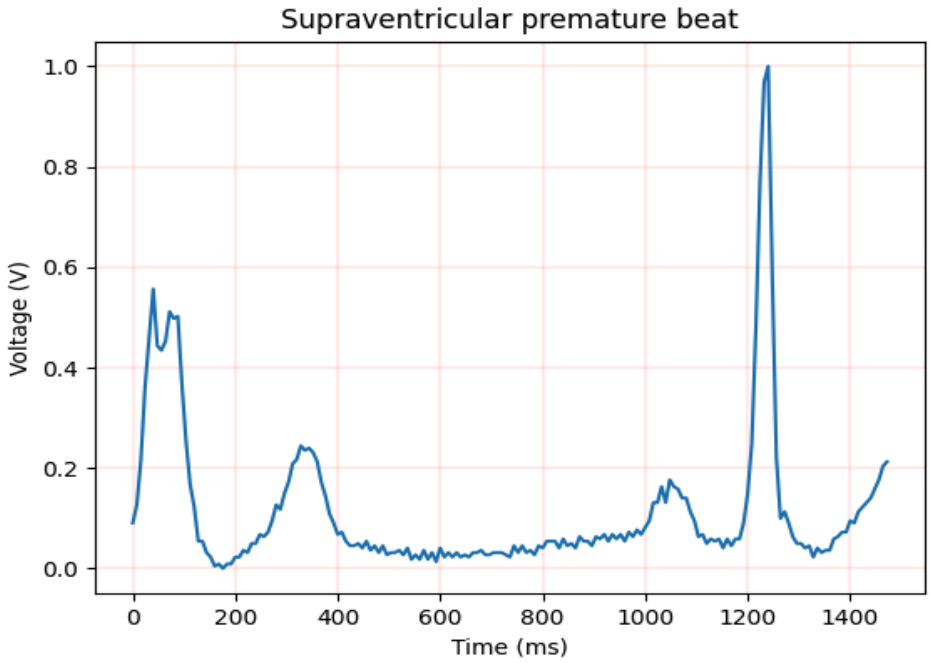

Рисунок 3.4 – Зразок ЕКГ позначений як Supraventricular premature beat

При візуальному аналізі цього запису, виявляється наявність Supraventricular premature beat, так як перший зубець R містить «впадину» і є поділеним на дві частини.

Візуалізація запису із категорії 2 для Premature ventricular contraction. У даному датасеті, перший запис для цієї категорії має порядковий номер 74697 і представлений на рис. 3.5.

Аналізуючи цей ЕКГ запис візуально, можна зауважити, що тут присутній Supraventricular premature beat, бо другий зубець R має «впадину» і поділений на 2 частини.

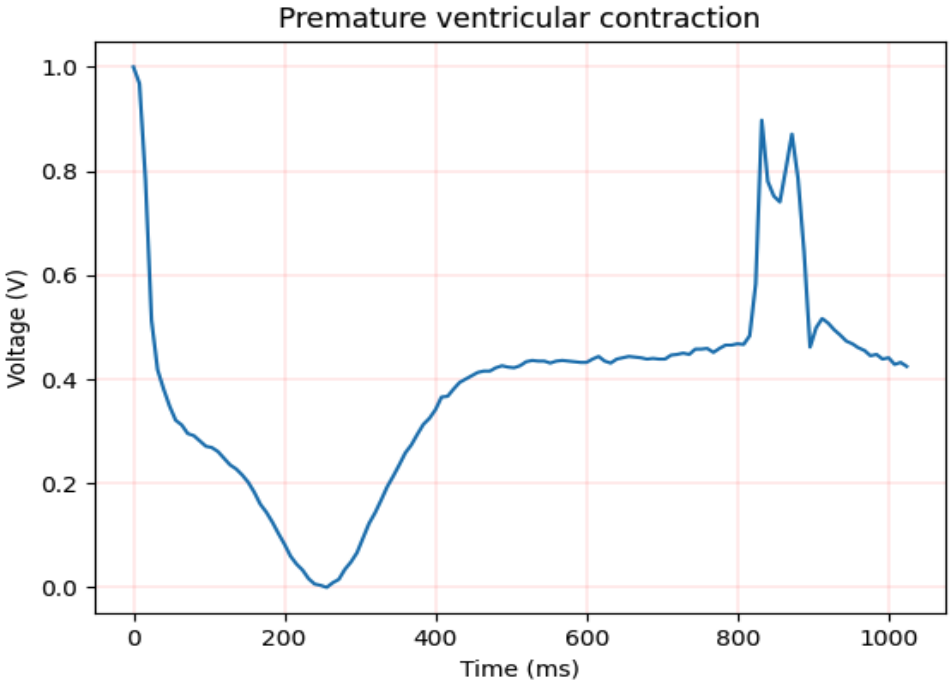

Рисунок 3.5 – Зразок ЕКГ позначений як Premature ventricular contraction

Візуалізація запису із категорії 3: Fusion of ventricular & normal beat. У даному датасеті, перший запис для цієї категорії має порядковий номер 80485 і показаний на рис. 3.6.

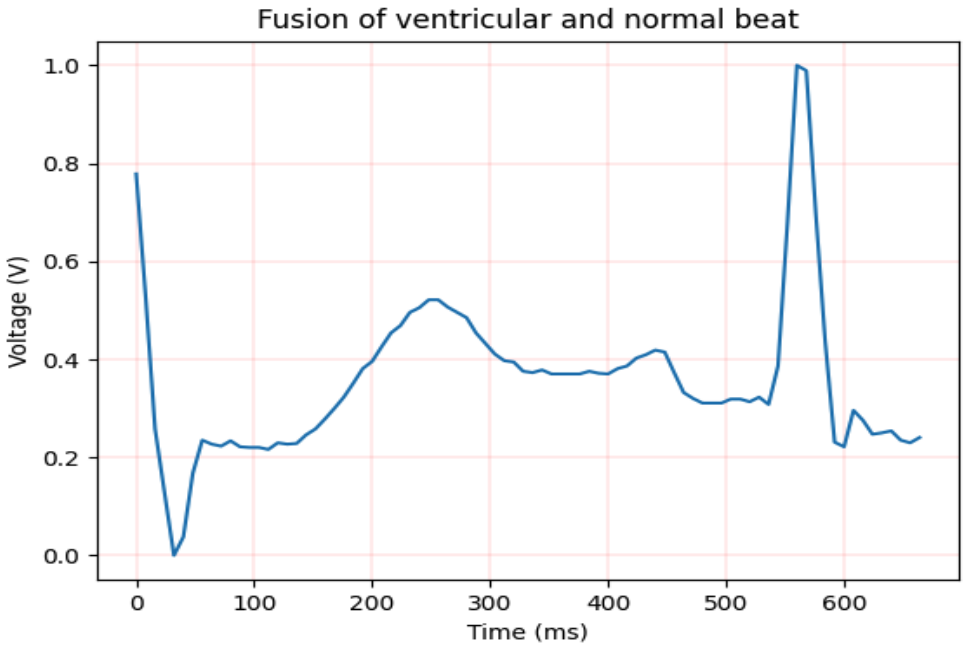

Рисунок 3.6 – Зразок ЕКГ позначений як Fusion of ventricular & normal beat

Цей тип кардіоциклу виникає, коли синусовий імпульс і збудження зі шлуночків активують міокард майже одночасно, створюючи гібридну морфологію сигналу.

Аналізуючи цей ЕКГ запис візуально, можна впевнено віднести сигнал до класу «Fusion of ventricular & normal beat», оскільки на кардіограмі спостерігається специфічний деформований комплекс, що має ознаки обох типів збудження. Зокрема, комплекс виглядає ширшим за нормальний (синусовий), проте він часто має зубець P, що передує йому, на відміну від класичної шлуночкової екстрасистоли. Це підтверджує одночасну активність двох різних джерел електричних імпульсів, що призводить до накладання векторів деполяризації.

Таким чином, проведений етап візуалізації підтверджує репрезентативність обраних даних та демонструє суттєві морфологічні відмінності між класами часових рядів у межах датасету. Ідентифіковані аномалії, такі як деформація зубця R та наявність реципрокних змін, є ключовими предикторами для алгоритмів глибокого навчання.

Отримана графічна інтерпретація специфічних патологій обґрунтовує доцільність використання архітектур нейронних мереж, здатних до екстракції нелінійних ознак у складних біосигналах. Верифікація вхідних даних на цьому етапі мінімізує ризики помилкової класифікації та створює підґрунтя для переходу до етапу розробки архітектури інтелектуального класифікатора.

РОЗДІЛ 4.
РЕЗУЛЬТАТИ ПОРІВНЯННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

4.1 Підготовка даних та їх попередня обробка

Нормалізація – це процес перетворення числових ознак у стандартний діапазон, зазвичай від 0 до 1 або від -1 до 1 [5,28]. Це критично важливо для алгоритмів, що базуються на обчисленні відстані, таких як K-NN або SVM. Без нормалізації ознака з великими абсолютними значеннями (наприклад, річний дохід у гривнях) матиме набагато більший вплив на результат моделі, ніж ознака з малим діапазоном (наприклад, вік пацієнта), навіть якщо остання є більш значущою для прогнозу. Математично найпоширенішим методом є Min-Max Scaling, де кожне значення x перетворюється за формулою:

$$X' = \frac{(X - Xmin)}{(Xmax - Xmin)}, \tag{4.1}$$

де *X'* – нормалізоване значення; *X* – початкове значення; *Xmax* – максимальне значення *X*; *Xmin* – мінімальне значення *X*;

Окрім покращення точності, нормалізація суттєво прискорює процес навчання моделі, особливо для алгоритмів, що використовують градієнтний спуск (як-от логістична регресія або нейронні мережі). Коли всі ознаки масштабовані однаково, поверхня функції втрат стає більш симетричною, що дозволяє алгоритму оптимізації швидше знаходити глобальний мінімум. Проте варто пам'ятати, що звичайна нормалізація чутлива до викидів (outliers): якщо в даних є одне аномально велике значення, воно «стисне» всі інші корисні дані у дуже вузький проміжок, тому перед її застосуванням важливо провести аналіз аномалій або використовувати стійку стандартизацію (StandardScaler).

Для того, щоб отримати об'єктивну оцінку якості моделі, набір даних зазвичай розділяють на три частини: тренувальну, валідаційну та тестову. Тренувальний набір використовується безпосередньо для навчання алгоритму, де модель шукає закономірності. Валідаційна вибірка потрібна для «тонкого налаштування» моделі: саме на ній перевіряються різні комбінації гіперпараметрів (наприклад, глибина дерева чи кількість сусідів у K-NN), щоб обрати найкращу конфігурацію, не перенавчаючись під тестові дані. Тестовий набір залишається повністю ізольованим до самого кінця і слугує «фінальним іспитом», який показує, як модель працюватиме в реальних умовах [8,9].

Конструювання додаткових параметрів є творчим етапом, на якому дані перетворюються з сирого вигляду у форму, що краще розкриває суть проблеми для алгоритму. Це може включати створення нових ознак на основі існуючих: наприклад, маючи дату народження, ми вираховуємо вік, або, маючи координати, розраховуємо відстань до центру міста. Такий підхід дозволяє моделі вловити складні нелінійні зв'язки, які не були очевидними спочатку. Часто якісно сконструйовані параметри дають більший приріст до точності моделі, ніж зміна самого алгоритму машинного навчання.

Виділення ознак (Feature Selection) спрямоване на те, щоб залишити в моделі лише найбільш інформативні змінні, відкинувши «шум» та дублюючу інформацію. Велика кількість зайвих ознак не лише сповільнює навчання, а й може призвести до «прокляття розмірності», коли модель губиться в надлишковій інформації. Для цього використовуються методи статистичного аналізу або алгоритми зменшення розмірності (наприклад, PCA – метод головних компонент), які стискають простір ознак, зберігаючи при цьому максимум корисної варіативності даних. Це робить модель простішою, швидшою та стійкішою до помилок [8,13].

Тривалість інтервалу R-R, представленого на рис. 4.1, виступає ключовим біомаркером при класифікації ЕКГ, оскільки він відображає часову динаміку

серцевого циклу та варіабельність ритму. Для машинного навчання цей інтервал виділяється шляхом детекції піків зубців R, що дозволяє перетворити складний безперервний сигнал у набір дискретних числових значень, які легко інтерпретуються алгоритмами на кшталт Random Forest або SVM. Статистичні метрики R-R інтервалів, такі як середнє квадратичне відхилення (RMSSD) або амплітуда коливань, стають ключовими предикторами для автоматизованого виявлення аритмій, оскільки вони чітко вказують на патологічні відхилення у часовій структурі серцебиття, недоступні для прямого візуального аналізу сирого сигналу.

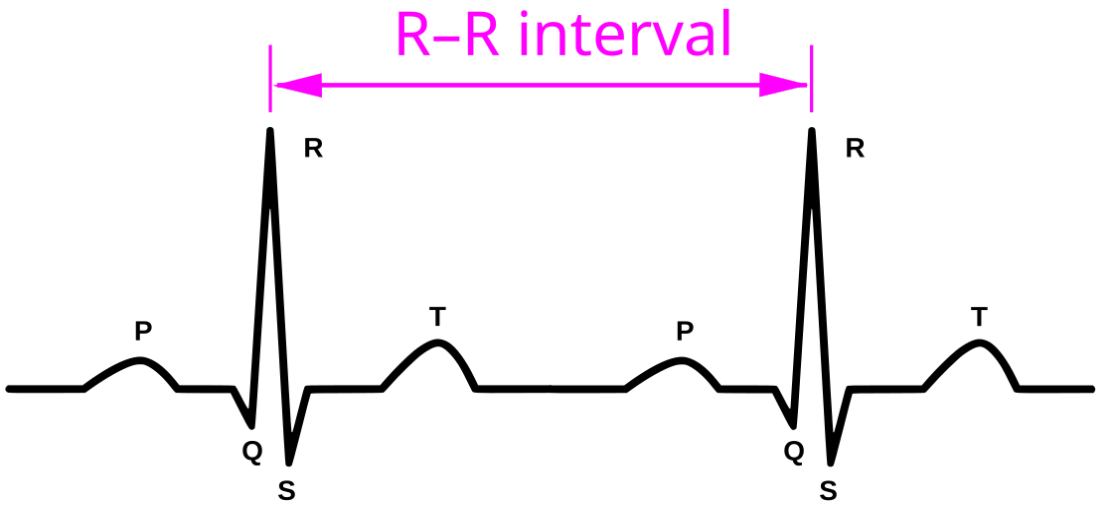

Рисунок 4.1 – Інтервал RR

Щоб отримати інтервал RR для кожного запису електрокардіограми (ЕКГ), була розроблена функція compute_rr_interval(X), розміщена у додатку Б. Ця функція обчислює різницю між індексами, на яких розміщені зубці R у даному сигналі електрокардіограми. Оскільки із документації відомо, що зчитування значень напруги відбувалось з періодом 8 мс, ми можемо легко розрахувати інтервал RR, скориставшись формулою (t = n / times 0,008) сек.,

де (t) – це інтервал RR визначений у секундах, (n), в свою чергу різниця між індексами, яку визначає функція compute_rr_interval(X) на рис 4.2.

```
def compute_rr_intervals(self, data_frame):
        rr_values = []
        for _, row in data_frame.iterrows():
            signal_values = row.values
            peak_positions = self._locate_r_peaks(signal_values)
            interval_samples = abs(peak_positions[0] -
                peak_positions[1])
            interval_seconds = interval_samples * self
                .sampling_period
            rr_values.append(interval_seconds)
        return np.array(rr_values)
```

Рисунок 4.2 – Функція обчислення інтервалу R-R

Наприклад, якщо зубці R знаходяться на індексах 250 і 375, тоді їх різниця становитиме 125.

Комплекс QRS, зображений на рис. 4.3, є значущим критерієм для діагностики, оскільки зображає проходження електроімпульсу через шлуночки серця під час їх деполяризації. Відхилення від норми, такі як розширення або деформація QRS комплексу у грудних правих відведеннях (V1, V2), можуть свідчити про наявну блокаду правої ніжки пучка Гіса. У свою чергу, зміни у QRS комплексі у стандартних та грудних лівих відведеннях можуть вказувати на блокаду лівої ніжки пучка Гіса.

Крім того, патологічні зміни у комплексі QRS є ознакою гіпертрофії шлуночків. Це проявляється у такі способи: збільшення вольтажу комплексу QRS і його розширення, а також відхилення електричної осі комплексу QRS.

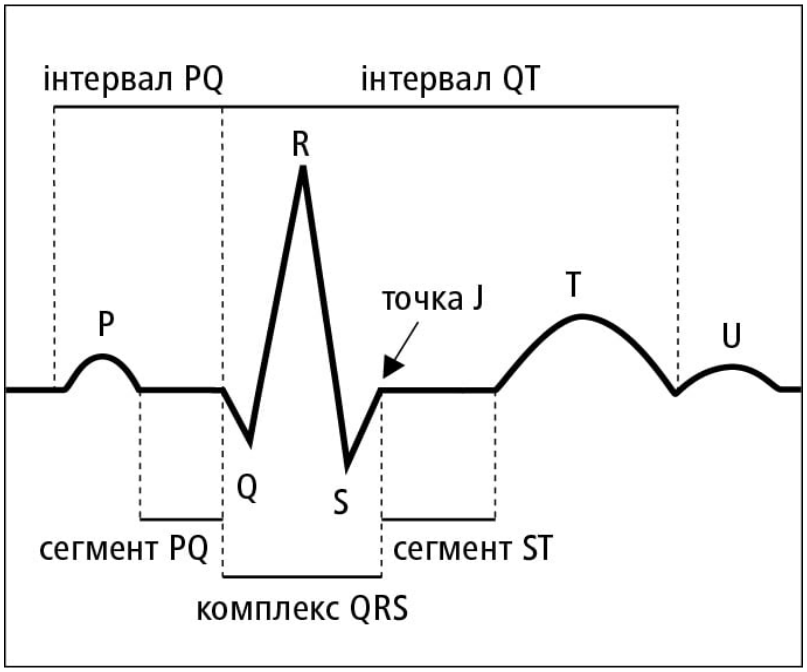

Рисунок 4.3 – Схема ЕКГ (Комплекс QRS)

Некроз міокарда також відбивається на ЕКГ у зміні комплексу QRS. При трансмуральному (наскрізному) некрозі спостерігається зникнення всіх позитивних відхилень під активним електродом, що на ЕКГ проявляється у вигляді комплексу QS. Якщо некроз охоплює частину стінки серця (найчастіше біля ендокарда), а ознакою некрозу стає комплекс QR. У таких випадках зубець R відображає збудження збережених ділянок, а Q вказує на взяття до уваги векторів зони некрозу.

Щоб визначити комплекс QRS у кожному записі електрокардіограми (ЕКГ), була створена функція compute_qrs_complexes (X)), представлена у додатку Б. Ця функція обчислює різницю між індексами, на яких знаходяться зубці S і Q у сигналі ЕКГ. Зважаючи на те, що період зчитування значень напруги становить 8 мс, комплекс QRS можна легко розрахувати за формулою (t = n /0,008) сек., де (t) – це тривалість комплексу QRS в секундах, а (n) – різниця між індексами, яку визначає функція compute_qrs_complexes (X), зображена на рис. 4.4.

```
def compute_qrs_complexes(self, data_frame, r_peak_indices):
        qrs_durations = []
        for idx, (_, row) in enumerate(data_frame.iterrows()):
            signal_values = self._clean_signal(row.values)
            r_position = r_peak_indices[idx][0]
            baseline = np.median(signal_values)
            q_start = self._find_q_wave_start(signal_values,
                r_position, baseline)
            s_end = self._find_s_wave_end(signal_values,
                r_position, baseline)
            duration_samples = s_end - q_start
            duration_seconds = duration_samples * self
                .sampling_period
            qrs_durations.append(duration_seconds)
        return np.array(qrs_durations)

    def _find_q_wave_start(self, signal, r_idx, baseline_level):
        idx = r_idx - 1
        found_peak = False
        while idx > 0:
            current_val = signal[idx]
            next_val = signal[idx + 1]
            if current_val > next_val: found_peak = True
            if found_peak and (current_val < next_val or
                current_val >= baseline_level): return idx
            idx -= 1
        return 0
```

Рисунок 4.4 – Функція обчислення комплексу QRS

Наприклад, якщо зубець Q знаходиться на індексі 372, зубець S – на індексі 382, то їхня різниця становитиме 10. Використовуючи формулу, можна обчислити, що це відповідає 80 мілісекундам.

Програмний код для розрахунків часових інтервалів RR та QRS наведено в додатку Б.

Далі розглянемо статистичні вимірювання. Застосування методів математичної статистики є фундаментом для аналізу електрокардіографічних (ЕКГ) сигналів. Статистичні метрики дозволяють отримати кількісну оцінку сигналу, сформувати його описовий профіль та ідентифікувати закономірності, що

відрізняють фізіологічну норму від патологічних станів. У межах даного дослідження ці параметри використовуються як інформативні ознаки (features) для навчання класифікаційних моделей, оскільки вони відображають специфіку розподілу, амплітудну варіабельність та морфологічні особливості значень напруги.

Для формування вектору ознак кожного запису було обрано такий набір статистичних показників, який включав в себе середнє арифметичне, що визначає центральну тенденцію сигналу та загальний рівень амплітуди. Медіана, що характеризує стійке середнє значення, що є менш чутливим до поодиноких сплесків або артефактів запису. Середньоквадратичне відхилення, це кількісний показник варіабельності сигналу, що демонструє ступінь розсіювання значень напруги навколо середнього рівня. Коефіцієнт асиметрії відображає ступінь нерівномірності розподілу амплітуд відносно центру, що важливо для детекції специфічних деформацій зубців ЕКГ. Коефіцієнт ексцесу вказує на «піковість» або «площинність» розподілу, дозволяючи оцінити вираженість екстремальних змін у сигналі. В свою чергу розмах визначає амплітудний діапазон між піковими точками максимуму та мінімуму. Перцентильний аналіз, що передбачає розрахунок значень, нижче яких розташована певна частка відліків сигналу (зокрема 10-й та 90-й перцентилі), дає змогу оцінити структуру хвостів розподілу

Для автоматизації процесу обчислення зазначених параметрів для кожного фрагмента ЕКГ було розроблено алгоритм, реалізований у функції extract_statistical_metrics(X) у додатку Б.

4.2 Розробка та верифікація прогностичних моделей машинного навчання

У процесі проектування програмного комплексу для автоматизованої діагностики кардіологічних станів критичне значення має вибір релевантного

математичного апарату. З огляду на специфіку вхідних даних та багатогранність клінічних проявів, пріоритетним напрямом дослідження визначено застосування парадигми мультикласової класифікації. Даний підхід забезпечує можливість диференціації вхідних ЕКГ-сигналів за декількома категоріями, що відповідають як фізіологічній нормі, так і специфічним нозологічним формам серцево–судинних патологій.

Для порівняльного аналізу ефективності та вибору оптимальної моделі було сформовано перелік алгоритмів, що демонструють високу репрезентативність у задачах біомедичного аналізу: K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Machine, Naive Bayes Algorithm.

Програмна реалізація моделей на базі перелічених вище алгоритмів міститься у додатку В, а програмний код, для візуалізації матриці помилок у додатку Г.

4.2.1 Застосування методу K-Nearest Neighbor

Алгоритм методу K-Nearest Neighbor (K-NN) працює шляхом порівняння нового зразка з навчальними зразками на основі деякої «відстані» між ними. У контексті цього проекту це свідчить про те, що алгоритм шукатиме у навчальному наборі даних серцеві ритми, які найбільше нагадують ритм серця конкретного пацієнта. Наперед оцінити, наскільки цей алгоритм буде точним у цьому випадку, складно. З одного боку, серцевий ритм кожного пацієнта є унікальним, а з іншого боку, серцеві патології часто мають подібні ознаки.

Точність моделі, розробленої на основі K-NN на тестовому датасеті рівна 83,7% (рис. 4.5).

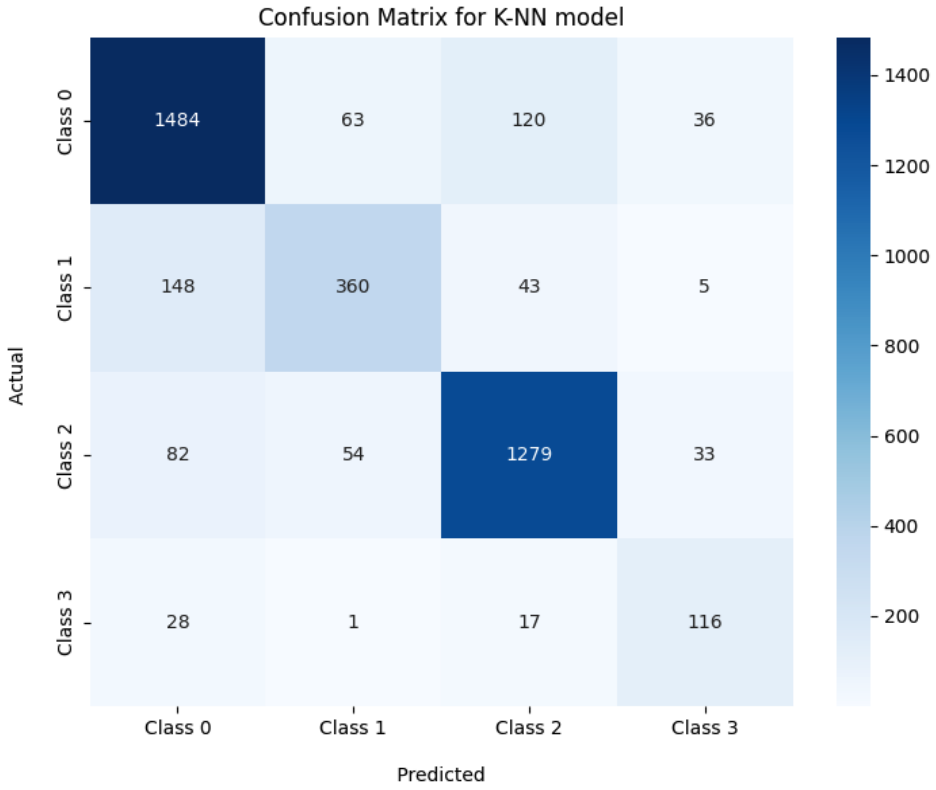

Рисунок 4.5 – Матриця помилок для алгоритму K–найближчих сусідів

Отриманий результат демонструє, що алгоритм успішно вловлює спільні патерни патологій, проте певна частка помилок може бути зумовлена саме надмірною чутливістю K-NN до індивідуальних особливостей сигналу, які метод іноді трактує як хибні ознаки схожості. Таким чином, попри високу загальну точність, обмеженням моделі залишається її залежність від щільності навчальної вибірки та складність у розрізненні дуже подібних за морфологією ритмів у граничних випадках.

4.2.2 Застосування дерева рішень

Дерево рішень являє собою структуру, яка моделює процес прийняття рішень, де кожен вузол відповідає за перевірку певної характеристики, а кожна гілка демонструє можливий результат перевірки.

Гіпотетично, такий підхід може забезпечити хороші результати, оскільки він здатен визначити, чи електричні сигнали в зубцях, інтервалах і сегментах відповідають нормі. Якщо dataset є достатньо масштабним і навчання проведено адекватно, дерево рішень здатне навіть відтворювати правила, що застосовуються в людській інтерпретації ЕКГ.

З певної перспективи можна вважати, що лікарі щодня використовують метод дерев рішень для трактування ЕКГ на рівні клінічного мислення. Наприклад, алгоритм може починати аналіз із перевірки наявності зубця P (визначення ритму), потім переходити до оцінки інтервалу RR: якщо він більший за норму, це вказує на брадикардію, якщо менший – на тахікардію. Подібна послідовна логіка робить цей алгоритм одним із найбільш природних для впровадження в системи медичної підтримки прийняття рішень.

Точність моделі розробленої на основі алгоритму Decision Tree на тестовому датасеті рівна 87,051% (рис. 4.6)

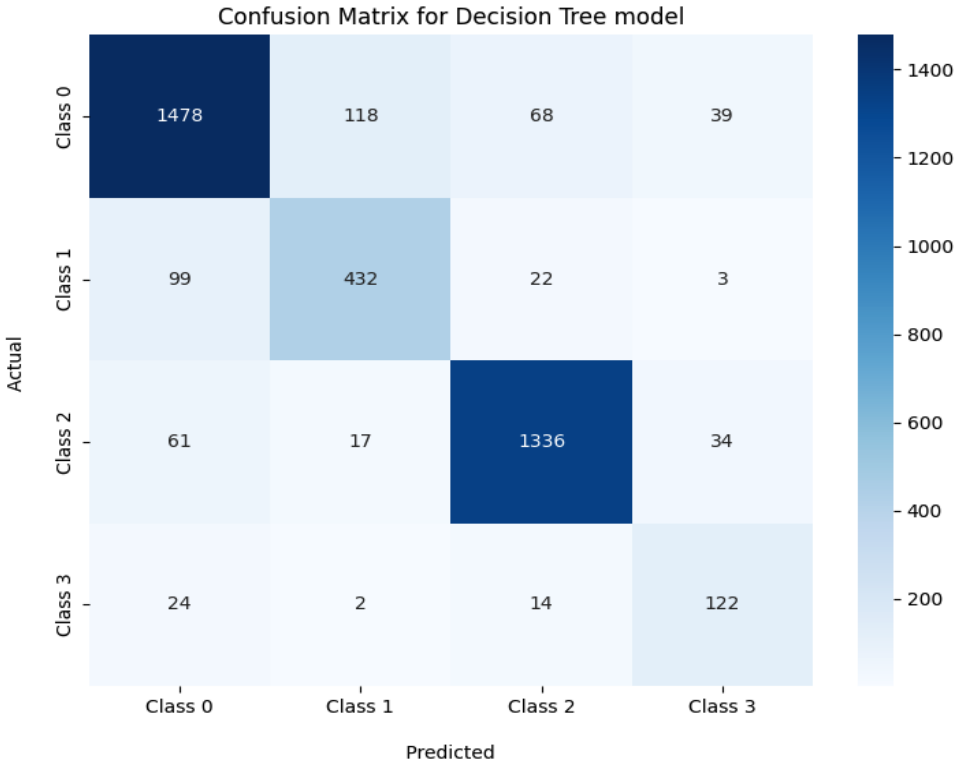

Рисунок 4.6 – Матриця помилок для алгоритму дерева рішень

З одного боку, такий високий показник точності підтверджує головну перевагу методу – його інтерпретованість, адже модель фактично вибудовує прозору логіку діагностики, максимально наближену до клінічного мислення лікаря. Проте, з іншого боку, дерева рішень схильні до перенавчання на специфічних шумах конкретного датасету, що може призвести до втрати гнучкості при аналізі нестандартних кардіограм, які не ідеально вписуються в раніше сформовані жорсткі правила

4.2.3 Застосування випадкового лісу

Випадковий ліс (Random Forest) – це класифікатор, що складається з кількох дерев рішень, які формуються для різних підмножин одного і того ж набору даних. Він використовує середнє значення з цих дерев для підвищення точності прогнозування. Алгоритм працює у два етапи: спочатку формують випадковий ліс, об’єднуючи N дерев рішень, а потім на основі цих дерев здійснюються прогнози.

Точність моделі, розробленої на основі методу Random Forest Classifier на тестовому датасеті рівна 93,2% (рис. 4.7)

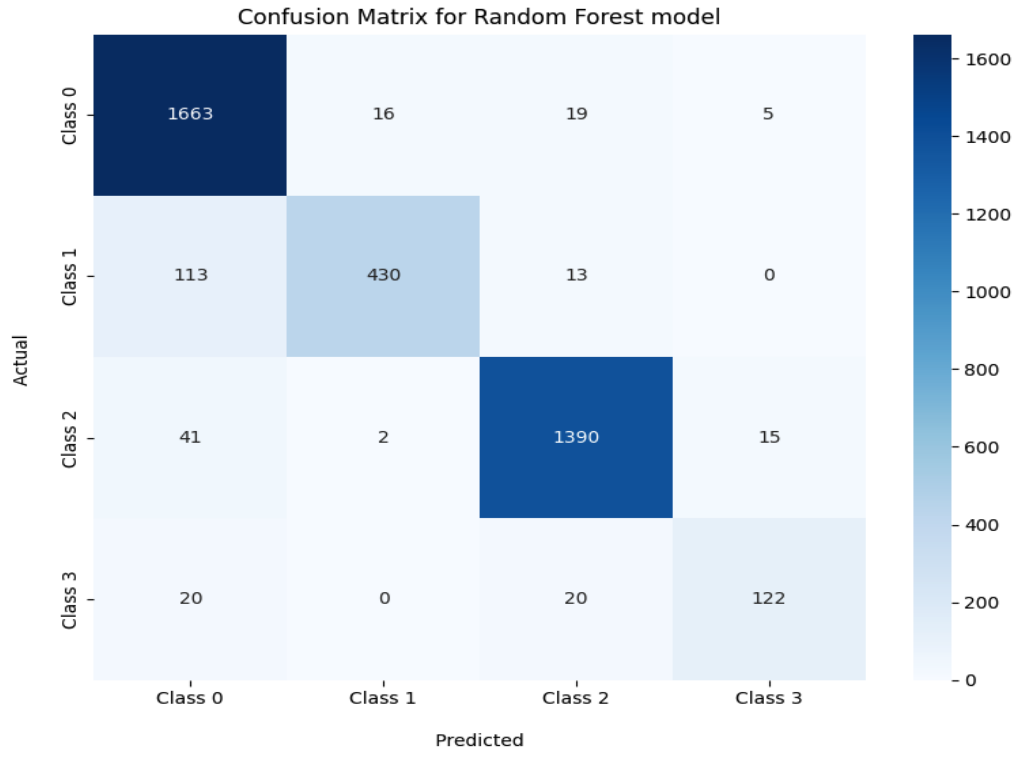

Рисунок 4.7 – Матриця помилок для алгоритму випадкового лісу

Високий показник точності підтверджує, що агрегація багатьох дерев дозволяє значно знизити ризик помилок, характерних для поодиноких моделей, та ефективно обробляти складні нелінійні залежності в ЕКГ-сигналах. Водночас зворотним боком такої ефективності є втрата простоти інтерпретації, адже логіка прийняття рішення стає менш прозорою для лікаря, а сама модель вимагає значно більше обчислювальних ресурсів для навчання та збереження порівняно з поодиноким деревом

4.2.4 Застосування методу опорних векторів

Метою алгоритму Support Vector Machine (SVM) є побудова оптимальної гіперплощини в n-вимірному просторі ознак, яка забезпечує максимальний відступ між об’єктами різних класів. Такий підхід дозволяє мінімізувати помилку узагальнення та ефективно класифікувати нові, раніше не опрацьовані дані, відносячи їх до відповідних категорій.

Я виділяю певні переваги застосування алгоритму SVM у межах дослідження. Такі як високу роздільну здатність у просторі ознак: Алгоритм демонструє високу ефективність при роботі з багатовимірними векторами даних. У контексті аналізу ЕКГ, де кожне спостереження містить 187 індикаторів напруги, SVM дозволяє ідентифікувати оптимальну межу рішення для точного диференціювання патологічних станів. Також адаптивність до багатокласової класифікації, тобто використання стратегій архітектури SVM дозволяє проводити комплексну діагностику, розпізнаючи декілька типів серцевих патологій одночасно, що є критично важливим для клінічного аналізу кардіограм. Обчислювальна ефективність та робота з опорними векторами є перевагою завдяки орієнтації на найбільш інформативні точки даних (опорні вектори), алгоритм забезпечує

стійкість моделі до надлишковості даних. Це гарантує стабільну швидкість обробки великих масивів інформації без втрати точності класифікації.

За результатами тестування на контрольній вибірці, точність (accuracy) розробленої моделі склала 89,6%. Візуалізацію результатів класифікації та матрицю помилок представлено на рис. 4.8.

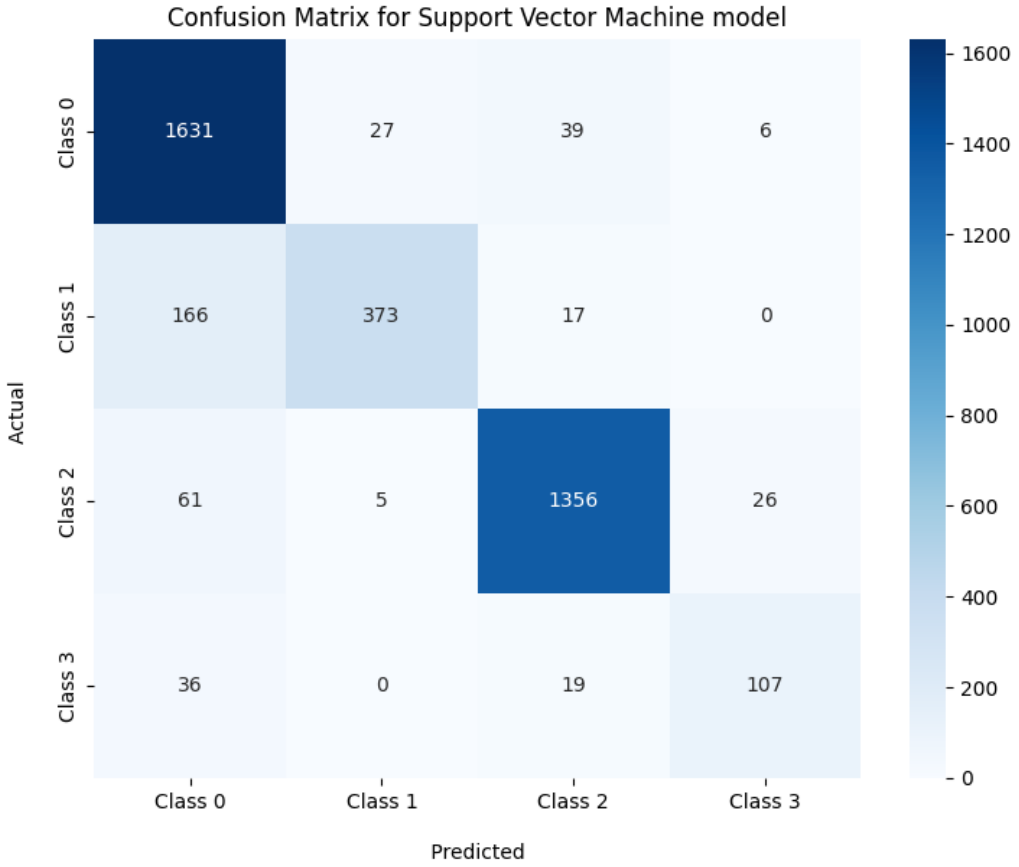

Рисунок 4.8 – Матриця помилок для методу опорних векторів

Такий високий показник точності підтверджує ефективність SVM у роботі з даними високої розмірності, оскільки алгоритм успішно знаходить оптимальну гіперплощину для розділення складних станів серцевого ритму навіть за наявності нелінійних залежностей. Проте варто враховувати, що метод є досить чутливим до

вибору параметрів ядра та потребує ретельної попередньої обробки даних, оскільки наявність значного шуму в сигналах ЕКГ може призвести до зміщення розділової межі та зниження загальної надійності класифікації.

4.2.5 Наївний Байєсівський класифікатор

В основі функціонування наївного баєсівського класифікатора лежить розрахунок імовірності того, що об’єкт стосується певного класу, який здійснюється згідно з теоремою Баєса. Головною особливістю цього алгоритму є припущення про повну автономність кожної ознаки відносно інших. Своєю чергою, під час тестування методу опорних векторів (Support Vector Machine) було зафіксовано рівень точності, що дорівнює 62,7% (рис. 4.9).

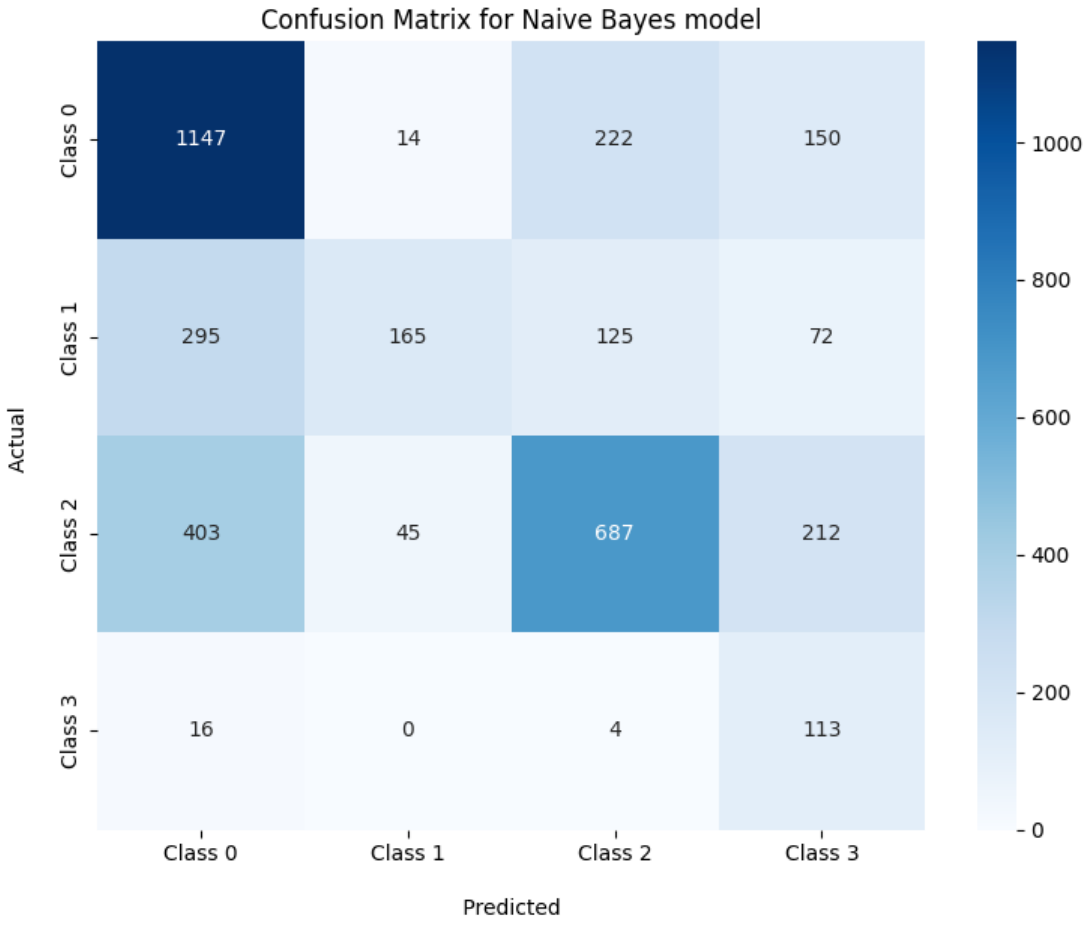

Рисунок 4.9 – Матриця помилок для Naïve Bayes алгоритму

Такий результат підкреслює головний недолік методу, а саме його надмірну «наївність», тобто припущення про повну незалежність ознак, що у випадку зі складними взаємозв'язками параметрів ЕКГ призводить до значної втрати прогностичної потужності. З іншого боку, цей алгоритм залишається найбільш невибагливим до обчислювальних потужностей і може слугувати базовим рівнем для порівняння, демонструючи, що для діагностики серцевих патологій прості імовірнісні моделі без урахування залежностей між симптомами є недостатніми.

4.3 Узагальнення результатів порівняння алгоритмів машинного навчання

Аналіз результатів моделювання підтвердив, що ефективність класифікації критично залежить від обраного набору ознак. Використання комбінації сирих даних з електродів ЕКГ разом із розрахованими клінічними показниками (RR-інтервалами та характеристиками QRS-комплексів) дозволило моделям краще ідентифікувати специфічні патології. Зокрема, параметри QRS-комплексу забезпечили точне розпізнавання шлуночкових порушень, тоді як варіабельність RR-інтервалів стала ключовим маркером для діагностики аритмій. Такий гібридний підхід до формування вектора ознак дозволив алгоритму Random Forest досягти пікової точності 93,2%, оскільки він зміг ефективно зважити значущість кожного амплітудно-часового параметра.

Незважаючи на високі показники окремих алгоритмів, таких як SVM (89,6%) та Decision Tree (87,05%), дослідження вказує на доцільність їхньої інтеграції в єдину систему. Комбінація методів дозволяє нівелювати індивідуальні недоліки моделей: наприклад, «прозорість» дерев рішень допомагає лікарю зрозуміти логіку виявлення відхилень у комплексі QRS, тоді як потужні ансамблеві методи підтверджують остаточний діагноз у складних клінічних випадках.

Для подальшого підвищення надійності діагностики було спроектовано ансамблеву модель на основі мажоритарного голосування (Majority Voting), яка об'єднує три найкращі алгоритми: Random Forest (93,2%), SVM (89,6%) та Decision Tree (87,051%). Математично, точність такої комбінації P_{ens} можна оцінити за формулою біноміального розподілу. Припускаючи, що помилки моделей є відносно незалежними, імовірність того, що ансамбль прийме правильне рішення (тобто принаймні дві з трьох моделей дадуть вірну відповідь), розраховується як:

$$P_{ens} = p_1 p_2 p_3 + (1 - p_1) p_2 p_3 + p_1 (1 - p_2) p_3 + p_1 p_2 (1 - p_3) \quad (4.2)$$

Підставивши значення точності (p_1=0,932; p_2=0,896; p_3=0,870), отримуємо теоретичну точність ансамблю на рівні 95,4%. Це на 2,2% вище за показник найкращого окремого алгоритму (Random Forest). Такий підхід дозволяє нівелювати випадкові помилки окремих класифікаторів: наприклад, якщо SVM помилково класифікує сигнал через шум, Random Forest та Decision Tree, що базуються на ієрархічних правилах, можуть скоригувати фінальний результат, забезпечуючи вищу стабільність системи у складних клінічних випадках.

Узагальнюючи отримані дані, можна стверджувати, що розроблений програмний комплекс демонструє високу адаптивність до реальних біомедичних сигналів. Завдяки акценту на найбільш інформативних ділянках кардіограми, система забезпечує надійну диференціацію між нормою та патологією. Це створює міцне підґрунтя для розробки систем підтримки прийняття медичних рішень, які працюють у режимі реального часу, полегшуючи процес первинної діагностики та підвищуючи точність інтерпретації електрокардіограм у клінічній практиці.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Нормативно-правове забезпечення охорони праці та безпеки у надзвичайних ситуаціях в ІТ сфері

Фундамент нормативно-правового регулювання питань безпеки життєдіяльності та охорони праці під час розробки системи автоматизованого аналізу ЕКГ базується на законодавчому базисі України. Дані норми є обов'язковими для виконання всіма суб'єктами господарювання, що займаються програмуванням та впровадженням інтелектуальних систем, незалежно від їхньої організаційно-правової форми. Ключовим документом у цій царині виступає Закон України «Про охорону праці». Він детермінує засади державної політики щодо створення безпечного робочого середовища, гарантує право розробників на захист їхнього здоров'я в процесі професійної діяльності, а також встановлює відповідальність роботодавця за дотримання стандартів безпеки.

Додатково правовідносини у цій сфері регулюються Кодексом законів про працю України, який закріплює обов'язок проведення систематичних інструктажів, навчання персоналу методам безпечної роботи та регламентує відповідальність за недотримання правил техніки безпеки. Для команди, що працює над алгоритмами машинного навчання, ці нормативи передбачають створення належних мікрокліматичних умов, відповідного рівня освітленості та гарантування електробезпеки при експлуатації обчислювальної техніки [16,18].

Специфічні особливості розробки програмних продуктів для аналізу медичних даних полягають у тривалій взаємодії фахівців із відеодисплейними терміналами, серверними потужностями та периферійним обладнанням. Головним спеціалізованим нормативним актом тут є Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями (що прийшли на зміну

НПАОП 0.00-1.31-99). Цей документ встановлює чіткі параметри організації робочих місць ІТ-спеціалістів, визначає оптимальні режими праці й відпочинку, а також вимоги до заземлення та вентиляції приміщень, де проводиться навчання нейромереж.

Важливе значення мають також санітарно-гігієнічні стандарти, зокрема ДсанПіН 3.3.2.007–98, які, попри свій вік, залишаються орієнтиром для встановлення допустимих рівнів випромінювання, тривалості безперервної роботи за монітором та ергономічного розміщення меблів. Ці критерії є критичними для запобігання професійним захворюванням програмістів та інженерів з підготовки даних для ЕКГ-моніторингу [15].

Окремим пріоритетом є забезпечення пожежної безпеки в приміщеннях, де розміщено високопродуктивне обладнання для обробки великих масивів кардіоданих та вузли безперебійного живлення. Правове регулювання цього аспекту здійснюється на основі «Правил пожежної безпеки в Україні» (НАПБ А.01.001–2014). Вони регламентують порядок утримання об’єктів, правила експлуатації електроустановок та вимоги до систем автоматичного пожежогасіння. Деталізовані вимоги до приміщень, де безпосередньо розробляється система та розміщуються сервери для аналізу ЕКГ, наведено у таблиці 5.1.

Оскільки розроблювана система автоматизованого аналізу ЕКГ може функціонувати на базі серверних центрів, дотримання цих норм забезпечує вибухо-пожежну стійкість інфраструктури. Це включає правильний підбір вогнегасників, коректне прокладання мережевих кабелів та встановлення сповіщувачів, що мінімізує ризик втрати критично важливих медичних даних у разі виникнення пожежі [16].

Таблиця 5.1 – Протипожежні вимоги до приміщень з високопродуктивними обчислювальними обладнаннями для обробки медичних даних

Параметр	Вимога згідно з НАПБ А.01.001–2014	Примітка для системи ЕКГ
Категорія приміщення	Категорія В (пожежонебезпечна)	Через наявність великої кількості кабелів та ПЕОМ.
Засоби гасіння	Вуглекислотні вогнегасники (типу ВВ)	Не пошкоджують електроніку при гасінні.
Сигналізація	Димові сповіщувачі	Найефективніші для виявлення перегріву кабелів.
Електроживлення	ІІ категорія надійності (UPS)	Забезпечує збереження медичних даних при відключенні.

Захист персоналу та технічних ресурсів у надзвичайних ситуаціях (НС) регулюється Кодексом цивільного захисту України. Даний кодекс визначає заходи щодо запобігання техногенним та природним катастрофам, а також алгоритми дій у разі збройних конфліктів чи аварій. Для організацій, що займаються розробкою інтелектуальних систем моніторингу здоров’я, це передбачає створення планів евакуації, забезпечення стійкості ІТ-інфраструктури до тривалих відключень енергопостачання та захист баз даних. У цьому контексті розроблена система автоматизованого аналізу ЕКГ набуває додаткової значущості: вона не лише виконує діагностичну функцію, а й має бути інтегрована у загальну схему безпеки медичного закладу, забезпечуючи безперервний моніторинг стану пацієнтів навіть в умовах надзвичайних ситуацій. Своєчасне виявлення критичних станів за допомогою алгоритмів машинного навчання дозволяє оперативно реагувати на загрози життю людей, що є невід’ємною частиною системи цивільного захисту в сучасних умовах.

5.2 Аналіз умов праці розробника інтелектуальної системи аналізу ЕКГ та ідентифікація шкідливих виробничих факторів

Процес розробки системи автоматизованого аналізу електрокардіограм передбачає тривалу роботу фахівця (інженера-програміста, спеціаліста з Machine Learning) у середовищі, насиченому електронно-обчислювальною технікою. Специфіка діяльності полягає у високому когнітивному навантаженні, необхідності візуального контролю за процесом навчання нейромереж та роботі з великими масивами графічної медичної інформації.

Згідно з ГОСТ 12.0.003-74, на розробника протягом робочої зміни впливає комплекс небезпечних та шкідливих виробничих факторів (НШВФ), які можна класифікувати за групами, такими як фізичні фактори, психофізіологічні фактори та хімічні фактори [16].

Параметри мікроклімату: Оскільки навчання моделей машинного навчання вимагає значних обчислювальних ресурсів (відеокарт, серверів), обладнання виділяє значну кількість надлишкового тепла. Порушення температурного режиму або недостатня вентиляція призводять до зниження працездатності та швидкої втомлюваності [18].

Освітленість: Робота з графіками ЕКГ вимагає високої точності візуального сприйняття. Недостатнє освітлення, наявність відблисків на екрані монітора або надмірна яскравість спричиняють астенопію (зорову втому).

Електромагнітні поля та випромінювання: Експлуатація системних блоків, моніторів та джерел безперебійного живлення створює електромагнітне поле, тривалий вплив якого може погіршувати стан нервової та серцево-судинної систем.

Небезпека ураження електричним струмом: Робота з обладнанням під напругою 220 В при порушенні ізоляції або відсутності заземлення створює пряму загрозу життю.

Зорове напруження: Зумовлене потребою чітко розрізняти дрібні елементи кардіограми (такі як зубці R, Q, P, S, T) та переглядати вихідний код програми.

Інтелектуальні та емоційні перевантаження: Висока складність розробки алгоритмів, необхідність прийняття відповідальних рішень (оскільки помилка алгоритму може вплинути на діагностику пацієнта) та тривала концентрація уваги.

Гіподинамія та вимушена робоча поза: Тривале перебування у сидячому положенні призводить до застійних явищ у кровообігу та захворювань опорно-рухового апарату.

Хоча ІТ-сфера не пов’язана з прямим використанням хімікатів, в робочій зоні може спостерігатися підвищена концентрація озону, оксидів азоту та пилу, що накопичується в системних блоках через електростатичний ефект, що може подразнювати дихальні шляхи.

Таким чином, для мінімізації негативного впливу ідентифікованих факторів під час розробки системи аналізу ЕКГ, необхідно передбачити заходи з раціональної організації робочого місця, дотримання режиму праці та відпочинку, а також впровадження засобів колективного та індивідуального захисту.

5.3 Організаційно-технічні заходи із забезпечення безпечних умов експлуатації обчислювальних засобів та систем безперебійного живлення

Для забезпечення безпечної роботи під час супроводження та експлуатації системи автоматизованого аналізу ЕКГ необхідно впровадити комплекс організаційних заходів. Оскільки система базується на алгоритмах машинного навчання, особлива увага приділяється стабільності електроживлення та запобіганню аварійним ситуаціям в серверних приміщеннях і на робочих місцях розробників.

Експлуатація серверів для навчання моделей та систем безперебійного живлення (ДБЖ) вимагає суворого дотримання «Правил технічної експлуатації

електроустановок споживачів». Перелік основних контрольних заходів, терміни їх виконання та розподіл відповідальності наведено в таблиці 5.2.

Таблиця 5.2 – План-графік заходів з технічної безпеки та гігієни праці

Об’єкт контролю	Захід	Періодичність	Відповідальна особа
Електрообладнання	Перевірка цілісності заземлення та ізоляції	1 раз на рік	Інженер з охорони праці
ДБЖ та акумулятори	Візуальний огляд на предмет здуття/окислення	Щомісяця	Системний адміністратор
Серверне обладнання	Моніторинг температури CPU/GPU	Безперервно (автоматично)	Програмний модуль захисту
Робоче місце	Регламентовані перерви (10–15 хв)	Кожні 2 години	Розробник/ тестувальник
Виробниче середовище	Вологе прибирання та провітрювання	Щоденно	Технічний персонал

Усе обладнання (системні блоки, монітори, ДБЖ) повинно мати надійне захисне заземлення відповідно до ДСТУ. Це запобігає ураженню персоналу струмом у разі пробою ізоляції на корпус.

Акумуляторні батареї джерел безперебійного живлення мають проходити регулярний огляд. Необхідно контролювати відсутність здуття корпусів та окислення клем, оскільки це може призвести до короткого замикання або виділення шкідливих газів.

До технічного обслуговування серверної інфраструктури допускаються особи, що пройшли навчання та мають відповідну групу з електробезпеки (не нижче III для обслуговуючого персоналу).

Для мінімізації впливу шкідливих факторів, ідентифікованих у попередньому розділі, впроваджуються наступні заходи.

Режим праці та відпочинку: згідно з нормами, при роботі з ЕОМ встановлюються регламентовані перерви по 10–15 хвилин через кожні 2 години роботи. Це критично важливо при аналізі великих масивів графіків ЕКГ для запобігання зоровій перевтомі [16].

Ергономіка: використання крісел з регульованою висотою та кутом нахилу спинки, а також розміщення монітора на відстані 600–700 мм від очей (нижче рівня очей на 10–20°).

Оскільки обробка ЕКГ-даних у реальному часі вимагає безперервної роботи обладнання, ризик перегріву є підвищеним:

Автоматичний моніторинг: впровадження програмно-апаратних засобів контролю температури процесорів (CPU) та графічних прискорювачів (GPU). У разі критичного перегріву система повинна автоматично завершувати обчислення.

Засоби пожежогасіння: приміщення, де розміщено сервери з медичними базами даних, мають бути укомплектовані вуглекислотними вогнегасниками (типу ВВ), які не пошкоджують електроніку при гасінні.

Вентиляція: для відведення тепла від потужних обчислювальних вузлів необхідно забезпечити припливно-витяжну вентиляцію або систему кондиціонування, що підтримує температуру в межах 18 - 20°С та вологість 40 - 60%.

Вологе прибирання: проводиться щоденно для зменшення концентрації пилу, що накопичується під впливом статичної електрики від обладнання.

5.4 Технічні заходи щодо покращення умов праці, зміцнення безпеки та алгоритми дій у надзвичайних ситуаціях

Для забезпечення стабільної роботи системи автоматизованого аналізу ЕКГ та мінімізації ризиків для персоналу й інфраструктури, розроблено наступний комплекс технічних рішень:

З метою зниження зорового та нервового напруження фахівця, що працює з біосигналами, передбачено використання високоякісних моніторів із матрицями типу IPS або OLED, частотою оновлення не менше 75 Гц та підтримкою технологій фільтрації синього світла, що є критично важливим при тривалій візуальній інтерпретації кардіограм, а сама суть впровадження системи на основі алгоритмів машинного навчання виступає заходом з охорони праці, оскільки автоматичне виділення критичних аномалій на записі ЕКГ суттєво знижує когнітивне навантаження на лікаря. Для забезпечення електробезпеки та безперебійності обчислень, що можуть тривати годинами, застосовуються джерела безперебійного живлення онлайн-типу для нівелювання стрибків напруги та коректного збереження стану нейромережі, а також встановлюються пристрої захисного відключення та автоматичні вимикачі, що спрацьовують при струмах витоку понад 30 мА. Заходи на випадок надзвичайних ситуацій включають систему моніторингу температури обладнання, яка при досягненні критичних показників у 85°C на ядрі графічного процесора ініціює безпечне завершення процесів, та оснащення приміщень автономними системами газового або порошкового пожежогасіння, що не пошкоджують електроніку. У разі раптового припинення електропостачання механізм автоматичного збереження прогресу обробки кожні 5 хвилин запобігає втраті результатів аналізу, тоді як в умовах воєнного стану чи техногенних аварій захищені канали зв’язку та географічне резервування даних на хмарних серверах дозволяють персоналу працювати дистанційно з укриттів, гарантуючи збереження інформації навіть при фізичному руйнуванні локальної інфраструктури.

ВИСНОВКИ

У межах даного дослідження мною було проведено комплексний аналіз можливостей застосування сучасних методів машинного навчання для автоматизованої інтерпретації електрокардіографічних записів. Робота охопила теоретичні засади електрокардіографії, огляд існуючих підходів до автоматизації, методологію машинного навчання, практичну реалізацію та всебічну оцінку ефективності різних алгоритмів класифікації.

Основні досягнення та наукові результати дослідження:

1. Теоретичний внесок: Було проведено систематичний огляд існуючих підходів до автоматизованої інтерпретації ЕКГ, що дозволило визначити найбільш перспективні напрямки дослідження та обґрунтувати вибір методів машинного навчання для практичної реалізації.

2. Методологічний внесок: Розроблено комплексну методологію виділення та обробки ключових ознак з ЕКГ-сигналів, включаючи інтервали RR, комплекси QRS та статистичні параметри (середнє значення, медіана, стандартне відхилення, асиметрія, ексцес, діапазон, перцентилі). Ці ознаки значно покращили якість класифікації порівняно з використанням лише сирих даних, що підтверджує важливість доменних знань при роботі з медичними даними.

3. Практичний внесок: Було протестовано п’ять різних алгоритмів машинного навчання (KNN, Decision Tree, Random Forest, SVM, Naive Bayes) на великому наборі даних, що містить понад 109 тисяч записів ЕКГ від 47 пацієнтів, розподілених на 5 класів. Це дозволило провести об’єктивне порівняння ефективності різних підходів з використанням комплексного набору метрик якості.

4. Найвищу точність (93,2%) продемонстрував алгоритм Random Forest, що підтверджує переваги ансамблевих методів при роботі зі складними біомедичними сигналами. Також високі результати показали SVM (89,6%) та Decision Tree (87,1%).

5. Підраховано, що застосування ансамблевого підходу на основі мажоритарного голосування (Majority Voting) дозволяє підвищити загальну точність діагностики до 95,4%. Це підтверджує гіпотезу про те, що комбінація моделей з різною архітектурою (Random Forest, SVM та Decision Tree) мінімізує специфічні помилки кожної з них, забезпечуючи вищу стабільність системи при аналізі зашумлених або нетипових клінічних сигналів.

Отримані результати підтверджують гіпотезу про доцільність застосування методів машинного навчання як допоміжного інструменту в системах підтримки прийняття клінічних рішень. Високий рівень точності моделей дозволяє суттєво автоматизувати процес інтерпретації електрокардіограм, мінімізуючи вплив людського фактору та прискорюючи діагностику, що особливо важливо в умовах підвищеного навантаження на медичну систему.

Проте важливо зазначити, що розроблена система не призначена для повної заміни лікаря-кардіолога, а слугує як допоміжний інструмент, що допомагає швидше виявляти потенційні проблеми та зосереджувати увагу фахівця на найбільш критичних випадках. Остаточне рішення щодо діагнозу та лікування завжди повинно прийматися кваліфікованим медичним фахівцем на основі комплексної оцінки стану пацієнта.

Для подальшого вдосконалення системи необхідні додаткові дослідження, спрямовані на:

1. Розширення набору даних за рахунок включення записів від пацієнтів різного віку, статі, етнічної приналежності та з різними коморбідними станами для підвищення узагальнюючої здатності моделей.

2. Дослідження більш складних архітектур, зокрема глибоких нейронних мереж (Deep Learning), які можуть показати ще вищу точність завдяки здатності автоматично виявляти складні залежності в даних.

3. Розширення набору використаних ознак за рахунок додавання частотних характеристик сигналу (через перетворення Фур’є), вейвлет-аналізу або часово-частотних представлень.

4. Проведення клінічної валідації розроблених моделей шляхом порівняльного тестування результатів автоматизованої інтерпретації з експертними оцінками кардіологів.

5. Розробку зручного користувацького інтерфейсу для практичного застосування та інтеграцію з існуючими медичними інформаційними системами.

6. Детальний аналіз помилок класифікації для виявлення систематичних проблем та покращення моделей.

Проведене дослідження демонструє значний потенціал застосування методів машинного навчання для автоматизованої інтерпретації електрокардіографічних записів та відкриває перспективи для подальшого розвитку цього напрямку в медичній діагностиці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гемптон Дж., Гемптон Дж. Основи ЕКГ. 9-те вид. Київ : Всеукраїнське спеціалізоване видавництво «Медицина», 2020. 192 с.

2. Хемптон Дж., Едлем Д. ЕКГ у практиці. 7-ме вид. Київ : Всеукраїнське спеціалізоване видавництво «Медицина», 2020. 384 с.

3. Machine Learning. [Електронний ресурс]. Режим доступу: https://www.it.ua/knowledge–base/technology–innovation/machine–learning, дата звернення: жовт., 12. 2025.

4. Борне П., Баркін Я., Руз К. Книга Штучний інтелект і нейромережі. Моноліт–Bizz, 2024. 216 с.

5. Mac Namee B., D'Arcy A., Kelleher J. D. Fundamentals of Machine Learning for Predictive Data Analytics. 2nd ed. 2015.

6. ECG Heartbeat Categorization Dataset – версія датасета – 12.06.2018 р. [Електронний ресурс]. Режим доступу : https://www.kaggle.com/datasets/shayanfazeli/heartbeat, дата звернення : лип., 12. 2025.

7. Moody G. B. The impact of the MIT-BIH Arrhythmia Database . G. B. Moody, R. G. Mark. IEEE Engineering in Medicine and Biology Magazine. 2001.

8. Bishop C. M. Pattern Recognition and Machine Learning. New York : Springer–Verlag, 2006. 738 P.179–210.

9. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning: with Applications in Python. Springer, 2023. 607 P. 321–350.

10. McKinney W. Python for Data Analysis. 3rd ed. O'Reilly Media, 2022. 578 P. 135-160.

11. Oppenheim A. V., Schafer R. W. Discrete–Time Signal Processing. 3rd ed. Pearson, 2010. 1120 P. 165–510

12. Pedregosa F., et al. Scikit–learn: Machine Learning in Python: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

13. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень : навч. посіб. Запоріжжя : ЗНТУ, 2008. 341 Р. 82–105.

14. Moody G. B., Mark R. G. The impact of the MIT–BIH Arrhythmia Database. IEEE Engineering in Medicine and Biology Magazine. 2001. Vol. 20(3). P. 45–50.

15. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : НПАОП 0.00–7.15–18. Затв. Наказом Міністерства соціальної політики України № 207 від 14.02.2018: https://zakon.rada.gov.ua/laws/show/z0508–18#Text

16. Загальні вимоги стосовно забезпечення роботодавцями охорони праці працівників : НПАОП 0.00–7.11–12. Затв. Наказом МНС України № 67 від 25.01.2012: https://zakon.rada.gov.ua/laws/show/z0226–12#Text

17. Державні санітарні норми та правила «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» : Наказ МОЗ України № 248 від 08.04.2014: https://zakon.rada.gov.ua/laws/show/z0472–14#Text

18. Жидецький В. Ц. Основи охорони праці : підручник. Львів : Афіша, 2005. 351 с.

19. Черкасов В. Г., Кравчук С. Ю. Анатомія людини : підручник. Вінниця : Нова Книга, 2011. 640 с.

20. Чорнокульський С. Т. Анатомія внутрішніх органів (спланхнологія). Київ : Книга-плюс, 2020. 189 с.

21. Шевчук В. Г., Мороз В. М., Бєлан С. М. Фізіологія : підручник для студ. вищ. мед. навч. закладів. 2–ге вид. Вінниця : Нова Книга, 2015. 448 с.

22. Awan A. Performance Analysis of Machine Learning Algorithms for Heart Disease Prediction A. Awan et al. Journal of Physics: Conference Series. 2021. https://www.itmconferences.org/articles/itmconf/pdf/2021/05/itmconf_icacc2021_03007.pdf.

23. Luz E. ECG heartbeat classification: A systematic review E. Luz, W. Schwartz, L.A. e Silva, L.S. Menotti Neurocomputing. 2016 (updated 2020).

24. Sahoo S. On-chip Healthcare Diagnostics System using Wavelet-based ECG Signal Processing S. Sahoo, B. Kanungo, S. Behera, S. Sabut Expert Systems with Applications. 2020.

25. Chollet F. Deep Learning with Python / F. Chollet. 2nd ed. Shelter Island, NY : Manning Publications, 2021. 504 p.

26. Ismail Fawaz H. Deep learning for time series classification: a review / H. Ismail Fawaz, G. Forestier, I. D. Weber et al. // Data Mining and Knowledge Discovery. 2019 (updated 2023). Vol. 33, No. 4. P. 917-963. DOI: 10.1007/s10618-019-00619-1.

27. Geron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow / A. Geron. 3rd ed. Sebastopol, CA : O'Reilly Media, 2022. 856 p.

28. Zheng A. Feature Engineering for Machine Learning / A. Zheng, A. Casari. Sebastopol, CA : O'Reilly Media, 2018 (reprint 2022). 218 p.

ДОДАТКИ

ДОДАТОК А

Модуль попередньої обробки та візуалізації даних

У цьому додатку ми описуємо структуру вхідних міток, завантажуємо дані з CSV, розділяємо їх на вибірки (train/test), проводимо нормалізацію (StandardScaler) та реалізуємо інструмент для візуальної перевірки ЕКГ–сигналу.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.naive_bayes import MultinomialNB

# Dictionary for classification of heartbeat types
HEARTBEAT_TYPES = {
    0: 'Normal beat',
    1: 'Supraventricular premature beat',
    2: 'Premature ventricular contraction',
    3: 'Fusion of ventricular and normal beat',
    4: 'Unclassifiable beat'
}

class ECGDataProcessor:
```

```
    """Class for processing and preparing electrocardiographic data"""

    def __init__(self, sampling_period_ms=8):
        self.sampling_period = sampling_period_ms / 1000.0  # Convert to seconds

    def visualize_ecg_recording(self, data_path, sample_idx=0):
        dataset = pd.read_csv(data_path)
        class_label = dataset.iloc[sample_idx, 187]
        heartbeat_category = HEARTBEAT_TYPES.get(class_label, 'Unknown')
        signal_data = dataset.iloc[sample_idx, 0:186].values
        cleaned_signal = np.trim_zeros(signal_data, 'b')
        time_points = np.arange(len(cleaned_signal)) * self.sampling_period * 1000

        plt.figure(figsize=(12, 4))
        plt.plot(time_points, cleaned_signal, linewidth=1.5)
        plt.grid(True, alpha=0.3, linestyle='—')
        plt.title(f'Electrocardiogram: {heartbeat_category}', fontsize=14, fontweight='bold')
        plt.xlabel('Time (ms)', fontsize=12)
        plt.ylabel('Voltage (V)', fontsize=12)
        plt.tight_layout()
        plt.show()

    def prepare_dataset(self, data_path, target_column_idx=187, test_ratio=0.2,
random_seed=42):
        full_data = pd.read_csv(data_path, header=None)
        feature_columns = list(range(target_column_idx))
        X = full_data.iloc[:, feature_columns]
        y = full_data.iloc[:, target_column_idx]
```

```
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=test_ratio, random_state=random_seed, stratify=y
    )
    return X_train, X_test, y_train, y_test

def normalize_features(self, X_train, X_test):
    normalizer = StandardScaler()
    X_train_scaled = normalizer.fit_transform(X_train)
    X_test_scaled = normalizer.transform(X_test)
    return X_train_scaled, X_test_scaled, normalizer
```

ДОДАТОК Б

Модуль інтелектуального аналізу та вилучення ознак

Реалізуємо алгоритми цифрової обробки сигналів. Знаходимо R–піки, розраховуємо часові інтервали (RR, QRS) та обчислюємо статистичні метрики (асиметрія, ексцес, процентилі) для формування вектора ознак.

```
class ECGFeatureExtractor:
    """Class for extracting features from ECG signals"""

    def __init__(self, sampling_period_ms=8):
        self.sampling_period = sampling_period_ms / 1000.0

    def _clean_signal(self, signal_array):
        return np.trim_zeros(signal_array, 'b')

    def _locate_r_peaks(self, signal_data):
        cleaned = self._clean_signal(signal_data)
        midpoint = len(cleaned) // 2
        left_segment = cleaned[:midpoint]
        right_segment = cleaned[midpoint:]
        left_peak_idx = np.argmax(left_segment)
        right_peak_idx = np.argmax(right_segment) + midpoint
        return [right_peak_idx, left_peak_idx]

    def compute_rr_intervals(self, data_frame):
        rr_values = []
        for _, row in data_frame.iterrows():
            signal_values = row.values
            peak_positions = self._locate_r_peaks(signal_values)
            interval_samples = abs(peak_positions[0] – peak_positions[1])
```

```
            interval_seconds = interval_samples * self.sampling_period
            rr_values.append(interval_seconds)
        return np.array(rr_values)

    def compute_qrs_complexes(self, data_frame, r_peak_indices):
        qrs_durations = []
        for idx, (_, row) in enumerate(data_frame.iterrows()):
            signal_values = self._clean_signal(row.values)
            r_position = r_peak_indices[idx][0]
            baseline = np.median(signal_values)
            q_start = self._find_q_wave_start(signal_values, r_position, baseline)
            s_end = self._find_s_wave_end(signal_values, r_position, baseline)
            duration_samples = s_end – q_start
            duration_seconds = duration_samples * self.sampling_period
            qrs_durations.append(duration_seconds)
        return np.array(qrs_durations)

    def _find_q_wave_start(self, signal, r_idx, baseline_level):
        idx = r_idx – 1
        found_peak = False
        while idx > 0:
            current_val = signal[idx]
            next_val = signal[idx + 1]
            if current_val > next_val: found_peak = True
            if found_peak and (current_val < next_val or current_val >= baseline_level):
return idx
            idx –= 1
        return 0
```

```
        def _find_s_wave_end(self, signal, r_idx, baseline_level):
            idx = r_idx + 1
            found_peak = False
            end_idx = len(signal) – 1
            while idx < len(signal):
                current_val = signal[idx]
                prev_val = signal[idx – 1]
                if current_val > prev_val: found_peak = True
                if found_peak and (current_val < prev_val or current_val >= baseline_level):
return idx
                idx += 1
            return end_idx

        def extract_statistical_metrics(self, data_frame):
            metrics_list = []
            for _, row in data_frame.iterrows():
                signal_values = self._clean_signal(row.values)
                signal_mean = np.mean(signal_values)
                signal_median = np.median(signal_values)
                signal_std = np.std(signal_values)
                signal_skew = scipy.stats.skew(signal_values)
                signal_kurt = scipy.stats.kurtosis(signal_values)
                p10 = np.percentile(signal_values, 10)
                p90 = np.percentile(signal_values, 90)
                metrics_list.append([signal_mean, signal_median, signal_std, signal_skew,
signal_kurt, p10, p90])
            return np.array(metrics_list)
```

ДОДАТОК В

Модуль машинного навчання

Описуємо процес навчання ансамблевих та класичних моделей (Random Forest, SVM, K–NN, Decision Treen, Random Forest, Naïve Bayes).

```
class MLModelTrainer:
    """Class for training and evaluating machine learning models"""

    def __init__(self):
        self.models = {}
        self.predictions = {}
        self.accuracies = {}

    def train_knn_classifier(self, X_train, y_train, n_neighbors=50):
        classifier = KNeighborsClassifier(n_neighbors=n_neighbors)
        classifier.fit(X_train, y_train)
        self.models['knn'] = classifier
        return classifier

    def train_decision_tree(self, X_train, y_train, random_seed=0):
        classifier = DecisionTreeClassifier(random_state=random_seed)
        classifier.fit(X_train, y_train)
        self.models['decision_tree'] = classifier
        return classifier

    def train_random_forest(self, X_train, y_train):
        classifier = RandomForestClassifier(n_estimators=100, random_state=42)
        classifier.fit(X_train, y_train)
        self.models['random_forest'] = classifier
        return classifier
```

```
def train_svm_classifier(self, X_train, y_train):
    classifier = svm.SVC(kernel='rbf', random_state=42)
    classifier.fit(X_train, y_train)
    self.models['svm'] = classifier
    return classifier

def train_naive_bayes(self, X_train, y_train):
    classifier = MultinomialNB()
    classifier.fit(X_train, y_train)
    self.models['naive_bayes'] = classifier
    return classifier

def evaluate_models(self, X_test, y_test):
    results = {}
    for model_name, model in self.models.items():
        predictions = model.predict(X_test)
        accuracy = model.score(X_test, y_test) * 100
        self.predictions[model_name] = predictions
        self.accuracies[model_name] = accuracy
        results[model_name] = {'accuracy': accuracy, 'predictions': predictions}
        print(f"Model {model_name} accuracy: {accuracy:.2f}%")
    return results

def get_model_accuracy(self, model_name):
    return self.accuracies.get(model_name, None)

def get_predictions(self, model_name):
    return self.predictions.get(model_name, None)
```

ДОДАТОК Г

Модуль оцінки якості

Реалізуємо функцію оцінки точності та візуалізації матриці помилок (Confusion Matrix) для контролю якості класифікації.

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

class MLModelTrainer:
  def create_confusion_matrix_plot(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=list(HEARTBEAT_TYPES.values()),
                yticklabels=list(HEARTBEAT_TYPES.values()))
    plt.title(f'Confusion Matrix: {model_name}', fontsize=14, fontweight='bold')
    plt.ylabel('True Classes', fontsize=12)
    plt.xlabel('Predicted Classes', fontsize=12)
    plt.tight_layout()
    plt.show()
```