Міністерство освіти і науки України
Криворізь кий національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня вищої освіти магістра

за спеціальністю 121 – Інженерія програмного забезпечення

На тему: Дослідження та розробка програмного забезпечення для аналізу гідроакустичних сигналів

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-23-1м _____ /А. О. Поляєв /

Керівник кваліфікаційної роботи _____ / Н.Х. Саітгареєв /

Економіко-організаційна частина _____ / О. В Шамрай /

Нормоконтроль _____ / Н.Х. Саітгареєв /

Завідувач кафедри _____ / А.М Стрюк /

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. Кафедри

_____ А. М. Стрюк

«___» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-23-1м Поляєву Антону Олександровичу

1. Тема: Дослідження та розробка програмного забезпечення для аналізу гідроакустичних сигналів затверджено наказом по КНУ № 277с від «15» квітня 2024 р.

2. Термін подання студентом закінченої роботи: «01» грудня 2024 р.

3. Вихідні дані по роботі:порівняльний аналіз методів штучного інтелекту для задачі класифікації гідроакустичних сигналів та розроблений додаток для інтеграції отриманої моделі, мінімальна точність моделі – 80%.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): виконати аналіз існуючих методів розв`язання задачі, розробити математичні моделі подання знань, спроектувати інтелектуальний програмний комплекс розпізнавання гідроакустичних сигналів, дослідити вплив різних методів та моделей розпізнавання гідроакустичних сигналів на роботу комплексу, виконати аналіз економічної ефективності розроблювального комплексу.

5. Перелік ілюстративного матеріалу: схеми та візуалізація досліджень, знімок екрану додатку.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Огляд літератури за тематикою та збір даних	25.04.2024 – 10.05.2024
2	Проведення порівняльного аналізу існуючих рішень аналізу гідроакустичних сигналів	11.05.2024 – 24.05.2024
3	Підготовка матеріалів для першого розділу роботи	25.05.2024 – 07.06.2024
4	Розробка програмного комплексу для аналізу гідроакустичних сигналів	08.06.2024 – 19.06.2024
5	Підготовка матеріалів для другого розділу роботи	20.06.2024 – 02.07.2024
6	Підготовка матеріалів для третього розділу роботи	03.07.2024 – 23.07.2024
7	Розробка веб-додатку для інтерактивної роботи з даними	24.07.2024 – 28.07.2024
8	Тестування та виправлення програмного комплексу	29.07.2024 – 18.08.2024
9	Підготовка матеріалів для четвертого розділу роботи	20.08.2024 – 10.10.2024
10	Розрахунок економічної ефективності	10.10.2024 – 25.10.2024
11	Оформлення пояснювальної записки	02.11.2024 – 29.11.2024

Дата видачі завдання: «09» лютого 2024 р.
Студент _____ / А.О. Поляєв /
Керівник роботи _____ / Н.Х. Саітгареєв /

РЕФЕРАТ

ПРОГРАМНИЙ КОМПЛЕКС АНАЛІЗУ ГІДРОАКУСТИЧНИХ СИГНАЛІВ, КЛАСИФІКЦІЯ ТА АНАЛІЗ СИГНАЛІВ СИГНАЛІВ.

Пояснювальна записка: 120 с., 1 дод., 14 рис., 5 табл., 14 джерел.

Метою кваліфікаційної роботи є дослідження та розробка програмного забезпечення для класифікації гідроакустичних сигналів, який підвищить ефективність та безпеку виявлення підводних об`єктів.

У роботі проведено аналіз сучасних методів класифікації гідроакустичних сигналів, розглянуто існуючі підходи та інструменти для їх реалізації. Було розглянуто різні методи та алгоритми для класифікації сигналів.

У результаті роботи розроблено програмний комплекс, що включає серверну частину для взаємодії з моделями машинного навчання та веб-додаток для користувачів. В процесі розробки було проведено тестування програмного забезпечення, що дозволило забезпечити коректність його роботи та відповідність вимогам проєкту.

Здійснено розрахунок собівартості розробки програмного комплексу, а також проведено оцінку економічної доцільності його впровадження. Результати показали, що впровадження розробленого комплексу дозволяє автоматизувати процес аналізу сигналів та значно зменшити витрати.

ABSTRACT

SOFTWARE SYSTEM FOR ANALYSIS OF HYDROACOUSTIC SIGNALS, CLASSIFICATION AND ANALYSIS OF SIGNALS.

Thesis in: 120 p., 1 app., 14 fig., 5 tab., 14 references.

The purpose of the qualification work is to research and develop software for classification of hydroacoustic signals, which will increase the efficiency and safety of detection of underwater objects.

The paper analyzes modern methods of classification of hydroacoustic signals, considers existing approaches and tools for their implementation. Different methods and algorithms for signal classification were considered.

As a result of the work, a software system was developed, including a server part for interaction with machine learning models and a web application for users. During the development process, the software was tested to ensure that it worked correctly and met the project requirements.

The cost of developing the software system was calculated, and the economic feasibility of its implementation was assessed. The results showed that the implementation of the developed complex allows automating the signal analysis process and significantly reducing costs.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРИНЦИПІВ ОБРОБКИ ТА РОЗПІЗНАВАННЯ ГІДРОАКУСТИЧНИХ СИГНАЛІВ 10
1.1 Аналіз теоретичних основ та проблеми розпізнавання гідроакустичних сигналів 10
1.2 Аналіз підходів до побудови та застосування систем штучного інтелекту для аналізу гідроакустичних сигналів 11
1.3 Аналіз сучасних програмних систем розпізнавання гідроакустичних сигналів, їх функціональні можливості та принципи дії 14
1.4 Актуальність розробки системи для аналізу гідроакустичних сигналів із застосуванням сучасних методів штучного інтелекту 16
1.5 Постановка основних задач розробки системи для обробки гідроакустичних сигналів 17
1.6 Розробка функціональної та структурної схеми програмного комплексу…. 19
2 РОЗРОБКА МАТЕМАТИЧНИХ МОДЕЛЕЙ ТА ОГЛЯД ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В РАМКАХ АНАЛІЗУ ГІДРОАКУСТИЧНИХ СИГНАЛІВ 22
2.1 Розробка математичних моделей для аналізу та обробки даних……..... 22
2.2 Детальний огляд методів штучного інтелекту для аналізу та обробки даних……..... 26
2.3 Порівняння методів штучного інтелекту для аналізу гідроакустичних сигналів 32
3 РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ АНАЛІЗУ ГІДРОАКУСТИЧНИХ СИГНАЛІВ……. 37
3.1 Порівняльний аналіз та вибір мови програмування і інструментів для розробки програмного комплексу 37
3.2 Попередня обробка даних 38

3.3 Реалізація моделей аналізу гідроакустичних сигналів 41
3.4 Розробка API для використання моделей 54
3.5 Розробка веб-додатку 55
3.6 Розробка інструкції користувача 60
4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ 62
4.1 Наукова новизна 62
4.2 Розробка методик проведення дослідження 63
4.3 Обробка результатів дослідження, аналіз адекватності моделей та оцінка технічної ефективності 65
4.4 Формулювання і узагальнення основних науково-технічних результатів дослідження 80
5 АНАЛІЗ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ІННОВАЦІЇ 82
5.1 Розрахунок собівартості програмної інновації 82
5.2 Розрахунок ефективності впровадження програмної інновації 86
ВИСНОВКИ 89
ПЕРЕЛІК ПОСИЛАНЬ 91
Додаток А 92

ВСТУП

Сучасні технології та розвиток цифрових методів обробки сигналів відкрили шлях до нових можливостей для досліджень у галузі аналізу гідроакустичних даних. Гідроакустичні сигнали є ключовими для підводної навігації, захисту морських кордонів та безпеки судноплавства. Зростання обсягів морських перевезень та необхідність захисту критичної інфраструктури обумовлюють потребу у створенні автоматизованих систем для ідентифікації підводних об`єктів, таких як: природні утворення та штучні об`єкти. Проте класифікація гідроакустичних сигналів є складною задачею через їх варіативність, що залежить від фізичних умов середовища, таких як: глибина, склад води, рельєф дна та наявність шумів. Це визначає актуальність досліджень, спрямованих на покращення аналізу таких сигналів. [1]

Сучасні методи машинного навчання широко використовуються для аналізу гідроакустичних даних, забезпечуючи автоматизовану обробку великих обсягів складних сигналів. Штучний інтелект дозволяє створювати системи з високою точністю обробки, які адаптуються до різних умов середовища. Проте все ще актуальним залишається покращення точності та адаптивності цих систем до характеристик об`єктів. У сучасних дослідженнях помітна тенденція до створення комплексних гібридних моделей, які поєднують класичні алгоритми класифікації з глибокими нейронними мережами та методами бустингу, що дозволяє ефективніше вирішувати задачі класифікації у реальному часі. [2, 3]

Актуальність роботи полягає в розробці адаптивних моделей, здатних аналізувати гідроакустичні сигнали з високою точністю та автоматизувати процес ідентифікації підводних об`єктів. Використання ШІ для аналізу гідроакустичних сигналів дозволяє знизити ризик помилок та підвищити точність обробки інформації в складних підводних середовищах.

Об`єктом дослідження є процес автоматизованого аналізу гідроакустичних сигналів для ідентифікації підводних об`єктів, що передбачає комплексний підхід до обробки даних.

Предметом дослідження є алгоритми машинного навчання для класифікації об`єктів як «камінь» або «міна» на основі даних з сонару, зосереджуючи увагу на методах, що дозволяють автоматизувати та оптимізувати цей процес із максимальною точністю.

Мета роботи - дослідження та розробка програмного забезпечення для класифікації гідроакустичних сигналів, який підвищить ефективність та безпеку виявлення підводних об`єктів. Результати роботи можуть бути застосовані в системах підводного моніторингу, судноплавстві, військовій галузі та екологічному моніторингу морських екосистем. Основні завдання роботи включають: аналіз сучасних методів машинного навчання для класифікації гідроакустичних сигналів; визначення оптимальних моделей для їх класифікації; розробку програмного забезпечення, яке аналізує дані.

Таким чином, розробка та дослідження методів для автоматизації аналізу гідроакустичних сигналів є важливим завданням, яке сприятиме підвищенню ефективності систем підводного моніторингу. Використання ШІ дозволяє покращити точність і швидкість ідентифікації об`єктів, що є важливим для безпеки та навігації. Запропонована система, що поєднує сучасні алгоритми машинного навчання, може внести значний вклад у розвиток підводних технологій та забезпечити новий рівень автоматизації та надійності для вирішення завдань підводного аналізу.

1 АНАЛІЗ ПРИНЦИПІВ ОБРОБКИ ТА РОЗПІЗНАВАННЯ ГІДРОАКУСТИЧНИХ СИГНАЛІВ

1.1 Аналіз теоретичних основ та проблеми розпізнавання гідроакустичних сигналів

Гідроакустичні сигнали є важливим джерелом інформації для вивчення підводного середовища, моніторингу морського транспорту, досліджень морської флори та фауни, а також для вирішення завдань оборонного характеру. Їх особливістю є здатність до поширення на значні відстані у водному середовищі, що робить їх цінними для задач, які потребують дистанційного спостереження та збору даних у воді. Гідроакустичні сигнали формуються за рахунок генерації та передачі звукових хвиль, які залежать від ряду фізичних характеристик: частоти, амплітуди, швидкості розповсюдження у воді та здатності до відбиття від перешкод.

Середовище поширення таких сигналів характеризується значною відмінністю від повітряного, зокрема через вищу швидкість звуку у воді, яка в середньому становить близько 1500 м/с. На швидкість поширення також впливають такі чинники, як температура, тиск і солоність, що ускладнює точне вимірювання параметрів сигналу. Крім того, розповсюдження сигналів супроводжується рефракцією, розсіюванням і поглинанням, що впливає на їх форму та інтенсивність. Процес розпізнавання гідроакустичних сигналів вимагає обліку їх спектрального складу, тривалості, частоти повторюваних імпульсів і рівня сигналу щодо рівня шуму. Зміна цих характеристик залежить як від джерел сигналів (підводні човни, риба, природні об`єкти), так і від умов середовища, що часто призводить до значного спотворення. [1, 4]

Процес аналізу гідроакустичних сигналів супроводжується низкою труднощів, обумовлених природними особливостями підводного середовища та технологічними обмеженнями. Однією з основних проблем є вплив шумів, які можуть маскувати цільові сигнали. Підводний шум створюється як за допомогою природних джерел, наприклад: вітри, хвилі, підводні вулкани,

тварини, так і антропогенних, наприклад: морський транспорт, будівельна активність, тощо. Високий рівень шуму значно ускладнює виділення корисних сигналів та підвищує ймовірність помилкової класифікації.

Крім того, мінливість підводного середовища змінює властивості сигналів залежно від температури, глибини, солоності і наявності об`єктів, які можуть створювати відбиття або додаткові шуми. Сигнали також можуть змінюватися з часом через нестабільність середовища, що ускладнює їх розпізнавання в реальному часі. Зрештою, обробка великих обсягів даних, характерних для таких досліджень, вимагає високої обчислювальної потужності. Інструменти для аналізу гідроакустичних сигналів повинні обробляти дані швидко і точно, тому для ефективного виділення та класифікації сигналів використовуються сучасні методи машинного навчання.

Складність обробки гідроакустичних сигналів обумовлює актуальність розвитку спеціалізованого програмного забезпечення, яке може вирішувати проблеми автоматизації аналізу, підвищення точності і швидкості обробки даних у водному середовищі. Впровадження таких інструментів є важливим кроком для покращення якості моніторингу та безпеки в морському середовищі, наукових досліджень і вирішення прикладних задач, що потребують обробки даних гідроакустичних сигналів.

Таким чином, розпізнавання гідроакустичних сигналів є складним завданням, яке вимагає застосування різних методів обробки та інтелектуальних алгоритмів для ефективного виділення та класифікації сигналів у водному середовищі, що обумовлює актуальність подальших досліджень у цій сфері.

1.2 Аналіз підходів до побудови та застосування систем штучного інтелекту для аналізу гідроакустичних сигналів

Системи штучного інтелекту (ШІ) є важливим інструментом для аналізу гідроакустичних сигналів, оскільки дозволяють автоматизувати обробку великих обсягів даних і значно підвищують точність розпізнавання підводних

об`єктів та явищ. Використання ШІ в цій галузі зумовлене необхідністю швидкої й надійної обробки інформації, яка надходить з різних джерел і ускладнена шумами підводного середовища. Основні методи, що використовуються в аналізі гідроакустики, включають глибоке навчання, класичне машинне навчання та стандартні методи обробки сигналів. Кожен з них має свої переваги й обмеження, які впливають на загальну ефективність системи.

Для розуміння можливостей та обмежень застосування ШІ в аналізі гідроакустики важливо врахувати природу гідроакустичних сигналів. Вони являють собою звукові коливання, що поширюються у воді, несучи інформацію про джерело звуку або характеристики підводного середовища. Завдяки фізичним властивостям води, сигнал може поширюватися на великі відстані, що робить його цінним для моніторингу. Водночас складні умови середовища, такі як рефракція, розсіювання та поглинання, можуть спотворювати сигнал і ускладнювати точне розпізнавання джерел. Це створює особливі виклики для системи ШІ, яка повинна враховувати ці спотворення для надійної обробки та класифікації сигналів.

Одним із найперспективніших підходів для вирішення цих проблем є використання глибокого навчання, зокрема згорткових нейронних мереж (CNN). Завдяки здатності автоматично виділяти специфічні ознаки сигналів, CNN ефективно працюють із двовимірними представленнями сигналів, такими як спектрограми, що показують їхні частотні складові в часі. Це дозволяє нейромережам навчатися розпізнавати різні типи сигналів (наприклад, природні звуки або антропогенні шуми), використовуючи великі обсяги даних. Однак CNN мають значну обчислювальну складність, що може бути проблемою для роботи в реальному часі, коли потрібна миттєва обробка сигналів.

Крім глибокого навчання, для аналізу гідроакустичних сигналів часто застосовують класичні методи машинного навчання, такі як Random Forest, підтримка векторних машин (SVM) та XGBoost. Ці алгоритми відрізняються

своєю стійкістю та здатністю обробляти великі масиви даних, що є корисним для роботи з шумними підводними сигналами. Наприклад, SVM має високу точність при роботі з шумними даними, а Random Forest ефективно запобігає перевчанню. Проте класичні методи потребують ретельного налаштування параметрів і попередньої обробки даних для адаптації до особливостей гідроакустичних сигналів.

Важливим етапом у побудові системи ШІ для гідроакустичного аналізу є попередня обробка даних, яка включає фільтрацію, нормалізацію та виділення характеристик. Фільтрація допомагає знизити рівень шуму, що підвищує якість даних і дозволяє моделі ШІ точніше визначати ознаки сигналу. Нормалізація забезпечує стабільність сигналу, а виділення специфічних характеристик (наприклад, спектральних і часових компонентів) допомагає зосередитися на найбільш значущих аспектах для класифікації. Без якісної попередньої обробки даних точність і надійність системи ШІ можуть бути значно знижені.

Особливий інтерес у задачах аналізу гідроакустичних сигналів становлять гібридні моделі, які поєднують у собі переваги глибокого і класичного машинного навчання. Наприклад, CNN можна використовувати для виділення складних патернів, а такі алгоритми, як SVM або Random Forest, виконують роль кінцевих класифікаторів, що підвищує загальну точність і швидкість обробки даних. Такий підхід є особливо корисним у режимі реального часу, де важлива висока продуктивність у поєднанні з точністю розпізнавання.

Таким чином, застосування систем штучного інтелекту для аналізу гідроакустичних сигналів є перспективним напрямом, який вимагає комплексного підходу та адаптації під специфіку підводного середовища. Різні методи, такі як глибоке навчання, класичне машинне навчання та гібридні моделі, дозволяють знаходити оптимальні рішення для задач аналізу сигналів, забезпечуючи високу точність і надійність результатів навіть у складних умовах середовища.

1.3 Аналіз сучасних програмних систем розпізнавання гідроакустичних сигналів, їх функціональні можливості та принципи дії

Аналіз сучасних програмних систем розпізнавання гідроакустичних сигналів є важливим для розуміння функціональних можливостей, алгоритмічної основи та принципів дії, які застосовуються у цій галузі. Сучасні системи охоплюють широкий спектр завдань, включаючи виявлення підводних об`єктів, моніторинг екосистем, діагностику підводної інфраструктури та вивчення підводного середовища. Усі вони побудовані на основі передових алгоритмів обробки сигналів та використовують різні підходи до автоматизації аналізу.

Одним із найбільш технологічно розвинених напрямків є військові сонарні системи, наприклад, AN/SQQ-89, що використовується ВМС США. Їх функціональні можливості включають виявлення та класифікацію підводних човнів, морських мін та інших об`єктів, що є критично важливими для забезпечення безпеки. Система працює у двох режимах: активному (емісія сигналів і прийом відбитих хвиль) та пасивному (аналіз навколишніх шумів). Програмне забезпечення таких систем використовує алгоритми класифікації для відокремлення корисних сигналів від шуму і забезпечення високої точності навіть в умовах складного акустичного середовища.

Системи для моніторингу морських екосистем, такі як BioSonics DT-X та Fish Hunter PRO, також мають широкий спектр функцій. Вони дозволяють розпізнавати звуки морської фауни та класифікувати їх за видами. Принцип дії цих систем базується на аналізі акустичних патернів, характерних для окремих видів тварин. Це дає можливість екологам оцінювати стан популяцій та отримувати дані про біорізноманіття в досліджуваних районах. Функціонал таких систем передбачає автоматичне формування звітів про чисельність тварин, їх розташування та поведінку.

Серед програмних систем, які забезпечують моніторинг підводних інфраструктур, слід виділити Kongsberg EM 2040 і Teledyne RESON SeaBat. Їх основна функція полягає у виявленні дефектів у трубопроводах, кабелях та

інших підводних об`єктах. Завдяки використанню багатопроменевих ехолотів, ці системи можуть створювати детальні тривимірні моделі морського дна та підводних споруд. Програмне забезпечення аналізує відбиті сигнали, визначаючи параметри об`єктів і виявляючи пошкодження. Це дає змогу швидко реагувати на можливі ризики та мінімізувати витрати на ремонт.

Дослідницькі системи, як-от Edgetech 6205 і Innomar SES-2000, орієнтовані на вивчення підводного середовища. Їхні функції включають дослідження геологічної структури морського дна, аналіз шарів ґрунту та виявлення сейсмічної активності. Програмне забезпечення таких систем дозволяє створювати високоточні зображення дна за допомогою аналізу відбитих сигналів. Особливістю цих систем є здатність розпізнавати аномалії в структурі дна, що може бути корисним для прогнозування геологічних явищ або визначення місць для інженерних робіт.

Комерційні сонарні системи для риболовецьких флотів, такі як Simrad ES70 та Furuno CHIRP, спрямовані на виявлення риби та оцінку її кількості. Ці системи використовують технологію CHIRP, яка забезпечує широкий спектр частот для підвищення роздільної здатності сигналу. Програмне забезпечення таких систем аналізує щільність зграї, розмір риб та їх розташування, дозволяючи оптимізувати риболовецькі операції. Вбудовані алгоритми допомагають визначати найбільш вигідні місця для вилову, що сприяє зменшенню витрат і підвищенню ефективності.

Наукові системи, такі як Wildlife Acoustics Song Meter та Aquarian Audio H2a Hydrophone, спеціалізуються на вивченні звуків морських ссавців. Їх функціонал дозволяє реєструвати звуки китів, дельфінів та інших морських тварин, а також відокремлювати їх від техногенних шумів. Програмне забезпечення таких систем виконує класифікацію звуків, що дає можливість дослідникам отримувати цінну інформацію про поведінку морської фауни та оцінювати вплив людської діяльності на підводне середовище.

Отже, провівши аналіз сучасних програмних систем для розпізнавання гідроакустичних сигналів, я дійшов висновку, що кожна з них має свої сильні

сторони, які спрямовані на вирішення конкретних завдань. Вони використовують передові технології, такі як багатопроменеві ехолоти, алгоритми класифікації та розпізнавання, а також адаптивні методи обробки сигналів, що робить їх ефективними в реальному часі. Для мене було важливо зрозуміти, як різні системи реалізують функціонал залежно від умов та цілей застосування. Це дало змогу краще усвідомити, які підходи варто використовувати для розробки власного програмного забезпечення, а також які методи та алгоритми можуть забезпечити максимальну ефективність у вирішенні задач аналізу гідроакустичних сигналів.

1.4 Актуальність розробки системи для аналізу гідроакустичних сигналів із застосуванням сучасних методів штучного інтелекту

Розробка системи для аналізу гідроакустичних сигналів із використанням сучасних методів штучного інтелекту є важливим і актуальним завданням. Гідроакустичні сигнали містять цінну інформацію про підводне середовище, але їх обробка ускладнюється через вплив шумів, змінні умови середовища та великий обсяг даних. Це створює значний виклик для традиційних методів обробки сигналів, які часто не забезпечують необхідної точності та швидкості аналізу. Саме тому створення автоматизованої системи, здатної вирішувати ці проблеми, є особливо актуальним.

Значною перевагою сучасних методів штучного інтелекту є їх здатність автоматизувати складні процеси, зокрема аналіз сигналів. Завдяки цьому можна досягти високої точності класифікації навіть у складних умовах, таких як значне шумове забруднення чи зміна фізичних властивостей середовища. Штучний інтелект дозволяє виділяти ключові характеристики сигналів, ідентифікувати джерела звуків та розпізнавати об`єкти. Ці можливості роблять такі системи надзвичайно ефективними порівняно з традиційними підходами. Особливо важливо, що такі системи можуть працювати з великими обсягами даних у реальному часі, що робить їх незамінними у практичних задачах.

Практичне значення таких систем охоплює широкий спектр галузей. Наприклад, в обороні вони використовуються для точного виявлення підводних човнів, мінування та інших об`єктів, що дозволяє значно підвищити рівень безпеки. У промисловості подібні системи забезпечують моніторинг стану підводної інфраструктури, зокрема трубопроводів чи кабелів, що допомагає своєчасно виявляти та усувати пошкодження. У наукових дослідженнях ці системи сприяють детальному вивченню морських екосистем, аналізу поведінки морських тварин та дослідженню структури морського дна. Таким чином, застосування штучного інтелекту значно розширює можливості роботи з гідроакустичними сигналами.

Особливу увагу привертає можливість автоматизації аналізу, яку забезпечують такі системи. Завдяки цьому знижується навантаження на фахівців і підвищується швидкість обробки даних. Автоматизація також дозволяє проводити аналіз у реальному часі, що є критично важливим у багатьох практичних сценаріях. Додатковою перевагою є здатність адаптивних моделей навчатися на нових даних, що з часом робить їхню роботу ще більш ефективною.

Таким чином, розробка системи для аналізу гідроакустичних сигналів із застосуванням сучасних методів штучного інтелекту є надзвичайно актуальним завданням. Вона дозволяє вирішити багато існуючих проблем, підвищити ефективність обробки даних та розширити можливості застосування гідроакустичних технологій у різних галузях.

1.5 Постановка основних задач розробки системи для обробки гідроакустичних сигналів

Аналіз предметної області та існуючих підходів до обробки гідроакустичних сигналів показав, що для ефективного вирішення завдань класифікації сигналів і автоматизації процесу ідентифікації підводних об`єктів необхідно використовувати сучасні методи штучного інтелекту. На основі

цього визначено, що система повинна бути адаптивною, забезпечувати високу точність і надійність аналізу.

Метою розробки є створення системи, яка автоматизує обробку гідроакустичних сигналів, здатна точно аналізувати та класифікувати їх та забезпечує її використання в оборонній, промисловій та науково-дослідницькій сферах.

Виділенні завдання кваліфікаційної роботи:

— збір даних**;**

— оптимізація даних;

— зібрані дані повинні пройти попередню обробку: видалення шумів та нормалізацію;

— створення адаптивної моделі для обробки та класифікації сигналів. Створення моделі, яка забезпечуватиме точне виділення ключових характеристик гідроакустичних сигналів та їх класифікацію;

— створення програмного інтерфейсу користувача;

— для зручності використання система повинна включати інтуїтивно зрозумілий інтерфейс, який забезпечуватиме доступ до основних функцій, таких як завантаження даних, запуск аналізу, перегляд і візуалізація результатів;

— забезпечення масштабованості системи;

— система повинна бути здатною обробляти великі обсяги даних і працювати в режимі реального часу, а також підтримувати можливість інтеграції з іншими інформаційними системами;

— перевірка та оптимізація точності моделі.

Система має бути протестована на реальних даних, щоб оцінити її точність.

Таким чином, основні задачі спрямовані на побудову комплексної, адаптивної та надійної системи для аналізу гідроакустичних сигналів, яка дозволить вирішити існуючі проблеми в обробці таких даних та забезпечити високу якість результатів у різних сферах застосування.

1.6 Розробка функціональної та структурної схеми програмного комплексу

Розглянемо рисунок 1.1, який описує контекстну функціональну схему програмного комплексу.

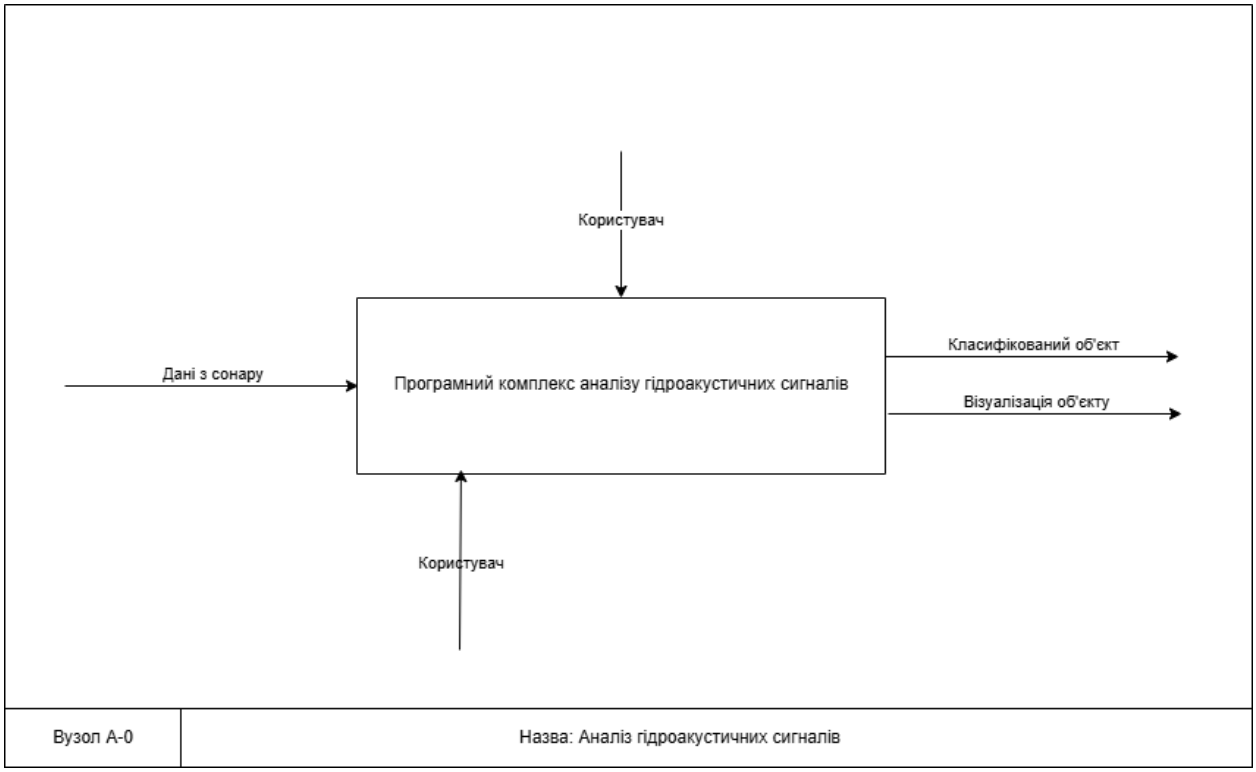

Рисунок 1.1 – Контекстна функціональна схема програмного комплексу

На рисунку 1.1 зображена контекстна функціональна схема програмного комплексу для аналізу гідроакустичних сигналів, яка описує його основні компоненти та взаємодію.

Вхідні дані включають дані, отримані з сонару, які надходять для обробки до програмного комплексу. Дані з сонару містять інформацію про акустичні сигнали, які підлягають класифікації та візуалізації.

Вихідні дані представлені у вигляді класифікованого об'єкта (наприклад, "Міна" чи "Камінь") та візуалізації об'єкта, що дозволяє користувачеві проаналізувати результат обробки сигналу.

Зовнішня взаємодія забезпечується користувачем, який вводить початкові дані та отримує результати аналізу.

Дана схема наочно демонструє основний процес роботи комплексу: від отримання даних з сонару до надання результатів класифікації та їх візуалізації користувачеві.

Розглянемо рисунок 2.2, який описує декомпозицію функціональної схеми програмного комплексу.

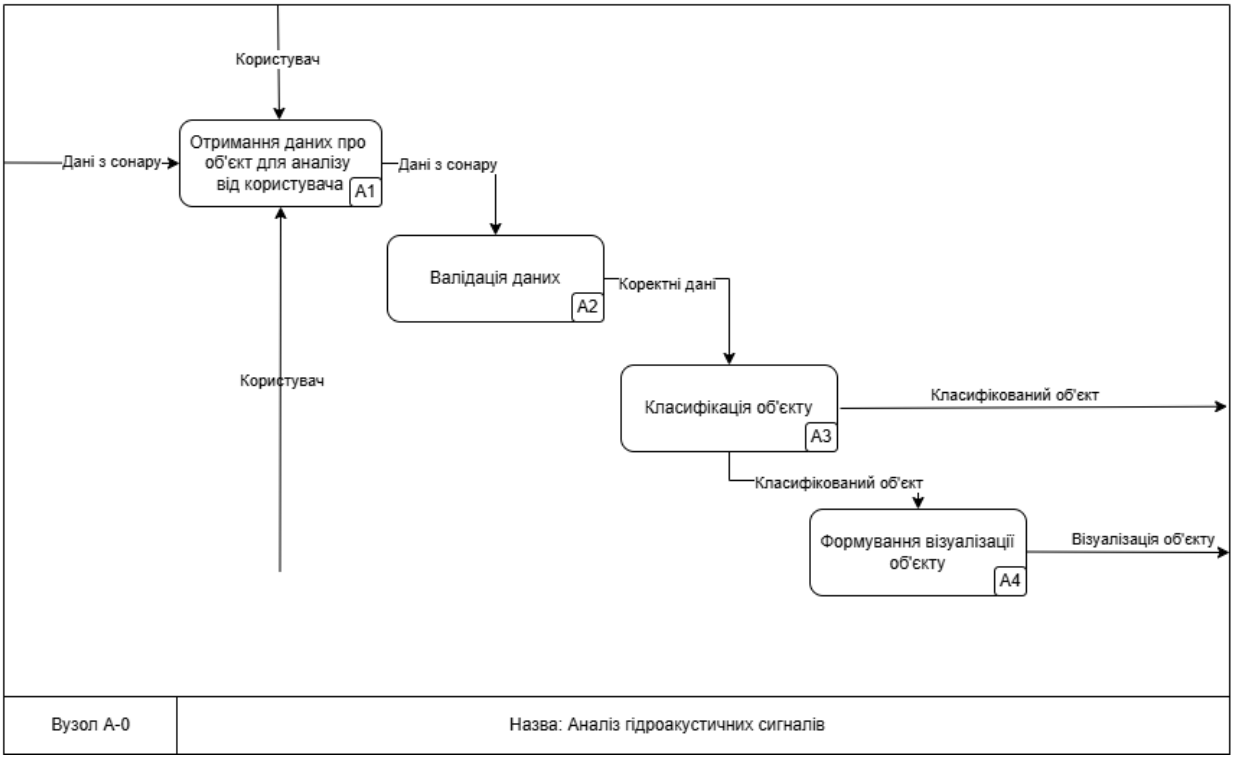

Рисунок 1.2 – Декомпозиція функціональної схеми програмного комплексу

На рисунку 1.2 зображена декомпозиція функціональної схеми програмного комплексу для аналізу гідроакустичних сигналів, яка описує роботу програми більш детально.

Першим етапом є отримання даних про об'єкт для аналізу від користувача. На цьому етапі користувач передає дані з сонару, які є вхідною інформацією для подальшої обробки.

Далі відбувається валідація даних, під час якої перевіряються отримані дані на коректність і відповідність встановленим критеріям. У результаті цього етапу формується набір коректних даних, які передаються на наступний крок.

Після валідації дані надходять на етап класифікації об'єкта, де програмний комплекс виконує аналіз вхідних даних і визначає клас об'єкта, наприклад: міна або камінь. Результатом цього етапу є класифікований об'єкт.

Останнім етапом є формування візуалізації об'єкта, де створюється графічне зображення або інша візуалізація, яка допомагає користувачеві зрозуміти результати класифікації. Візуалізація та класифікований об'єкт надаються користувачеві як вихідні дані.

Ця декомпозиція дозволяє детально описати основні етапи обробки гідроакустичних сигналів у межах роботи програмного комплексу.

Розглянемо рисунок 1.3, який описує структурну схему програмного клмплексу.

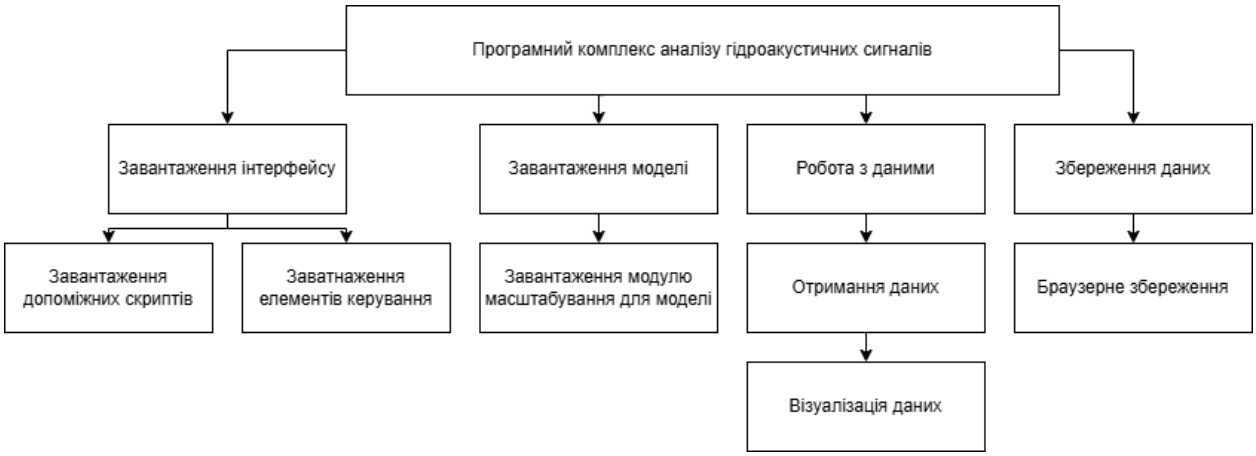

Рисунок 1.3 – Структурна схема програмного комплексу

На рисунку 1.3 зображена структурна схема, яка описує підпрограми та алгоритми, які повинні бути реалізовані.

2 РОЗРОБКА МАТЕМАТИЧНИХ МОДЕЛЕЙ ТА ОГЛЯД ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В РАМКАХ АНАЛІЗУ ГІДРОАКУСТИЧНИХ СИГНАЛІВ

2.1 Розробка математичних моделей для аналізу та обробки даних

Для ефективного аналізу гідроакустичних сигналів важливо правильно вибрати математичну модель, яка відповідає поставленій задачі. Основними типами задач у цій галузі є класифікація, регресія та кластеризація. Кожна з них має свої специфічні алгоритми, які доцільно застосовувати в залежності від мети дослідження, характеристик даних та особливостей гідроакустичних сигналів.

Класифікація є найбільш поширеною задачею при роботі з гідроакустичними сигналами. Вона дозволяє визначати тип сигналу, класифікувати об`єкти чи стан середовища на основі вхідних даних. Математично задача класифікації виглядає як:

$$f: X \rightarrow Y, \tag{2.1}$$

де X – вхідний простір ознак;

Y - скінченна множина класів.

Мета моделі полягає в знаходженні функції f, яка для кожного $x \in X$ повертає правильний клас $y \in Y$.

Для оцінки якості моделі використовується функція втрат $L(y, \hat{y})$, яка порівнює реальний клас y з передбаченим $\hat{y}$. Наприклад, при використанні логістичної регресії часто застосовується логарифмічна функція втрат (log loss):

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\widehat{y_i}) + (1 - y_i) \log(1 - \widehat{y_i})], \tag{2.2}$$

де N - кількість зразків;

y_i - реальний клас;

$\widehat{y_i}$ - ймовірність належності зразка до класу.

Таким чином, основна ідея класифікації полягає в мінімізації функції втрат L, щоб покращити точність передбачень.

Для розв`язання задачі класифікації використовуються різні алгоритми, такі як:

— Support Vector Machine (SVM). Алгоритм для задач із чіткими межами між класами, навіть у високо розмірних просторах;

— Random Forest. Алгоритм для задач класифікації з шумними даними або складними залежностями;

— XGBoost. Алгоритм для складних класифікаційних задач, в яких потрібно мати високу точність і стабільність;

— KNeighbors. Простий і ефективний алгоритм, що класифікує зразок на основі класів його найближчих сусідів. Особливо ефективний у задачах із невеликими наборами даних;

— Logistic Regression. Базовий метод класифікації, що добре працює у задачах з лінійною роздільністю класів. Простий у реалізації та швидкий у роботі;

— Gaussian Naive Bayes. Імовірнісний алгоритм, що базується на теоремі Байєса і припускає нормальний розподіл ознак. Ефективний у задачах з великим обсягом даних та слабкою кореляцією між ознаками. [2, 5]

Щодо задачі регресії, вона використовується для прогнозування числових значень, наприклад, інтенсивності сигналу, швидкості руху об`єкта або глибини джерела сигналу, моделюючи залежність між вхідними ознаками X та цільовою змінною y.

Математично задача регресії виглядає як:

$$y = f(X) + \varepsilon, \tag{2.3}$$

де *f(X)* - функція, яка відображає залежність між ознаками та цільовим значенням;

ε - випадкова похибка.

Найбільш поширеною формою регресії є лінійна регресія, у якій *f(X)* виражається як лінійна комбінація ознак:

$$f(X) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i, \quad (2.4)$$

де β_0 – вільний член;

β_i - коефіцієнти, що вказують на вплив кожної ознаки x_i на цільову змінну *y*.

Для розв`язання задачі регресії використовуються різні алгоритми, такі як:

— Linear Regression. Алгоритм простий та швидкий у обробці, але обмежений задачами з лінійними залежностями;

— Decision Trees. Алгоритм який забезпечує інтуїтивно зрозумілий прогноз та працює з нелінійними залежностями в даних;

— Support Vector Regression (SVR). Потужний метод, який дозволяє моделювати складні нелінійні залежності. Ефективний для задач із високою розмірністю даних, але може бути обчислювально інтенсивним;

— Random Forest. Алгоритм, який створює ансамбль дерев рішень для прогнозування, забезпечуючи точність і стійкість до перенавчання. Особливо добре працює зі складними, багатофакторними задачами;

— XGBoost. Алгоритм, що базується на техніці градієнтного бустингу, дозволяє моделювати складні залежності та забезпечує високу точність прогнозування. Його оптимізована архітектура гарантує швидкість і ефективність роботи навіть із великими наборами даних. [5, 6]

Остання на розгляді задача кластеризації. Вона застосовуєтся для виявлення груп схожих сигналів без наявності міток, наприклад, для розпізнавання невідомих джерел сигналів чи виявлення аномалій.

Математично ця задача формулюється як розподіл N зразків $\{ x_1, x_2, \dots, x_N \}$ на K кластерів $\{ C_1, C_2, \dots, C_K \}$, де кожен кластер C_K відповідає підмножині даних.

Класичний метод кластеризації K-Means мінімізує відстань між точками даних і центроїдом їх кластера:

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2, \tag{2.5}$$

де K - кількість кластерів;

C_k множина точок у кластері k, μ_k центр кластера k.

Для методів, таких як Gaussian Mixture Models (GMM), замість детермінованого підходу використовується ймовірнісний:

$$P(x|\lambda_k) = \frac{1}{\sqrt{(2\pi)^d |\sum k|}} \exp\left(-\frac{1}{2} (x - \mu_k)^T \sum_K^{-1} (x - \mu_k) \right), \tag{2.6}$$

де μ_k і $\sum k$ відповідно центр і коваріаційна матриця кластеру k.

Для розв`язання задачі кластеризації використовуються різні алгоритми, такі як:

— K-Means. Алгоритм для виділення кластерів, коли вони мають сферичну форму. Hierarchical Clustering. Алгоритм, який використовується для ієрархічної структури кластерів, що може бути корисно при аналізі багаторівневих систем;

— DBSCAN (Density-Based Spatial Clustering). Алгоритм для даних із нерівномірною щільністю та для виявлення аномалій;

— Gaussian Mixture Models (GMM). Алгоритм, який враховує перекриття між кластерами;

— Agglomerative Clustering. Ієрархічний підхід до кластеризації, який послідовно об`єднує точки в кластери, ґрунтуючись на їхній подібності. Підходить для аналізу складних структур даних із різною формою кластерів.

Отже, розглянувши задачі та алгоритми їх розв`язання, можна сказати що залежно від задачі аналізу гідроакустичних сигналів, доцільність використання певного методу визначається особливостями даних. Таким чином, вибір математичної моделі має ґрунтуватися на природі даних і конкретній задачі, щоб забезпечити максимально точний і надійний аналіз.

2.2 Детальний огляд методів штучного інтелекту для аналізу та обробки даних

У попередньому розділі ми розглянули основні математичні моделі та методи, які застосовуються для аналізу та обробки різноманітних видів та типів даних, серед них ми згадували алгоритми класифікації, регресії та кластеризації. У цьому розділі детально зупинимося на характеристиках цих методів, їхніх перевагах, обмеженнях та доцільності застосування для задач аналізу гідроакустичних сигналів.

Розглянемо метод класифікації. Класифікація є однією з найпоширеніших задач при аналізі гідроакустичних сигналів. Метою є ідентифікація типу об`єкта або стану середовища, що генерує сигнал.

Алгоритми класифікації працюють шляхом визначення залежності між ознаками сигналу та класами, до яких вони належать, а саме:

— Support Vector Machine (SVM)**.** Алгоритм опорних векторів (SVM) є одним із найефективніших методів для класифікації, особливо коли дані мають складні межі розподілу. Його основна ідея полягає у побудові гіперплощини, яка розділяє класи з максимальним можливим відступом між найближчими точками обох класів - опорними векторами. Якщо дані не можна розділити лінійно, використовується механізм ядерних функцій, які перетворюють простір ознак у вищу розмірність, що дозволяє розділити класи. SVM чудово працює із задачами високої розмірності ознак і забезпечує високу точність у класифікації. Однак алгоритм є обчислювально важким для великих наборів даних. Доцільність використання: SVM ідеально підходить для задач, де

важливі точність і чіткість меж між класами, таких як аналіз гідроакустичних сигналів із складною структурою;

— Random Forest. Алгоритм Random Forest - це ансамблевий метод, що складається з безлічі дерев рішень. Кожне дерево будується на випадковій підмножині даних, а прогноз визначається голосуванням дерев для класифікації або усередненням результатів для регресії. Такий підхід забезпечує стійкість до перенавчання та стабільні результати навіть для даних, які мають шум. Алгоритм легко інтерпретується, добре працює з великою кількістю ознак і забезпечує точність при обробці складних сигналів, таких як частотні або часові характеристики. Доцільність використання: Метод підходить для задач із шумними даними та великим набором характеристик, роблячи його ефективним для аналізу гідроакустичних сигналів;

— XGBoost. Алгоритм XGBoost є одним із найпотужніших сучасних алгоритмів, який використовується як для класифікації, так і для регресії. Він заснований на техніці градієнтного бустингу, де кожне наступне дерево навчається виправляти помилки попередніх. Завдяки вбудованим механізмам регуляризації, XGBoost забезпечує високу точність і стійкість до перенавчання. Алгоритм оптимізований для швидкої роботи з великими наборами даних і є чудовим вибором для складних задач, таких як аналіз великих обсягів гідроакустичних сигналів. Доцільність використання: Використовується для складних класифікаційних задач, де потрібна висока точність і стабільність прогнозів;

— KNeighbors. Цей алгоритм базується на ідеї визначення класу зразка шляхом голосування серед класів його найближчих сусідів. Кількість сусідів (k) задається заздалегідь, а відстань до них обчислюється за певною метрикою, наприклад, евклідовою або мангеттенською. Цей алгоритм є простим у реалізації та не вимагає попереднього навчання, але може бути повільним на великих наборах даних через необхідність порівняння кожного нового зразка з усією навчальною вибіркою. Доцільність використання: KNeighbors добре підходить для задач із невеликими наборами даних, де важлива локальна

структура, наприклад, для класифікації гідроакустичних сигналів за схожими характеристиками;

— Logistic Regression. Це базовий метод класифікації, який моделює ймовірність належності зразка до певного класу за допомогою логістичної функції. Алгоритм працює на основі зважування ознак і оцінки їхнього впливу на клас за допомогою коефіцієнтів, що оптимізуються методом максимізації правдоподібності. Логістична регресія є швидкою, простою в реалізації та добре працює у задачах, де класи є лінійно роздільними, але може бути менш ефективною при нелінійних залежностях. Доцільність використання: Цей метод підходить для швидких і простих рішень задач класифікації з чіткою лінійною залежністю, наприклад, для базового аналізу гідроакустичних сигналів;

— Gaussian Naive Bayes. Алгоритм базується на теоремі Байєса і припускає, що всі ознаки є незалежними одна від одної (умовно незалежні), а їхній розподіл є нормальним (гаусовим). Кожна ознака оцінюється окремо для кожного класу, що дозволяє швидко обчислювати ймовірність належності зразка до певного класу. Незважаючи на спрощення про незалежність ознак, метод часто показує хороші результати навіть у випадках, коли ознаки сильно корелюють. Доцільність використання: Gaussian Naive Bayes є ефективним для класифікації великих обсягів даних з невеликою кількістю ознак, наприклад, для початкового аналізу або класифікації гідроакустичних сигналів із простими розподілами ознак.

Розглянемо метод регресії. Регресія використовується для прогнозування числових характеристик, наприклад, інтенсивності сигналу, глибини джерела або швидкості об`єкта.

— Linear Regression**.** Лінійна регресія - найпростіший метод прогнозування, який моделює лінійну залежність між ознаками та цільовою змінною. Цей метод є швидким і легко інтерпретується, проте його ефективність обмежена задачами з нелінійними залежностями. У контексті гідроакустичних сигналів він підходить для прогнозування простих числових

значень. Доцільність використання: Використовується, коли залежність між характеристиками сигналу та цільовою змінною є лінійною;

— Random Forest. Алгоритм є ансамблевим методом, що базується на поєднанні численних дерев рішень. Кожне дерево будується на основі випадкової підмножини даних і ознак, що зменшує ризик перенавчання та підвищує стабільність моделі. Прогнозування здійснюється шляхом усереднення результатів усіх дерев, що дозволяє моделювати як лінійні, так і нелінійні залежності. Random Forest є стійким до шуму в даних і здатним обробляти великі набори ознак. Доцільність використання: Random Forest підходить для складних задач регресії з великою кількістю змінних, таких як моделювання параметрів гідроакустичних сигналів у нелінійних середовищах;

— Support Vector Regression (SVR). Алгоритм є розширенням методу опорних векторів (SVM) для задач регресії. Його основна ідея полягає в побудові моделі, яка мінімізує відхилення прогнозу від фактичного значення в межах допустимого порогу (ε). Алгоритм дозволяє ефективно моделювати як лінійні, так і нелінійні залежності завдяки використанню ядерних функцій. Хоча SVR є потужним інструментом, він може бути обчислювально важким при роботі з великими наборами даних. Доцільність використання: SVR добре підходить для задач, де потрібна висока точність у моделюванні складних залежностей, таких як прогнозування параметрів сигналу;

— XGBoost. Алгоритм є вдосконаленням техніки градієнтного бустингу, що побудований на послідовному тренуванні ансамблю моделей, кожна з яких коригує помилки попередньої. Алгоритм забезпечує високу точність завдяки регуляризації та ефективній обробці великих обсягів даних. Він також включає оптимізації, які прискорюють навчання та зменшують використання пам`яті. Доцільність використання: XGBoost ідеально підходить для складних задач регресії з великою кількістю ознак і високими вимогами до точності, наприклад, для прогнозування складних параметрів гідроакустичних сигналів;

Розглянемо метод кластеризації. Кластеризація дозволяє групувати схожі сигнали без попередньої інформації про мітки. Це корисно для виявлення аномалій чи нових типів сигналів.

— K-Means. K-Means - це алгоритм кластеризації, який мінімізує відстань між точками даних і центроїдами кластерів. Метод є простим у реалізації та ефективним для даних із чітко визначеними групами. Однак він менш ефективний для кластерів нерівномірної щільності або складної форми. Доцільність використання: Використовується для простого групування сигналів;

— DBSCAN (Density-Based Spatial Clustering). DBSCAN кластеризує дані на основі щільності, що дозволяє автоматично визначати кількість кластерів і виявляти аномалії. На відміну від K-Means, цей алгоритм добре працює з шумними даними та кластеризацією нерівномірної щільності. DBSCAN є ефективним для виявлення рідкісних або аномальних сигналів у гідроакустичних даних. Доцільність використання: Використовується для виявлення аномалій або сигналів із нерівномірною щільністю;

— Gaussian Mixture Models (GMM). Gaussian Mixture Models (GMM) базується на ймовірнісному підході, моделюючи дані як суму кількох нормальних розподілів. Цей метод забезпечує гнучкість у випадках, коли кластери мають перекриття. Однак GMM є чутливим до початкових умов і потребує значних обчислювальних ресурсів. Доцільність використання: Використовується для даних із комплексними розподілами, що містять перекриття між кластерами;

— Agglomerative Clustering. Алгоритм Agglomerative Clustering є ієрархічним методом кластеризації, який побудований на принципі послідовного об`єднання точок даних у кластери. Спочатку кожна точка розглядається як окремий кластер, а далі, на кожному етапі, найближчі кластери об`єднуються на основі обраної метрики відстані (наприклад, евклідової). Цей процес триває, доки не залишиться задана кількість кластерів або всі точки не об`єднаються в один кластер. Алгоритм ефективний для задач,

де важливо зрозуміти ієрархічну структуру даних. Доцільність використання: Agglomerative Clustering добре підходить для аналізу даних із багаторівневою структурою, наприклад, для виявлення ієрархічних залежностей між гідроакустичними сигналами.

Отже, нами було детально розглянуто різноманітні методи штучного інтелекту, які застосовуються для аналізу та обробки різних видів і типів даних. Кожен із методів має свої сильні сторони, обмеження та області ефективного застосування, що дозволяє вибирати оптимальний підхід залежно від задачі та характеристик даних.

Класифікація є однією з ключових задач в аналізі гідроакустичних сигналів. Алгоритми, такі як SVM**,** Random Forest**,** KNeighbors**,** Logistic Regression **і** Gaussian Naive Bayes, дозволяють точно ідентифікувати джерела сигналів або класифікувати їх за типом. Зокрема, SVM ефективний для задач із чіткими межами між класами, Random Forest добре працює з шумними даними, а KNeighbors є простим і ефективним для малих наборів даних. Logistic Regression дає гарні результати для задач із лінійними залежностями, а Gaussian Naive Bayes підходить для задач з великими наборами даних, де ознаки є умовно незалежними.

Для задач регресії, де необхідно прогнозувати числові характеристики, такі як інтенсивність сигналу або швидкість об`єкта, використовуються методи, як Linear Regression, Random Forest Regressor, SVR, XGBoost. Linear Regression підходить для простих задач з лінійними залежностями, тоді як Random Forest Regressor і XGBoost демонструють високу точність при роботі з нелінійними залежностями та великими наборами даних, а SVR ефективний для моделювання складних нелінійних залежностей.

Для задач кластеризації, де необхідно групувати дані без попередніх міток, розглянуті алгоритми, такі як K-Means, DBSCAN, Agglomerative Clustering і Gaussian Mixture Models (GMM). K-Means підходить для чітко визначених сферичних кластерів, DBSCAN є ідеальним для даних із нерівномірною щільністю та шуми, а Agglomerative Clustering дозволяє

працювати з ієрархічними структурами. GMM є гнучким методом для задач із перекриттям між кластерами.

Таким чином, вибір методу для аналізу гідроакустичних сигналів залежить від специфіки даних і задачі. Кожен алгоритм має свої переваги в залежності від типу сигналу, його структури та поставленої мети аналізу.

2.3 Порівняння методів штучного інтелекту для аналізу гідроакустичних сигналів

Порівнюючи методи класифікації, регресії та кластеризації які ми розглянули у минулому розділі, слід враховувати ключові аспекти, такі як точність, стійкість до шуму, здатність працювати з великими обсягами даних, здатність враховувати часові залежності та обчислювальна складність. У цьому розділі проведено детальне порівняння основних методів із підсумковим зведенням у таблицю.

Методи класифікації забезпечують точність у визначенні типу гідроакустичного сигналу або джерела його генерації. Найпоширенішими є:

— SVM забезпечує високу точність для невеликих наборів даних із чіткими межами між класами. Завдяки ядерним функціям він добре працює з нелінійно розділеними даними. Проте його обчислювальна складність значно зростає з розміром вибірки, що обмежує його застосування для великих обсягів даних;

— Random Forest є стійким до шуму і добре працює з великим набором ознак. Завдяки ансамблевій природі метод забезпечує стабільність результатів навіть для складних і неоднорідних сигналів. Однак час навчання може бути тривалим, особливо при збільшенні кількості дерев;

— XGBoost демонструє високу продуктивність завдяки градієнтному бустингу. Алгоритм ефективний для складних задач, де потрібно врахувати численні залежності між ознаками. Проте налаштування гіперпараметрів може бути трудомістким;

— KNeighbors**.** Цей метод класифікує об`єкт на основі класів його найближчих сусідів. Алгоритм простий у реалізації, не вимагає попереднього навчання та є ефективним для невеликих наборів даних. Проте його продуктивність знижується на великих вибірках через високу обчислювальну складність при визначенні відстаней до сусідів;

— Logistic Regression**.** Це базовий метод класифікації, який добре працює з лінійно розділеними даними. Він моделює ймовірність належності об`єкта до певного класу за допомогою логістичної функції. Метод є швидким, простим у реалізації та забезпечує інтерпретовані результати, проте менш ефективний для задач із нелінійними залежностями;

— Gaussian Naive Bayes**.** Імовірнісний алгоритм, який базується на теоремі Байєса та припущенні про умовну незалежність ознак. Метод швидкий у навчанні та виконанні, забезпечує гарну продуктивність на великих наборах даних, навіть якщо припущення про незалежність не завжди дотримуються. Проте його ефективність може знижуватися при значній кореляції між ознаками.

Методи регресії забезпечують прогнозування числових характеристик гідроакустичних сигналів, таких як інтенсивність, глибина джерела або швидкість об`єкта. Найпоширенішими є:

— Linear Regression**.** Це базовий метод регресії, який моделює лінійну залежність між вхідними ознаками та цільовою змінною. Алгоритм швидкий, простий у реалізації та добре працює з невеликими наборами даних і лінійними залежностями. Однак його ефективність значно знижується для задач із нелінійними взаємозв`язками між ознаками;

— Random Forest**.** Алгоритм є ансамблевим методом, що створює декілька дерев рішень для прогнозування. Прогноз обчислюється як середнє значення результатів окремих дерев. Random Forest демонструє високу точність, добре працює з нелінійними залежностями та стійкий до перенавчання, проте може бути обчислювально інтенсивним для великих наборів даних;

— Support Vector Regression (SVR)**.** Алгоритм базується на методі опорних векторів і моделює залежності в даних, враховуючи відхилення в межах заданого порогу. SVR ефективно працює як із лінійними, так і з нелінійними залежностями завдяки ядерним функціям. Проте обчислювальна складність алгоритму обмежує його застосування для великих вибірок;

— XGBoost**.** Цей метод використовує градієнтний бустинг для побудови послідовності моделей, кожна з яких виправляє помилки попередньої. XGBoost демонструє високу точність, добре працює з великими наборами даних і складними нелінійними залежностями. Завдяки оптимізованій архітектурі алгоритм є одним із найефективніших для задач регресії, хоча його використання потребує значних обчислювальних ресурсів.

Кластеризація дозволяє групувати сигнали без попередньої інформації про їхню класифікацію. Найбільш поширеними методами є:

— K-Means є простим і швидким методом, який працює добре, якщо кластери мають сферичну форму та однакову щільність. Проте цей метод погано справляється з даними, які містять шум або нерівномірну щільність кластерів;

— DBSCAN дозволяє виявляти кластери на основі щільності, автоматично визначаючи їхню кількість. Алгоритм ефективний для шумних даних, але налаштування параметрів може бути складним;

— GMM моделює дані як сукупність нормальних розподілів, дозволяючи враховувати перекриття між кластерами. Проте метод є чутливим до початкових умов і вимагає значних обчислювальних ресурсів;

— Agglomerative Clustering. Цей алгоритм базується на ієрархічному підході, поступово об`єднуючи точки даних у кластери. Спочатку кожна точка розглядається як окремий кластер, а далі вони об`єднуються, ґрунтуючись на обраній метриці відстані, поки не залишиться задана кількість кластерів. Agglomerative Clustering ефективно працює з багаторівневими структурами даних і дозволяє отримати ієрархію залежностей між об`єктами. Однак

алгоритм є обчислювально складним для великих наборів даних і вимагає попереднього визначення кількості кластерів.

Провівши детальне порівняння методів, можна узагальнити результати в таблиці, яка систематизує ключові аспекти кожного підходу, допомагаючи вибрати оптимальну модель для аналізу гідроакустичних сигналів.

Таблиця 2.1 - Порівняння методів штучного інтелекту для аналізу гідроакустичних сигналів

Метод	Тип	Точність	Доцільність використання
SVM	Класифікація	Висока для чітких меж між класами.	Чіткі межі між класами.
Random Forest	Класифікація та Регресія	Висока для шумних даних.	Складні, шумні дані.
XGBoost	Класифікація та Регресія	Висока для великих обсягів даних.	Великі дані, складні залежності.
KNeighbors	Класифікація	Помірна для невеликих даних.	Невеликі набори даних.
Logistic Regression	Класифікація	Висока для лінійно розділених даних.	Лінійно розділені дані.
Gaussian Naive Bayes	Класифікація	Висока для великих наборів із незалежними ознаками.	Великі набори з незалежними ознаками.
Linear Regression	Регресія	Висока для лінійних залежностей.	Прості числові прогнози.
Support Vector Regression	Регресія	Висока для нелінійних залежностей.	Прогнози складних числових залежностей.
K-Means	Кластеризація	Висока для сферичних кластерів.	Початковий аналіз структурованих даних.
DBSCAN	Кластеризація	Висока для даних із шумом та аномаліями.	Аномалії, нерівномірна щільність.
GMM	Кластеризація	Висока для складних кластерів.	Моделювання складних кластерів.

Продовження таблиці 2.1

Метод	Тип	Точність	Доцільність використання
Agglomerative Clustering	Кластеризація	Висока для багаторівневих структур.	Багаторівневі структури.

Результати проведеного порівняння показують, що вибір моделі для аналізу гідроакустичних сигналів значною мірою визначається специфікою задачі та особливостями даних. Для класифікаційних задач, що включають розпізнавання типу сигналу або джерела його генерації, найефективнішими є SVM та Random Forest. SVM забезпечує високу точність для чітко розділених класів, тоді як Random Forest демонструє стійкість до шуму. Для задач із простими залежностями добре підходять Logistic Regression та KNeighbors, а Gaussian Naive Bayes є чудовим вибором для великих наборів даних із умовно незалежними ознаками.

У задачах регресії, які потребують прогнозування числових характеристик сигналів, XGBoost виявляється найефективнішим завдяки здатності працювати з великими обсягами даних і складними нелінійними залежностями. Random Forest також є потужним методом, що поєднує високу точність і стійкість до шуму. Linear Regression залишається базовим вибором для задач із лінійними залежностями, а SVR є оптимальним для задач із нелінійними зв`язками.

Щодо кластеризації, DBSCAN є найкращим вибором для аналізу даних із нерівномірною щільністю, оскільки він автоматично визначає кількість кластерів і ефективно працює із шумом. Для даних зі складним перекриттям між кластерами оптимальним є GMM завдяки його гнучкості. K-Means залишається ефективним методом для простих сферичних кластерів, а Agglomerative Clustering дозволяє вивчати багаторівневі структури даних.

Таким чином, вибір алгоритму залежить від конкретних вимог аналізу, особливостей сигналів і задачі, що стоїть перед дослідником. Це дозволяє забезпечити високу точність і ефективність у виконанні поставлених завдань.

3 РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ АНАЛІЗУ ГІДРОАКУСТИЧНИХ СИГНАЛІВ

3.1 Порівняльний аналіз та вибір мови програмування і інструментів для розробки програмного комплексу

У процесі розробки програмного забезпечення для аналізу гідроакустичних сигналів важливим етапом є вибір мови програмування та інструментів, які забезпечать ефективну реалізацію моделей, створення API та розробку веб-інтерфейсу. Цей вибір ґрунтується на аналізі доступних можливостей та їх відповідності завданням, що поставлені в межах роботи.

Для створення та навчання моделей використовуються різні мови програмування, серед яких Python, R, JavaScript, Java та C++. Після аналізу було визначено, що Python є найкращим варіантом для реалізації цієї частини проекту. Основною перевагою Python є його універсальність і широкий набір бібліотек для машинного навчання, серед яких scikit-learn**,** NumPy та Pandas**.** Крім того, Python має простий синтаксис, що значно прискорює розробку, і велику підтримку спільноти, яка забезпечує доступ до численних готових рішень і документації. [7, 12]

Для створення API було обрано фреймворк Fast API. Його вибір зумовлений тим, що він є легким і швидким у налаштуванні, але водночас достатньо потужним для інтеграції з ML-моделями. Фреймворк дозволяє створювати API, яке може приймати дані від користувача, виконувати обчислення за допомогою моделей і повертати результати. Додатковою перевагою є те, що він легко масштабується для проєктів невеликого та середнього розміру.

Для реалізації веб-інтерфейсу було обрано React. Це сучасна бібліотека для створення динамічних веб-додатків. Основними причинами такого вибору стали його гнучкість, швидкість розробки та можливість інтеграції з Fast API. React забезпечує побудову зручного та інтуїтивно зрозумілого інтерфейсу для

кінцевого користувача, дозволяючи завантажувати дані та переглядати результати аналізу. [11, 13]

Обрані бібліотеки та інструменти для кожного з етапів розробки також були ретельно підібрані. Для роботи з моделями класифікації, регресії та кластеризації використовується scikit-learn, який надає широкий вибір алгоритмів і функцій для навчання моделей. Для обробки та підготовки даних використовуються NumPy і Pandas, а для візуалізації результатів - Matplotlib та Seaborn. Для збереження моделей і стандартизаторів обрано бібліотеку joblib, яка забезпечує швидке збереження та завантаження даних.

Отже, в результаті порівняльного аналізу для реалізації завдань аналізу гідроакустичних сигналів були обрані такі інструменти:

— для створення та навчання моделей було обрано Python з використанням бібліотек scikit-learn, NumPy, Pandas, Matplotlib та Seaborn. Цей вибір забезпечує ефективну реалізацію алгоритмів класифікації, регресії та кластеризації, а також обробку й візуалізацію даних;

— для створення API було обрано Fast API, який дозволяє інтегрувати машинні моделі в серверний додаток і забезпечує обробку запитів від користувача;

— для розробки веб-інтерфейсу було обрано React, що надає можливість створення динамічних і зручних веб-додатків для взаємодії з API та відображення результатів аналізу.

Обрані інструменти відповідають вимогам проекту, забезпечують простоту розробки, високу продуктивність і гнучкість, а також інтеграцію між усіма компонентами системи. [13, 14]

3.2 Попередня обробка даних

Для побудови ефективних моделей аналізу гідроакустичних сигналів важливим етапом є підготовка даних. У цьому підрозділі описано джерело даних, структуру датасету, процес зчитування та основні методи попередньої обробки, включаючи нормалізацію.

Дані представлені у вигляді набору числових характеристик. Цей набір містить вимірювання сигналів, відбитих від двох об`єктів: камню та міни. Кожен запис представлений у вигляді 60 числових характеристик, що відображають частотні ознаки сигналів, і одного цільового стовпця, який позначає клас об`єкта (M - Міна, R - Камінь). Датасет є збалансованим: кількість зразків для кожного класу приблизно однакова, що дозволяє уникнути проблеми класового дисбалансу при побудові моделей.

Для зчитування даних було використано бібліотеку Pandas, яка дозволяє швидко завантажити датасет і переглянути його структуру.

Розглянемо лістинг коду програмної реалізації зчитування датасету з файлу формату CSV.

```
import pandas as pd
sonar_data = pd.read_csv(`/content/sonar_data.csv`,
    header=None)
sonar_data.head()
```

У цьому лістингу для зчитування датасету використовується метод read_csv, який забезпечує доступ до даних, які готові для попереднього аналізу. За допомогою методу head можна переглянути перші п`ять записів, щоб переконатися в коректності структури даних.

Наступним кроком стало забезпечення якості даних шляхом попередньої обробки. Спочатку було перевірено наявність пропущених значень у датасеті. За допомогою методу isnull вдалося підтвердити, що в наборі даних немає пропусків.

Розглянемо лістинг коду програмної реалізації перевірки наявності пропущених значень у датасеті.

```
sonar_data.isnull().sum().sum()
```

Цей етап гарантує, що всі значення в датасеті повні, що спрощує подальший аналіз.

Після цього датасет було поділено на ознаки та цільову змінну.

Розглянемо лістинг коду програмної реалізації розділення датасету на ознаки та цільову змінну.

```
X = sonar_data.iloc[:, :-1]
y = sonar_data.iloc[:, -1]
```

У цьому лістингу усі числові характеристики сигналів були збережені у змінній X, а цільова змінна, що позначає клас об`єкта, - у змінній y.

Далі дані було розділено на навчальну та тестову вибірки, щоб забезпечити можливість оцінки продуктивності моделей.

Розглянемо лістинг коду програмної реалізації розділення даних.

```
from sklearn.model_selection import
    train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.25,
    random_state=42)
```

Для цього використано функцію train_test_split з бібліотеки scikit-learn. Співвідношення розподілу даних було встановлено як 75% для навчання та 25% для тестування.

Щоб покращити роботу моделей, було застосовано нормалізацію даних.

Розглянемо лістинг коду програмної реалізації нормалізації даних.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

У цьому лістингу код гарантує стандартизацію значень, тобто, всі ознаки мають середнє значення 0 та стандартне відхилення 1. Це зменшує вплив ознак із великою дисперсією на процес навчання моделей. Для цього використано StandardScaler з бібліотеки scikit-learn.

Таким чином, підготовка даних включала зчитування, перевірку якості, поділ на вибірки та нормалізацію. Завдяки цим крокам датасет готовий до використання для побудови моделей машинного навчання.

У результаті, попередня обробка даних забезпечила надійність і якість інформації, що передається моделям. Це дозволяє перейти до наступного етапу - реалізації та оцінки моделей.

3.3 Реалізація моделей аналізу гідроакустичних сигналів

Процес аналізу гідроакустичних сигналів передбачає побудову моделей машинного навчання, які можуть виконувати задачі класифікації, регресії та кластеризації. У цьому підрозділі описано розробку моделей з використанням бібліотеки scikit-learn. Було реалізовано два підходи: базова реалізація моделей без налаштування гіперпараметрів та оптимізована реалізація з використанням методу GridSearchCV та ParameterGrid для підбору гіперпараметрів.

Для реалізації моделі класифікації були використані наступні алгоритми: Kneighbors, LogisticRegression, RandomForest, SVC, XGBoost та Gaussian Naive Bayes.

Розглянемо лістинг коду програмної реалізації моделі класифікації без оптимізації гіперпараметрів.

```
# Моделі
models = {
    "KNeighbors": KNeighborsClassifier(),
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(probability=True),
    "XGBoost": XGBClassifier(eval_metric=`logloss`)
    "Gaussian Naive Bayes": GaussianNB()
}
# Навчання і передбачення
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
```

У цьому лістингу кожна модель навчається на навчальних даних X_train та y_train за допомогою методу fit. Після навчання модель передбачає класи для тестових даних X_test за допомогою методу predict. На завершення обчислюється базова точність моделі за метрикою accuracy.

Щоб підвищити продуктивність моделей, була застосувана оптимізація гіперпараметрів за допомогою методу GridSearchCV. Цей інструмент автоматично перебирає різні комбінації параметрів і обирає найкращу модель на основі результатів крос-валідації.

Для кожної моделі я визначив набір параметрів, які потенційно можуть покращити її продуктивність.

Розглянемо лістинг коду програмної реалізації моделі класифікації з оптимізацією гіперпараметрів.

```
# Словник для гіперпараметрів
param_grids = {
    "KNeighbors": {
        "n_neighbors": [3, 5, 7],
        "weights": ["uniform", "distance"],
        "metric": ["euclidean", "manhattan"]
    },
    "Logistic Regression": {
        "C": [0.1, 1, 10],
        "solver": ["liblinear", "lbfgs"]
    },
    "Random Forest": {
        "n_estimators": [50, 100],
        "max_depth": [10, 20],
        "min_samples_split": [2, 5]
    },
    "SVM": {
```

```
            "C": [0.1, 1, 10],
            "kernel": ["linear", "rbf"],
            "gamma": ["scale", "auto"]
        },
        "XGBoost": {
            "n_estimators": [50, 100, 200],
            "max_depth": [3, 6, 10],
            "learning_rate": [0.01, 0.1, 0.2]
        }
    }
    # Calculate metrics for each model
    metrics_optimized_results = {
        "Model": [],
        "Accuracy": [],
        "Precision": [],
        "Recall": [],
        "F1-Score": []
    }
    optimized_results = {}
    # Оптимізація гіперпараметрів для кожної моделі
    for model_name, model in models.items():
        if model_name in param_grids:
            grid_search = GridSearchCV(
                model,
                param_grids[model_name],
                scoring="accuracy",
                cv=5,
                n_jobs=-1,
                verbose=1,
                refit=True)
```

```
            grid_search.fit(X_train, y_train)
            best_model = grid_search.best_estimator_
            y_pred = best_model.predict(X_test)
        else:
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
```

У цьому лістингу метод GridSearchCV перебирає комбінації гіперпараметрів, задаючи для кожної моделі оптимальний набір параметрів. Але, алгоритм Gaussian Naive Bayes немає можливості для оптимізації гіперпараметрів, тому його результати залишаються незміненими.

Для реалізації моделі регресії були використані наступні алгоритми: Linear Regression, Random Forest, SVR, XGBoost.

Розглянемо лістинг коду програмної реалізації моделі регресії без оптимізації гіперпараметрів.

```
# Моделі
models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(),
    "SVR": SVR(),
    "XGBoost": XGBRegressor(eval_metric="rmse")
}
# Навчання моделей
results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
```

```
        results[model_name] = {"MSE": mse, "MAE": mae,
        "R2": r2}
```

У цьому лістингу кожна модель навчається на навчальних даних X_train та y_train за допомогою методу fit. Після навчання модель прогнозує значення для тестових даних X_test за допомогою методу predict. Результати прогнозів порівнюються з реальними значеннями тестової вибірки y_test шляхом обчислення метрик оцінки: середньоквадратичної помилки, середньої абсолютної помилки та коефіцієнта детермінації. Ці метрики дозволяють оцінити якість передбачень для кожної моделі. Усі результати зберігаються в словнику results, де для кожної моделі зазначені значення відповідних метрик.

Щоб покращити точність та адаптивність моделей регресії, було проведено оптимізацію їх гіперпараметрів за допомогою методу GridSearchCV. Цей метод дозволяє систематично перебирати всі можливі комбінації заданих параметрів, навчаючи моделі на різних підмножинах даних та обираючи найкращі параметри на основі результатів п`ятиразової крос-валідації.

Для кожної моделі я визначив окремий набір гіперпараметрів, які мають найбільший вплив на її продуктивність.

Розглянемо лістинг коду програмної реалізації моделі регресії з оптимізації гіперпараметрів.

```
    # Словник гіперпараметрів
    param_grids = {
        "Linear Regression": {},
        "Random Forest": {
            "n_estimators": [50, 100, 200],
            "max_depth": [None, 10, 20],
            "min_samples_split": [2, 5]
        },
        "SVR": {
            "C": [0.1, 1, 10],
            "kernel": ["linear", "rbf"],
```

```
            "gamma": ["scale", "auto"]
        },
        "XGBoost": {
            "n_estimators": [50, 100, 200],
            "max_depth": [3, 6, 10],
            "learning_rate": [0.01, 0.1, 0.2]
        }
}
# Оптимізації Моделей
optimized_results = {}
for model_name, model in models.items():
    grid = GridSearchCV(model,
    param_grids[model_name],
    scoring="neg_mean_squared_error", cv=5,
    n_jobs=-1, verbose=1)
    grid.fit(X_train, y_train)
    best_model = grid.best_estimator_
    y_pred = best_model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    optimized_results[model_name] = {"MSE": mse,
    "MAE": mae, "R2": r2}
```

У цьому лістингу здійснюється оптимізація гіперпараметрів моделей регресії для підвищення точності передбачень. Спочатку створюється словник param_grids, у якому визначені можливі значення гіперпараметрів для кожної моделі.

Далі кожна модель оптимізується за допомогою GridSearchCV, який автоматично перебирає всі комбінації гіперпараметрів із заданої сітки та обирає найкращу модель за метрикою негативної середньоквадратичної

помилки (neg_mean_squared_error). Процес оптимізації здійснюється через п`ятиразову крос-валідацію (cv=5), що дозволяє оцінити стабільність моделі на різних підмножинах даних.

Для виконання кластеризації були використані три основні алгоритми: KMeans, DBSCAN, AgglomerativeClustering та GMM. Ці методи дозволяють групувати дані без попередньо визначених міток, що є корисним для виявлення схожих груп сигналів або аномалій.

Розглянемо лістинг коду програмної реалізації моделі кластеризації без оптимізації гіперпараметрів.

```
# Моделі
models = {
    "KMeans": KMeans(n_clusters=4,
    random_state=42),
    "DBSCAN": DBSCAN(eps=2, min_samples=2),
    "Agglomerative":
    AgglomerativeClustering(n_clusters=4),
    "GMM": GaussianMixture(n_components=4,
    random_state=42)  # Додано GMM
}
# Результати для кожної моделі
clustering_results = {
    "Model": [],
    "Silhouette Score": [],
    "Davies-Bouldin Index": [],
    "Calinski-Harabasz Score": []
}
# Оновлений цикл з метриками
for model_name, model in models.items():
    if model_name == "GMM":
```

```
    labels = model.fit_predict(X_scaled)  # GMM
використовує fit_predict
else:
    labels = model.fit_predict(X_scaled)
n_clusters = len(set(labels)) - (1 if -1 in
labels else 0)  # Враховуємо шум (-1 для
DBSCAN)
if n_clusters > 1:  # Обчислюємо метрики лише
якщо кластери більше одного
    silhouette = silhouette_score(X_scaled,
labels)
    davies_bouldin =
davies_bouldin_score(X_scaled, labels)
    calinski_harabasz =
calinski_harabasz_score(X_scaled, labels)
    clustering_results["Model"].append(model_na
me)
    clustering_results["Silhouette
Score"].append(silhouette)
    clustering_results["Davies-Bouldin
Index"].append(davies_bouldin)
    clustering_results["Calinski-Harabasz
Score"].append(calinski_harabasz)
else:
    clustering_results["Model"].append(model_na
me)
    clustering_results["Silhouette
Score"].append("Only 1 cluster")
    clustering_results["Davies-Bouldin
Index"].append("Only 1 cluster")
```

```
            clustering_results["Calinski-Harabasz
        Score"].append("Only 1 cluster")
```

У цьому лістингу реалізовано розрахунок якості кластеризації для чотирьох алгоритмів: KMeans, DBSCAN, Agglomerative Clustering та Gaussian Mixture Model (GMM). Кожна модель навчається на масштабованих даних X_scaled за допомогою методу fit_predict, який визначає кластери для кожного зразка. Для алгоритму DBSCAN враховується можливість існування шумових точок, які отримують мітку -1.

У випадку, якщо формується лише один кластер, метрики не обчислюються, і результат позначається як "Only 1 cluster". Усі результати зберігаються у словнику clustering_results для подальшого аналізу. Код дозволяє порівняти якість кластеризації для різних алгоритмів на одному наборі даних.

Для досягнення кращої якості кластеризації було проведено оптимізацію гіперпараметрів. Ця оптимізація виконувалася шляхом перебору різних комбінацій параметрів і оцінки результатів за допомогою коефіцієнта силуета.

Розглянемо лістинг коду програмної реалізації моделі кластеризації з оптимізацією.

```
# Моделі
models = {
    "KMeans": KMeans(random_state=42),
    "DBSCAN": DBSCAN(),
    "Agglomerative": AgglomerativeClustering(),
    "GMM": GaussianMixture(random_state=42)
}
# Параметри оптимізації
param_grids = {
    "KMeans": {"n_clusters": [2, 3, 4, 5, 6]},
    "DBSCAN": {"eps": [0.3, 0.5, 0.7],
     "min_samples": [3, 5, 10]},
```

```
    "Agglomerative": {"n_clusters": [2, 3, 4, 5,
    6]},
    "GMM": {"n_components": [2, 3, 4, 5, 6]}
}

# Оптимізація
optimized_results = {}
best_models = {}
for model_name, model in models.items():
    if model_name in param_grids:  # Перевірка
    наявності параметрів
        best_score = -1
        best_params = None
        best_labels = None
        for params in
    ParameterGrid(param_grids[model_name]):
            model.set_params(**params)
            if model_name == "GMM":
                labels =
    model.fit_predict(X_scaled)
            else:
                labels =
    model.fit_predict(X_scaled)
            n_clusters = len(set(labels)) - (1 if -
    1 in labels else 0)
            if n_clusters > 1:
                score = silhouette_score(X_scaled,
    labels)
                if score > best_score:
                    best_score = score
```

```
                    best_params = params
                    best_labels = labels
        optimized_results[model_name] = {
            "Silhouette Score": best_score if
    best_score != -1 else "Only 1 cluster",
            "Best Params": best_params,
        }
        best_models[model_name] = best_labels
# Додаткові метрики для найкращих моделей
clustering_results = {
    "Model": [],
    "Silhouette Score": [],
    "Davies-Bouldin Index": [],
    "Calinski-Harabasz Score": [],
    "Best Params": []
}
for model_name, result in
    optimized_results.items():
    labels = best_models.get(model_name)
    if labels is not None and len(set(labels)) > 1:
        silhouette = silhouette_score(X_scaled,
    labels)
        davies_bouldin =
    davies_bouldin_score(X_scaled, labels)
        calinski_harabasz =
    calinski_harabasz_score(X_scaled, labels)
    else:
        silhouette = "Only 1 cluster"
        davies_bouldin = "Only 1 cluster"
        calinski_harabasz = "Only 1 cluster"
```

```
        clustering_results["Model"].append(model_name)
        clustering_results["Silhouette
        Score"].append(silhouette)
        clustering_results["Davies-Bouldin
        Index"].append(davies_bouldin)
        clustering_results["Calinski-Harabasz
        Score"].append(calinski_harabasz)
        clustering_results["Best
        Params"].append(result["Best Params"])
```

У цьому лістингу реалізовано оптимізацію чотирьох алгоритмів кластеризації: KMeans, DBSCAN, Agglomerative Clustering та Gaussian Mixture Model (GMM). Для кожного алгоритму використовується набір гіперпараметрів, які підбираються за допомогою повного перебору (ParameterGrid). Оптимізація базується на коефіцієнті силуета, який розраховується лише за умови наявності двох або більше кластерів.

На першому етапі для кожної моделі викликається метод fit_predict, що визначає кластери для зразків даних. Найкращі параметри обираються на основі максимального значення коефіцієнта силуета. Якщо модель формує лише один кластер, оптимізація пропускається для таких параметрів.

Результати оптимізації, включно з найкращими параметрами та значеннями метрик, зберігаються у словнику та подаються у вигляді таблиці. Цей код дозволяє проводити порівняння різних алгоритмів кластеризації, а також визначати оптимальні параметри для кожного з них.

Для подальшого використання моделі та стандартизатора вони зберігаються у файл. Це дозволяє завантажувати їх у майбутньому без повторного навчання чи масштабування даних. Розглянемо лістинг коду програмної реалізації цього процесу.

```
import joblib
# Збереження моделі та стандартизатора у файл
```

```
joblib.dump(models["*"], `*.joblib`)  # Збереження
    моделі
joblib.dump(scaler, `scaler.joblib`)  # Збереження
    StandardScaler
```

У цьому лістингу використовується бібліотека joblib для серіалізації об`єктів. Метод joblib.dump зберігає модель, яка відповідає ключу models["*"], у файл із розширенням .joblib.

Окрім моделі, зберігається об`єкт scaler, що відповідає за масштабування даних. Його збереження у файлі scaler.joblib гарантує, що нові дані будуть стандартизовані аналогічно тим, які використовувалися під час навчання.

Отже, нами було реалізовано три основні типи моделей: класифікації, регресії та кластеризації. Для кожної з цих задач було використано сучасні алгоритми машинного навчання. Реалізація моделей включала етапи навчання, оптимізації гіперпараметрів.

Для класифікації було розроблено алгоритми, які дозволяють точно визначати тип об`єктів або середовища на основі даних сигналів. Реалізація моделей регресії спрямована на прогнозування числових параметрів, таких як інтенсивність або глибина сигналу. Алгоритми кластеризації забезпечили можливість групування сигналів і виявлення аномалій без наявності попередньо заданих міток.

Особливу увагу було приділено оптимізації гіперпараметрів, що дозволило суттєво покращити точність моделей та адаптувати їх до специфіки даних. Також були розроблені підходи для збереження моделей і стандартизаторів, що спрощує їх подальше використання.

Усі ці етапи заклали основу для інтеграції моделей у виробниче середовище та створення API для автоматизації аналізу гідроакустичних сигналів.

3.4 Розробка API для використання моделей

Для інтеграції моделі класифікації в програмне середовище було створено API за допомогою фреймворку Fast API. Це API забезпечує взаємодію між користувачем і моделлю, дозволяючи надсилати дані для аналізу та отримувати результат класифікації.

Розглянемо лістинг коду програмної реалізації API:

```
from flask import Flask, request, jsonify
import joblib
import numpy as np
loaded_svm_model =
    joblib.load(`models/model.joblib`)
loaded_scaler = joblib.load(`models/scaler.joblib`)
app = Flask(__name__)
@app.route(`/predict`, methods=[`POST`])
def predict():
    data = request.json[`input_data`]
    input_data_as_numpy_array =
    np.asarray(data).reshape(1, -1)
    std_data =
    loaded_scaler.transform(input_data_as_numpy_arr
    ay)
    prediction = loaded_svm_model.predict(std_data)
    result = `Міна` if prediction[0] == `M` else
    `Камінь`

    return jsonify({`prediction`: result})
if __name__ == `__main__`:
    app.run(debug=True)
```

У цьому коді реалізовано RESTful API, яке забезпечує інтерактивність між користувачем і моделлю класифікації. Спочатку завантажуються

збережені об`єкти: модель класифікації model.joblib і стандартизатор scaler.joblib. Це дозволяє забезпечити ідентичність попередньої обробки нових даних із тією, що застосовувалася під час навчання моделі.

Для роботи API використовується фреймворк Fast API, який створює сервер для обробки HTTP-запитів. Основний маршрут /predict реалізує функціонал прийому POST-запитів із даними у форматі JSON та їх стандартизацію.

Далі стандартизовані дані передаються до моделі, яка виконує класифікацію та повертає передбачений клас. Результат класифікації інтерпретується як "Міна" для класу `M` або "Камінь" для іншого класу. Відповідь формується у вигляді JSON-об`єкта з ключем `prediction`, що дозволяє клієнту отримати чіткий результат.

3.5 Розробка веб-додатку

Для взаємодії з сервісом був розроблений веб-додаток, який забезпечує аналіз гідроакустичних сигналів у зручному інтерфейсі. Додаток дозволяє користувачу взаємодіяти з вхідними даними, запускати аналіз сигналів за допомогою класифікаційної моделі, отримувати результати у текстовому вигляді та візуалізувати оброблені дані.

Розглянемо лістинг коду програмної реалізації сторінки веб-додатку.

```
function App() {
  const [predictionResult, setPredictionResult] =
    useState({ prediction: '', spectrogram: '' });
  const [currentSonarData, setCurrentSonarData] =
    useState(sonarData[0]);
  async function clickHandler() {
    const randomIndex = Math.floor(Math.random() *
    (41 - 0 + 1) + 0);
    const inputData = {
      input_data: sonarData[randomIndex],
```

```
    };
    setCurrentSonarData(sonarData[randomIndex]);
    const result = await
    fetch('http://127.0.0.1:8000/predict', {
      method: 'POST',
      body: JSON.stringify(inputData),
      headers: {
        'Content-Type': 'application/json',
      },
    });
    const data = await result.json();
    setPredictionResult(data);
    const sonarHistory =
    localStorage.getItem('predictionHistory');
    localStorage.setItem(
      'predictionHistory',
      JSON.stringify([...JSON.parse(sonarHistory ||
    '[]'), { sonarData: sonarData[randomIndex],
    result: data.prediction }]),
    );
  }
  return (
    <>
      <main>
        <Container className="px-20 py-4">
          <h1 className="text-center text-lg">
            <span className="font-
    semibold">Аналізатор гідроакустичних
    сигналів</span>
          </h1>
```

```
            <TopBar data={currentSonarData} />
            <Container className="flex items-center
        gap-x-4">
              <Button
        onClick={clickHandler}>Проаналізувати</Button>
              <p>Результат аналізу оброблених
        сигналів: {predictionResult.prediction}</p>
            </Container>
            <hr className="my-4" />
            <Container>
              <Container>
                <h2 className="font-
        semibold">Візуалізація результатів</h2>
              </Container>
             <img
        src={`${predictionResult.
        spectrogram}`} className="w-1/2 "
        alt="Спектрограма" />
            </Container>
          </Container>
        </main>
      </>
    );
  }
```

У цьомі лістингу реалізовано веб-додаток, який дає можливість зручно використовувати розроблену API для аналізу гідроакустичних сигналів. Даний додаток реалізовано за допомогою бібліотеки React та має низку функцій, які забезпечують інтерактивність для користувача.

Функціональність додатка починається з оголошення двох основних станів: predictionResult та currentSonarData. Стан predictionResult

використовується для зберігання результатів аналізу сигналу, таких як прогнозований клас і спектрограма, тоді як currentSonarData відповідає за відображення поточного набору даних із сонару.

Ключовий елемент додатка - це функція clickHandler, яка викликається при натисканні кнопки "Проаналізувати". Ця функція генерує випадковий індекс, що використовується для вибору випадкових даних із сонару, які передаються серверу для аналізу. Перед надсиланням запиту серверу функція формує об'єкт inputData та змінює стан currentSonarData відповідно до обраного набору даних.

Запит до серверу виконується через функцію fetch, яка надсилає HTTP-запит методом POST на адресу локального сервера http://127.0.0.1:8000/predict. У тілі запиту передаються дані в форматі JSON. Відповідь сервера містить результат прогнозу (клас об'єкта) та спектрограму у форматі base64. Ці дані зберігаються в стані predictionResult.

Окрім цього, функція clickHandler зберігає історію попередніх прогнозів у локальному сховищі браузера (localStorage). Для цього отримується поточна історія прогнозів, яка оновлюється новим записом, і зберігається у форматі JSON.

Інтерфейс додатка побудовано таким чином, щоб забезпечити користувачеві зручну взаємодію. Заголовок "Аналізатор гідроакустичних сигналів" відображає призначення програми. Компонент TopBar виводить поточні дані з сонару, які змінюються після кожного запиту. Основна частина інтерфейсу включає кнопку "Проаналізувати", результат аналізу (прогнозований клас) та блок для візуалізації спектрограми.

Спектрограма виводиться як зображення, URL якого динамічно формується на основі даних із predictionResult. Це дозволяє користувачу одразу побачити результат аналізу в графічному вигляді.

Таким чином, даний додаток забезпечує інтерактивний та зрозумілий функціонал для аналізу гідроакустичних сигналів, дозволяючи зручно працювати з даними сонару, отримувати прогнози та візуалізувати результати.

Розглянемо рисунок, на якому зображено інтерфейс розробленого веб-додатку.

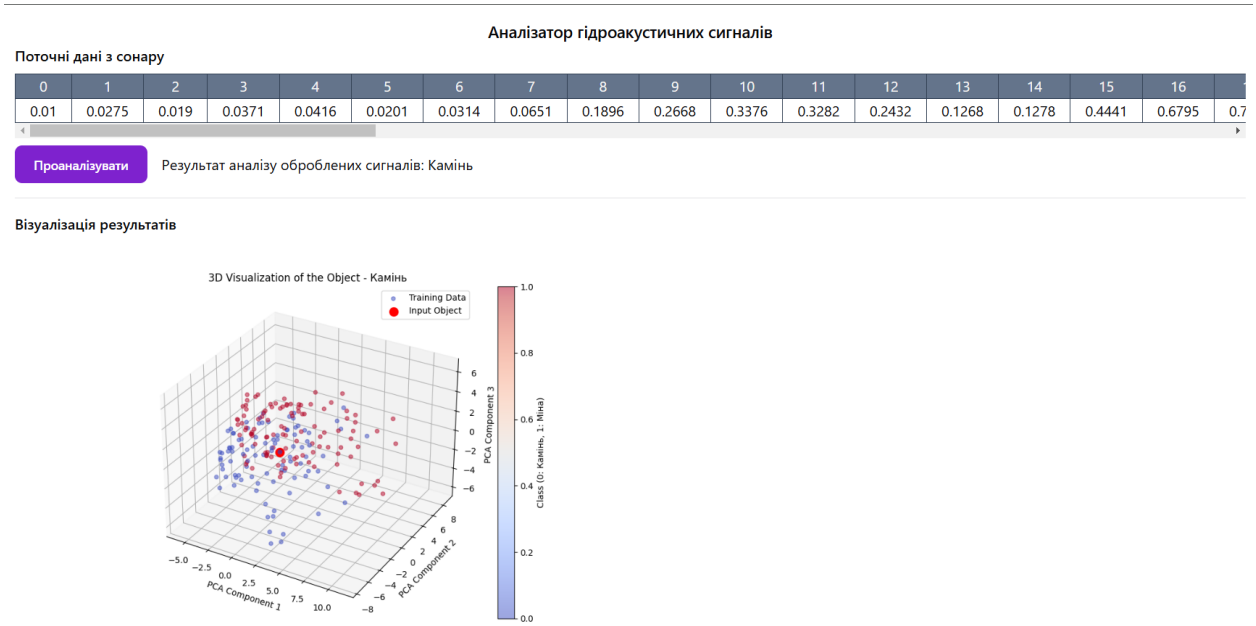

Рисунок 3.1 – Інтерфейс розробленого веб-додатку

На рисунку 3.1 зображено інтерфейс розробленого веб-додатку. На головному екрані є таблиця з числовими даними сигналу, які будуть аналізуватися. Кнопка "Проаналізувати" дозволяє запустити процес аналізу, після чого виводиться текстовий результат, наприклад, "Камінь" чи "Міна", що вказує на класифікацію об'єкта.

Результати аналізу доповнюються візуалізацією у вигляді тривимірного графіка, де велика червона точка представляє вхідний сигнал, а всі інші точки - тренувальні дані. Шкала кольорів додатково допомагає зрозуміти класифікацію сигналу. Додаток автоматично зберігає історію аналізів у локальному сховищі браузера, дозволяючи користувачу повернутися до попередніх результатів.

Інтерфейс додатку простий і інтуїтивний, що дозволяє легко працювати навіть користувачам без технічних знань. Він поєднує ефективність алгоритмів обробки з зручністю візуалізації, надаючи потужний інструмент для аналізу гідроакустичних сигналів.

Отже, нами виконано розробку програмного комплексу для аналізу гідроакустичних сигналів, який складається з кількох ключових компонентів: вибору технологій, підготовки даних, реалізації моделей машинного навчання, створення API для взаємодії з моделями та розробки веб-додатку.

Було проведено порівняльний аналіз мов програмування і інструментів, результатом якого став вибір Python для реалізації моделей, FastAPI для створення API та React для розробки веб-інтерфейсу. Цей вибір забезпечив ефективну інтеграцію всіх компонентів системи.

Для роботи з даними виконано їх попередню обробку: перевірено якість, виконано поділ на навчальні і тестові вибірки та застосовано нормалізацію. Це забезпечило підготовку якісного набору даних для побудови моделей. Реалізовано моделі класифікації, регресії та кластеризації, які включали як базові версії, так і оптимізовані варіанти з використанням гіперпараметрів. Показано, що оптимізація значно покращує продуктивність моделей.

На основі створених моделей розроблено API, яке дозволяє взаємодіяти з моделями класифікації. Завдяки цьому створено веб-додаток, який забезпечує простий та інтуїтивний інтерфейс для аналізу даних. Додаток дозволяє користувачу отримувати результати класифікації, переглядати спектрограми і зберігати історію запитів для подальшого використання.

Розроблений програмний комплекс дозволяє ефективно вирішувати завдання аналізу гідроакустичних сигналів, поєднуючи надійність моделей машинного навчання з зручністю роботи кінцевого користувача через веб-інтерфейс.

3.6 Розробка інструкції користувача

Для того, щоб розпочати роботу з веб-додатком для аналізу гідроакустичних сигналів, необхідно відкрити головну сторінку додатка. На цій сторінці знаходиться таблиця з даними сонару, які використовуються для аналізу, а також кнопка «Проаналізувати».

Натискання на кнопку «Проаналізувати» запускає процес вибору випадкового сигналу із таблиці та передає його для аналізу через сервер. Після завершення обробки користувач отримує текстовий результат класифікації, який відображається під кнопкою, наприклад, «Камінь» або «Міна». Крім текстового результату, на сторінці відображається графічна візуалізація у вигляді тривимірного графіка, де червоною точкою позначено вхідний сигнал, а синіми точками — тренувальні дані. Шкала кольорів допомагає зрозуміти розподіл класифікації.

Для збереження результатів аналізу веб-додаток автоматично записує дані до локального сховища браузера, що дозволяє переглянути історію запитів у подальшому. Інтерфейс додатка інтуїтивно зрозумілий, що дає можливість зручно працювати навіть без технічних знань.

Для роботи з API як окремим сервісом користувач може надсилати POST-запити на сервер. Щоб отримати прогноз, потрібно підготувати JSON-запит із даними сигналу у форматі:

```
{ "input_data": [0.01, 0.02, 0.03, ..., 0.07] }
```

Запит надсилається на адресу http://127.0.0.1:8000/predict, після чого сервер повертає відповідь у вигляді JSON-об’єкта:

```
{ "prediction": "Камінь", visualization: image }
```

API дозволяє інтегрувати функціонал аналізу сигналів у сторонні програми або автоматизувати аналіз через запити HTTP. Це зручно для розробників, які можуть використовувати API для обробки даних у своїх власних системах або автоматизації процесів.

4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ

4.1 Наукова новизна

Наука як динамічна система знань постійно формує нові підходи, методи та результати, що сприяють її розвитку. Важливим аспектом наукового процесу є новизна, яка визначає якісні зміни у дослідженнях та їх вплив на науково-технічний прогрес. Новизна є основою інноваційного розвитку, адже саме вона сприяє не лише отриманню нових знань, але й їх практичному застосуванню. У контексті цієї роботи новизна полягає у створенні нового програмного комплексу для аналізу гідроакустичних сигналів із застосуванням сучасних методів машинного навчання.

Новизна дослідження має кілька рівнів, які відображають основні елементи виконаної роботи. По-перше, розглядається новий об’єкт дослідження: аналіз та ідентифікація гідроакустичних сигналів для класифікації об'єктів, таких як "Камінь" і "Міна". Ця задача поставлена та вирішена вперше, з урахуванням специфічних особливостей сигналів і необхідності інтеграції різних інструментів обробки даних.

По-друге, в роботі запропоновано нові підходи до розв'язання задачі класифікації, що включають оптимізацію гіперпараметрів моделей машинного навчання. Це дозволило підвищити точність та узгодженість результатів, що є важливим для практичного застосування. Використання методів класифікації доповнено реалізацією моделей регресії та кластеризації, які розширюють можливості програмного комплексу.

По-третє, новизна проявляється в розробці програмного забезпечення, яке об'єднує автоматизовану обробку даних, інтеграцію моделей через API та інтерактивний веб-додаток. Такий підхід забезпечує доступність і зручність роботи з результатами для кінцевого користувача, що важливо для впровадження в реальні умови.

Особливу увагу приділено питанням діагностики ступеня новизни. Отримані результати не лише підтверджують ефективність запропонованих

підходів, але й демонструють їх адаптованість до специфіки гідроакустичних даних. Це дозволяє зробити висновок про те, що створений комплекс має потенціал для подальшого розвитку й застосування у різних галузях, пов’язаних із аналізом акустичних сигналів.

Таким чином, наукова новизна роботи проявляється у вирішенні нової наукової задачі, розробці комплексного підходу до аналізу даних та інтеграції сучасних технологій у практичне програмне забезпечення. Це закладає основи для подальших досліджень та практичної реалізації результатів у суміжних галузях науки й техніки.

4.2 Розробка методик проведення дослідження

У ході розробки методик для аналізу гідроакустичних сигналів було створено структурований підхід, що включає чотири основні етапи: попередню обробку даних, створення моделей класифікації, регресії та кластеризації, оптимізацію моделей та інтерпретацію отриманих результатів. Цей підхід дозволяє забезпечити надійність, точність та адаптивність розроблених алгоритмів у реальних умовах.

На етапі попередньої обробки даних було забезпечено підготовку даних до подальшого навчання моделей. Основою для дослідження став датасет, що складається з 60 числових характеристик, які описують частотні ознаки гідроакустичних сигналів. Класи цільової змінної відповідають двом об`єктам: міна (позначена як "M") та камінь (позначений як "R"). Структура датасету є збалансованою, що забезпечує рівні умови для навчання моделей, мінімізуючи ризик переважання одного класу.

Для роботи з даними було виконано їх зчитування, базовий аналіз структури, перетворення текстових міток у числові значення.

Щоб уникнути перенавчання та оцінити узагальнюючу здатність моделей, дані було поділено на тренувальний набір та тестовий набір. Цей підхід дозволив створити чіткий розподіл між даними, на яких моделі навчаються, та даними, на яких вони тестуються.

Завершальним кроком обробки стало масштабування ознак. Усі числові параметри були приведені до стандартного масштабу з нульовим середнім значенням і одиничним стандартним відхиленням. Це забезпечило однакову вагу для кожної ознаки, що особливо важливо для алгоритмів, чутливих до масштабу.

Після етапу обробки даних, наступним був етап створення моделей включав розробку та навчання алгоритмів для трьох основних типів задач: класифікації, регресії та кластеризації.

Для класифікації використовувалися моделі, спрямовані на визначення типу об`єкта за характеристиками сигналів. У задачах регресії моделі були налаштовані на прогнозування числових параметрів сигналів. У кластеризації застосовувалися підходи, що дозволяють групувати дані на основі схожих характеристик.

Наступним етапом стала оптимізація моделей, яка була спрямована на підвищення їхньої точності та продуктивності шляхом налаштування гіперпараметрів. Для цього використовувалися сучасні методи оптимізації, такі як Grid Search і ParameterGrid.

Оптимізація дала змогу підвищити точність і узагальнювальну здатність моделей, зробивши їх придатними для використання у реальних задачах, але для деяких алгоритмів, оптимізація могла погіршити результат.

Після аналізу результатів було виконано інтерпретацію результатів. Було проведено оцінювання моделей на основі метрик точності, чутливості та специфічності. Візуалізація результатів, зокрема графіки та матриці плутанини, дозволила оцінити здатність моделей до правильного аналізу даних та правильного вибору необхідної моделі для аналізу гідроакустичних сигналів.

4.3 Обробка результатів дослідження, аналіз адекватності моделей та оцінка технічної ефективності

У цьому підрозділі проведено детальний аналіз результатів роботи моделей класифікації, регресії та кластеризації, їх адекватність для поставлених задач, а також технічна ефективність. Розглядаються метрики до і після оптимізації гіперпараметрів із зазначенням точних числових значень, що дозволяє оцінити якість кожної моделі.

Для класифікаційних моделей основними метриками виступають точність та F1-міра. Для їх аналізу було використано дані до і після оптимізації гіперпараметрів. Також важливою складовою є візуалізація матриці плутанини, яка дозволяє оцінити кількість правильних і хибних передбачень для кожного класу.

Розглянемо рисунок, на якому зображений графік результату роботи моделі класифікації без оптимізації гіперпараметрів.

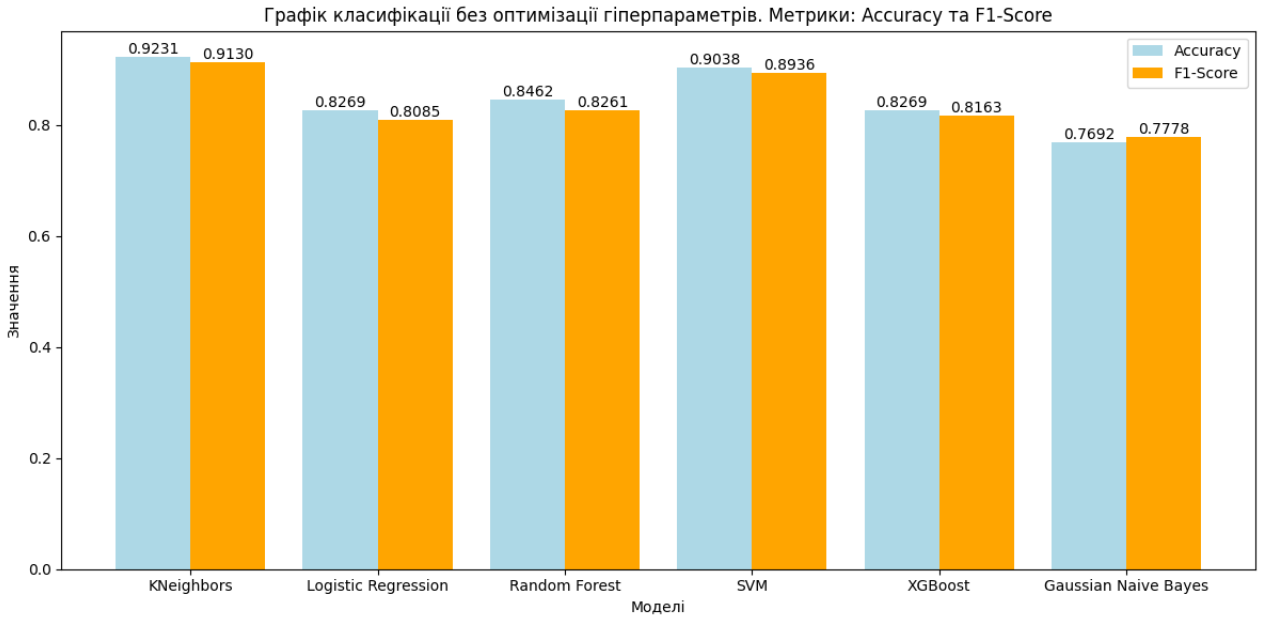

Рисунок 4.1 – Графік результату роботи моделі класифікації без оптимізації гіперпараметрів

На рисунку 4.1 зображено графік, який описує метрики точності та F1-міри для моделей класифікації до оптимізації гіперпараметрів. Видно, що

модель KNeighbors має найвищі показники точності, яка дорівнює 0.92 та F1-міри, яка дорівнює 0.91. На другому місці за результативністю знаходиться модель SVM з точністю 0.90 і F1-мірою 0.89, що свідчить про її здатність ефективно розпізнавати класи. Інші моделі, такі як Logistic Regression, Random Forest і XGBoost, мають схожі результати з точністю близько 0.82, але поступаються за обома метриками. Найгірший результат демонструє Gaussian Naive Bayes з точністю 0.76 і F1-мірою 0.77.

Ці результати свідчать, що навіть без оптимізації моделі KNeighbors та SVM демонструють високу ефективність у класифікації даних.

Розглянемо рисунок, на якому зображені матриці плутанини для моделей класифікації без оптимізації гіперпараметрів.

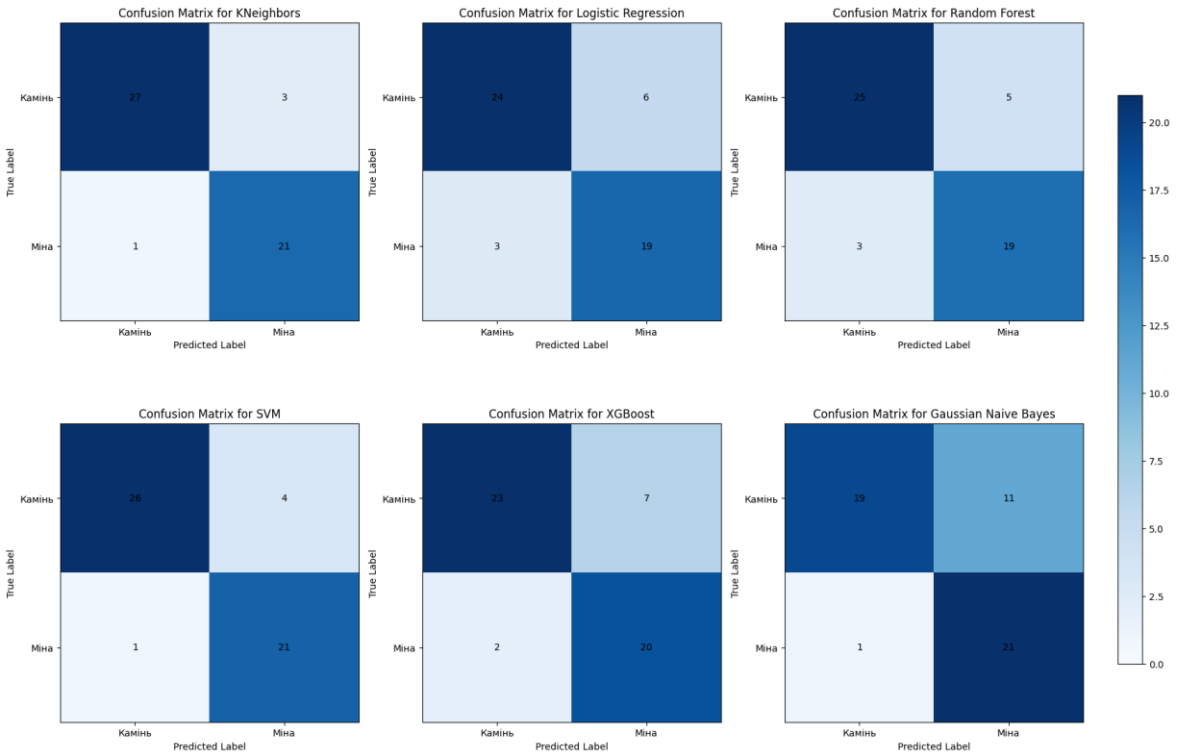

Рисунок 4.2 – Матриці плутанини для моделей класифікації без оптимізації гіперпараметрів

На рисунку 4.2 зображені матриці плутанини для кожної класифікаційної моделі без оптимізації гіперпараметрів. Ці матриці дозволяють оцінити кількість правильних і хибних передбачень для кожного класу.

— KNeighbors: Модель правильно класифікувала 27 із 30 об`єктів класу "Камінь" та 21 із 22 об`єктів класу "Міна", допустивши лише 3 хибних передбачення для класу "Камінь" та 1 для класу "Міна". Це підтверджує високу точність моделі навіть без оптимізації.

— Logistic Regression: Модель допустила 6 хибних передбачень для класу "Камінь" і 3 для класу "Міна". Загалом, цей алгоритм працює добре, але поступається моделям KNeighbors та SVM.

— Random Forest: Модель правильно класифікувала 25 об`єктів класу "Камінь" і 19 об`єктів класу "Міна", але допустила 5 і 3 хибних передбачення відповідно. Це свідчить про необхідність оптимізації для підвищення точності.

— SVM: Алгоритм допустив лише 4 хибних передбачення для класу "Камінь" та 1 для класу "Міна". Це один із найкращих результатів серед моделей до оптимізації.

— XGBoost: Модель зробила 7 хибних передбачень для класу "Камінь" та 2 для класу "Міна". Цей результат є середнім і свідчить про потребу в налаштуванні гіперпараметрів.

— Gaussian Naive Bayes: Алгоритм допустив 11 хибних передбачень для класу "Камінь", що є найгіршим результатом серед усіх моделей. Водночас для класу "Міна" кількість хибних передбачень становить 1, але загальна точність є недостатньою для ефективної класифікації.

Матриця плутанини до оптимізації показує, що моделі KNeighbors і SVM є найкращими з точки зору точності передбачень. Обидві моделі демонструють високу точність навіть без оптимізації. Водночас Gaussian Naive Bayes та XGBoost потребують суттєвого покращення через значну кількість хибних передбачень, що робить їх менш придатними для задач класифікації в їх базовій конфігурації.

Ці результати підкреслюють важливість налаштування гіперпараметрів для досягнення максимальної точності моделей, особливо для алгоритмів, які мають середню продуктивність у базовому варіанті.

Розглянемо рисунок, на якому зображений графік результату роботи моделі класифікації з оптимізацією гіперпараметрів.

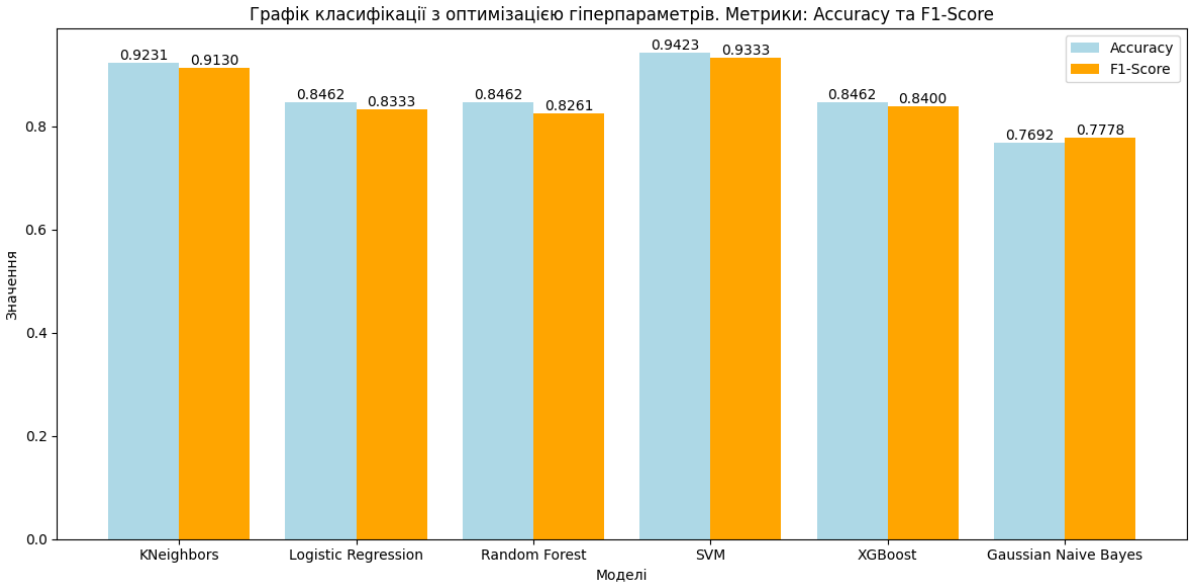

Рисунок 4.3 - Графік результату роботи моделі класифікації з оптимізацією гіперпараметрів

На рисунку 4.3 зображено графік, який описує точність та F1-міру для моделей класифікації після оптимізації гіперпараметрів. Видно, що модель SVM досягла найвищих показників точності 0.94 та F1-міри 0.93, що свідчить про її здатність ефективно розпізнавати об`єкти після налаштування параметрів. Ця модель демонструє стабільно високі результати серед усіх розглянутих.

Модель KNeighbors зберегла стабільно високі результати, маючи точність 0.92 та F1-міру 0.91. Це вказує на те, що її продуктивність була високою навіть без оптимізації, проте після налаштування вона змогла утримати ці показники.

Random Forest також продемонструвала значне покращення, піднявши точність до 0.86 та F1-міру до 0.86. Це свідчить про ефективність оптимізації для цієї моделі, яка стала більш надійною в класифікації.

Модель XGBoost покращила свої результати, досягнувши точності 0.84 та F1-міри 0.84. Хоча її продуктивність не є найвищою, вона залишається стабільною та надійною після оптимізації.

Найгірший результат знову показала модель Gaussian Naive Bayes з точністю 0.76 та F1-мірою 0.77. Це свідчить про обмежену здатність цієї моделі обробляти складні набори даних, навіть після налаштування гіперпараметрів.

Ці результати підкреслюють, що моделі SVM та KNeighbors залишаються найбільш ефективними для задач класифікації навіть після оптимізації. Оптимізація гіперпараметрів також виявилася надзвичайно корисною для моделей Random Forest і XGBoost, які покращили свої показники, тоді як Gaussian Naive Bayes залишилася найслабшою серед усіх розглянутих моделей.

Розглянемо рисунок, на якому зображена матриця плутанини для моделі класифікації з оптимізацією гіперпараметрів.

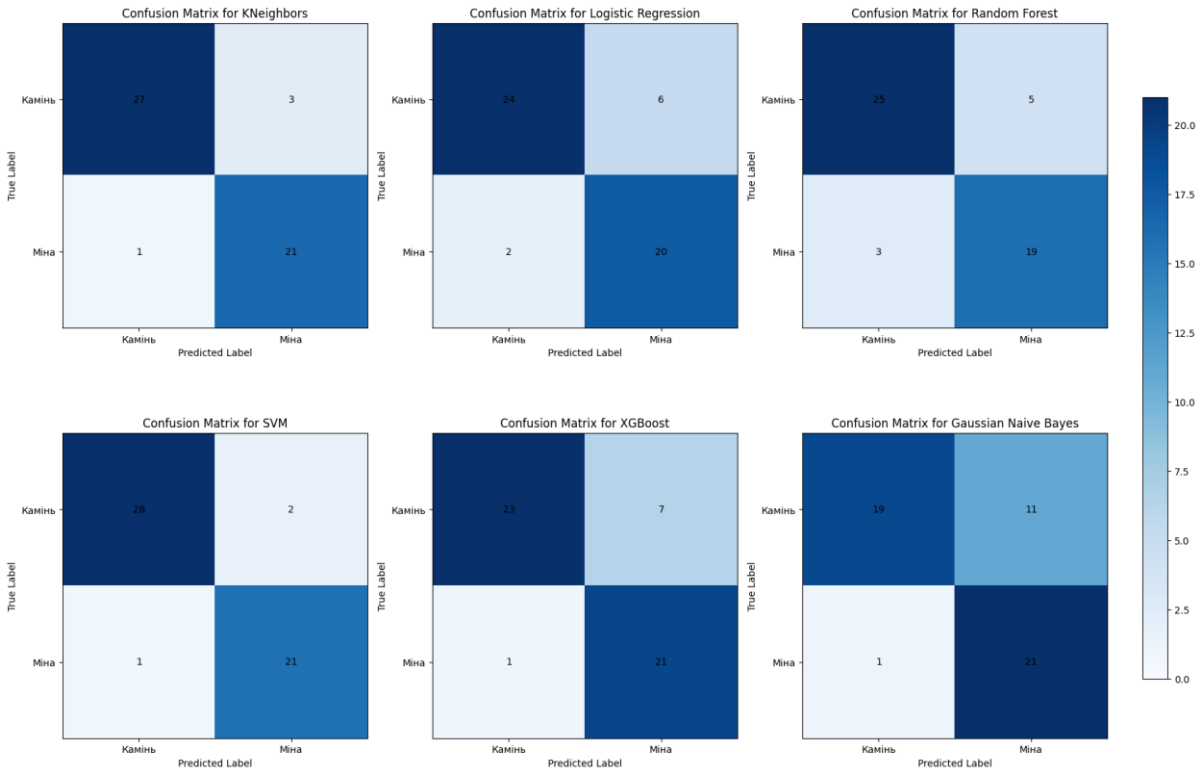

Рисунок 4.4 - Матриця плутанини для моделі класифікації з оптимізацією гіперпараметрів

На рисунку 4.4 зображені матриці плутанини для кожної класифікаційної моделі з оптимізацією гіперпараметрів. Ці матриці дозволяють оцінити кількість правильних і хибних передбачень для кожного класу.

— SVM: Після оптимізації ця модель допустила лише 2 хибних передбачення для класу "Камінь" і 1 для класу "Міна", правильно класифікувавши 28 об`єктів класу "Камінь" і 21 об`єкт класу "Міна". Це найкращий результат серед усіх моделей.

— KNeighbors: Модель показала схожі до SVM результати, правильно класифікувавши 27 об`єктів класу "Камінь" і 21 об`єкт класу "Міна", допустивши 3 та 1 хибних передбачення відповідно.

— Random Forest: Після оптимізації модель покращила свої результати, правильно класифікувавши 25 об`єктів класу "Камінь" і 20 об`єктів класу "Міна", допустивши 5 та 2 хибних передбачення.

— XGBoost: Модель правильно класифікувала 23 об`єкти класу "Камінь" і 21 об`єкт класу "Міна", але допустила 7 і 1 хибних передбачення відповідно.

— Gaussian Naive Bayes: Залишилася найгіршою моделлю з 11 хибними передбаченнями для класу "Камінь" і 1 для класу "Міна", правильно класифікувавши лише 19 об`єктів класу "Камінь" і 21 об`єкт класу "Міна".

Аналіз матриць плутанини після оптимізації підтверджує значне покращення продуктивності моделей, таких як SVM, Random Forest та XGBoost. Найбільше покращення спостерігається у SVM, яка стала лідером серед усіх моделей, зменшивши кількість хибних передбачень до мінімуму. Gaussian Naive Bayes, попри свою простоту, залишилася найслабшою моделлю, що підтверджує її низьку ефективність для задач класифікації даного типу даних.

Ці результати демонструють важливість оптимізації гіперпараметрів для підвищення точності та стабільності передбачень, особливо для моделей, які мають середню продуктивність у базовій конфігурації.

Для моделей регресії основними метриками виступають середньоквадратична помилка (MSE), середня абсолютна помилка (MAE) та коефіцієнт детермінації (R^2). Ці метрики дозволяють оцінити точність, узгодженість та адекватність моделей у прогнозуванні числових значень. Також важливою складовою є візуалізація розподілу залишків моделей, яка описує різницю між фактичними та передбаченими значеннями.

Розглянемо рисунок, на якому зображений графік результату роботи моделі регресії без оптимізації гіперпараметрів.

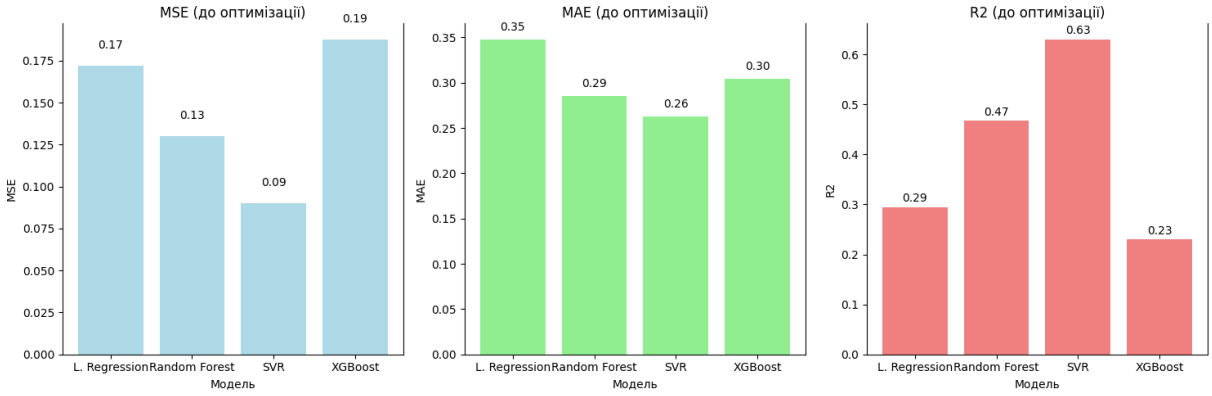

Рисунок 4.5 – Графік результату роботи моделі регресії без оптимізації гіперпараметрів

На рисунку 4.5 зображено графік, який описує основні метрики для моделей регресії до оптимізації гіперпараметрів: середньоквадратична помилка (MSE), середня абсолютна помилка (MAE) та коефіцієнт детермінації (R^2).

Видно, що модель SVR демонструє найкращі результати з MSE = 0.09, MAE = 0.26 та $R^2 = 0.63$. Ці показники вказують на те, що навіть без оптимізації модель має високу здатність точно описувати залежності в даних.

На другому місці за результативністю знаходиться модель Random Forest, яка досягла MSE = 0.13, MAE = 0.29 та $R^2 = 0.47$. Це свідчить про її потенціал, який можна покращити шляхом налаштування гіперпараметрів.

Модель XGBoost демонструє посередні результати з MSE = 0.19, MAE = 0.30 та R^2 = 0.23. Такий рівень точності свідчить про необхідність додаткової оптимізації параметрів.

Найгірший результат спостерігається у моделі Лінійна регресія, яка має MSE = 0.17, MAE = 0.35 та R^2 = 0.29. Ця модель демонструє обмежену здатність до обробки складних залежностей у даних, що робить її менш ефективною у порівнянні з іншими алгоритмами.

Ці результати показують, що навіть до оптимізації модель SVR є найкращим вибором для задач регресії, а оптимізація може покращити результати моделі.

Розглянемо рисунок, на якому зображений розподіл залишків моделей регресії без оптимізації гіперпараметрів.

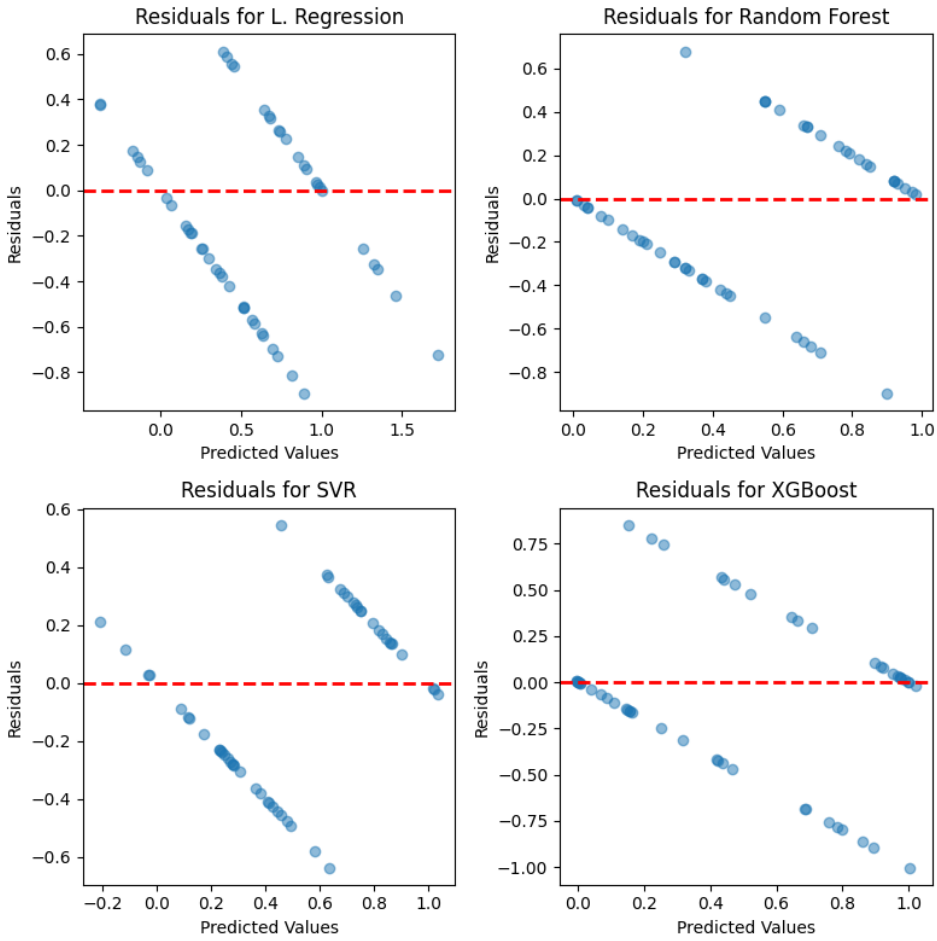

Рисунок 4.6 - Розподіл залишків моделей регресії без оптимізації гіперпараметрів

На рисунку 4.6 представлено розподіл залишків (різниці між передбаченими та фактичними значеннями) для кожної моделі регресії до оптимізації гіперпараметрів. Цей аналіз дозволяє оцінити, наскільки адекватно моделі прогнозують результати та чи існують систематичні відхилення.

— Linear Regression (L. Regression): Розподіл залишків демонструє значні відхилення, які мають тенденцію до групування в певних діапазонах. На графіку помітно, що залишки далеко від нульової лінії, що свідчить про низьку точність прогнозів і наявність систематичних похибок. Це підтверджує обмежену здатність цієї моделі працювати зі складними даними.

— Random Forest: Розподіл залишків є більш рівномірним у порівнянні з лінійною регресією. Основна частина залишків знаходиться поблизу нульової лінії, проте все ще помітні деякі відхилення, особливо для високих і низьких значень. Це вказує на потенціал моделі, який може бути розкритий через оптимізацію.

— SVR (Support Vector Regression): Розподіл залишків для цієї моделі є найкращим серед усіх представлених. Залишки рівномірно розподілені навколо нульової лінії, що свідчить про високу точність моделі навіть до оптимізації гіперпараметрів.

— XGBoost: Розподіл залишків менш збалансований у порівнянні зі SVR та Random Forest. Деякі залишки значно відхиляються від нульової лінії, що вказує на наявність похибок. Це свідчить про потребу в налаштуванні параметрів моделі.

На основі розподілу залишків видно, що модель SVR демонструє найкращі результати до оптимізації, тоді як лінійна регресія має найгірші показники. Моделі Random Forest та XGBoost демонструють помірну точність.

Розглянемо рисунок, на якому зображений графік метрик моделей регресії після оптимізації гіперпараметрів.

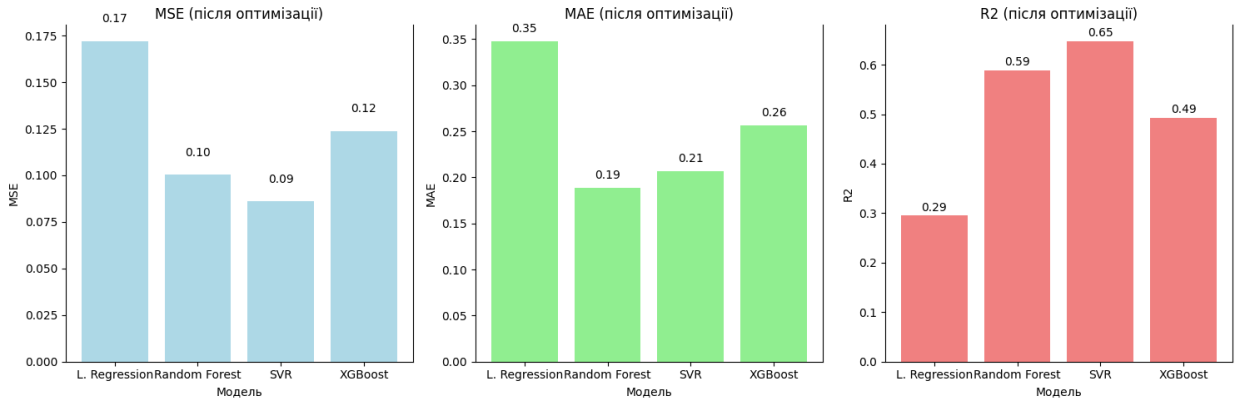

Рисунок 4.7 - Графік результату роботи моделі регресії з оптимізацією гіперпараметрів

На рисунку 4.7 зображені результати моделей регресії після оптимізації гіперпараметрів, представлені на рисунку 4.7. Для аналізу використовувалися такі основні метрики, як середньоквадратична помилка (MSE), середня абсолютна помилка (MAE) та коефіцієнт детермінації (R^2). Ці метрики дозволяють оцінити точність, стабільність та адекватність моделей у прогнозуванні.

Після оптимізації модель SVR показала найкращі результати серед усіх розглянутих алгоритмів. Її значення метрик складають MSE = 0.09, MAE = 0.21 та R^2 = 0.65. Такі показники свідчать про високу точність прогнозування та здатність моделі ефективно відображати залежності в даних після налаштування параметрів.

Другий за результативністю є алгоритм Random Forest, який після оптимізації досягнув MSE = 0.10, MAE = 0.19 та R^2 = 0.59. Оптимізація дозволила цій моделі значно покращити свої показники точності, що робить її ефективним вибором для регресії.

Модель XGBoost також продемонструвала позитивні зміни після оптимізації, зокрема MSE знизився до 0.12, MAE до 0.26, а R^2 підвищився до 0.49. Хоча її продуктивність поступається SVR та Random Forest, вона залишається стабільною і показує хороші результати.

Лінійна регресія, незважаючи на проведену оптимізацію, продовжує демонструвати найгірші результати з-поміж усіх моделей. Її метрики залишаються відносно низькими: MSE = 0.17, MAE = 0.35 та R^2 = 0.29. Це вказує на те, що модель має обмежену здатність обробляти складні залежності у даних і поступається більш сучасним алгоритмам.

Загалом, результати оптимізації показують значне покращення продуктивності для SVR і Random Forest, які є лідерами серед моделей регресії. Модель XGBoost демонструє середню ефективність, тоді як лінійна регресія потребує суттєвих покращень для застосування в задачах зі складними залежностями.

Розглянемо рисунок, на якому представлено розподіл залишків для моделей регресії після оптимізації гіперпараметрів.

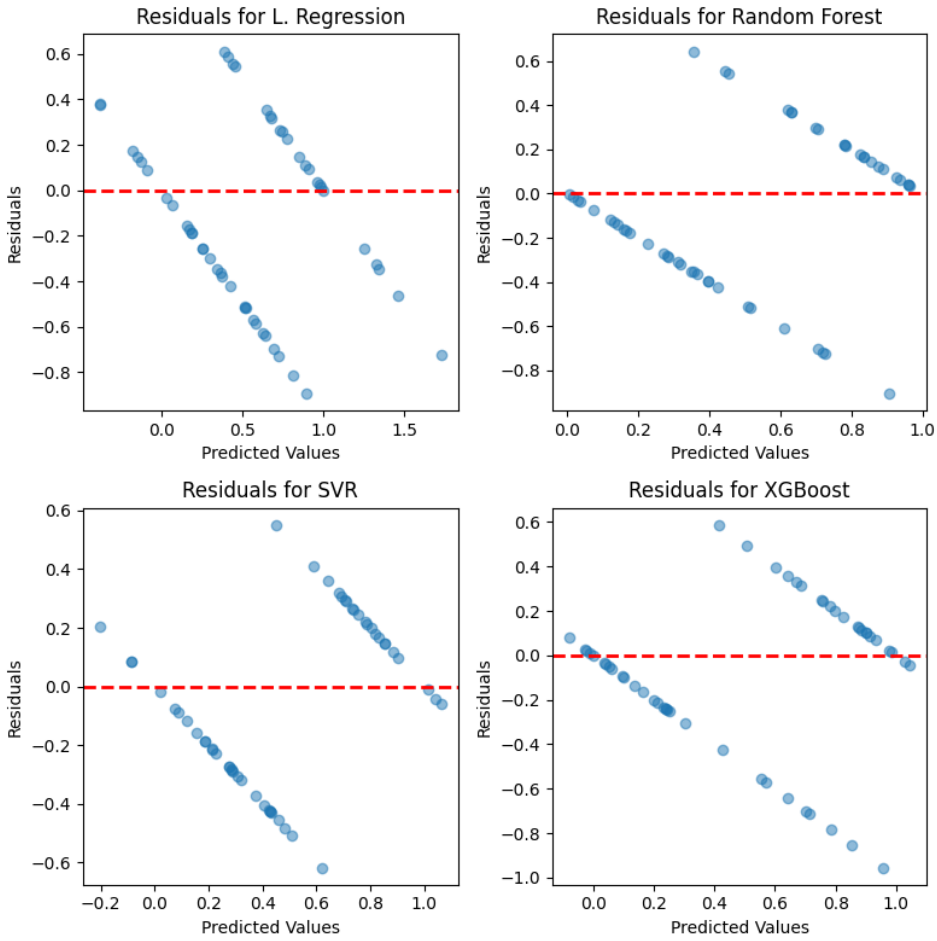

Рисунок 4.8 - Розподіл залишків моделей регресії після оптимізації гіперпараметрів

На рисунку 4.8 представлено графіки залишків для кожної моделі регресії після оптимізації. Залишки, які відображають різницю між передбаченими та фактичними значеннями, дозволяють оцінити точність моделі та виявити можливі систематичні похибки.

— Лінійна регресія (L. Regression): Після оптимізації залишки для лінійної регресії все ще демонструють значні систематичні відхилення. Розподіл залишків має помітну тенденцію до лінійної залежності, що свідчить про недостатню адаптацію моделі до складних залежностей у даних. Оптимізація суттєво не покращила результати цієї моделі.

— Random Forest: Для Random Forest після оптимізації залишки розподілені значно рівномірніше навколо нульової лінії. Це вказує на покращення точності моделі. Проте все ще спостерігаються деякі відхилення для крайніх значень, що може бути викликано складністю даних.

— SVR (Support Vector Regression): Модель SVR після оптимізації демонструє один із найкращих розподілів залишків. Вони рівномірно розподілені навколо нульової лінії, що вказує на відсутність систематичних похибок. Цей результат свідчить про здатність моделі точно описувати залежності навіть після налаштування.

— XGBoost: Після оптимізації розподіл залишків у XGBoost покращився, але все ще спостерігаються відхилення для крайніх значень. Це вказує на те, що модель добре адаптується до даних, але може бути чутливою до їх складності.

Аналіз залишків після оптимізації показує значне покращення у моделях SVR та Random Forest. Ці моделі демонструють найбільш рівномірний розподіл залишків навколо нульової лінії, що свідчить про високу точність і надійність прогнозів. Лінійна регресія, навіть після оптимізації, демонструє систематичні похибки, що підтверджує її обмежену здатність до обробки складних даних. XGBoost також демонструє покращення, проте залишається чутливим до крайніх значень.

Для оцінки моделей кластеризації використовувалися три основні метрики: скоригований індекс RAND (ARI), який вимірює точність групування порівняно з реальними даними; гомогенність (Homogeneity), яка оцінює однорідність кластерів; та V-міра (V-measure), що є середнім гармонічним значенням гомогенності та повноти.

Розглянемо рисунок, на якому зображено графік метрик моделей кластеризації до оптимізації гіперпараметрів.

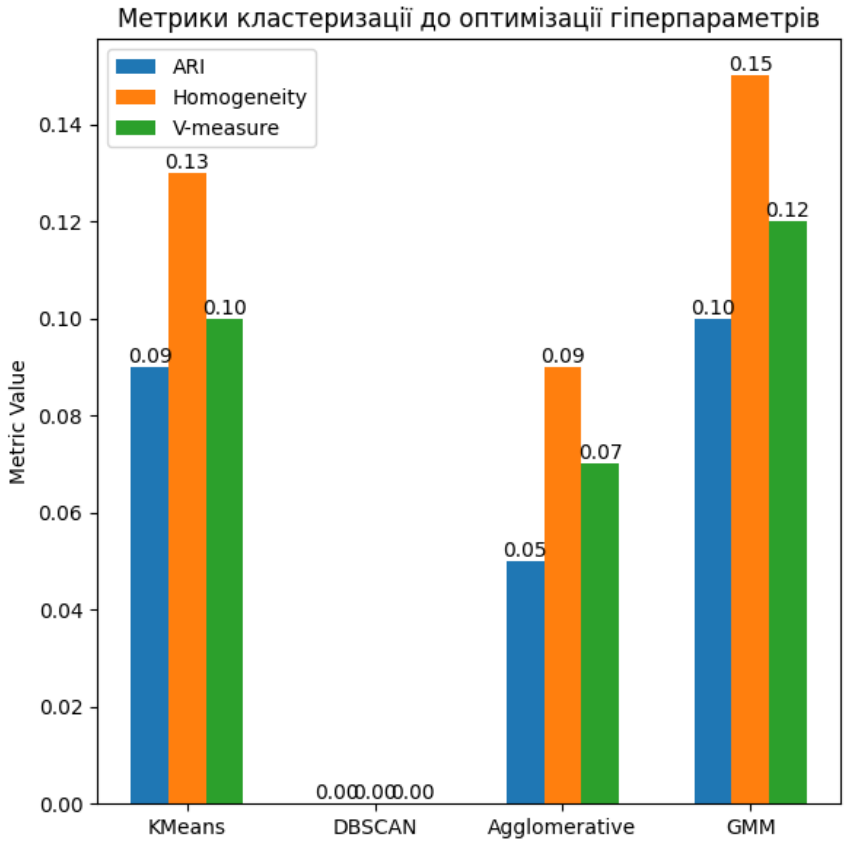

Рисунок 4.9 - Графік метрик моделей кластеризації до оптимізації гіперпараметрів

На рисунку 4.9 представлено результати моделей кластеризації до оптимізації гіперпараметрів. Видно, що модель KMeans демонструє найвищі результати серед усіх моделей за всіма трьома метриками: ARI = 0.09, Homogeneity = 0.13, V-measure = 0.10. Хоча ці показники є низькими, вони свідчать про часткову здатність моделі групувати дані.

Модель GMM показує дещо кращі результати, зокрема Homogeneity = 0.15, що є найвищим показником серед усіх моделей до оптимізації. Її ARI дорівнює 0.10, а V-measure становить 0.12. Це свідчить про те, що GMM є другою за якістю кластеризації.

Модель Agglomerative Clustering демонструє посередні результати (ARI = 0.05, Homogeneity = 0.09, V-measure = 0.07), що вказує на її менш ефективну роботу. Найгірший результат має модель DBSCAN, яка не змогла сформувати жодного валідного кластеру, тому її метрики дорівнюють нулю.

Таким чином, моделі KMeans та GMM є найбільш ефективними серед розглянутих до оптимізації, хоча їх показники свідчать про низьку якість кластеризації.

Розглянемо рисунок, на якому зображено графік метрик моделей кластеризації після оптимізації гіперпараметрів.

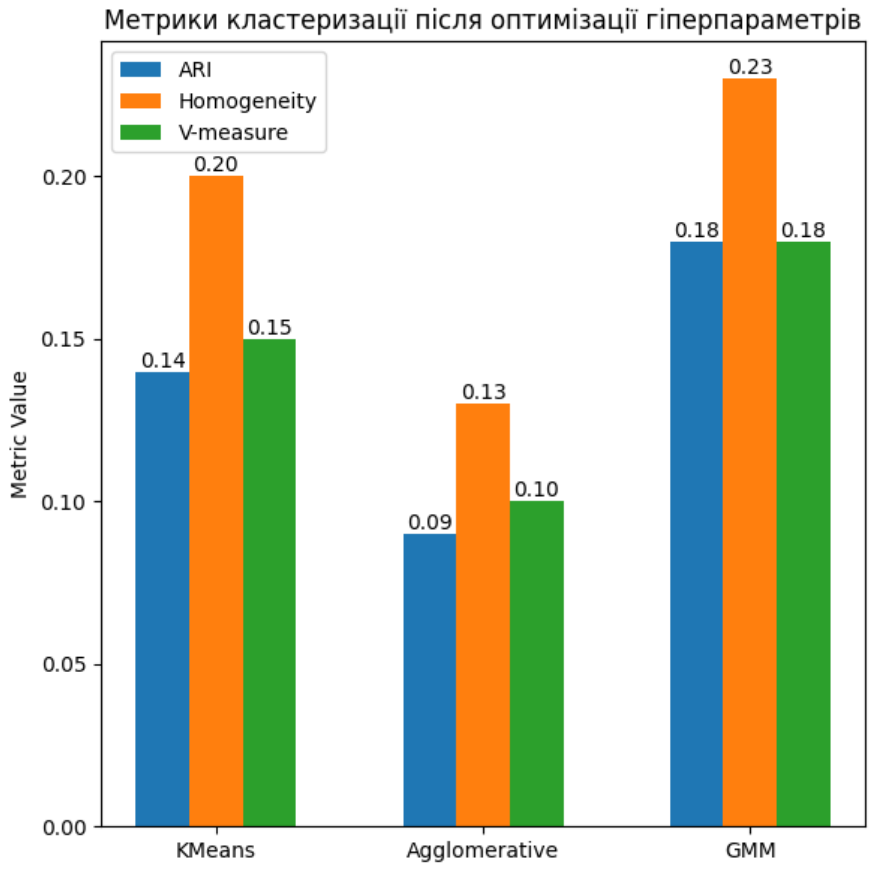

Рисунок 4.10 - Графік метрик моделей кластеризації після оптимізації гіперпараметрів

На рисунку 4.10 зображені результати кластеризації після оптимізації гіперпараметрів. Видно суттєве покращення якості кластеризації для моделей KMeans та GMM.

Модель KMeans після оптимізації демонструє помітне зростання метрик: ARI = 0.14, Homogeneity = 0.20, V-measure = 0.15. Це вказує на підвищення якості групування даних і поліпшення узгодженості кластерів.

Модель GMM досягає найвищих показників серед усіх моделей: ARI = 0.18, Homogeneity = 0.23, V-measure = 0.18. Це свідчить про те, що модель стала найбільш ефективною після оптимізації, особливо за рахунок підвищення однорідності кластерів.

Модель Agglomerative Clustering також покращила свої показники, але залишається слабшою порівняно з лідерами: ARI = 0.09, Homogeneity = 0.13, V-measure = 0.10. DBSCAN, попри оптимізацію, все ще не змогла виконати ефективну кластеризацію, тому її метрики залишаються нульовими.

Таким чином, провівши оптимізації гіперпараметрів, ми отримали, що вона мала найбільший вплив на моделі KMeans та GMM, які суттєво покращили свої показники за всіма метриками, але не змінили всієї картини роботи моделі кластеризації. Зокрема, GMM досягла найкращих результатів серед усіх розглянутих моделей і є найбільш придатною для розв’язання задач кластеризації. KMeans також показує хороші результати, зберігаючи стабільність і значно покращуючи якість кластеризації після налаштування.

Модель Agglomerative Clustering продемонструвала незначне покращення, залишаючись середньою за якістю кластеризації. DBSCAN показала найгірші результати як до, так і після оптимізації, що свідчить про її непридатність для даного типу задач без додаткової адаптації або трансформації даних.

Отже, результати аналізу показали, що оптимізація гіперпараметрів значно покращує точність і ефективність моделей для задач класифікації, регресії та кластеризації. Моделі SVM та KNeighbors у класифікації продемонстрували стабільно високі результати, що робить їх придатними для

поставлених задач кваліфікаційної роботи. У той же час моделі регресії, навіть після оптимізації, показали обмежену здатність точно прогнозувати дані через складність і нелінійність залежностей, властивих гідроакустичним сигналам. Для кластеризації навіть оптимізовані моделі, зокрема GMM, продемонстрували недостатню ефективність, що свідчить про обмеженість їхнього застосування для задач цієї роботи. Таким чином, основний акцент у дослідженні доцільно робити на класифікаційних моделях, які найкраще відповідають вимогам аналізу гідроакустичних сигналів.

4.4 Формулювання і узагальнення основних науково-технічних результатів дослідження

У процесі виконання дослідження було досягнуто низку важливих науково-технічних результатів, які мають значення як для розуміння характеристик гідроакустичних сигналів, так і для побудови ефективних моделей аналізу даних. Проведений аналіз дозволив оцінити можливості сучасних методів машинного навчання у вирішенні задач класифікації, регресії та кластеризації, а також визначити їхні сильні і слабкі сторони в контексті роботи з даними цієї специфіки.

Наукова цінність роботи полягає у вивченні здатності різних алгоритмів до адекватного відображення складних залежностей, характерних для гідроакустичних сигналів. Було виявлено, що класифікаційні моделі, зокрема SVM та KNeighbors, демонструють високу ефективність у задачах розпізнавання класів. Їхня здатність точно розрізняти об'єкти після оптимізації гіперпараметрів свідчить про те, що вони можуть бути ефективним інструментом для автоматизації процесів аналізу гідроакустичних даних. Це підтверджує їхню наукову цінність як перспективних методів для задач цього класу.

Технічна цінність дослідження полягає у визначенні обмежень регресійних та кластеризаційних моделей у контексті поставленої задачі. Було показано, що моделі регресії, такі як лінійна регресія та XGBoost, мають

недостатню точність через складність та нелінійність залежностей у гідроакустичних сигналах. Навіть після оптимізації ці моделі не продемонстрували високих результатів, що свідчить про обмеженість їх застосування у таких задачах. Кластеризаційні моделі, попри оптимізацію, також виявилися недостатньо ефективними для коректного розподілу даних по групах, що пояснюється низькою однорідністю даних та складністю виділення структурних особливостей сигналів.

Узагальнюючи результати дослідження, можна зробити висновок, що для аналізу гідроакустичних сигналів найбільш перспективними є класифікаційні моделі. Їхня здатність точно і стабільно працювати після оптимізації гіперпараметрів дозволяє рекомендувати їх для подальшої розробки та інтеграції в програмні засоби. Регресійні та кластеризаційні моделі, хоча й продемонстрували певний потенціал, потребують додаткових досліджень для адаптації до специфіки задач, пов’язаних із гідроакустичними сигналами.

Таким чином, науково-технічні результати дослідження підтвердили можливість застосування сучасних моделей машинного навчання для вирішення задач класифікації гідроакустичних сигналів. Отримані дані можуть бути використані для подальшого вдосконалення методик аналізу даних, а також для створення програмного забезпечення, яке спирається на найефективніші алгоритми класифікації.

5 АНАЛІЗ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ІННОВАЦІЇ

5.1 Розрахунок собівартості програмної інновації

Згідно із завданням, проведено розрахунок собівартості розробленого програмного комплексу для аналізу гідроакустичних сигналів. Основними статтями витрат стали супутні та непередбачувані витрати. Для виконання розрахунку використані вихідні дані, наведені в таблиці 5.1.

Таблиця 5.1 - Початкові дані для визначення собівартості

Найменування початкових даних	Показник	Джерело отримання
Тривалість розробки, місяців	3	Дані кваліфікаційної роботи (90 днів по 3 дні в годину)
Тривалість розробки, годин	270	Дані кваліфікаційної роботи
Одноразова виплата розробнику, грн	45000	Дані кваліфікаційної роботи
Супутні витрати, грн	2966	Дані кваліфікаційної роботи
ПДВ (податок на додану вартість), %	20	Дані кваліфікаційної роботи
Непередбачувані витрати, %	10	Дані кваліфікаційної роботи

Стаття 1. Розробимо розрахунок супутніх витрат.

Витрати на електроенергію раховується як сукупна вартість споживаної енергії під час розробки програмного комплексу. Враховуючи ціну за електроенергію у 4.32 грн. за кВт годину, маємо наступну формулу:

$$\mathrm{B_E} = T * C_{\text{ГОД}}, \tag{5.1}$$

де T – вартість кВт години;

$C_{\text{ГОД}}$ – кількість годин.

Визначимо витрати на електроенергію за формулою 5.1:

$$\text{В}_\text{Е} = 4{,}32 * 270 \approx 1166 \text{ грн} \quad (5.2)$$

Витрати на електроенергію склали 1166 грн.

Витрати на хостинг та інтернет враховуються як сукупна вартість витрат на їх використання. Враховуючи ціну хостингу у 400 грн. за місяць та інтернету у 200 грн. за місяць, маємо наступну формулу:

$$\text{В}_N = (I * C_I) + (H * C_H), \quad (5.3)$$

де I – ціна інтернету за місяць;

H – ціна хостингу за місяць;

C_H, C_H – кількість місяців використання;

Визначимо витрати на хостинг та інтернет за формулою 5.3:

$$\text{В}_N = (200 * 3) + (400 * 3) \approx 1800 \text{ грн} \quad (5.4)$$

Витрати на хостинг та інтернет склали 1800 грн.

Загальні супутні витрати обчислюються за формулою:

$$\text{В}_{\text{супутні}} = B_E + B_N, \quad (5.6)$$

де, B_E – витрати на електроенергію;

B_N – витрати на хостинг та інтернет.

Визначимо супутні витрати за формулою 5.6:

$$\text{В}_{\text{супутні}} = 1166 + 1800 \approx 2966 \text{ грн} \quad (5.7)$$

Супутні витрати склали 2966 грн.

Стаття 2. Непередбачувані витрати визначаються як відсоток від суми одноразової виплати та супутніх витрат. Розрахунок проводиться за формулою:

$$В_{нп} = \left(X + B_{сутупні}\right) * \frac{N}{100}, \quad (5.8)$$

де, $В_{супутні}$ – супунті витрати;

X – одноразова виплата розробнику;

N – відсоток непередбачуваних витрат.

Визначимо непередбачувані витрати за формулою 5.8:

$$В_{нп} = (45000 + 2966) * \frac{10}{100} \approx 4796 \text{ грн} \quad (5.9)$$

Непередбачувані витрати склали 4796 грн.

Стаття 3. ПДВ розраховується як 20% від загальної суми витрат (включаючи непередбачувані витрати). Формула розрахунку:

$$В_{ПДВ} = \left(X + B_{сутупні} + В_{нп}\right) * \frac{P}{100}, \quad (5.10)$$

де, $В_{супутні}$ – супунті витрати;

$В_{нп}$ – непередбачувані витрати;

X – одноразова виплата розробнику;

N – відсоток ПДВ.

Визначимо ПДВ за формулою 5.10:

$$В_{ПДВ} = (45000 + 2966 + 4796) * \frac{20}{100} \approx 10552 \text{ грн} \quad (5.11)$$

ПДВ склало 10552 грн.

Стаття 4. Загальна собівартість визначається як сума одноразової виплати розробнику, супутніх витрат, непередбачуваних витрат та ПДВ. Формула розрахунку:

$$В_{загал} = \left(X + B_{сутупні} + В_{нп} + В_{ПДВ}\right), \quad (5.12)$$

де, $В_{супутні}$ – супунті витрати;

$В_{нп}$ – непередбачувані витрати;

$В_{ПДВ}$ – ПДВ;

X – одноразова виплата розробнику.

Визначемо загальну собівартість за формулою 5.12:

$$В_{загал} = (45000 + 2966 + 4796 + 10552) = 63287 \text{ грн} \quad (5.13)$$

Загальна собівартість склала 63287 грн. Отримані результати виведемо в окрему таблицю розрахунків, яка наведена у таблиці 5.2.

Таблиця 5.2 – Таблиця розрахунків

Стаття розрахунку	Сума в грн.
Стаття 1. Супутні витрати	2966
Стаття 2. Неперебачені витрати	4796
Стаття 3. ПДВ	3952
Витрати на розробника	45000
Стаття 4. Собівартість	63287

Робота по розробці програмного комплексу проводилась протягом трьох місяців, таким чином розрахунок показав, що розробленого програмного комплексу для аналізу гідроакустичних сигналів становить 63287 грн. Ця сума

включає одноразову виплату розробнику, супутні витрати, непередбачувані витрати та ПДВ.

5.2 Розрахунок ефективності впровадження програмної інновації

Розрахунок економічної ефективності впровадження програмного комплексу для аналізу гідроакустичних сигналів здійснюється з урахуванням собівартості його розробки, супутніх витрат під час експлуатації та очікуваної економії від його використання. Ефективність визначається шляхом обчислення загальної економії коштів, чистого економічного ефекту та терміну окупності.

Параметри, які використовувалися у розрахунках, наведені у таблиці 5.3.

Таблиця 5.3 – Таблиця параметрів для розрахунків

Найменування показника	Значення, грн	Джерело отримання
Загальна собівартість розробки	63287	Розрахунки підрозділу 5.1
Щомісячна економія витрат	8000	Підвищення ефективності та автоматизація
Щомісячні супутні витрати	700	Витрати на електроенергію, хостинг, інтернет
Термін використання програмного комплексу, місяців	24	Прогнозований період експлуатації

Розрахуємо щомісячну чисту економію, яка визначається як різниця між щомісячною економією витрат та щомісячними супутніми витратами:

$$В_{міс\_чист} = (Е_{міс} - В_{суп}), \quad (5.13)$$

де, $В_{міс}$ – щомісячна економія;

$В_{суп}$ – щомісячні супутні витрати.

Визначемо щомісячну чисту економію за формулою 5.13:

$$В_{міс\_чист} = (8000 - 3000) = 5000 \text{ грн} \quad (5.14)$$

Щомісячна чиста економія склала 5000 грн.

Загальна економія витрат визначається як добуток щомісячної чистої економії на термін використання:

$$Е_{загальна} = (Е_{міс\_чист} * Т_{вик}), \quad (5.15)$$

де, $Е_{міс\_чист}$ – щомісячна чиста економія;

$Т_{вик}$ – термін використання у місяцях.

Визначемо загальну економію витрат за формулою 5.15:

$$Е_{загальна} = (5000 * 24) = 120000 \text{ грн} \quad (5.16)$$

Загальна економія за 24 місяці використання становить 120000 грн.

Термін окупності програмного комплексу розраховується як відношення собівартості розробки до щомісячної чистої економії:

$$Т_{окуп} = \left(\frac{С_{заг}}{Е_{міс\_чист}}\right), \quad (5.17)$$

де, $Е_{міс\_чист}$ – щомісячна чиста економія;

$С_{заг}$ – собівартість програмного комплексу.

Визначемо термін окупності витрат за формулою 5.17:

$$Т_{окуп} = \left(\frac{63287}{5000}\right) \approx 12.66 \text{ місяців} \quad (5.18)$$

Термін окупності витрат програмного комплексу становить приблизно 12.66 місяців.

На основі отриманих розрахунків, створимо таблицю 5.4, яка буде містити числові дані розрахунків економічної ефективності.

Таблиця 5.4 - Таблиця розрахунків економічної ефективності

Показник	Значення
Щомісячна економія	8000
Щомісячні супутні витрати	3000
Щомісячна чиста економія	5000
Загальна економія	120000
Термін окупності, місяців	12.66

Отже, нами було проведено аналіз економічної ефективності розробки та впровадження програмного комплексу для аналізу гідроакустичних сигналів. Було визначено собівартість розробки, яка включає супутні витрати, непередбачувані витрати та податок на додану вартість. Загальна собівартість проекту становить 63287 грн.

Також було розраховано щомісячну економію від використання комплексу, яка досягає 8000 грн завдяки підвищенню ефективності аналізу сигналів та автоматизації процесів. При цьому щомісячні супутні витрати, що включають витрати на електроенергію, хостинг та інтернет, склали 3000 грн.

Прогнозована загальна економія витрат за рік експлуатації комплексу становить 120000 грн. Розрахунки підтвердили, що впровадження даного програмного комплексу є економічно виправданим, оскільки дозволяє оптимізувати ресурси та зменшити витрати, пов'язані з аналізом гідроакустичних сигналів.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено всебічне дослідження проблеми аналізу гідроакустичних сигналів, що має актуальне значення для забезпечення безпеки морських перевезень, захисту морських кордонів та навігації. На основі аналізу сучасних методів та технологій були сформовані теоретичні та практичні підходи до автоматизації обробки та класифікації підводних сигналів.

У рамках роботи виконано такі ключові етапи:

— аналіз проблемної області та визначення задачі дослідження. Був виконаний огляд сучасних методів аналізу гідроакустичних сигналів, включаючи традиційні статистичні методи та підходи на основі машинного навчання. Було сформульовано цілі та завдання розробки програмного комплексу;

— розробка програмного комплексу. Було здійснено вибір мов програмування та інструментів, що забезпечують ефективну реалізацію поставленої задачі. На основі бібліотек Python, таких як scikit-learn, NumPy та Pandas, були реалізовані моделі класифікації, регресії та кластеризації. Для створення API використовувався фреймворк FastAPI, а для розробки веб-додатку – бібліотека React;

— експериментальні дослідження. Були проведені експерименти щодо ефективності моделей, зокрема їх точності, швидкодії та здатності адаптуватися до нових даних. Особливу увагу приділено оптимізації гіперпараметрів моделей, що дозволило досягти високих результатів у задачі класифікації;

— економічна оцінка. Виконано розрахунок собівартості розробки програмного комплексу, враховуючи супутні та непередбачувані витрати. Оцінено ефективність впровадження розробки у виробничу діяльність, що підтвердило її доцільність та практичну цінність.

Розроблений програмний комплекс дозволяє автоматизувати аналіз гідроакустичних сигналів, що значно спрощує роботу спеціалістів та підвищує

точність класифікації підводних об'єктів. Отримані результати дослідження підтверджують актуальність запропонованих методів, їх практичну цінність та можливість інтеграції в реальні системи.

Таким чином, виконана кваліфікаційна робота зробила внесок у розвиток методів аналізу гідроакустичних сигналів та продемонструвала можливості застосування сучасних технологій машинного навчання для вирішення складних задач у цій галузі. Отримані результати мають потенціал для подальших досліджень і вдосконалень.

ПЕРЕЛІК ПОСИЛАНЬ

1) Білецький В.С. Основи підводної акустики. – Київ: Вища школа, 2020. – 328 с.

2) Bishop C.M. Pattern Recognition and Machine Learning. – Springer, 2006. – 738 p.

3) Хасті Т., Тібшірані Р., Фрідман Дж. Елементи статистичного навчання. – Москва: ИНФРА-М, 2021. – 745 с.

4) Гусєв О.О. Методи аналізу багатовимірних даних. – Київ: Наукова думка, 2019. – 376 с.

5) Breiman L. Random Forests. Machine Learning. – 2001. P. 5–32.

6) Pedregosa F., Varoquaux G., Gramfort A., et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. – 2011. – №12. – P. 2825–2830.

7) Кузьменко І.В. Програмування на Python: теорія і практика. – Харків: ХНУРЕ, 2022. – 401 с.

8) VanderPlas J. Python Data Science Handbook. – O’Reilly Media, 2016. – 548 p.

9) Жуков О.М. Основи машинного навчання. – Одеса: ОНУ ім. Мечникова, 2021. – 312 с.

10) Шевченко А.В. Економіка розробки програмного забезпечення. – Київ: КНЕУ, 2019. – 278 с.

11) FastAPI Documentation [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com/.

12) Müller A.C., Guido S. Introduction to Machine Learning with Python. – O’Reilly Media, 2016. – 394 p.

13) Kuznetsov M.V. Web API Design for Machine Learning. – IEEE Proceedings, 2020. – P. 124–132.

14) Джонс М. Графічна візуалізація даних. – Київ: Видавничий дім, 2022. – 292 с.

Додаток А

Лістинг коду завантаження даних:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler,
    LabelEncoder
from sklearn.model_selection import
    train_test_split
data =
    pd.read_csv('/content/drive/MyDrive/datasets/so
    nar_dataset.csv', header=None)
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.25,
    random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Лістинг коду методу класифікації:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from xgboost import XGBClassifier
```

```
from sklearn.metrics import accuracy_score,
    precision_score, recall_score, f1_score,
    confusion_matrix
def visualize_confusion_matrices(metrics_results,
    n_cols=3):
    n_models = len(metrics_results["Model"])
    n_rows = (n_models + n_cols - 1) // n_cols
    fig, axes = plt.subplots(n_rows, n_cols,
    figsize=(6 * n_cols, 6 * n_rows))
    axes = axes.flatten()
    for idx, (model_name, cm) in
    enumerate(zip(metrics_results["Model"],
    metrics_results["Confusion Matrix"])):
        ax = axes[idx]
        im = ax.imshow(cm, cmap="Blues", vmin=0,
    vmax=max(cm.flatten()))
        ax.set_xticks(np.arange(2))
        ax.set_yticks(np.arange(2))
        ax.set_xticklabels(["Камінь", "Міна"])
        ax.set_yticklabels(["Камінь", "Міна"])
        ax.set_xlabel("Predicted Label")
        ax.set_ylabel("True Label")
        ax.set_title(f"Confusion Matrix for
    {model_name}")
        for i in range(2):
            for j in range(2):
                ax.text(j, i, cm[i, j],
    ha="center", va="center", color="black")
```

```
    for idx in range(len(metrics_results["Model"]),
    len(axes)):
        fig.delaxes(axes[idx])
    # Глобальна шкала кольорів
    cbar_ax = fig.add_axes([0.92, 0.15, 0.02, 0.7])
    fig.colorbar(im, cax=cbar_ax)
    plt.tight_layout(rect=[0, 0, 0.9, 1])
    plt.show()
# Моделі
models = {
    "KNeighbors": KNeighborsClassifier(),
    "Logistic Regression": LogisticRegression(),
    "Random Forest":
    RandomForestClassifier(random_state=42),
    "SVM": SVC(probability=True),
    "XGBoost":
    XGBClassifier(eval_metric='logloss'),
    "Gaussian Naive Bayes": GaussianNB()
}

# Метріки для кожної моделі
metrics_results = {
    "Model": [],
    "Accuracy": [],
    "Precision": [],
    "Recall": [],
    "F1-Score": [],
    "Confusion Matrix": []
}
```

```
# Навчання і оцінка
results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[model_name] = accuracy
    metrics_results["Model"].append(model_name)
    metrics_results["Accuracy"].append(accuracy_sco
    re(y_test, y_pred))
    metrics_results["Precision"].append(precision_s
    core(y_test, y_pred))
    metrics_results["Recall"].append(recall_score(y
    _test, y_pred))
    metrics_results["F1-
    Score"].append(f1_score(y_test, y_pred))
    metrics_results["Confusion
    Matrix"].append(confusion_matrix(y_test,
    y_pred))

metrics_df = pd.DataFrame(metrics_results)
visualize_confusion_matrices(metrics_results)
models_plt = metrics_df["Model"]
accuracy_plt = metrics_df["Accuracy"]
f1_score_plt = metrics_df["F1-Score"]
x = np.arange(len(models_plt))
width = 0.4
fig, ax = plt.subplots(figsize=(12, 6))
bars1 = ax.bar(x - width/2, accuracy_plt, width,
    label="Accuracy", color="lightblue")
```

```
bars2 = ax.bar(x + width/2, f1_score_plt, width,
    label="F1-Score", color="orange")
for bar in bars1:
    ax.text(bar.get_x() + bar.get_width()/2,
    bar.get_height(),
            f'{bar.get_height():.4f}', ha='center',
    va='bottom')
for bar in bars2:
    ax.text(bar.get_x() + bar.get_width()/2,
    bar.get_height(),
            f'{bar.get_height():.4f}', ha='center',
    va='bottom')
ax.set_xlabel("Моделі")
ax.set_ylabel("Значення")
ax.set_title("Графік класифікації без оптимізації
    гіперпараметрів. Метрики: Accuracy та F1-
    Score")
ax.set_xticks(x)
ax.set_xticklabels(models_plt)
ax.legend()
plt.tight_layout()
plt.show()
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
param_grids = {
    "KNeighbors": {
        "n_neighbors": [7],
        "weights": ["distance"],
        "metric": ["manhattan"]
    },
```

```
    "Logistic Regression": {
        "C": [0.1],
        "solver": ["liblinear"]
    },
    "Random Forest": {
        "n_estimators": [100],
        "max_depth": [10],
        "min_samples_split": [2]
    },
    "SVM": {
        "C": [10],
        "kernel": ["rbf"],
        "gamma": ["scale"]
    },
    "XGBoost": {
        "n_estimators": [100],
        "max_depth": [6],
        "learning_rate": [0.2]
    }
}
metrics_optimized_results = {
    "Model": [],
    "Accuracy": [],
    "Precision": [],
    "Recall": [],
    "F1-Score": [],
    "Confusion Matrix": []
}

optimized_results = {}
```

```
best_params_results = {}
for model_name, model in models.items():
    if model_name in param_grids:
        grid_search = GridSearchCV(
            model,
            param_grids[model_name],
            scoring="accuracy",
            cv=5,
            n_jobs=-1,
            verbose=1,
            refit=True
        )
        grid_search.fit(X_train, y_train)
        best_model = grid_search.best_estimator_
        y_pred = best_model.predict(X_test)
        best_params_results[model_name] =
    grid_search.best_params_
    else:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        best_params_results[model_name] = "Default
    Parameters"

    # Обчислення метрик
    metrics_optimized_results["Model"].append(model
    _name)
    metrics_optimized_results["Accuracy"].append(ac
    curacy_score(y_test, y_pred))
    metrics_optimized_results["Precision"].append(p
    recision_score(y_test, y_pred))
```

```
        metrics_optimized_results["Recall"].append(reca
        ll_score(y_test, y_pred))
        metrics_optimized_results["F1-
        Score"].append(f1_score(y_test, y_pred))
        metrics_optimized_results["Confusion
        Matrix"].append(confusion_matrix(y_test,
        y_pred))
    metrics_optimized_df =
        pd.DataFrame(metrics_optimized_results)
    visualize_confusion_matrices(metrics_optimized_df)
    print(best_params_results)
    models_plt = metrics_optimized_df["Model"]
    accuracy_plt = metrics_optimized_df["Accuracy"]
    f1_score_plt = metrics_optimized_df["F1-Score"]
    x = np.arange(len(models_plt))
    width = 0.4
    fig, ax = plt.subplots(figsize=(12, 6))
    bars1 = ax.bar(x - width/2, accuracy_plt, width,
        label="Accuracy", color="lightblue")
    bars2 = ax.bar(x + width/2, f1_score_plt, width,
        label="F1-Score", color="orange")
    for bar in bars1:
        ax.text(bar.get_x() + bar.get_width()/2,
        bar.get_height(),
                f'{bar.get_height():.4f}', ha='center',
        va='bottom')
    for bar in bars2:
        ax.text(bar.get_x() + bar.get_width()/2,
        bar.get_height(),
```

```
            f'{bar.get_height():.4f}', ha='center',
    va='bottom')
ax.set_xlabel("Моделі")
ax.set_ylabel("Значення")
ax.set_title("Графік класифікації з оптимізацією
    гіперпараметрів. Метрики: Accuracy та F1-
    Score")
ax.set_xticks(x)
ax.set_xticklabels(models_plt)
ax.legend()
plt.tight_layout()
plt.show()
```

Лістинг коду методу регресії:

```
def visualize_regression_plots(model_names, X_test,
    y_preds, y_test, n_cols=2):
    n_models = len(model_names)
    n_rows = (n_models + n_cols - 1)
    fig, axes = plt.subplots(n_rows, n_cols,
    figsize=(4 * n_cols, 4 * n_rows))
    axes = axes.flatten()
    for idx, (model_name, y_pred) in
    enumerate(zip(model_names, y_preds)):
        residuals = y_test - y_pred
        ax = axes[idx]
        ax.scatter(y_pred, residuals, alpha=0.5)
        ax.axhline(0, color='r', linestyle='--',
    linewidth=2)
        ax.set_title(f"Residuals for {model_name}")
        ax.set_xlabel("Predicted Values")
        ax.set_ylabel("Residuals")
```

```
    for idx in range(len(model_names), len(axes)):
        fig.delaxes(axes[idx])
    plt.tight_layout()
    plt.show()
import numpy as np
import pandas as pd
from sklearn.model_selection import
    train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error,
    mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
import matplotlib.pyplot as plt
models = {
    "L. Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(),
    "SVR": SVR(),
    "XGBoost": XGBRegressor(eval_metric="rmse")
}
results = {
    "Model": [],
    "MSE": [],
    "MAE": [],
    "R2": [],
    "YPred": []
}
for model_name, model in models.items():
```

```
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        results["Model"].append(model_name)
        results["MSE"].append(mean_squared_error(y_test
        , y_pred))
        results["MAE"].append(mean_absolute_error(y_tes
        t, y_pred))
        results["R2"].append(r2_score(y_test, y_pred))
        results["YPred"].append(y_pred)
results_df = pd.DataFrame(results)
print("Результаты без оптимизации:")
print(results_df)
visualize_regression_plots(results["Model"],
    X_test, results["YPred"], y_test)
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].bar(results_df["Model"], results_df["MSE"],
    color="lightblue")
axes[0].set_title("MSE (до оптимізації)")
axes[0].set_ylabel("MSE")
axes[0].set_xlabel("Модель")
for i, v in enumerate(results_df["MSE"]):
    axes[0].text(i, v + 0.01, f"{v:.2f}",
    ha='center')
axes[0].spines['top'].set_visible(False)
axes[1].bar(results_df["Model"], results_df["MAE"],
    color="lightgreen")
axes[1].set_title("MAE (до оптимізації)")
axes[1].set_ylabel("MAE")
axes[1].set_xlabel("Модель")
for i, v in enumerate(results_df["MAE"]):
```

```
    axes[1].text(i, v + 0.01, f"{v:.2f}",
    ha='center')
axes[1].spines['top'].set_visible(False)
axes[2].bar(results_df["Model"], results_df["R2"],
    color="lightcoral")
axes[2].set_title("R2 (до оптимізації)")
axes[2].set_ylabel("R2")
axes[2].set_xlabel("Модель")
for i, v in enumerate(results_df["R2"]):
    axes[2].text(i, v + 0.01, f"{v:.2f}",
    ha='center')
axes[2].spines['top'].set_visible(False)
plt.tight_layout()
plt.show()
from sklearn.model_selection import GridSearchCV
param_grids = {
    "L. Regression": {},
    "Random Forest": {
        "n_estimators": [50, 100, 200],
        "max_depth": [None, 10, 20],
        "min_samples_split": [2, 5]
    },
    "SVR": {
        "C": [0.1, 1, 10],
        "kernel": ["linear", "rbf"],
        "gamma": ["scale", "auto"]
    },
    "XGBoost": {
        "n_estimators": [50, 100, 200],
        "max_depth": [3, 6, 10],
```

```
        "learning_rate": [0.01, 0.1, 0.2]
    }
}
optimized_results = {
    "Model": [],
    "MSE": [],
    "MAE": [],
    "R2": [],
    "YPred": []
}
for i, (model_name, model) in
    enumerate(models.items()):
    grid = GridSearchCV(model,
    param_grids[model_name],
    scoring="neg_mean_squared_error", cv=5,
    n_jobs=-1, verbose=1)
    grid.fit(X_train, y_train)
    best_model = grid.best_estimator_
    y_pred = best_model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    optimized_results["Model"].append(model_name)
    optimized_results["MSE"].append(mse)
    optimized_results["MAE"].append(mae)
    optimized_results["R2"].append(r2)
    optimized_results["YPred"].append(y_pred)
    print(f"Лучшие параметры для {model_name}:
    {grid.best_params_}")
```

```
optimized_results_df =
    pd.DataFrame(optimized_results)
print("Результаты после оптимизации:")
print(optimized_results_df)
visualize_regression_plots(optimized_results["Model
    "], X_test, optimized_results["YPred"], y_test)
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].bar(optimized_results_df["Model"],
    optimized_results_df["MSE"], color="lightblue")
axes[0].set_title("MSE (після оптимізації)")
axes[0].set_ylabel("MSE")
axes[0].set_xlabel("Модель")
for i, v in enumerate(optimized_results_df["MSE"]):
    axes[0].text(i, v + 0.01, f"{v:.2f}",
    ha='center')
axes[0].spines['top'].set_visible(False)
axes[1].bar(optimized_results_df["Model"],
    optimized_results_df["MAE"],
    color="lightgreen")
axes[1].set_title("MAE (після оптимізації)")
axes[1].set_ylabel("MAE")
axes[1].set_xlabel("Модель")
for i, v in enumerate(optimized_results_df["MAE"]):
    axes[1].text(i, v + 0.01, f"{v:.2f}",
    ha='center')
axes[1].spines['top'].set_visible(False)
axes[2].bar(optimized_results_df["Model"],
    optimized_results_df["R2"], color="lightcoral")
axes[2].set_title("R2 (після оптимізації)")
axes[2].set_ylabel("R2")
```

```
axes[2].set_xlabel("Модель")
for i, v in enumerate(optimized_results_df["R2"]):
    axes[2].text(i, v + 0.01, f"{v:.2f}",
    ha='center')
axes[2].spines['top'].set_visible(False)
plt.tight_layout()
plt.show()
```

Лістинг коду моделі кластеризації:

```
X = data.iloc[:, :-1]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
from sklearn.cluster import KMeans, DBSCAN,
    AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import adjusted_rand_score,
    homogeneity_score, completeness_score,
    v_measure_score
import matplotlib.pyplot as plt
models = {
    "KMeans": KMeans(n_clusters=2,
    random_state=42),
    "DBSCAN": DBSCAN(eps=2, min_samples=2),
    "Agglomerative":
    AgglomerativeClustering(n_clusters=2),
    "GMM": GaussianMixture(n_components=2,
    random_state=42)
}
true_labels = data.iloc[:, -1]
if true_labels.dtype == 'object':
```

```
    true_labels =
    LabelEncoder().fit_transform(true_labels)
metrics_results = {"Model": [], "ARI": [],
    "Homogeneity": [], "V-measure": []}
for model_name, model in models.items():
    if model_name == "GMM":
        predicted_labels =
    model.fit_predict(X_scaled)
    else:
        predicted_labels =
    model.fit_predict(X_scaled)
    ari = adjusted_rand_score(true_labels,
    predicted_labels)
    homogeneity = homogeneity_score(true_labels,
    predicted_labels)
    v_measure = v_measure_score(true_labels,
    predicted_labels)
    metrics_results["Model"].append(model_name)
    metrics_results["ARI"].append(ari)
    metrics_results["Homogeneity"].append(homogenei
    ty)
    metrics_results["V-measure"].append(v_measure)
metrics_df = pd.DataFrame(metrics_results)
fig, ax = plt.subplots(figsize=(6, 6))
x = range(len(metrics_df["Model"]))
ari_bars = ax.bar(x, metrics_df["ARI"], width=0.2,
    label="ARI", align="center")
homogeneity_bars = ax.bar([p + 0.2 for p in x],
    metrics_df["Homogeneity"], width=0.2,
    label="Homogeneity", align="center")
```

```
v_measure_bars = ax.bar([p + 0.4 for p in x],
    metrics_df["V-measure"], width=0.2, label="V-
    measure", align="center")
for bar in ari_bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2.0,
    height, f"{height:.2f}", ha='center',
    va='bottom', fontsize=10)
for bar in homogeneity_bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2.0,
    height, f"{height:.2f}", ha='center',
    va='bottom', fontsize=10)
for bar in v_measure_bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2.0,
    height, f"{height:.2f}", ha='center',
    va='bottom', fontsize=10)
ax.set_xticks([p + 0.2 for p in x])
ax.set_xticklabels(metrics_df["Model"])
ax.set_ylabel("Metric Value")
ax.set_title("Метрики кластеризації до оптимізації
    гіперпараметрів")
ax.legend()
plt.tight_layout()
plt.show()
from sklearn.cluster import KMeans, DBSCAN,
    AgglomerativeClustering
from sklearn.mixture import GaussianMixture
```

```
from sklearn.metrics import silhouette_score,
    davies_bouldin_score, calinski_harabasz_score
from sklearn.model_selection import ParameterGrid
import pandas as pd
models = {
    "KMeans": KMeans(n_clusters=4,
    random_state=42),
    "DBSCAN": DBSCAN(eps=2, min_samples=2),
    "Agglomerative":
    AgglomerativeClustering(n_clusters=4),
    "GMM": GaussianMixture(n_components=4,
    random_state=42)
}
param_grids = {
    "KMeans": {"n_clusters": [2, 3, 4, 5, 6]},
    "DBSCAN": {"eps": [0.3, 0.5, 0.7],
    "min_samples": [3, 5, 10]},
    "Agglomerative": {"n_clusters": [2, 3, 4, 5,
    6]},
    "GMM": {"n_components": [2, 3, 4, 5, 6]}
}
optimized_metrics_results = {"Model": [], "ARI":
    [], "Homogeneity": [], "V-measure": []}
best_models = {}
for model_name, model in models.items():
    if model_name in param_grids:
        best_score = -1
        best_params = None
        best_labels = None
        best_metrics = {}
```

```
    if (model_name == 'DBSCAN'):
      continue
    for params in
ParameterGrid(param_grids[model_name]):
        if model_name == "KMeans":
            model.set_params(**params)
        elif model_name == "DBSCAN":
            model = DBSCAN(**params)
        elif model_name == "Agglomerative":
            model =
AgglomerativeClustering(**params)
        elif model_name == "GMM":
            model.set_params(**params)
        if model_name == "GMM":
            labels =
model.fit_predict(X_scaled)
        else:
            labels =
model.fit_predict(X_scaled)
        n_clusters = len(set(labels)) - (1 if -
1 in labels else 0)
        if n_clusters > 1:
            ari =
adjusted_rand_score(true_labels, labels)
            homogeneity =
homogeneity_score(true_labels, labels)
            completeness =
completeness_score(true_labels, labels)
            v_measure =
v_measure_score(true_labels, labels)
```

```
                score = silhouette_score(X_scaled,
    labels)
                if score > best_score:
                    best_score = score
                    best_params = params
                    best_labels = labels
                    best_metrics = {
                        "ARI": ari,
                        "Homogeneity": homogeneity,
                        "Completeness":
    completeness,
                        "V-measure": v_measure,
                        "Silhouette Score": score,
                    }
        optimized_metrics_results["Model"].append(m
    odel_name)
        optimized_metrics_results["ARI"].append(bes
    t_metrics.get("ARI", -1))
        optimized_metrics_results["Homogeneity"].ap
    pend(best_metrics.get("Homogeneity", -1))
        optimized_metrics_results["V-
    measure"].append(best_metrics.get("V-measure",
    -1))
opt_metrics_df =
    pd.DataFrame(optimized_metrics_results)
fig, ax = plt.subplots(figsize=(6, 6))
x = range(len(opt_metrics_df["Model"]))
ari_bars = ax.bar(x, opt_metrics_df["ARI"],
    width=0.2, label="ARI", align="center")
```

```
homogeneity_bars = ax.bar([p + 0.2 for p in x],
    opt_metrics_df["Homogeneity"], width=0.2,
    label="Homogeneity", align="center")
v_measure_bars = ax.bar([p + 0.4 for p in x],
    opt_metrics_df["V-measure"], width=0.2,
    label="V-measure", align="center")
for bar in ari_bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2.0,
    height, f"{height:.2f}", ha='center',
    va='bottom', fontsize=10)
for bar in homogeneity_bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2.0,
    height, f"{height:.2f}", ha='center',
    va='bottom', fontsize=10)
for bar in v_measure_bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2.0,
    height, f"{height:.2f}", ha='center',
    va='bottom', fontsize=10)
ax.set_xticks([p + 0.2 for p in x])
ax.set_xticklabels(opt_metrics_df["Model"])
ax.set_ylabel("Metric Value")
ax.set_title("Метрики кластеризації після
    оптимізації гіперпараметрів")
ax.legend()
plt.tight_layout()
plt.show()
```

Лістинг коду API:

```
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
import joblib
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from io import BytesIO
import pandas as pd
import base64
loaded_svm_model =
    joblib.load('models/svm_model.joblib')
loaded_scaler = joblib.load('models/scaler.joblib')
df = pd.read_csv("sonar_dataset.csv")
training_data = df.iloc[:, :-1].values
training_labels = df.iloc[:, -1].values
app = FastAPI()
origins = [
    "http://localhost",
    "http://localhost:5173",
]
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

```
class InputData(BaseModel):
    input_data: list[float]
def generate_3d_visualization(input_data,
    prediction, training_data, training_labels):
    training_data_scaled =
    loaded_scaler.transform(training_data)
    input_data_scaled =
    loaded_scaler.transform(np.array(input_data).re
    shape(1, -1))
    training_labels_numeric =
    np.where(training_labels == 'R', 0, 1)
    input_data_numeric = 0 if prediction[0] == 'R'
    else 1
    pca = PCA(n_components=3)
    training_data_pca =
    pca.fit_transform(training_data_scaled)
    input_data_pca =
    pca.transform(input_data_scaled)
    fig = plt.figure(figsize=(10, 7))
    ax = fig.add_subplot(111, projection='3d')
    scatter = ax.scatter(training_data_pca[:, 0],
    training_data_pca[:, 1], training_data_pca[:,
    2],
                         c=training_labels_numeric
    , cmap='coolwarm', alpha=0.5, label='Training
    Data')
    ax.scatter(input_data_pca[0, 0],
    input_data_pca[0, 1], input_data_pca[0, 2],
               c='red', s=100, label="Input
    Object")
```

```
    ax.set_title(f"3D Visualization of the Object -
    {'Міна' if prediction[0] == 'M' else
    'Камінь'}")
    ax.set_xlabel("PCA Component 1")
    ax.set_ylabel("PCA Component 2")
    ax.set_zlabel("PCA Component 3")
    plt.legend()
    plt.colorbar(scatter, ax=ax, label="Class (0:
    Камінь, 1: Міна)")
    buf = BytesIO()
    plt.savefig(buf, format="png")
    buf.seek(0)
    plt.close()
    image_base64 =
    base64.b64encode(buf.getvalue()).decode('utf-
    8')
    buf.close()
    return image_base64
@app.post("/predict")
async def predict(input_data: InputData):
    print(input_data)
    try:
        input_data_as_numpy_array =
    np.asarray(input_data.input_data).reshape(1, -
    1)
        std_data =
    loaded_scaler.transform(input_data_as_numpy_arr
    ay)
        prediction =
    loaded_svm_model.predict(std_data)
```

```
        result = 'Міна' if prediction[0] == 'M'
    else 'Камінь'
        spectrogram =
    generate_spectrogram(input_data.input_data,
    prediction)
        visualization =
    generate_3d_visualization(input_data.input_data
    , prediction, training_data, training_labels)
        return {"prediction": result,
    "spectrogram": visualization}
    except Exception as e:
        raise HTTPException(status_code=400,
    detail=f"Помилка обробки: {str(e)}")
```

Лістинг коду веб-додатку:

```
import { useState } from 'react';
import './App.css';
import { sonarData } from './common/constants';
import TopBar from './components/TopBar/TopBar';
import Container from
    './components/Container/Container';
import Button from './components/ui/Button/Button';
function App() {
  const [predictionResult, setPredictionResult] =
    useState({ prediction: '', spectrogram: '' });
  const [currentSonarData, setCurrentSonarData] =
    useState(sonarData[0]);
  async function clickHandler() {
    const randomIndex = Math.floor(Math.random() *
    (41 - 0 + 1) + 0);
    const inputData = {
```

```
      input_data: sonarData[randomIndex],
    };
    setCurrentSonarData(sonarData[randomIndex]);
    const result = await
    fetch('http://127.0.0.1:8000/predict', {
      method: 'POST',
      body: JSON.stringify(inputData),
      headers: {
        'Content-Type': 'application/json',
      },
    });
    const data = await result.json();
    setPredictionResult(data);
    const sonarHistory =
    localStorage.getItem('predictionHistory');
    localStorage.setItem(
      'predictionHistory',
      JSON.stringify([...JSON.parse(sonarHistory ||
    '[]'), { sonarData: sonarData[randomIndex],
    result: data.prediction }]),
    );
  }
  return (
    <>
      <main>
        <Container className="px-20 py-4">
          <h1 className="text-center text-lg">
            <span className="font-
    semibold">Аналізатор гідроакустичних
    сигналів</span>
```

```
            </h1>
            <TopBar data={currentSonarData} />
            <Container className="flex items-center
      gap-x-4">
              <Button
      onClick={clickHandler}>Проаналізувати</Button>
              <p>Результат аналізу оброблених
      сигналів: {predictionResult.prediction}</p>
            </Container>
            <hr className="my-4" />
            <Container>
              <Container>
                <h2 className="font-
      semibold">Візуалізація результатів</h2>
              </Container>
              <img
      src={`${predictionResult.
      spectrogram}`} className="w-1/2 "
      alt="Спектрограма" />
            </Container>
          </Container>
        </main>
      </>
    );
}
export default App;
import React from 'react';

interface ITableProps {
  data: any[];
```

```
  header: any[];
}
export default function Table(props: ITableProps) {
  return (
    <div className="overflow-x-scroll">
      <table className="border-collapse">
        <thead>
          <tr className="bg-slate-500">
            {props.header.map((item, idx) => (
              <td key={idx} className="border
    border-slate-700 px-4 text-center text-stone-
    100">
                {item}
              </td>
            ))}
          </tr>
        </thead>
        <tbody>
          <tr>
            {props.data.map((item, idx) => (
              <td key={idx} className="border
    border-slate-700 px-4">
                {item}
              </td>
            ))}
          </tr>
        </tbody>
      </table>
    </div>
  );
```

```
}
import { cn } from '@/lib/utils';
import React from 'react';
interface IButtonProps extends
    React.HTMLAttributes<HTMLButtonElement> {}
export default function Button(props: IButtonProps)
    {
  return (
    <button
      className={cn(
        props.className,
        'focus:outline-none text-white bg-purple-
    700 hover:bg-purple-800 focus:ring-4
    focus:ring-purple-300 font-medium rounded-lg
    text-sm px-5 py-2.5 dark:bg-purple-600
    dark:hover:bg-purple-700 dark:focus:ring-
    purple-900',
      )}
      {...props}
    >
      {props.children}
    </button>
  );
}
```