МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп’ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри,
д-р техн. наук, проф.
_____Ірина ЖУРАВСЬКА
« __ » _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Комплекс для моніторингу інтенсивності землетрусів та оцінки наслідків на базі сенсорів IoT

Спеціальність 123 Комп’ютерна інженерія

Освітня програма «Комп’ютерна інженерія»

Здобувач _____Максим ЖУЛАНОВ
підпис
«__» _____ 2024 р.

Керівник канд. техн. наук, доцент _____Ярослав КРАЙНИК
підпис
«__» _____ 2024 р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної магістерської роботи

Факультет	Комп'ютерних наук
Кафедра	Комп'ютерної інженерії
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	123 Комп'ютерна інженерія
Освітня програма	Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерної інженерії
_____ Ірина ЖУРАВСЬКА

«_____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Жуланова Максима Олеговича
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Комплекс для моніторингу інтенсивності землетрусів та оцінки наслідків на базі сенсорів IoT
Затверджена наказом ректора ЧНУ ім. Петра Могили від 16.09.2024 № 236.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є: апаратне та програмне забезпечення системи для моніторингу інтенсивності землетрусів та оцінки наслідків на базі сенсорів IoT. Вхідними даними роботи є специфікація вимог, що описують характеристики зазначеного апаратного та програмного забезпечення.

4. Перелік питань, що підлягають розробці
1) аналіз існуючих підходів до прогнозування землетрусів та оцінка їх ефективності;
2) розробка алгоритмів обробки та аналізу даних сенсорів IoT для прогнозування магнітуди землетрусів;
3) можливості застосування машинного навчання для покращення точності прогнозування сейсмічної активності;

4) експериментальна оцінка розроблених алгоритмів та порівняння їх з існуючими методами прогнозування землетрусів; _____

5. Перелік графічних матеріалів

_слайди презентації_____

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Ярослав КРАЙНИК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Максим ЖУЛАНОВ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_ __» __ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної магістерської роботи

Тема: Комплекс для моніторингу інтенсивності землетрусів та оцінки наслідків на базі сенсорів IoT

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КМР	01.09.2024	16.09.2024	Виконано
2	Огляд літератури за темою роботи	17.09.2024	22.09.2024	Виконано
3	Складання календарного плану КМР	23.09.2024	26.09.2024	Виконано
4	Аналіз предметної області	26.09.2024	29.09.2024	Виконано
5	Розробка проектних рішень	30.09.2024	07.10.2024	Виконано
6	Моделювання апаратної частини	08.10.2024	20.10.2024	Виконано
7	Оформлення КМР та презентації	21.10.2024	29.10.2024	Виконано
8	Перший передзахист КМР	30.10.2024	31.10.2024	Виконано
9	Розробка програмного забезпечення та тестування	01.11.2024	22.11.2024	Виконано
10	Відгук керівника КМР	23.11.2024	26.11.2024	Виконано
11	Другий передзахист КМР	28.11.2024	29.11.2024	Виконано
12	Рецензування	30.11.2024	05.12.2024	Виконано
13	Завершення оформлення КМР та презентації	06.12.2024	12.12.2024	Виконано
14	Захист кваліфікаційної роботи	19.12.2024	20.12.2024	Виконано

Керівник роботи _____ Ярослав КРАЙНИК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Максим ЖУЛАНОВ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Комплекс для моніторингу інтенсивності землетрусів та оцінки наслідків на базі сенсорів IoT»

Студент гр. 605м Жуланов Максим Олегович

Керівник: канд. техн. наук, доцент Крайник Ярослав Михайлович

Актуальністю теми є дефіцит систем та засобів моніторингу сейсмічної активності у сучасному світі. Землетруси є одними з найбільш руйнівних природних катастроф, які спричиняють значні людські втрати, руйнування інфраструктури та колосальні економічні збитки. В умовах глобальних кліматичних змін і активізації тектонічних процесів, що спостерігаються в багатьох регіонах світу, питання вдосконалення систем прогнозування та реагування на сейсмічні явища стає критично важливим.

Традиційні методи моніторингу, які використовують сейсмографи або супутникові технології, мають низку обмежень, зокрема високу вартість, складність у розгортанні, особливо у віддалених регіонах, і недостатню оперативність у обробці даних. Ці недоліки унеможливлюють своєчасне попередження та ефективну оцінку наслідків стихійних лих, що посилює ризики для життя людей та критичної інфраструктури.

Об’єкт дослідження (розробки): процеси прогнозування землетрусів та оцінки їх наслідків для інфраструктури.

Предмет дослідження (розробки): методи та алгоритми прогнозування землетрусів на основі даних, зібраних за допомогою сенсорів IoT, а також моделі оцінки наслідків сейсмічних подій для критичної інфраструктури.

Мета: розробка системи для моніторингу інтенсивності землетрусів та прогнозування їх наслідків із використанням сенсорів IoT та методів машинного навчання.

Для досягнення поставленої мети було поставлено такі завдання:

1) проаналізувати існуючі підходи до прогнозування землетрусів та оцінити їх ефективність;
2) розробити алгоритми обробки та аналізу даних сенсорів IoT для прогнозування магнітуди землетрусів;
3) дослідити можливості застосування машинного навчання для покращення точності прогнозування сейсмічної активності.
4) спроєктувати систему для моніторинку і прогнозування землетрусі;
5) провести експериментальну оцінку розроблених алгоритмів та порівняти їх з існуючими методами прогнозування землетрусів.

Кваліфікаційна робота містить: перелік скорочень, вступ, чотири розділи, висновки, перелік джерел посилання та чотири додатка.

Вступ містить основні обґрунтування актуальності розробки обраної теми, об’єкт, предмет дослідження, мету та завдання які необхідно виконати для досягнення поставленої мети.

В першому розділі розглянуто основні теоретичні принципи та проведено аналіз існуючих рішень.

Другий розділ містить опис алгоритмів оцінки інтенсивності землетрусів та моделювання їх наслідків.

У третьому розділі наведено опис процесу розробки та проектування системи: вибір компонентів, архітектура, розробка математичної моделі, вибір технологій, опис реалізацій серверної частини та на стороні контролера.

Четвертий розділ присвячено налаштуванню середовища для тестування системи, експерементальному дослідженню та аналізу результатів.

У висновку описано результати виконання кваліфікаційної роботи.

Додатки містять код програмного забезпечення.

Кваліфікаційна робота містить 130 сторінок (без додатків), 61 рис., 18 табл., 43 літературні джерела та 4 додатки.

Ключові слова: *моніторинг інтенсивності землетрусів, моделі оцінки наслідків, сенсори IoT, алгоритми обробки та аналізу даних, машинне навчання*.

ABSTRACT

of the Master's Thesis

" IoT-Based System for Monitoring Earthquake Intensity and Assessing Consequences"

Applicant: Zhulanov Maksym Olehovych

Supervisor: Ph.D. (Techn.), Associate Professor Kraynyk Yaroslav Mykhailovych

The relevance of the topic lies in the global deficit of systems and tools for monitoring seismic activity. Earthquakes are among the most destructive natural disasters, causing significant human casualties, infrastructure damage, and substantial economic losses. In the context of global climate changes and increasing tectonic activity observed in various regions worldwide, improving systems for earthquake prediction and response has become critically important.

Traditional monitoring methods utilizing seismographs or satellite technologies face several limitations, such as high costs, complexity of deployment (especially in remote areas), and insufficient data processing speed. These shortcomings hinder timely warnings and effective assessment of disaster impacts, increasing risks to human life and critical infrastructure.

The object of the study: processes of earthquake prediction and assessment of their consequences for infrastructure.

Subject of research: methods and algorithms for earthquake prediction based on data collected by IoT sensors, as well as models for assessing the impact of seismic events on critical infrastructure.

Objective: to develop a system for monitoring earthquake intensity and predicting its consequences using IoT sensors and machine learning methods.

To achieve this objective, the following tasks were set:

1) to analyze existing approaches to earthquake prediction and evaluate their effectiveness;

2) to develop algorithms for processing and analyzing IoT sensor data to predict earthquake magnitude;

3) to investigate the potential of machine learning to improve the accuracy of seismic activity prediction;

4) to design a system for earthquake monitoring and prediction;

5) to conduct an experimental evaluation of the developed algorithms and compare them with existing earthquake prediction methods.

The qualification work contains a list of abbreviations, an introduction, four chapters, a conclusion, a list of references, and four appendices.

The introduction provides the main justification for the relevance of the chosen topic, the research object and subject, the objective, and the tasks necessary to achieve it.

The first chapter examines the fundamental theoretical principles and analyzes existing solutions.

The second chapter describes algorithms for assessing earthquake intensity and modeling their consequences.

The third chapter focuses on the development and design of the system, including component selection, architecture, mathematical modeling, technology choices, and implementation of both the server and controller sides.

The fourth chapter is dedicated to setting up the testing environment, conducting experimental research, and analyzing the results.

The conclusion describes the results of the qualification work.

The appendices contain the software code.

The qualification work contains 130 pages (without appendices), 61 figures, 18 tables, 43 references and 4 appendices.

***Keywords**: earthquake intensity monitoring, impact assessment models, IoT sensors, data processing and analysis algorithms, machine learning.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВЛ – метод випадкового лісу
МН – машинне навчання
МНК – метод найменших квадратів
ОВ – метод опорних векторів
СКЖ – система контролю живлення
СППК – метод стохастичного подвійного підйому координат
ШІ – штучний інтелект

GIS – Geographic Information System
GPS – Global Positioning System
I²C – Inter-Integrated Circuit
IoT – Internet of Things
MAE – Mean Absolute Error
MMI – Modified Mercalli Intensity
MQTT – Message Queue Telemetry Transport
NTP – Network Time Protocol
QLARM – Quake Loss Assessment for Response and Mitigation
RNN – Recurrent Neural Network
SDCA – Standardize Do Check Act
SPI – Serial Peripheral Interface
SSID – Service Set Identifier
SSL – Secure Sockets Layer
TLS – Transport Layer Security
UART – Universal Asynchronous Receiver-Transmitter
USB – Universal Serial Bus
Wi-Fi – Wireless Fidelity
WPF – Windows Presentation Foundation
ZHA – Zigbee Home Automation

ВСТУП

Сучасний світ постійно стикається з різноманітними природними катастрофами, серед яких землетруси є одними з найбільш руйнівних та непередбачуваних. Ці події можуть призвести до значних втрат серед населення, руйнувань інфраструктури та серйозних економічних наслідків. Традиційні методи моніторингу сейсмічної активності не завжди можуть забезпечити швидку та точну оцінку інтенсивності землетрусу та його потенційних наслідків [1]. У зв'язку з цим зростає потреба у розробці сучасних технологій, які дозволять не лише оперативно виявляти землетруси, але й своєчасно інформувати відповідні служби для запобігання масштабним втратам та ефективного управління наслідками.

Одним із перспективних напрямів, що здатен значно покращити моніторинг сейсмічної активності, є використання сенсорних технологій на основі Інтернету речей (англ. Internet of Things, IoT) [2]. Сенсори IoT можуть працювати в режимі реального часу, збирати дані про зміну геологічних параметрів і передавати їх на централізовані сервери для подальшого аналізу. Це дозволяє створювати розподілені системи моніторингу, які відрізняються високою точністю, надійністю та доступністю. Такі системи можуть оперативно реагувати на зміни в земній корі, забезпечуючи своєчасне попередження про можливі землетруси та оцінку їх інтенсивності.

На сьогодні існує декілька підходів до моніторингу землетрусів, які базуються на використанні сейсмографів, супутникових систем та інших технологій. Проте ці рішення мають низку недоліків, таких як висока вартість, обмежена доступність та недостатня швидкість реакції на стихійні явища. Зокрема, більшість із них не можуть забезпечити моніторинг у віддалених регіонах, де інфраструктура або відсутня, або знаходиться у ненадійному стані. Крім того, сучасні технології виявлення землетрусів часто не встигають

вчасно опрацювати отримані дані, що унеможливлює оперативне реагування на стихійні лиха.

Враховуючи ці проблеми, розробка системи моніторингу інтенсивності землетрусів на основі IoT набуває особливої актуальності. Поєднання сенсорних технологій з можливостями мереж Інтернету речей дозволить створити розподілену мережу датчиків, яка забезпечить своєчасне виявлення сейсмічної активності, швидке оновлення даних та можливість масштабування системи під конкретні потреби регіону чи об'єкта. Зокрема, така система може стати важливою складовою інфраструктури безпеки України, особливо у сейсмічно активних регіонах, таких як Карпати чи Крим.

Окрім того, впровадження IoT-сенсорів у процес моніторингу дозволяє знизити витрати на установку та обслуговування обладнання, оскільки вони можуть функціонувати автономно протягом тривалого часу, використовуючи малі ресурси енергії. Це робить такі системи доступнішими та придатними для використання як на державному, так і на локальному рівні. Впровадження подібних технологій дозволить значно покращити системи цивільного захисту та підвищити рівень готовності населення до стихійних лих. У той час, як Україна активно розвиває свою технологічну базу, впровадження IoT-систем для моніторингу природних явищ може стати важливим кроком у напрямку забезпечення національної безпеки та сталого розвитку.

Об'єкт дослідження – процеси прогнозування землетрусів та оцінки їх наслідків для інфраструктури.

Предмет дослідження – методи та алгоритми прогнозування землетрусів на основі даних, зібраних за допомогою сенсорів IoT, а також моделі оцінки наслідків сейсмічних подій для критичної інфраструктури.

Метою даного дослідження є розробка системи для моніторингу інтенсивності землетрусів та прогнозування їх наслідків із використанням сенсорів IoT та методів машинного навчання.

Виходячи із поставленої мети, виділено наступні задачі:

– проаналізувати існуючі підходи до прогнозування землетрусів та оцінити їх ефективність;

– розробити алгоритми обробки та аналізу даних сенсорів IoT для прогнозування магнітуди землетрусів;

– дослідити можливості застосування машинного навчання для покращення точності прогнозування сейсмічної активності.

– провести експериментальну оцінку розроблених алгоритмів та порівняти їх з існуючими методами прогнозування землетрусів.

Для досягнення поставленої мети в даному дослідженні були використані наступні методи:

– аналіз літературних джерел та існуючих рішень – застосовувався для дослідження сучасних підходів до моніторингу землетрусів та прогнозування їх магнітуди. Це дозволило оцінити сильні та слабкі сторони існуючих методів, а також визначити напрями вдосконалення;

– математичне моделювання – використовувалось для створення моделей, що описують зв'язок між параметрами сейсмічної активності та магнітудою землетрусів. Завдяки цьому було можливо розробити алгоритми для точного прогнозування на основі даних сенсорів IoT;

– алгоритми машинного навчання – застосовувались для обробки великих обсягів даних, отриманих від сенсорів IoT. Використання методів машинного навчання дозволило поліпшити точність прогнозування магнітуди землетрусів, враховуючи попередні історичні дані та патерни поведінки земної кори;

– експериментальне тестування – проводилось для оцінки ефективності запропонованих алгоритмів прогнозування. Тестування включало порівняння результатів прогнозів із реальними даними про землетруси, що дозволило визначити точність і надійність розроблених моделей;

– статистичний аналіз – застосовувався для обробки результатів тестування та оцінки ефективності розроблених алгоритмів у порівнянні з іншими підходами. Статистичні методи допомогли виявити закономірності та тренди, що впливають на точність прогнозування.

Практичне значення одержаних результатів полягає в можливості використання розробленого підходу для підвищення точності прогнозування землетрусів і оперативного реагування на їх наслідки. Використання алгоритму стохастичного подвійного підйому координат у поєднанні з даними сенсорів IoT дозволяє створити систему, яка може бути впроваджена в реальних умовах для моніторингу сейсмічної активності. Така система сприятиме швидшій оцінці потенційної загрози та зменшенню збитків від землетрусів завдяки своєчасному попередженню відповідних служб і підготовці до можливих наслідків.

Дана кваліфікаційна робота пройшла апробацію на XXI Міжнародній науковій конференції «Ольвійський форум – 2024: стратегії країн Причорноморського регіону в геополітичному просторі». Матеріали доповіді були опубліковані у збірнику конференції [43].

1 ОСНОВНІ ТЕОРЕТИЧНІ ПРИНЦИПИ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ МОНІТОРИНГУ ІНТЕНСИВНОСТІ ЗЕМЛЕТРУСІВ

1.1 Сейсмологічні системи моніторингу: принципи та характеристики

Сейсмологічні системи моніторингу є складними технологічними рішеннями, що призначені для реєстрації сейсмічних подій та аналізу їхніх параметрів. Вони відіграють ключову роль у зменшенні наслідків землетрусів, забезпечуючи своєчасне інформування про сейсмічну активність та оцінку її потужності. Принципи роботи таких систем ґрунтуються на зборі, обробці та аналізі сейсмічних даних, які отримуються від мережі сейсмометрів, розташованих у сейсмічно активних регіонах [3].

На рис. 1.1 зображено діаграму активності процесу роботи сейсмологічної системи. Діаграма ілюструє основні етапи процесу моніторингу сейсмічної активності, від збору даних до надання сповіщень про загрозу землетрусу.

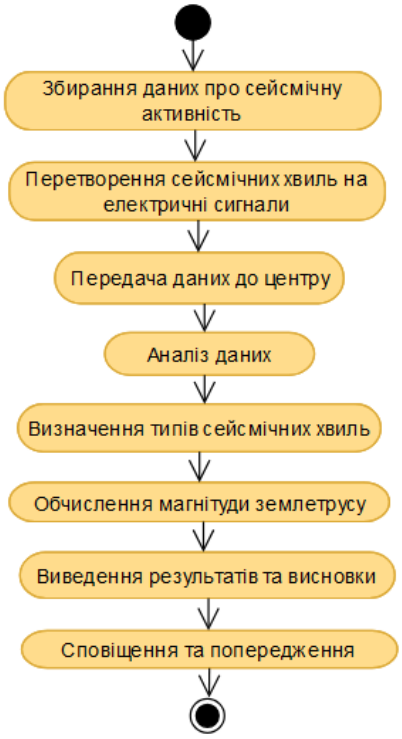


Рисунок 1.1 – Процес роботи сейсмологічної системи

Процес роботи сейсмологічної системи починається зі збору сейсмічних даних: мережа сейсмометрів, розташованих у сейсмічно активних регіонах, реєструє коливання ґрунту, спричинені сейсмічними хвилями, і перетворює їх на електричні сигнали, придатні для подальшої обробки та аналізу. Мережа сейсмометрів, встановлених у різних регіонах, фіксує коливання ґрунту, що виникають внаслідок сейсмічних хвиль [4]. Ці механічні коливання перетворюються на електричні сигнали, які передаються до обчислювального центру через супутникові або наземні комунікаційні мережі.

В обчислювальному центрі дані обробляються і аналізуються для визначення типів сейсмічних хвиль. Система ідентифікує поздовжні P-хвилі, поперечні S-хвилі та поверхневі хвилі для отримання детальної інформації про подію. На основі цього аналізу встановлюється точне місце епіцентру та глибина джерела землетрусу.

Розрахунок магнітуди відображає кількість вивільненої енергії під час землетрусу, а інтенсивність оцінює вплив на поверхню та інфраструктуру. Отримані результати, включаючи магнітуду, інтенсивність і координати епіцентру, виводяться для подальшого використання в ухваленні рішень [5]. Система автоматично надсилає сповіщення та попередження організаціям або населенню для своєчасного реагування на потенційну загрозу.

Основні характеристики сейсмологічних систем моніторингу визначають їхню здатність до ефективного виявлення та аналізу сейсмічних подій [6]. Важливими аспектами таких систем є їхня чутливість, швидкодія, точність, надійність та здатність до швидкого реагування на зміни в сейсмічній активності. Ці параметри дозволяють системам працювати в реальному часі, забезпечуючи постійний моніторинг та своєчасне попередження про землетруси. Ефективність сейсмологічних систем визначається також їхньою здатністю до обробки великих обсягів даних, отриманих від мереж сейсмометрів, і швидкістю передачі результатів до центрів обробки інформації. Таким чином, основні характеристики є критичними для

досягнення високого рівня точності та надійності у прогнозуванні та оцінці землетрусів.

На рис. 1.2 представлено основні характеристики сейсмологічних систем моніторингу.

Рисунок 1.2 – Основні характеристики сейсмологічних систем

Чутливість систем визначається їхньою здатністю фіксувати найменші коливання ґрунту. Сучасні сейсмометри можуть реєструвати навіть мікросейсмічні рухи, що дозволяє виявляти землетруси низької магнітуди [7]. Швидкодія системи забезпечується здатністю миттєво обробляти отримані дані та передавати їх для аналізу в режимі реального часу. Точність полягає в здатності системи точно визначати епіцентр, магнітуду та інтенсивність землетрусу, що дозволяє оцінити вплив події на певний регіон. Надійність характеризує стійкість системи до зовнішніх факторів, таких як пошкодження під час землетрусу, що забезпечує безперервне функціонування.

1.2 Технології IoT для моніторингу сейсмічної активності

Технології IoT стрімко стають незамінним інструментом для моніторингу природних катастроф, зокрема землетрусів. Вони дозволяють здійснювати постійний збір та передачу даних про сейсмічну активність у реальному часі, що значно підвищує ефективність оцінки потенційних ризиків

і мінімізує час реагування на катастрофи. Використання IoT дозволяє автоматизувати процеси збору, аналізу та передачі інформації, що є критичним для забезпечення оперативної оцінки сейсмічних подій.

Одна з ключових переваг IoT полягає у безперервному зборі даних за допомогою сейсмічних сенсорів, що можуть бути розміщені у віддалених або важкодоступних місцях [8]. Сенсори IoT постійно вимірюють коливання ґрунту та інші геологічні параметри, передаючи ці дані через супутникові чи інші бездротові мережі на центральні сервери для подальшого аналізу. Це дозволяє забезпечити всебічне покриття зони ризику, включно з регіонами, де традиційні системи моніторингу можуть бути недоступними.

Збір даних у реальному часі є одним з основних факторів, які роблять технології IoT надзвичайно ефективними для моніторингу землетрусів. Дані, які надходять у режимі реального часу, дозволяють сейсмологам та екстреним службам своєчасно реагувати на сейсмічні події та прогнозувати їх наслідки. Багато сенсорів IoT використовують сучасні технології, такі як підключені сейсмометри, акселерометри та геофони, які здатні виявляти навіть найменші коливання земної поверхні.

Автоматизація процесу аналізу даних є важливою характеристикою IoT для моніторингу землетрусів. Дані, отримані від сенсорів, автоматично обробляються у спеціалізованих програмних системах, що дає можливість одразу оцінити магнітуду землетрусу, його інтенсивність та вплив на інфраструктуру. Така автоматизація дозволяє значно скоротити час між виявленням події та її аналізом, що може бути вирішальним для забезпечення своєчасного попередження.

Окрім точності та швидкодії, технології IoT мають високу надійність у складних умовах. Вони використовують різноманітні комунікаційні канали для передачі даних, такі як супутникові, мобільні та Wi-Fi мережі, що дозволяє забезпечити безперервний зв'язок навіть у разі пошкодження традиційних засобів комунікації під час сильних поштовхів. Ця багатоканальність робить

системи більш стійкими до збоїв та гарантує стабільну передачу даних у будь-яких умовах.

IoT дозволяє координувати роботу різних сенсорів і забезпечує централізований збір даних для їх обробки в єдиній системі (рис. 1.3). Це сприяє не лише швидкій оцінці сейсмічної активності, але й координації дій між різними відомствами, що займаються реагуванням на стихійні лиха.

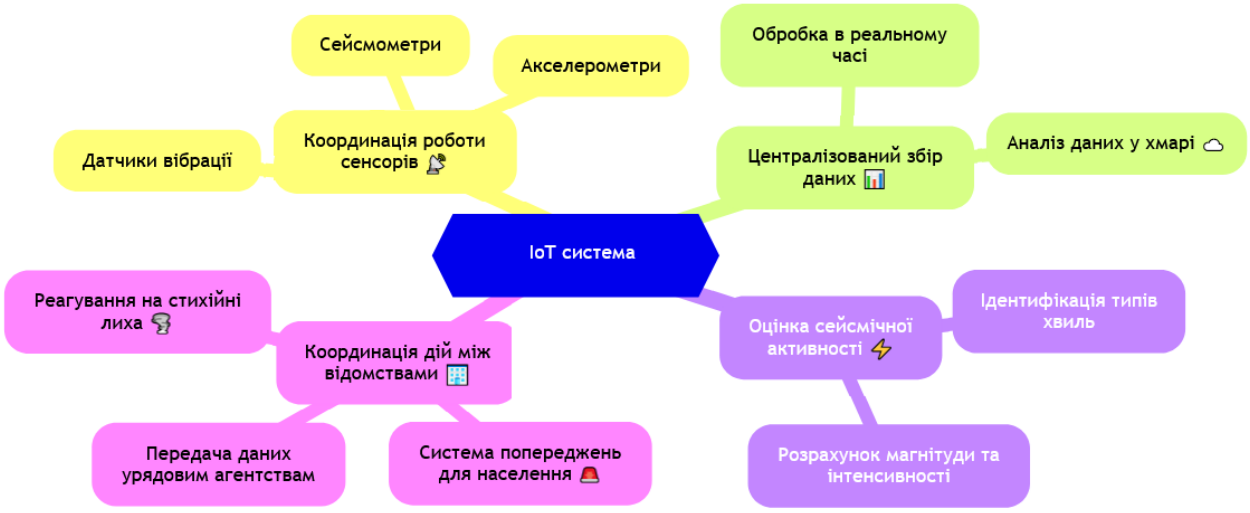


Рисунок 1.3 – Приклад централізованого збору даних за допомогою IoT

Технології IoT не тільки збільшують швидкість отримання даних, але й сприяють точнішому визначенню параметрів землетрусів, що є критичним для прогнозування їхніх наслідків та підготовки до подальших дій.

Важливою перевагою технологій IoT у сейсмологічному моніторингу є здатність забезпечувати точніше та швидше реагування на землетруси [9]. Завдяки безперервному оновленню даних з різноманітних джерел можна складати більш точні прогнози щодо подальших поштовхів або повторних землетрусів, що допомагає оптимізувати евакуаційні заходи та дії з реагування на надзвичайні ситуації.

Також значною перевагою є можливість моделювати наслідки на основі отриманих даних. Сенсори IoT можуть не лише фіксувати сейсмічні коливання, але й збирати інформацію про стан інфраструктурних об'єктів, таких як мости, дороги та будівлі. Це дозволяє точно оцінити масштаби можливих збитків і розробити плани відновлення пошкоджених об'єктів.

Застосування IoT у сейсмології має значний потенціал у поєднанні з іншими технологіями, такими як штучний інтелект (ШІ) та машинне навчання (МН) [10]. Алгоритми машинного навчання можуть аналізувати великі обсяги даних, отримані з сенсорів IoT, і виявляти складні патерни у сейсмічній активності. Це дозволяє покращити прогнози землетрусів та мінімізувати кількість хибних сповіщень.

Впровадження технологій IoT у систему моніторингу сейсмічної активності має величезний потенціал для забезпечення безпеки населення та зниження ризиків під час землетрусів [11]. Завдяки можливості безперервного збору даних, автоматизації обробки та надійній передачі інформації в реальному часі, технології IoT стають незамінними інструментами для боротьби зі стихійними лихами.

1.3 Огляд існуючих систем для оцінки наслідків землетрусів

Оцінка наслідків землетрусів є ключовим завданням для мінімізації збитків і ефективного реагування на стихійні лиха. Існуючі системи для оцінки наслідків землетрусів використовують широкий спектр технологій, включаючи моделювання руйнувань, аналіз інфраструктури та прогнозування ризиків для населення. Ці системи дозволяють не лише фіксувати сейсмічні події, але й надавати детальні дані про потенційні пошкодження будівель, інженерних споруд та критичних об'єктів, а також оцінювати можливі загрози для життя людей. Вони інтегрують сейсмологічні дані з іншими джерелами інформації, такими як географічні інформаційні системи (англ. Geographic Information System, GIS) та технології IoT, для створення всебічної картини наслідків землетрусу.

Сучасні системи оцінки наслідків включають як глобальні, так і локальні рішення, здатні адаптуватися до специфіки різних регіонів. Важливими критеріями ефективності таких систем є швидкість обробки даних, точність

прогнозів, а також здатність оперативно надавати рекомендації щодо евакуації та розподілу ресурсів для ліквідації наслідків.

SeisComP3 – це сейсмологічна система, призначена для реєстрації, обробки та поширення даних про землетруси в реальному часі (рис. 1.4) [12]. Вона використовується для моніторингу сейсмічної активності та швидкої оцінки сейсмічних подій, що дозволяє оперативно реагувати на землетруси та надавати попередження.

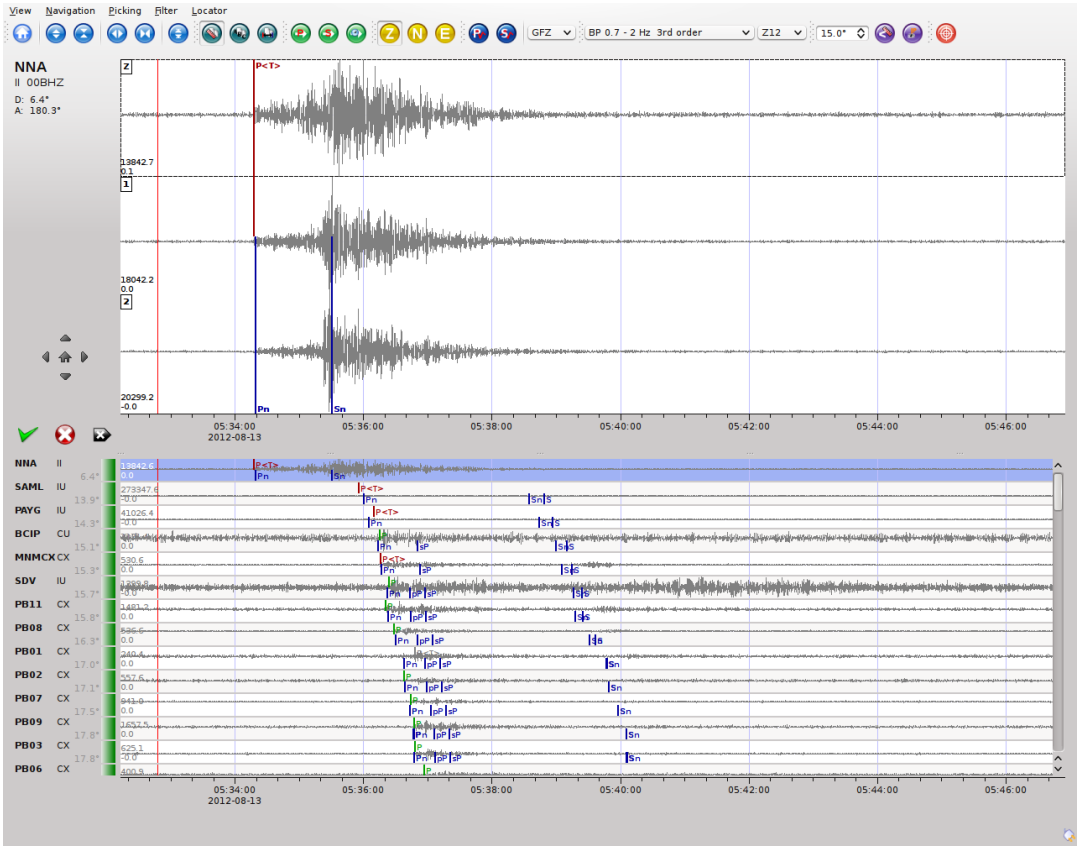


Рисунок 1.4 – Приклад інтерфейсу для роботи із «SeisComP3» [13]

Архітектура системи SeisComP3 складається з кількох компонентів, які взаємодіють між собою для забезпечення безперебійного збору та аналізу даних. Основою є мережа сейсмометрів, які фіксують сейсмічні коливання та передають їх до центральних серверів для обробки [14]. Система використовує спеціалізовані програми для автоматичної обробки сигналів, визначення епіцентру землетрусу, його магнітуди та інших параметрів. Оброблені дані зберігаються в базі даних, доступній для подальшого аналізу. SeisComP3 також підтримує розподілену обробку даних, що дозволяє підключати нові

сейсмічні станції та масштабувати систему для моніторингу землетрусів у різних регіонах світу.

Переваги SeisComP3 складаються із:

– робота у реальному часі. Система забезпечує обробку даних про землетруси в режимі реального часу, що дозволяє оперативно реагувати на події;

– масштабованість. Архітектура SeisComP3 підтримує розподілену обробку даних і може бути легко розширена за рахунок підключення нових сейсмічних станцій;

– автоматична обробка. Система здатна автоматично визначати основні параметри землетрусу, зменшуючи потребу в ручному втручанні;

– гнучка архітектура. SeisComP3 підтримує різноманітні формати даних і може інтегруватися з іншими сейсмологічними системами.

Недоліки SeisComP3 включають:

– складність налаштування. Система потребує високого рівня технічних знань для налаштування та підтримки;

– обмежена підтримка для початкових користувачів. Документація може бути складною для новачків;

– високі вимоги до інфраструктури. Для належного функціонування потрібна надійна комунікаційна інфраструктура;

– витрати на обслуговування. Розгортання системи та її підтримка можуть бути досить дорогими для деяких організацій.

Отже, SeisComP3 є потужною системою для моніторингу землетрусів, яка забезпечує швидку та ефективну обробку даних у режимі реального часу. Завдяки своїй масштабованій та гнучкій архітектурі, система підходить для великих сейсмологічних мереж, але вимагає значних технічних ресурсів і знань для налаштування та обслуговування.

OpenQuake – це програмна платформа, призначена для оцінки сейсмічних ризиків та моделювання наслідків землетрусів (рис. 1.5) [15]. Вона

розроблена для науковців, інженерів та урядових організацій з метою проведення сейсмічних досліджень та створення прогнозів щодо потенційних пошкоджень інфраструктури в разі землетрусу. OpenQuake допомагає оцінити сейсмічні загрози, розрахувати ймовірні збитки та підготувати заходи щодо зниження ризиків.

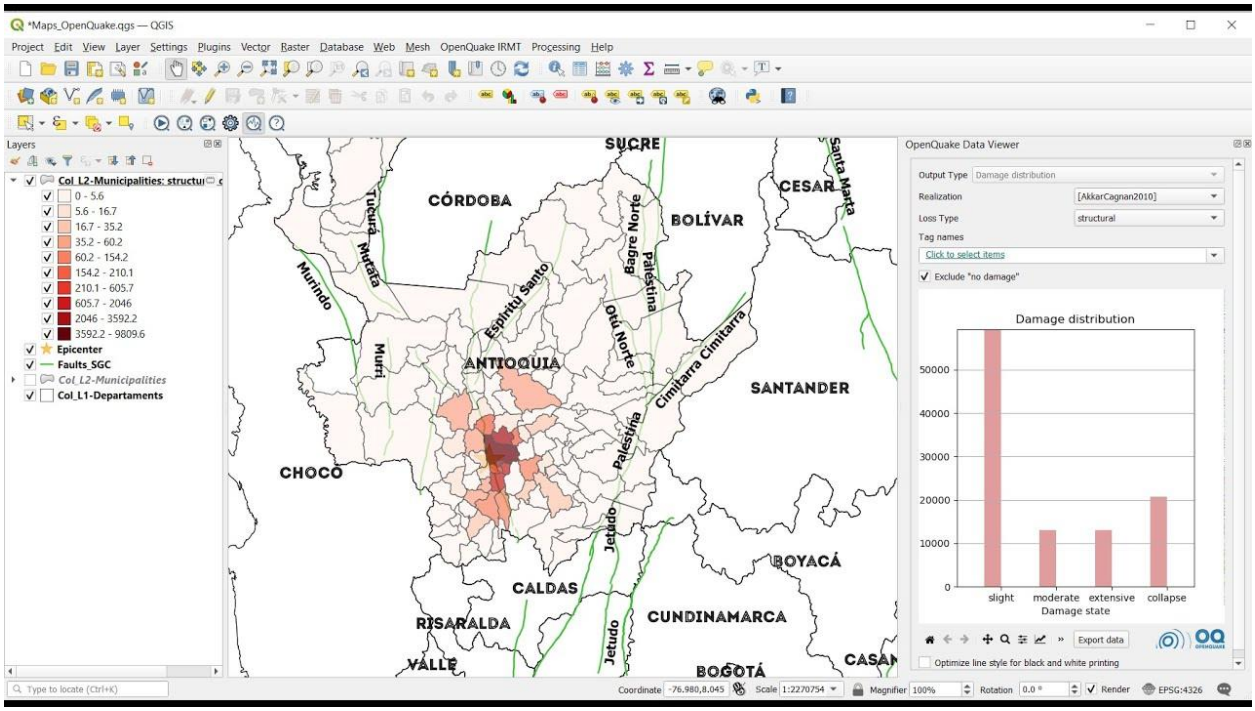


Рисунок 1.5 – Приклад інтерфейсу для роботи із «OpenQuake» [16]

Архітектура OpenQuake базується на модульній структурі, що дозволяє користувачам обирати різні моделі та інструменти для оцінки ризиків. Система складається з ядра, яке відповідає за обчислення та моделювання сейсмічних подій, баз даних для збереження інформації про ризики, та інтерфейсів для взаємодії з користувачем і зовнішніми системами. OpenQuake підтримує обробку великих обсягів даних та може виконувати складні розрахунки за допомогою розподілених обчислень на кластерних або хмарних платформах. Це дозволяє проводити масштабні сейсмічні дослідження та створювати детальні прогнози з використанням передових наукових моделей.

Переваги OpenQuake [17]:

– гнучкість. Підтримка різних моделей оцінки ризиків та сценаріїв землетрусів дозволяє адаптувати систему під конкретні завдання;

– відкритий код. Платформа з відкритим вихідним кодом, що дає можливість користувачам вільно модифікувати та вдосконалювати систему;

– розподілені обчислення. Підтримує обробку великих обсягів даних за допомогою кластерних і хмарних платформ, що підвищує продуктивність і дозволяє проводити масштабні дослідження;

– детальна оцінка збитків. Система забезпечує точні прогнози щодо руйнувань інфраструктури та можливих збитків від землетрусів.

Недоліки OpenQuake:

– складність для початківців. Система вимагає значних знань у галузі сейсмології та програмування, що ускладнює її використання новими користувачами;

– високі вимоги до ресурсів. Для обробки великих обсягів даних потрібні значні обчислювальні ресурси, що може бути проблемою для невеликих організацій;

– необхідність спеціальної підтримки. Встановлення та налаштування системи потребує професійних знань, а також можуть виникнути труднощі з інтеграцією;

– залежність від якості вхідних даних. Точність оцінки залежить від повноти та якості даних, які вводяться в систему.

Отже, OpenQuake є потужним інструментом для моделювання сейсмічних ризиків та прогнозування наслідків землетрусів, який вирізняється гнучкістю, відкритістю та підтримкою розподілених обчислень. Однак система може бути складною для початківців та вимагає значних ресурсів для обробки великих обсягів даних, що робить її більш придатною для великих науково-дослідних або урядових організацій.

QLARM (Quake Loss Assessment for Response and Mitigation) – це потужна система, призначена для детальної оцінки втрат та наслідків, викликаних землетрусами, включаючи прогнозування кількості жертв, пошкоджень інфраструктури та соціально-економічних збитків (рис. 1.6) [18].

Вона здатна аналізувати параметри землетрусу в режимі реального часу, такі як магнітуда, епіцентр і глибина, і на основі цих даних передбачати наслідки для регіону. Завдяки цьому система дозволяє швидко оцінити ситуацію та допомагає у прийнятті оперативних рішень для мінімізації збитків та організації рятувальних робіт. QLARM також враховує щільність населення, типи будівель та інші характеристики інфраструктури, що робить прогнози більш точними та деталізованими. Система розроблена спеціально для екстреного реагування, щоб забезпечити максимально ефективне управління ресурсами в перші години після землетрусу.

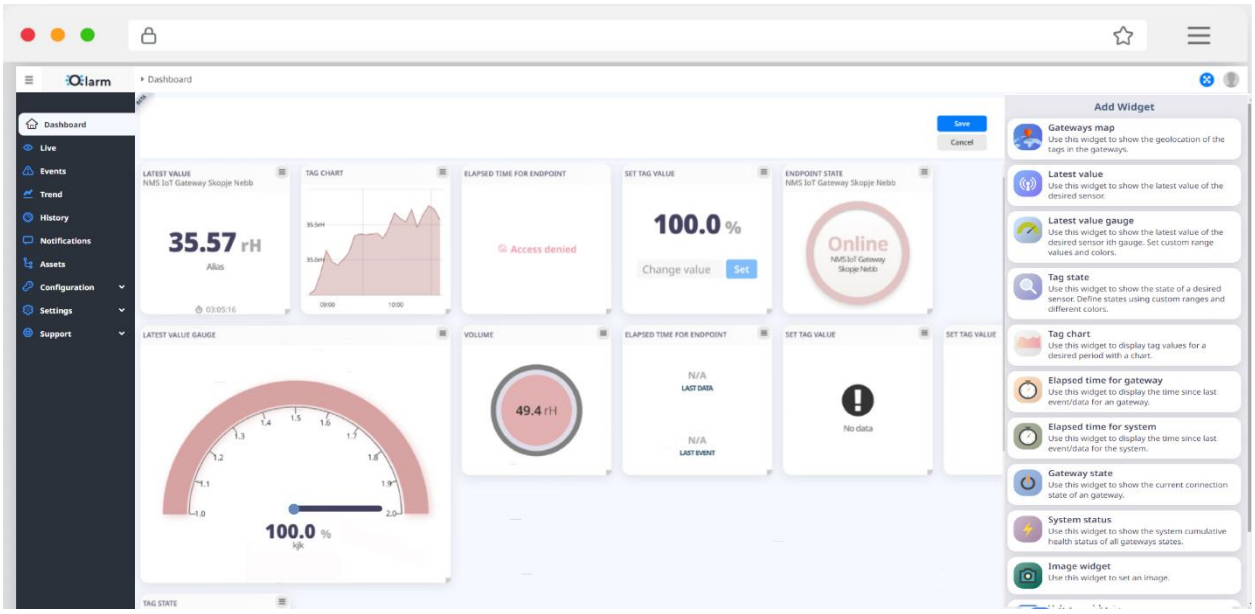


Рисунок 1.6 – Приклад інтерфейсу для роботи із «QLARM» [19]

Архітектура QLARM базується на поєднанні моделей оцінки втрат та алгоритмів прогнозування на основі даних про сейсмічні події та параметри територій, що піддаються впливу. Система використовує сейсмологічні дані про землетрус, такі як магнітуда, глибина та локація епіцентру, для швидкої оцінки можливих наслідків. Ці дані обробляються разом з інформацією про щільність населення, типи будівель та інфраструктури, що дозволяє точно оцінити можливі пошкодження та кількість жертв. QLARM також підтримує розподілену обробку даних для оперативної оцінки втрат у великих масштабах.

Переваги QLARM:

– оперативність. Система надає прогнози про втрати в режимі реального часу, що дозволяє швидко реагувати на землетруси;

– прогнозування жертв і пошкоджень [20]. QLARM здатен прогнозувати не лише руйнування інфраструктури, але й кількість потенційних жертв, що сприяє точнішій оцінці загрози;

– інтеграція з демографічними даними. Врахування щільності населення та типів будівель дозволяє робити більш точні прогнози щодо наслідків землетрусів;

– підтримка розподіленої обробки. Система може ефективно обробляти великі обсяги даних, що важливо для оцінки втрат на великих територіях.

Недоліки QLARM:

– залежність від якості вхідних даних. Точність оцінки безпосередньо залежить від точності та повноти даних про будівлі, населення та інфраструктуру;

– складність налаштування. Для використання системи потрібні спеціалізовані знання та професійна підготовка, що обмежує доступність для нових користувачів;

– високі вимоги до ресурсів. Обробка великого обсягу даних потребує потужних обчислювальних ресурсів, що може ускладнювати використання в менш розвинених регіонах;

– необхідність оновлення даних. Для отримання точних прогнозів потрібне постійне оновлення бази даних щодо населення та інфраструктури, що може бути складно в регіонах з обмеженими ресурсами.

Отже, QLARM є ефективною системою для прогнозування наслідків землетрусів, особливо в частині оцінки втрат серед населення та руйнувань інфраструктури. Вона вирізняється своєю оперативністю та точністю завдяки інтеграції демографічних даних, але потребує якісних вхідних даних та

значних обчислювальних ресурсів. Це робить її цінним інструментом для екстреного реагування, хоча її використання може бути складним для невеликих організацій або регіонів із обмеженими ресурсами.

Виходячи з аналізу існуючих рішень, таких як SeisComP3, OpenQuake та QLARM, можна зробити висновок, що ці системи мають високу ефективність у прогнозуванні та оцінці наслідків землетрусів, але вони також мають певні обмеження. Більшість з них потребують значних обчислювальних ресурсів, складного налаштування та спеціалізованих знань для роботи, що обмежує їх доступність для менш підготовлених користувачів. У зв'язку з цим доцільним є розробка нового рішення, яке використовуватиме штучний інтелект для аналізу даних, забезпечить простіший інтерфейс і функціонал, легкість впровадження навіть у невеликих організаціях та безкоштовну базову версію. Це дозволить зробити технологію доступнішою для ширшого кола користувачів, підвищивши ефективність прогнозування та реагування на землетруси в різних регіонах.

1.4 Порівняльний аналіз сучасних підходів до прогнозування землетрусів

Прогнозування землетрусів залишається однією з найбільш складних задач у галузі сейсмології, оскільки землетруси є непередбачуваними явищами, які можуть виникати без помітних попереджувальних ознак. Незважаючи на це, розвиток сучасних технологій, таких як IoT, штучний інтелект, машинне навчання та аналіз великих даних, відкриває нові можливості для покращення точності прогнозів. Існує декілька підходів до прогнозування землетрусів, які активно досліджуються та застосовуються у світовій практиці. Серед них можна виділити такі основні підходи: традиційні сейсмологічні методи, статистичні моделі, методи машинного навчання, а також новітні гібридні підходи, які комбінують різні технології.

Традиційні методи прогнозування землетрусів базуються на аналізі попередніх сейсмічних подій і сейсмологічних даних, отриманих за допомогою сейсмометрів. Вони включають вивчення сейсмічної активності у певних регіонах, зокрема визначення повторюваних землетрусів та вивчення активності розломів. Одним з основних інструментів у цьому підході є аналіз форшоків та афтершоків, які можуть бути передвісниками основного землетрусу [21]. Хоча ці методи є базовими і довгий час залишалися основними у сейсмології, їх точність є обмеженою, оскільки вони не дозволяють точно передбачити час і місце виникнення землетрусу.

Статистичні методу прогнозування землетрусів базуються на аналізі минулих сейсмічних подій з метою визначення закономірностей та побудови ймовірнісних прогнозів [22]. Одним із прикладів такого підходу є модель Гутенберга-Ріхтера, яка дозволяє оцінювати ймовірність виникнення землетрусу певної магнітуди на основі попередніх подій. Такі моделі корисні для довгострокового прогнозування в регіонах з високою сейсмічною активністю, однак їх ефективність для короткострокових прогнозів є обмеженою. Статистичні моделі надають загальне уявлення про сейсмічні ризики в регіоні, але не можуть точно передбачити конкретну подію.

З розвитком ШІ та МН з'явилася можливість аналізувати великі обсяги сейсмічних даних для прогнозування землетрусів з більшою точністю. Алгоритми машинного навчання, такі як нейронні мережі та методи кластеризації, дозволяють знаходити приховані закономірності у даних, які неможливо виявити за допомогою традиційних методів. Наприклад, використання рекурентних нейронних мереж (англ. Recurrent Neural Network, RNN) дозволяє аналізувати тимчасові послідовності сейсмічної активності для виявлення передвісників землетрусів [23]. Однією з основних переваг цього підходу є можливість швидкого аналізу великих масивів даних та виявлення патернів, що можуть свідчити про наближення землетрусу. Проте цей метод

також залежить від якості та обсягу навчальних даних, а також потребує потужних обчислювальних ресурсів.

Гібридні підходи поєднують різні методи для покращення точності прогнозування землетрусів. Це можуть бути комбінації традиційних сейсмологічних методів із сучасними технологіями аналізу даних, такими як машинне навчання або штучний інтелект. Наприклад, комбінування аналізу форшоків з методами машинного навчання дозволяє значно покращити точність короткострокових прогнозів [24]. Також активно досліджуються підходи, що використовують дані з IoT-сенсорів, які в реальному часі збирають інформацію про геологічні процеси та передають її для обробки за допомогою алгоритмів штучного інтелекту. Це дозволяє створювати динамічні моделі прогнозування, які постійно оновлюються на основі нових даних, забезпечуючи більш актуальну оцінку ризиків.

Нижче представлена порівняльна табл. 1.1, що ілюструє основні характеристики різних підходів до прогнозування землетрусів.

Таблиця 1.1 – Порівняння методів для прогнозування землетрусів

Характеристика	**Традиційні методи**	**Статистичні методи**	**ШІ та МН**	**Гібридні підходи**
Точність прогнозування	Низька для короткострокових прогнозів	Середня, залежить від історичних даних	Висока, особливо для аналізу великих даних	Висока, за рахунок поєднання різних підходів
Швидкість обробки даних	Висока, але з низькою деталізацією	Середня, потребує часу для побудови моделей	Висока, завдяки автоматизації та аналізу даних	Висока, завдяки поєднанню ШІ з іншими методами
Можливість обробки великих даних	Обмежена можливостями обчислювальних ресурсів	Працює з великими обсягами даних	Чудово підходить для великих обсягів даних	Чудово підходить для великих обсягів даних

Характеристика	Традиційні методи	Статистичні методи	ШІ та МН	Гібридні підходи
Адаптивність	Обмежена, важко інтегрується з новими даними	Залежить від моделі, необхідне оновлення	Висока, легко адаптується до нових даних	Висока, особливо з використанням ШІ
Складність впровадження	Низька, прості у використанні	Середня, потребує певної підготовки	Висока, потребує технічних знань	Висока, через поєднання кількох технологій

Як видно з таблиці, методи на основі штучного інтелекту та машинного навчання демонструють значні переваги в точності та швидкості обробки даних, особливо для аналізу великих обсягів інформації. Гібридні підходи, поєднуючи традиційні та інноваційні методи, забезпечують ще більшу точність прогнозування завдяки використанню сильних сторін кожного з підходів. Це робить їх перспективними для подальшого розвитку та впровадження в системи прогнозування землетрусів, дозволяючи підвищити ефективність реагування на природні катастрофи.

1.5 Постановка завдання

Для ефективної оцінки інтенсивності землетрусів та прогнозування їх наслідків необхідно побудувати комплексну систему, яка базується на використанні технологій IoT та машинного навчання. Основні аспекти вирішення цієї проблеми полягають в розробці алгоритмів збору та обробки даних, їх аналізу та прогнозування можливих наслідків.

Проблема моніторингу землетрусів у реальному часі полягає в необхідності безперервного отримання точних даних із сенсорів, їх обробки з метою оцінки сейсмічної активності, а також своєчасного реагування на загрози. Основним завданням є створення архітектури IoT-системи, яка

забезпечить передачу даних від сенсорів до контролера, їх запис у базу даних і подальшу обробку з використанням алгоритмів машинного навчання для прогнозування інтенсивності та наслідків землетрусів.

Основними завданнями цієї роботи є:

– розробка архітектури IoT-системи для збору даних із сенсорів та їх обробки контролером із записом у базу даних;

– створення алгоритмів обробки отриманих даних для оцінки інтенсивності землетрусів на основі сенсорної інформації;

– впровадження методів машинного навчання для прогнозування магнітуди землетрусів на основі зібраних даних;

– реалізація алгоритмів для оцінки потенційних наслідків землетрусів на основі параметрів подій та геоданих.

Специфікація вимог до апаратно-програмного забезпечення системи визначає ключові функціональні, технічні, нефункціональні вимоги, а також вимоги до комплектуючих, необхідних для успішної розробки та впровадження системи моніторингу та прогнозування землетрусів на основі IoT-технологій.

Функціональні вимоги:

– збір та обробка даних з сенсорів IoT. Система повинна виконувати збір даних з різних типів сенсорів (акселерометри, геофони, магнітометри, барометри, GPS) у режимі реального часу. Контролер повинен обробляти ці дані, записувати їх у базу даних для подальшого аналізу;

– алгоритми оцінки інтенсивності землетрусів. Впроваджені алгоритми повинні забезпечувати оцінку інтенсивності землетрусів на основі показників, зібраних сенсорами, таких як амплітуда коливань, глибина та інтенсивність сейсмічних хвиль;

– прогнозування магнітуди землетрусів. Використовувати методи машинного навчання для прогнозування магнітуди землетрусів на основі історичних даних та поточних показників;

– оцінка потенційних наслідків землетрусів. Реалізовані алгоритми повинні оцінювати можливі пошкодження інфраструктури (мости, будівлі, дороги) на основі магнітуди землетрусу, геоданих та типу інфраструктурних об'єктів.

Технічні вимоги:

– архітектура IoT-системи. Система повинна включати контролер Raspberry Pi Pico та модулі для збору даних з сенсорів. Контролер повинен підтримувати бездротові протоколи зв'язку (Wi-Fi, ZigBee) для з'єднання із сенсорами та передавання даних на сервер;

– база даних. Дані повинні зберігатися в централізованій базі даних для подальшого аналізу. База даних повинна підтримувати великі обсяги даних з можливістю їх швидкого доступу для обробки;

– методи машинного навчання. Система повинна використовувати алгоритми машинного навчання для тренування моделі прогнозування землетрусів. Обрано стохастичний метод подвійного підйому координат для його ефективності у прогнозуванні магнітуди;

– серверна інфраструктура. Система повинна включати сервер для обробки запитів, виконання алгоритмів машинного навчання та обробки великих обсягів даних.

Нефункціональні вимоги:

– надійність. Система повинна працювати безперебійно в умовах великих обсягів даних та підвищеної сейсмічної активності. Збої у зв'язку із сенсорами мають мінімізуватися шляхом автоматичного перепідключення;

– масштабованість. Архітектура системи повинна дозволяти додавання нових сенсорів без значних змін у системі;

– безпека. Дані, що передаються від сенсорів до бази даних і сервера, повинні бути захищені від несанкціонованого доступу. Використання SSL/TLS для забезпечення безпеки передачі даних;

– продуктивність. Система повинна забезпечувати обробку даних у реальному часі без затримок. Час обробки одного циклу збору та обробки даних не повинен перевищувати 1 секунди.

Вимоги до комплектуючих:

Контролер:

– Raspberry Pi Pico. Використовується для збору та обробки даних з сенсорів, виконання алгоритмів і передачі даних до серверу.

Сенсори:

1) Geospace Y-28 – для збору даних про сейсмічні хвилі;
2) ADXL345 (3-Axis Accelerometer) – для вимірювання коливань у трьох осях;
3) NEO-6M GPS – для визначення точного місця розташування під час сейсмічної події;
4) Geosense Seismic Hydrophone GD-25-11A – для моніторингу сейсмічної активності у водному середовищі;
5) BMP280 Barometric Pressure Sensor – для вимірювання тиску;
6) INFRA20 – для моніторингу інфразвукових коливань;
7) HMC5883L Triple Axis Magnetometer – для моніторингу магнітного поля;
8) DS18B20 Temperature Sensor – для вимірювання температури;
9) Portable Gravimeter Scintrex CG-5 – для вимірювання гравітаційних змін під час землетрусів.

Отже, система повинна забезпечувати автоматизоване збирання даних про землетруси, їх аналіз та прогнозування, що дозволить своєчасно оцінювати ризики та приймати відповідні заходи.

Висновки до розділу 1

У рамках даного розділу проведено детальний аналіз існуючих рішень та технологій для моніторингу інтенсивності землетрусів і оцінки їх наслідків.

Розглянуто принципи роботи сейсмологічних систем, що дозволило сформувати загальне розуміння ключових характеристик таких систем, включаючи їх чутливість, швидкодію та точність. Визначено, що технології IoT відіграють важливу роль у централізованому зборі та обробці даних, забезпечуючи ефективне керування великою кількістю сенсорів. Проаналізовано існуючі системи для оцінки наслідків землетрусів, такі як SeisComP3, OpenQuake і QLARM, що дозволило виявити їхні переваги та недоліки, а також обґрунтувати необхідність розробки нового рішення, яке поєднуватиме простіший функціонал, використання штучного інтелекту і машинного навчання, а також забезпечуватиме легкість впровадження.

Виконано порівняльний аналіз підходів до прогнозування землетрусів, де особливу увагу приділено методам машинного навчання та штучного інтелекту, які демонструють значну перевагу в точності та адаптивності порівняно з традиційними та статистичними методами. Переваги цих інноваційних підходів підтверджують доцільність їх використання в новій системі.

На основі проведеного аналізу було сформульовано постановку завдання, яка стане основою для подальшої розробки системи моніторингу та прогнозування землетрусів у наступному розділі, з урахуванням особливостей IoT, машинного навчання та геоданих для забезпечення більш точного прогнозування та оцінки наслідків.

2 АЛГОРИТМИ ОЦІНКИ ІНТЕНСИВНОСТІ ЗЕМЛЕТРУСІВ ТА МОДЕЛЮВАННЯ ЇХ НАСЛІДКІВ

2.1 Моделі інтенсивності землетрусів на основі даних сенсорів IoT

Оцінка інтенсивності землетрусів на основі даних сенсорів IoT дозволяє створювати точні математичні моделі, що забезпечують високу точність прогнозування землетрусів. Сенсори IoT забезпечують безперервний збір даних про коливання земної поверхні, які можуть використовуватись для оцінки інтенсивності землетрусу в реальному часі. Нижче розглянуто кілька найбільш поширених математичних моделей, що використовуються для обробки даних сенсорів IoT.

Сейсмічна модель прогнозування інтенсивності землетрусів на основі даних акселерометрів IoT використовує реальні дані прискорення ґрунту для визначення інтенсивності землетрусів [25]. Ця модель побудована на математичних залежностях, що дозволяють обчислювати інтенсивність землетрусу за його динамічними характеристиками.

Дані з акселерометрів надають показники прискорення ґрунту в трьох осях: a_x, a_y, a_z – прискорення в напрямках осей x, y, z.

Результуюче прискорення можна обчислити як векторну суму компонент прискорення:

$$a_{res} = \sqrt{a_x^2 + a_y^2 + a_x^2} \tag{2.1}$$

Формула дозволяє визначити загальне прискорення, яке відчувається на даній точці земної поверхні під час землетрусу. Сама інтенсивність землетрусу пов'язана з амплітудою прискорення. Відповідність між прискоренням ґрунту і рівнем інтенсивності (наприклад, за шкалою Меркаллі або Ріхтера) може бути представлена як залежність:

$$I = k \cdot \log(a_{res}), \tag{2.2}$$

де I – інтенсивність землетрусу;

k – емпіричний коефіцієнт, що залежить від характеристик ґрунту та будівель у регіоні;

a_{res} – результуюче прискорення.

Дані акселерометрів часто містять шум, тому перед розрахунками необхідно застосувати методи фільтрації сигналу, наприклад, фільтр Калмана або низькочастотний фільтр:

$$a_{filtered}(t) = \int_0^t \alpha a_{raw}(t) + (1-\alpha)\, a_{filtered}(t-1)dt, \qquad (2.3)$$

де $a_{filtered}(t)$ – відфільтроване прискорення у момент часу t;

$a_{raw}(t)$ – необроблені дані;

α – коефіцієнт фільтрації.

Для прогнозування інтенсивності землетрусу використовуються методи машинного навчання. Дані з акселерометрів подаються на вхід моделі, наприклад, нейронної мережі або регресійної моделі, яка передбачає рівень інтенсивності на основі тренувальних даних.

Переваги використання моделі прогнозування інтенсивності на основі даних акселерометрів IoT складаються із:

– масштабованість. Дані можуть збиратися з численних датчиків, покриваючи великі території;

– точність. Завдяки використанню реальних показників можна більш точно оцінити інтенсивність у кожній точці;

– прогнозування в реальному часі. Завдяки швидкому аналізу даних можна оперативно прогнозувати інтенсивність землетрусів.

Модель дозволяє інтегрувати дані з акселерометрів IoT для точного прогнозування інтенсивності землетрусів і може використовуватися як основа для сейсмічних систем раннього попередження.

У свою чергу модель розподілу енергії землетрусів на основі даних із сенсорів IoT використовується для оцінки енергії, яка вивільняється під час землетрусу, та її розподілу в різних точках на поверхні землі [26]. Основна

мета цієї моделі – передбачити інтенсивність сейсмічного впливу в залежності від енергетичних характеристик землетрусу та відстані до епіцентру.

Під час землетрусу вивільняється енергія, яка передається через сейсмічні хвилі. Сейсмічна енергія залежить від магнітуди землетрусу. Модель може використовувати відомий зв’язок між магнітудою і сейсмічною енергією:

$$E = 10^{1.5M+4.8}, \tag{2.4}$$

де E – вивільнена енергія, Дж;

M – магнітуда землетрусу.

Сенсори IoT, такі як акселерометри, вимірюють прискорення ґрунту. На основі виміряного прискорення можна оцінити енергію хвиль, що проходять через певну точку. Для цього використовується формула для кінетичної енергії:

$$E_{kin} = \frac{1}{2}mv^2, \tag{2.5}$$

де m – маса частинки ґрунту;

v – швидкість, яку можна отримати, інтегруючи прискорення, виміряне сенсорами:

$$v = \int a(t)dt. \tag{2.6}$$

Амплітуда сейсмічних хвиль зменшується з відстанню від епіцентру. Це загасання можна моделювати за допомогою експоненціального згасання енергії:

$$A(r) = A_0 \cdot e^{-\gamma r}, \tag{2.7}$$

де $A(r)$ – амплітуда хвиль на відстані r від епіцентру;

A_0 – амплітуда на епіцентрі;

γ – коефіцієнт загасання, який залежить від геологічних характеристик регіону.

Загальна енергія, вивільнена під час землетрусу, розподіляється через сейсмічні хвилі на велику площу. Сила хвилі на поверхні землі залежить не тільки від відстані до епіцентру, але й від типу ґрунту і топографії місцевості. Розподіл енергії можна описати як:

$$E(r) = \frac{E_0}{r^2} \cdot f(\theta, \phi), \quad (2.8)$$

де *E(r)* – енергія на відстані *r* від епіцентру;

E_0 – енергія вивільнення в епіцентрі;

$f(\theta, \phi)$ – функція, що враховує вплив місцевих умов (кутові координати θ, ϕ).

Інтенсивність землетрусу в різних точках поверхні може бути оцінена на основі енергії сейсмічних хвиль, що досягають цієї точки. Інтенсивність *I* пов'язана з енергією *E* за допомогою логарифмічної залежності:

$$I = k \cdot \log E(r), \quad (2.9)$$

де *k* – емпіричний коефіцієнт, що залежить від регіональних факторів;

E(r) – енергія на певній відстані від епіцентру.

До переваг моделі можна віднести:

– точне моделювання розподілу енергії. Дозволяє визначити інтенсивність землетрусу в різних точках на основі даних про енергію;

– гнучкість. Модель враховує вплив типу ґрунту, відстані від епіцентру та локальних геологічних умов;

– швидкість обробки даних. Використання сенсорів IoT дозволяє отримувати дані в реальному часі, що забезпечує оперативність прогнозів.

Дана модель надає можливість оцінювати вплив землетрусу в різних регіонах на основі енергетичних характеристик і даних сенсорів IoT.

Використання сенсорів IoT для побудови моделей інтенсивності землетрусів дозволяє значно підвищити точність прогнозування та скоротити час реакції на сейсмічні події. Завдяки великій кількості даних та використанню методів машинного навчання стає можливим передбачати не

лише загальну інтенсивність землетрусу, але й вплив на окремі території. Подальший розвиток таких моделей відкриває перспективи для побудови глобальних систем моніторингу та запобігання катастрофам.

2.2 Прогнозування магнітуди землетрусів з використанням методів машинного навчання

Машинне навчання є потужним інструментом для аналізу великих обсягів даних, що надходять з різних джерел, зокрема сенсорів IoT, та для прогнозування магнітуди землетрусів. Різні методи машинного навчання дозволяють будувати прогностичні моделі з різним ступенем точності та складності.

Метод стохастичного подвійного підйому координат (СППК) є ефективним методом оптимізації, який використовується для розв'язання задач регресії та класифікації у великих наборах даних. Він належить до методів стохастичної оптимізації, що дозволяють швидко знаходити рішення шляхом оновлення однієї або кількох змінних за ітерацію [27]. СППК широко використовується в машинному навчанні, зокрема для навчання моделей лінійної регресії, логістичної регресії та опорних векторів, особливо у випадках великих вибірок даних.

СППК працює на основі принципу подвійної оптимізації. У класичному методі градієнтного спуску одночасно оновлюються всі параметри моделі, що вимагає обчислення повного градієнта. Натомість у методі СППК для кожної ітерації вибирається лише одна змінна (координата), і здійснюється її оновлення. Це знижує обчислювальну складність кожної ітерації, що робить метод дуже ефективним для великих наборів даних.

Метод добре працює з великими наборами даних, оскільки дозволяє знижувати обчислювальні витрати без втрати точності результатів. Завдяки своїй здатності оновлювати параметри окремо, метод добре масштабується для задач високої розмірності. Його ефективно застосовувати для

прогнозування землетрусів, оскільки він здатен працювати з великими наборами сейсмічних даних, які містять численні параметри, такі як магнітуда попередніх землетрусів, глибина, відстань до епіцентру, кількість сейсмічних станцій та інші змінні, що характеризують сейсмічну активність.

Процес прогнозування магнітуди землетрусу за допомогою СППК полягає в навчанні моделі на основі історичних даних про землетруси. Для цього модель на кожній ітерації оновлює свої параметри на основі помилок у прогнозі магнітуди. Оскільки СППК зосереджений на стохастичних виборах змінних для оновлення, метод дозволяє ефективно навчати модель навіть на великих і різнорідних наборах даних.

Наприклад, сейсмологічні дані з IoT-сенсорів можуть включати великий обсяг інформації про кожен землетрус. СППК дозволяє швидко знаходити оптимальні значення параметрів моделі, знижуючи обчислювальні витрати і забезпечуючи точність прогнозів магнітуди землетрусу. Це робить СППК ефективним інструментом для задач, де швидкість і точність є критично важливими, особливо для реального часу.

Переваги методу СППК:

– висока ефективність для великих наборів даних. Метод добре масштабується, що дозволяє працювати з великими вибірками;

– швидке навчання. Використовує стохастичні оновлення координат, що знижує обчислювальні витрати;

– гнучкість. Добре підходить як для задач регресії, так і для класифікації.

Обмеженнями даного методу є:

– чутливість до параметрів навчання. Як і більшість методів оптимізації, СППК залежить від вибору коефіцієнта швидкості навчання;

– не завжди ефективний для малих наборів даних. На невеликих наборах даних стохастичні методи можуть не забезпечувати такої ж високої точності, як інші методи оптимізації;

– можливість потрапити у локальні мінімуми. Хоча метод часто є дуже ефективним, на складних поверхнях функцій втрат він може затримуватися у локальних мінімумів.

Отже, метод стохастичного подвійного підйому координат є потужним інструментом для прогнозування землетрусів на основі великих наборів даних. Завдяки своїй ефективності та здатності працювати з великими і складними наборами даних, СППК добре підходить для задач, де потрібна висока точність і швидке навчання моделі. У прогнозуванні землетрусів СППК може використовуватися для моделювання залежностей між сейсмічними параметрами та магнітудою події, що дозволяє точно передбачити ймовірну силу землетрусу та підготуватися до потенційних наслідків.

Метод опорних векторів (ОВ) шукає таку гіперплощину в багатовимірному просторі, яка максимально відділяє різні класи даних. У випадку лінійно роздільних даних, ОВ знаходить пряму (для двовимірних даних) або гіперплощину (для багатовимірних), що розділяє дані з максимальним відступом між класами [28]. Для цього ОВ використовує лише кілька точок з набору даних, які називаються опорними векторами. Ці точки найбільше впливають на положення гіперплощини. У нелінійних випадках метод використовує функцію ядра, яка дозволяє перенести дані в простір вищої вимірності, де їх можна лінійно розділити.

Для прогнозування землетрусів метод опорних векторів може бути використаний для оцінки магнітуди або ймовірності землетрусу на основі великої кількості сейсмічних параметрів. Вхідні дані можуть включати такі параметри, як: географічні координати (широта та довгота); глибина землетрусу; магнітуда попередніх поштовхів; кількість сейсмічних станцій, що фіксували подію; середньоквадратичне відхилення; розривність земної кори.

Параметри можуть бути як лінійно, так і нелінійно пов'язані з прогнозованою магнітудою або інтенсивністю землетрусу. ОВ дозволяє

знайти оптимальну залежність між цими змінними і результатом, особливо коли зв'язок є нелінійним. За допомогою функцій ядра, наприклад, радіально-базисної функції, метод може відображати дані в багатовимірний простір, де їх легше розділити або провести точну регресію. Це дає змогу будувати точні моделі для прогнозування землетрусів у різних регіонах, адаптуючи модель до локальних умов і специфіки сейсмічної активності.

ОВ також добре працює на відносно невеликих наборах даних, оскільки він оптимізує рішення на основі лише кількох ключових точок (опорних векторів), що робить його ефективним методом у випадках, коли дані про землетруси є обмеженими або недоступними в повному обсязі.

Переваги методу опорних векторів:

– висока точність. Здатний працювати з нелінійними залежностями, що робить його дуже точним у складних задачах прогнозування, таких як прогнозування магнітуди землетрусів;

– ефективність на малих вибірках. Завдяки використанню опорних векторів, МОВ може давати точні результати навіть на невеликих наборах даних, що є важливою перевагою для обмежених сейсмічних даних;

– гнучкість. Метод підтримує різні функції ядра (лінійні, нелінійні, радіально-базисні та інші), що дозволяє адаптувати його до різних типів залежностей у даних.

Недоліки методу опорних векторів складаються із:

– високі обчислювальні витрати. ОВ потребує значних обчислювальних ресурсів, особливо на великих наборах даних, що може збільшити час навчання;

– чутливість до вибору параметрів. Вибір функції ядра та її параметрів значно впливає на результат. Неправильна калібровка може призвести до погіршення точності моделі;

– проблеми з інтерпретацією. Хоча метод є потужним, результати МОВ не завжди легко інтерпретувати, особливо у випадках використання складних функцій ядра.

Отже, ОВ є потужним і гнучким інструментом для прогнозування магнітуди землетрусів. Його основні переваги полягають у здатності працювати з нелінійними даними та забезпечувати високу точність навіть на невеликих вибірках. Проте його застосування вимагає ретельного налаштування параметрів і значних обчислювальних ресурсів, що може бути обмеженням у деяких випадках. Незважаючи на це, МОВ залишається одним із провідних методів для задач, де складні залежності між даними є ключовими для точного прогнозування.

Метод випадкового лісу (ВЛ) є потужним алгоритмом машинного навчання, який використовується як для класифікаційних, так і для регресійних задач [29]. Випадковий ліс складається з великої кількості окремих дерев рішень, що працюють разом. Кожне дерево рішень створює прогноз, а метод випадкового лісу об'єднує ці прогнози для отримання остаточного результату.

ВЛ є ансамблевим методом, тобто він використовує кілька слабких моделей (дерев рішень) і об'єднує їх для побудови потужнішої моделі. Кожне дерево рішень у випадковому лісі створюється на основі різних підмножин даних та випадкових наборів ознак. Такий підхід сприяє зниженню варіативності та зменшенню ймовірності перенавчання моделі, що є основною проблемою для окремих дерев рішень. Модель випадкового лісу використовує принцип голосування для класифікації або усереднення для регресії, що дозволяє знижувати вплив шуму в даних.

Кожне дерево у ВЛ вивчає зв'язки між цими параметрами та магнітудою землетрусу або ймовірністю його виникнення. В результаті об'єднання результатів багатьох дерев, модель випадкового лісу може генерувати точніші

прогнози навіть при наявності складних нелінійних залежностей між параметрами.

Метод ВЛ добре працює з великими наборами даних, оскільки кожне дерево будується на частині вибірки, що дозволяє моделі залишатися стійкою до шуму в даних та уникати перенавчання. Крім того, ВЛ може обробляти відсутні або незначні дані, що часто зустрічається у сейсмічних спостереженнях, де не всі параметри можуть бути доступні.

Переваги методу ВЛ:

– стійкість до перенавчання. Використання кількох дерев рішень знижує ризик перенавчання моделі;

– гнучкість. Метод може працювати як для класифікації, так і для регресії, що робить його універсальним інструментом для різних типів задач;

– обробка великих наборів даних. ВЛ може ефективно працювати з великими обсягами даних, навіть якщо є корельовані ознаки або відсутні значення.

Недоліками методу є:

– високі обчислювальні витрати. Випадковий ліс може вимагати значних ресурсів для навчання, особливо при великій кількості дерев або великих наборах даних;

– складність інтерпретації. Хоча метод є дуже точним, інтерпретувати його результати може бути складно, оскільки він базується на численних окремих рішеннях дерев;

– нестійкість до незбалансованих даних. Якщо дані є дуже незбалансованими (наприклад, у задачах класифікації), випадковий ліс може демонструвати не найкращу точність без спеціальних модифікацій.

Отже, метод випадкового лісу є добре підходить для задач регресії, таких як прогнозування магнітуди землетрусу, або задач класифікації, пов'язаних із визначенням ймовірності землетрусу. Його гнучкість і стійкість

до перенавчання роблять випадковий ліс одним із найефективніших методів для аналізу складних даних у сейсмології.

Нижче наведено порівняльну табл. 2.2, що відображає основні характеристики методів стохастичного подвійного підйому координат, опорних векторів, та випадкового лісу.

Таблиця 2.2 – Порівняння методів машинного навчання

Характеристика	**СППК**	**ОВ**	**ВЛ**
Обчислювальна ефективність	Висока ефективність на великих наборах даних	Помірна обчислювальна складність	Високі витрати, особливо на великих даних
Масштабованість	Добре масштабується для великих наборів даних	Складно масштабується з великими даними	Добре масштабується, але потребує значних ресурсів
Швидкість навчання	Дуже швидке, особливо для великих даних	Помірна швидкість навчання	Повільніше через створення багатьох дерев
Простота реалізації	Відносно проста у реалізації	Потребує налаштування ядра	Більш складний в налаштуванні і побудові моделей
Стійкість до великого обсягу даних	Висока	Середня	Висока, але ресурсозатратна
Чутливість до параметрів	Менш чутливий до вибору параметрів	Висока чутливість до параметрів ядра	Низька чутливість до параметрів
Обробка нелінійних даних	Працює здебільшого з лінійними даними	Добре підходить для нелінійних даних	Добре працює з нелінійними залежностями
Точність прогнозів	Висока для лінійних залежностей	Висока для нелінійних задач	Дуже висока для складних і нелінійних даних

Характеристика	СППК	ОВ	ВЛ
Складність інтерпретації	Легка інтерпретація	Важка інтерпретація, особливо для складних ядер	Складна через велику кількість дерев

Метод СППК є оптимальним вибором для моніторингу інтенсивності землетрусів та прогнозування магнітуди завдяки своїй високій обчислювальній ефективності та швидкості навчання. При прогнозуванні землетрусів, особливо в реальному часі, необхідно працювати з великими обсягами даних, що включають сейсмічні показники, такі як глибина землетрусу, відстань до епіцентру, кількість сейсмічних станцій та інші параметри. СППК демонструє високу швидкість навчання, що дозволяє швидко адаптувати модель до нових даних без значних витрат на обчислення. Це особливо важливо для моніторингу землетрусів, коли необхідно оперативно реагувати на нові поштовхи.

Окрім того, СППК добре масштабується для великих наборів даних, що є важливою перевагою при моніторингу землетрусів, де дані надходять постійно і потребують швидкої обробки. Випадковий ліс та метод опорних векторів, хоч і показують хороші результати, значно поступаються СППК у швидкості обробки великих наборів даних, оскільки вони вимагають більше обчислювальних ресурсів для побудови моделі. СППК, на відміну від них, забезпечує високу продуктивність без необхідності тривалої калібровки або налаштувань.

Також, важливою перевагою СППК є менша чутливість до параметрів, що робить його простішим у налаштуванні та експлуатації для задач прогнозування магнітуди землетрусів. У випадку методу опорних векторів або випадкового лісу, користувачеві необхідно ретельно підбирати параметри ядра або кількість дерев для досягнення оптимальної точності. СППК, у свою

чергу, менш чутливий до цих аспектів, що спрощує процес підготовки моделі та дозволяє зосередитися на самих даних.

Зважаючи на ці фактори, СППК є чудовим вибором для задач, де швидкість, масштабованість та ефективність є критичними. Для прогнозування магнітуди землетрусів він дозволяє швидко обробляти великі набори даних і забезпечує високу точність, що робить його ідеальним рішенням для моніторингу інтенсивності сейсмічної активності в реальному часі.

2.3 Моделі оцінки наслідків землетрусів на основі геоданих та параметрів подій

Оцінка наслідків землетрусів базується на комбінації геопросторових даних і параметрів сейсмічної події, таких як магнітуда, глибина, відстань до епіцентру та тип ґрунту. Для точного прогнозування наслідків використовуються різні алгоритми, які дозволяють врахувати ці параметри та визначити можливий рівень руйнувань.

Сітковий метод є одним із основних підходів для оцінки наслідків землетрусів, що базується на геопросторових даних та параметрах сейсмічної події. Цей метод дозволяє поділити територію на рівномірні сіткові елементи (осередки) та оцінювати вплив землетрусу в кожному осередку на основі різних факторів, таких як магнітуда, глибина, тип ґрунту та відстань до епіцентру [30]. Використання геоданих дозволяє враховувати особливості кожного регіону, включаючи сейсмічну стійкість будівель та інфраструктури.

Для побудови моделі сіткового методу, спочатку розбивається територія на сіткові елементи розміру $\Delta x \times \Delta y$. Кожен осередок ii отримує свої локальні параметри: координати осередку, тип ґрунту, щільність забудови тощо. Оцінка наслідків землетрусу в кожному осередку ґрунтується на обчисленні рівня інтенсивності землетрусу та ймовірності пошкоджень інфраструктури. Модель включає такі основні параметри: магнітуда землетрусу (M); глибина

епіцентру (d); відстань до епіцентру r_i для кожного осередку і; тип ґрунту в осередку S_i, який характеризує здатність ґрунту передавати сейсмічні хвилі; щільність забудови в осередку D_i, що визначає ймовірність пошкоджень інфраструктури.

Рівень інтенсивності землетрусу в кожному осередку ii можна описати через наступну формулу:

$$I_i = a + b \cdot M - c \cdot \log_{10}(r_i + d) + S_i, \quad (2.10)$$

де I_i – інтенсивність землетрусу в осередку і;

a, b, c – емпіричні коефіцієнти, що налаштовуються на основі історичних даних;

M – магнітуда землетрусу;

r_i – відстань від епіцентру до осередку i, км;

d – глибина епіцентру, км;

S_i – коефіцієнт впливу ґрунту в осередку i.

Коефіцієнти a, b, c визначають вплив магнітуди та відстані на рівень інтенсивності. Врахування типу ґрунту S_i дозволяє моделювати різні геологічні умови, що можуть посилювати або послаблювати вплив землетрусу.

На основі інтенсивності землетрусу в кожному осередку I_i визначається рівень пошкоджень інфраструктури. Можливе пошкодження для кожного осередку можна оцінити за допомогою такої формули:

$$D_i = f(I_i) \cdot D_0, \quad (2.11)$$

де D_i – відсоток пошкодженої інфраструктури в осередку I_i;

$f(I_i)$ – функція ймовірності пошкодження залежно від інтенсивності I_i (наприклад, експоненціальна функція);

D_0 – базова вразливість забудови в осередку I_i, що враховує характеристики будівель та інфраструктури.

Функція $f(I_i)$ визначає зв'язок між рівнем інтенсивності та ймовірністю пошкоджень. Наприклад, для низьких рівнів інтенсивності I_i, значення $f(I_i)$ може бути близьким до нуля, а для високих рівнів – стрімко зростати, відображаючи більший ризик руйнувань.

Для отримання загальної оцінки наслідків землетрусу, рівень пошкоджень інфраструктури інтегрується по всій території:

$$D_{total} = \sum_{i=1}^{n} D_i \cdot A_i, \quad (2.12)$$

де D_{total} – загальний рівень пошкоджень у регіоні;

D_i – відсоток пошкоджень в осередку i;

A_i – площа осередку i;

n – кількість осередків.

Сітковий метод дозволяє врахувати як глобальні параметри землетрусу (магнітуда, глибина), так і локальні умови (ґрунти, забудова), що робить його дуже точним для оцінки наслідків землетрусів на великих територіях. Завдяки можливості розбивати територію на осередки, можна детально моделювати вплив землетрусу в різних зонах, що дозволяє точно прогнозувати пошкодження для окремих районів.

Також, варто розглянути регресійний аналіз, який дозволяє моделювати залежність між параметрами землетрусу та рівнем пошкоджень на основі геоданих [31]. Метою регресійного аналізу є визначення математичних залежностей між різними змінними, що впливають на наслідки землетрусу, такими як магнітуда, відстань до епіцентру, тип ґрунту та характеристики інфраструктури. Це дозволяє побудувати модель, що прогнозує ймовірність та рівень руйнувань у залежності від цих факторів.

Регресійний аналіз дозволяє оцінити наслідки землетрусів, використовуючи декілька незалежних змінних (факторів), які впливають на рівень руйнувань. Основні змінні, що зазвичай використовуються для побудови регресійної моделі: магнітуда землетрусу (M), відстань до епіцентру (r); глибина епіцентру (d); тип ґрунту (S); щільність забудови (D).

Для регресійного аналізу формується рівняння, що описує залежність між рівнем пошкоджень DD та вхідними параметрами землетрусу та геоданих. Основне рівняння регресійної моделі може бути записане у вигляді:

$$D = \beta_0 + \beta_1 \cdot M + \beta_2 \cdot r + \beta_3 \cdot d + \beta_4 \cdot S + \beta_5 \cdot \log(Density) + \varepsilon, \quad (2.13)$$

де D – рівень пошкоджень або ймовірність руйнувань;

M – магнітуда землетрусу;

r – відстань до епіцентру;

d – глибина епіцентру;

S – тип ґрунту в певній зоні;

$Density$ – щільність забудови;

$\beta_0,\beta_1,\ldots,\beta_5$ – коефіцієнти регресії, які визначають внесок кожної змінної у прогноз пошкоджень;

ε – похибка моделі, що враховує вплив неврахованих факторів.

Дана модель дозволяє кількісно оцінити вплив кожного параметра на рівень пошкоджень інфраструктури. Зазвичай коефіцієнти $\beta_0,\beta_1,\ldots,\beta_5$ отримуються за допомогою статистичних методів, таких як метод найменших квадратів (МНК), що мінімізує різницю між передбаченими та фактичними значеннями пошкоджень.

У випадках, коли необхідно оцінити ймовірність того, що інфраструктура в певній зоні зазнає руйнувань, модель регресії можна перетворити на логістичну регресію. Логістична регресія використовується для моделювання ймовірності події (в даному випадку – пошкоджень) і має наступний вигляд:

$$P(D = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot M + \beta_2 \cdot r + \beta_3 \cdot d + \beta_4 \cdot S + \beta_5 \cdot \log(Density))}}, \quad (2.14)$$

де $P(D = 1)$ – ймовірність того, що в зоні будуть руйнування.

Логістична регресія дає змогу оцінити ймовірність настання певних наслідків залежно від параметрів землетрусу та локальних умов, що може бути корисним для оцінки ризиків та планування заходів безпеки.

Отже, регресійний аналіз на основі геоданих є ефективним методом для оцінки наслідків землетрусів, що дозволяє будувати прогнозні моделі на основі кількісних залежностей між параметрами землетрусу та характеристиками території. Використовуючи магнітуду, відстань до епіцентру, глибину, тип ґрунту та щільність забудови, можна точно оцінювати рівень пошкоджень або ймовірність руйнувань для різних регіонів. Завдяки простоті інтерпретації та можливості швидкого оновлення моделі на основі нових даних, регресійний аналіз є потужним інструментом для прогнозування наслідків землетрусів і планування заходів для зниження ризиків.

2.4 Моделювання пошкоджень інфраструктури залежно від інтенсивності землетрусу

Оцінка пошкоджень інфраструктури під час землетрусів вимагає врахування ряду факторів, таких як інтенсивність землетрусу, геологічні особливості регіону та вразливість конструкцій. Для ефективного моделювання руйнувань застосовуються підходи, що поєднують сейсмологічні дані, зокрема шкалу інтенсивності Меркаллі, та інженерні параметри будівель і споруд [32]. Ці методи дозволяють точно оцінити рівень пошкоджень залежно від умов на конкретній території та сили поштовхів.

Шкала інтенсивності Меркаллі використовується для оцінки локального впливу сейсмічних хвиль, що допомагає визначати можливі наслідки для кожного регіону. Рівень інтенсивності землетрусу в кожному регіоні визначається як функція магнітуди землетрусу M, відстані до епіцентру r, глибини епіцентру d та геологічних умов ґрунту S. Формула для розрахунку інтенсивності землетрусу I на основі цих параметрів може бути представлена таким чином:

$$I = a + b \cdot M - c \cdot \log_{10}(r + d) + S, \quad (2.15)$$

де I – інтенсивність землетрусу за шкалою Меркаллі;

M – магнітуда землетрусу;

r – відстань до епіцентру;

d – глибина епіцентру;

S – коефіцієнт впливу геологічних умов ґрунту;

a, b, c – емпіричні коефіцієнти, які визначають вплив магнітуди та відстані на інтенсивність.

На основі обчисленої інтенсивності землетрусу за шкалою Меркаллі, можна моделювати пошкодження інфраструктури, враховуючи її вразливість до сейсмічних коливань. Для цього будується залежність між інтенсивністю землетрусу та ймовірністю пошкоджень будівель або інфраструктурних об'єктів.

Пошкодження моделюються за допомогою функції, яка визначає ймовірність пошкоджень $P(D)$ у залежності від інтенсивності I, вразливості інфраструктури V, та локальних геологічних умов T (тип ґрунту):

$$P(D) = f(I) \cdot V \cdot T, \tag{2.16}$$

де $P(D)$ – ймовірність пошкоджень;

$f(I)$ – функція залежності ймовірності руйнувань від інтенсивності землетрусу I;

V – вразливість будівлі або інфраструктури;

T – геологічні умови, що враховують тип ґрунту і його здатність передавати сейсмічні хвилі.

Функція $f(I)$ є нелінійною та може мати експоненціальну форму, оскільки ймовірність пошкоджень значно зростає з підвищенням інтенсивності:

$$f(I) = \frac{1}{1+e^{-\alpha(I-I_0)}}, \tag{2.17}$$

де α – параметр, що визначає крутизну зміни ймовірності;

I_0 – критична інтенсивність, при якій починають виникати суттєві пошкодження.

Щоб оцінити відсоток пошкоджень для кожної будівлі або інфраструктурного об'єкта, можна використати інтегральну формулу, що поєднує ймовірність пошкоджень та вразливість об'єкта:

$$D = P(D) \cdot S, \tag{2.18}$$

де D – загальний відсоток пошкоджень;

$P(D)$ – ймовірність пошкоджень для об'єкта;

S – площа або масштаб інфраструктурного об'єкта.

Коефіцієнт T, що враховує тип ґрунту, є критично важливим фактором для моделювання пошкоджень, оскільки різні ґрунти мають різну здатність передавати та ампліфікувати сейсмічні хвилі. Наприклад, м’які ґрунти можуть значно посилювати коливання, збільшуючи руйнування в регіоні. Для врахування цього впливу вводиться додатковий коефіцієнт у формулу, який визначає рівень ампліфікації сейсмічних хвиль залежно від типу ґрунту:

$$A(T) = 1 + k \cdot (T_m - T), \tag{2.19}$$

де $A(T)$ – коефіцієнт ампліфікації коливань;

k – емпіричний коефіцієнт, що визначає силу впливу ґрунту;

T_m – максимально можливий вплив м'яких ґрунтів;

T – тип ґрунту для певної зони.

Моделювання пошкоджень інфраструктури залежно від інтенсивності землетрусу є важливим інструментом для прогнозування наслідків сейсмічних подій. Шкала інтенсивності Меркаллі дозволяє визначити рівень впливу землетрусу на окремі регіони, а інженерні параметри інфраструктури дозволяють оцінити рівень пошкоджень залежно від вразливості будівель та геологічних умов. Такий підхід дозволяє будувати точні прогнози руйнувань та планувати заходи для зниження ризиків уразливості інфраструктури.

Висновки до розділу 2

У рамках даного розділу було розглянуто моделі інтенсивності землетрусів на основі даних сенсорів IoT, зокрема сейсмічну модель прогнозування інтенсивності на основі даних акселерометрів та модель розподілу енергії землетрусів, що базується на даних із сенсорів IoT. Ці моделі дозволяють детально аналізувати локальні інтенсивності сейсмічних подій та розподіляти енергію поштовхів, що надходять від сенсорів, забезпечуючи точне відображення сили землетрусів у конкретних регіонах.

Було проведено аналіз методів машинного навчання для прогнозування магнітуди землетрусів. Розглянуто методи стохастичного подвійного підйому координат, опорних векторів та випадкового лісу. На основі цього аналізу обґрунтовано вибір методу стохастичного подвійного підйому координат для моніторингу інтенсивності землетрусів, що забезпечує високу ефективність на великих наборах даних, швидкість навчання та гнучкість у роботі з різноманітними даними.

Описано моделі оцінки наслідків землетрусів на основі геоданих та параметрів подій. Сітковий метод дозволяє оцінювати руйнування інфраструктури для різних зон, використовуючи геопросторові дані, тоді як регресійний аналіз дозволяє точно моделювати вплив землетрусів на інфраструктуру, враховуючи такі фактори, як магнітуда, глибина епіцентру та тип ґрунту.

Підходи для моделювання пошкоджень інфраструктури залежно від інтенсивності землетрусу, зокрема з використанням шкали інтенсивності Меркаллі, забезпечують можливість оцінки ймовірності та масштабу пошкоджень на основі характеристик землетрусу та властивостей будівель. Ці методи будуть використані у системі для оцінки наслідків, що дозволить точно прогнозувати рівень пошкоджень інфраструктури на основі даних, що надходять із сенсорів IoT.

3 ПРОЄКТУВАННЯ СИСТЕМИ ДЛЯ МОНІТОРИНГУ ТА ПРОГНОЗУВАННЯ ЗЕМЛЕТРУСІВ

3.1 Вибір компонентів системи

Комплекс для моніторингу інтенсивності землетрусів та оцінки наслідків на базі сенсорів IoT включає апаратні й програмні компоненти, які забезпечують збір, обробку та аналіз даних у режимі реального часу. Основним завданням є вибір таких елементів, що дозволяють точно і швидко фіксувати сейсмічні коливання та передавати ці дані для подальшого аналізу та прогнозування.

Контролер є ключовим компонентом системи моніторингу інтенсивності землетрусів, оскільки він відповідає за збір, обробку та зберігання даних, отриманих від різних сенсорів. Його основні функції включають в себе:

– збір даних. Контролер отримує дані з підключених сенсорів, таких як сейсмометри, акселерометри, GPS-сенсори та інші. Ці дані можуть бути різної природи: аналогові сигнали (коливання ґрунту, прискорення) або цифрові дані (координати, температура, тиск);

– обробка даних. Після збору даних контролер виконує первинну обробку – фільтрацію, нормалізацію, перетворення аналогових сигналів у цифрові (за потреби). Це дозволяє підготувати дані для подальшого аналізу та прогнозування;

– зберігання даних. Контролер також має інтеграцію з базами даних, що дозволяє зберігати зібрану інформацію для подальшого аналізу та обробки в режимі реального часу або пізніше. Він може взаємодіяти з віддаленими серверами для збереження даних або використовувати локальні сховища.

У табл. 3.1 проведено порівняння контролерів, які можуть бути використані для побудови системи моніторингу інтенсивності землетрусів. Для порівняння було обрано три контролери: Raspberry Pi Pico, BeagleBone

Black та Arduino Mega 2560. Нижче наведено порівняльну таблицю за загальними характеристиками, де Raspberry Pi Pico переважає конкурентів.

Таблиця 3.1 – Порівняння контролерів

Параметр	**Raspberry Pi Pico**	**BeagleBone Black**	**Arduino Mega 2560**
Процесор	Dual-core ESP32 (240 МГц)	1 GHz ARM Cortex-A8	16 МГц ATmega2560
Оперативна пам'ять	520 кбайт SRAM, 4 Мбайт Flash	512 Мбайт DDR3	8 кбайт SRAM
Підтримка бездротових технологій	Wi-Fi, Bluetooth	Немає	Немає
Кількість пінів GPIO	36	92	54
Підтримка аналогових входів	18	7	16
Ціна	Низька	Середня	Низька
Енергоспоживання	Низьке	Високе	Низьке
Розширюваність	Висока	Висока	Середня

Контролер Raspberry Pi Pico був обраний для системи моніторингу інтенсивності землетрусів завдяки його високій продуктивності, низькому енергоспоживанню та підтримці бездротових технологій. Він дозволяє обробляти великі об'єми даних від сенсорів у реальному часі і передавати їх через Wi-Fi або Bluetooth без додаткового обладнання. Завдяки своїй компактності та гнучкості у розширенні, цей контролер легко інтегрується з різними сенсорами, забезпечуючи надійний збір даних. Крім того, його вартість є доступною, що робить цей варіант економічно вигідним для таких проєктів. Зовнішній вигляд контролера представлено на рис. 3.1, що підтверджує його компактність та зручність для використання в польових умовах.

Рисунок 3.1 – Зовнішній вигляд контролера «Raspberry Pi Pico» [33]

Для контролера були обрані наступні комплектуючі та модулі:

Geospace Y-28 (GS11-3D)

Сенсор Geospace Y-28 (GS11-3D) є високочутливим пристроєм для вимірювання сейсмічних коливань, що використовується для моніторингу тектонічної активності та виявлення землетрусів. Його застосовують у геофізичних дослідженнях та сейсмічному моніторингу, зокрема для фіксації малих коливань ґрунту в умовах підвищеної чутливості. У табл. 3.2 наведено порівняння цього сенсора з двома аналогами за основними характеристиками.

Таблиця 3.2 – Порівняльний аналіз сейсмічних сенсорів

Характеристика	**Geospace Y-28 (GS11-3D)**	**GS11-D Geophone**	**Lennartz LE-3Dlite**
Частотний діапазон, Гц	4,5–500	10–240	1–100
Чутливість	Висока	Середня	Висока
Вихідний сигнал	Аналоговий	Аналоговий	Аналоговий
Розміри	Компактні	Компактні	Середні

Geospace Y-28 (GS11-3D) має широкий частотний діапазон від 4,5 Гц до 500 Гц, що дозволяє йому ефективно фіксувати сейсмічні коливання різної інтенсивності. Як показано на рис. 3.2, сенсор має компактний корпус і легкий

у встановленні, що робить його зручним для використання у віддалених або важкодоступних регіонах.

Рисунок 3.2 – Зовнішній вигляд сенсора «Geospace Y-28 (GS11-3D)» [34]

Даний сенсор у системі моніторингу інтенсивності землетрусів Geospace Y-28 буде використовуватись для збору даних про сейсмічні коливання в реальному часі. Його висока чутливість дозволить фіксувати найменші рухи земної кори, передаючи ці дані на контролер для подальшої обробки та аналізу. Інтеграція цього сенсора зі створеною системою забезпечить надійний та точний моніторинг землетрусів на базі IoT технологій.

ADXL345 (3-Axis Accelerometer)

Сенсор ADXL345 є трьохосьовим акселерометром, який широко використовується для вимірювання лінійного прискорення у трьох напрямках: X, Y та Z. Він застосовується у різних галузях, включаючи моніторинг вібрацій, руху та нахилів у системах навігації, безпеки та контролю. У табл. 3.3 наведено порівняльний аналіз цього акселерометра з двома іншими аналогами.

Таблиця 3.3 – Порівняльний аналіз акселерометрів

Характеристика	**ADXL345**	**MPU6050**	**LSM303D**
Кількість осей	3	6 (акселерометр + гіроскоп)	3
Інтерфейси	I2C, SPI	I2C	I2C, SPI
Чутливість	Висока	Середня	Висока
Споживана потужність	Низька	Середня	Низька

Сенсор ADXL345, завдяки своїй підтримці інтерфейсів I2C і SPI, є сумісним як із Raspberry Pi, так і з ESP32, що забезпечує гнучкість у його використанні. Як показано на рис. 3.3, сенсор має компактні розміри та легкий у встановленні, що робить його ідеальним для інтеграції в мобільні та вбудовані системи.

Рисунок 3.3 – Зовнішній вигляд сенсора «ADXL345» [35]

У системі моніторингу інтенсивності землетрусів ADXL345 буде використовуватись для вимірювання лінійного прискорення ґрунту в трьох напрямах, що дозволить фіксувати найменші зміни руху під час сейсмічних подій. Ці дані будуть передаватися на контролер для подальшої обробки та аналізу, забезпечуючи точну інформацію про напрямок і силу землетрусу в режимі реального часу.

NEO-6M GPS

Модуль NEO-6M GPS забезпечує точне визначення географічних координат, використовуючи глобальну навігаційну супутникову систему GPS. Його основне застосування – це моніторинг місця розташування у реальному часі, що є важливим для навігаційних систем, геодезичних досліджень та різних IoT-проєктів. У табл. 3.4 представлено порівняльний аналіз модуля NEO-6M з двома аналогічними модулями.

Таблиця 3.4 – Порівняльний аналіз GPS-модулів

Характеристика	**NEO-6M GPS**	**UBlox NEO-M8N**	**Quectel L86**
Чутливість	Висока	Дуже висока	Висока
Інтерфейс	UART	I2C, UART	UART

Характеристика	**NEO-6M GPS**	**UBlox NEO-M8N**	**Quectel L86**
Час першого запуску (TTFF), с	27	24	25
Споживана потужність	Низька	Середня	Низька
Точність визначення, м	± 2,5	± 1,5	± 2,5

Модуль NEO-6M GPS легко підключається до контролерів, таких як Raspberry Pi та ESP32, через UART-інтерфейс, що забезпечує простоту інтеграції та швидке з'єднання для отримання даних про місцезнаходження. Як показано на рис. 3.4, цей модуль має компактний розмір та оснащений антеною для покращеного прийому сигналу GPS навіть у складних умовах.

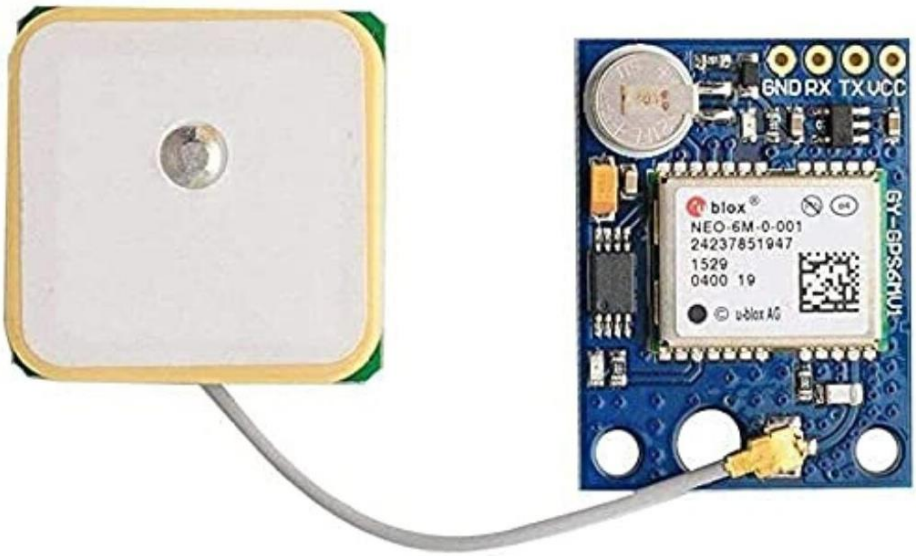


Рисунок 3.4 – Зовнішній вигляд сенсора «NEO-6M GPS» [36]

У системі моніторингу інтенсивності землетрусів NEO-6M GPS Module буде використовуватися для відстеження точних координат місця розташування сейсмічних подій. Це дозволить системі фіксувати географічні координати епіцентрів землетрусів у режимі реального часу, забезпечуючи високу точність визначення місця подій для подальшого аналізу та реагування на сейсмічні активності.

Geosense Seismic Hydrophone GD-25-11A

Сенсор Geosense Seismic Hydrophone GD-25-11A призначений для вимірювання підводних сейсмічних коливань і широко застосовується для

моніторингу сейсмічної активності на морських та океанічних днах. Завдяки високій чутливості, цей сенсор дозволяє виявляти навіть найменші зміни у підводних коливаннях, що робить його ідеальним для використання в наукових дослідженнях та системах раннього попередження про землетруси. У табл. 3.5 представлено порівняння цього сенсора з двома аналогічними моделями.

Таблиця 3.5 – Порівняльний аналіз гідрофонів

Характеристика	**GD-25-11A**	**Ocean Sonics icListen**	**AquaSound Hydrophone**
Чутливість	Висока	Дуже висока	Висока
Частотний діапазон, кГц	0,1–10	0,01–160	0,02–25
Вихідний сигнал	Аналоговий	Цифровий	Аналоговий
Енергоспоживання	Низьке	Середнє	Низьке
Інтерфейс	Аналоговий	Ethernet	Аналоговий

Сенсор GD-25-11A може бути інтегрований із Raspberry Pi через спеціальні плати перетворення сигналів, що дозволяють оцифровувати аналогові дані, отримані з підводних сейсмічних датчиків. Як показано на рис. 3.5, він має міцну конструкцію для роботи в екстремальних умовах підводного середовища, що забезпечує його надійність під час тривалих моніторингових операцій.

Рисунок 3.5 – Зовнішній вигляд сенсора «GD-25-11A» [37]

Для системи моніторингу інтенсивності землетрусів Geosense Seismic Hydrophone GD-25-11A буде використовуватись для збору даних про

сейсмічні коливання на морському дні. Дані, отримані від цього сенсора, дозволять фіксувати підводні землетруси, що є важливим для оцінки можливих цунамі та інших вторинних явищ, а також для аналізу сейсмічної активності в океанічних зонах.

BMP280 Barometric Pressure Sensor

Сенсор BMP280 є популярним барометричним датчиком, що використовується для вимірювання атмосферного тиску та температури навколишнього середовища. Завдяки своїй високій точності, він застосовується у різних системах моніторингу кліматичних умов, висотомірах та у метеорологічних дослідженнях. У табл. 3.6 проведено порівняльний аналіз цього сенсора з двома іншими аналогічними пристроями.

Таблиця 3.6 – Порівняльний аналіз барометричних датчиків

Характеристика	**BMP280**	**BME280**	**LPS25HB**
Вимірювані параметри	Тиск, температура	Тиск, температура, вологість	Тиск
Точність вимірювання тиску, гПа	± 1,0	± 1,0	± 1,5
Інтерфейс	I2C, SPI	I2C, SPI	I2C, SPI
Споживана потужність	Низька	Низька	Середня
Діапазон вимірювань тиску, гПа	300–1100	300–1100	260–1260

Сенсор BMP280 підключається через інтерфейс I2C або SPI, що дозволяє легко інтегрувати його як з Raspberry Pi, так і з ESP32 для вимірювання змін атмосферного тиску. Як показано на рис. 3.6, його компактний розмір і проста конструкція роблять його зручним для вбудованих систем, де важлива точність і стабільність вимірювань.

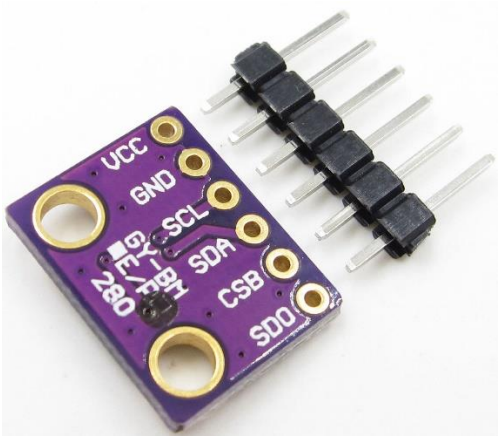


Рисунок 3.6 – Зовнішній вигляд сенсора «BMP280» [38]

У системі моніторингу інтенсивності землетрусів BMP280 Barometric Pressure Sensor використовуватиметься для фіксації змін тиску у реальному часі. Це допоможе виявити можливі зміни в атмосфері, які можуть бути пов'язані із сейсмічною активністю або інтенсивними коливаннями земної кори. Інтеграція цього сенсора дозволить системі аналізувати додаткові параметри, що можуть вказувати на тектонічні процеси.

INFRA20 від Chaparral Physics

Сенсор INFRA20 є високочутливим інфразвуковим датчиком, який спеціалізується на вимірюванні звукових хвиль низької частоти. Цей сенсор широко застосовується для моніторингу природних явищ, таких як землетруси, вулканічна активність, а також атмосферні процеси, що супроводжуються інфразвуковими коливаннями. У табл. 3.7 наведено порівняльний аналіз цього датчика з іншими інфразвуковими сенсорами.

Таблиця 3.7 – Порівняльний аналіз інфразвукових сенсорів

Характеристика	**INFRA20**	**ISF-10**	**GRAS 47AC**
Частотний діапазон, Гц	0,1–20	0,5–10	0,01–20
Чутливість	Висока	Середня	Дуже висока
Вихідний сигнал	Аналоговий	Аналоговий	Аналоговий
Розміри	Середні	Середні	Компактні
Споживана потужність	Низька	Низька	Низька

Сенсор INFRA20 підключається через аналоговий інтерфейс, що дозволяє його інтеграцію з Raspberry Pi або ESP32 за допомогою спеціальних перетворювачів сигналів для збору та аналізу даних у режимі реального часу. Як показано на рис. 3.7, цей сенсор має надійну конструкцію, що забезпечує його стабільну роботу навіть у складних погодних умовах.

Рисунок 3.7 – Зовнішній вигляд сенсора «INFRA20» [39]

Для системи моніторингу інтенсивності землетрусів INFRA20 буде використовуватися для виявлення інфразвукових хвиль, що супроводжують землетруси та інші природні явища. Ці хвилі не сприймаються людським вухом, але є важливими для раннього попередження про можливі катаклізми. Інтеграція цього датчика дозволить підвищити точність і швидкість реакції системи на сейсмічні події.

HMC5883L Triple Axis Magnetometer

Сенсор HMC5883L є трьохосьовим магнітометром, призначеним для вимірювання магнітного поля Землі. Його використовують у навігаційних системах, дослідженнях геомагнітних явищ та для моніторингу тектонічної активності. Цей сенсор дозволяє точно вимірювати зміни в магнітному полі у трьох координатах, що робить його корисним у проектах, які потребують високої точності магнітометричних даних. У табл. 3.8 наведено порівняння цього сенсора з двома аналогами.

Таблиця 3.8 – Порівняльний аналіз магнітометрів

Характеристика	**HMC5883L**	**LSM303D**	**QMC5883L**
Кількість осей	3	3	3
Інтерфейс	I2C	I2C, SPI	I2C
Чутливість	Висока	Середня	Висока
Споживана потужність	Низька	Низька	Дуже низька
Діапазон вимірювань магнітної індукції, Гс	± 8	± 16	± 8

Сенсор HMC5883L підключається через інтерфейс I2C, що дозволяє його просту інтеграцію як з Raspberry Pi, так і з ESP32 для реєстрації змін магнітного поля у реальному часі. Як видно на рис. 3.8, його компактний розмір і зручність у використанні роблять цей сенсор універсальним вибором для широкого спектру застосунків.

Рисунок 3.8 – Зовнішній вигляд сенсора «GY-273 HMC5883L DA5883» [40]

У системі моніторингу інтенсивності землетрусів HMC5883L буде використовуватися для вимірювання змін магнітного поля, які можуть супроводжувати тектонічні зрушення. Дані, отримані від цього сенсора, будуть допомагати в аналізі магнітних аномалій, що можуть передувати або супроводжувати землетруси, тим самим підвищуючи точність і ефективність системи виявлення сейсмічної активності.

Сенсор DS18B20 Temperature Sensor

Сенсор DS18B20 є цифровим датчиком температури, що широко використовується для точного вимірювання температури навколишнього середовища. Завдяки своїй простоті та надійності, він застосовується у різних системах моніторингу клімату, а також у проєктах IoT. Датчик підтримує підключення через 1-Wire інтерфейс, що дозволяє з'єднувати кілька датчиків в одну шину для збирання даних. У табл. 3.9 наведено порівняльний аналіз цього датчика з двома іншими аналогічними моделями.

Таблиця 3.9 – Порівняльний аналіз температурних сенсорів

Характеристика	**DS18B20**	**DHT22**	**TMP36**
Тип датчика	Цифровий	Цифровий	Аналоговий
Діапазон вимірювань	Від мінус 55°C до + 125°C	Від мінус 40°C до + 80°C	Від мінус 40°C до + 125°C
Точність, °C	± 0,5	± 0,5	± 2,0
Інтерфейс	1-Wire	1-Wire	Аналоговий
Споживана потужність	Низька	Середня	Низька

Сенсор DS18B20 легко інтегрується з такими платформами, як Raspberry Pi та ESP32, через 1-Wire інтерфейс, що забезпечує простоту налаштування та використання у багатьох проектах. Як показано на рис. 3.9, цей датчик має компактний корпус і може працювати в широкому діапазоні температур, що робить його ідеальним для тривалого моніторингу.

Рисунок 3.9 – Зовнішній вигляд сенсора «DS18B20» [41]

Використання у системі моніторингу інтенсивності землетрусів DS18B20 Temperature Sensor буде використовуватися для вимірювання температури навколишнього середовища та ґрунту. Це дозволить фіксувати зміни температури, які можуть бути корисними для виявлення аномальних процесів, пов'язаних із сейсмічною активністю, що додатково підвищить точність і надійність системи.

Portable Gravimeter Scintrex CG-5

Scintrex CG-5 – це компактний та високоточний портативний гравіметр, розроблений спеціально для вимірювання змін у гравітаційному полі Землі під час польових досліджень. Він відрізняється своєю зручністю та функціональністю, що робить його незамінним інструментом для геофізичних вимірювань у реальних умовах. Завдяки своїм невеликим розмірам і вазі, цей гравіметр ідеально підходить для використання в умовах обмеженого простору або складного рельєфу, де розміщення великогабаритного обладнання є неможливим. Це робить його ефективним інструментом для мобільних наукових експедицій, де важливими є легкість транспортування, швидкість розгортання та точність вимірювань. У табл. 3.10 наведено порівняльний аналіз цього гравіметра з двома іншими подібними пристроями.

Таблиця 3.10 – Порівняльний аналіз гравіметрів

Характеристика	**Scintrex CG-5**	**gPhoneX**	**ZLS Dynamic Gravimeter**
Точність	Висока	Дуже висока	Висока
Розміри, см	23 × 19 × 36	24 × 20 × 37	28 × 22 × 40
Вага, кг	5	10	8
Інтерфейс	USB	USB	RS-232
Енергоспоживання	Низьке	Середнє	Низьке

Scintrex CG-5 підключається через інтерфейс USB, що дозволяє його легко інтегрувати з Raspberry Pi для збору та обробки гравіметричних даних.

Як показано на рис. 3.10, цей прилад має зручний портативний корпус, що спрощує його транспортування і розгортання в польових умовах.

Рисунок 3.10 – Зовнішній вигляд магнітометра CG-5 [42]

У системі моніторингу інтенсивності землетрусів Portable Gravimeter Scintrex CG-5 буде використовуватися для вимірювання змін у гравітаційному полі Землі, які можуть супроводжувати тектонічні зрушення. Ці вимірювання допоможуть у виявленні ранніх ознак сейсмічної активності та прогнозуванні можливих землетрусів, тим самим підвищуючи ефективність системи.

3.2 Архітектура IoT–системи для збору та обробки даних

Побудова ефективної IoT-системи для моніторингу природних явищ вимагає чіткого визначення архітектурних елементів та взаємодії між ними. У такій системі ключовим завданням є забезпечення безперервного збору даних з сенсорів, їх подальшої обробки і передачі до центральної платформи для аналізу. Інфраструктура повинна забезпечувати надійність передачі даних у режимі реального часу, а також швидку реакцію на виявлені аномалії.

Система моніторингу інтенсивності землетрусів складається з кількох основних компонентів, що взаємодіють між собою для забезпечення збору, обробки, зберігання та аналізу даних (рис. 3.11).

Рисунок 3.11 – Загальна схема взаємодії компонентів системи

Система складається із наступних компонентів:

– сенсори та модулі. Сенсори розміщені в різних локаціях і відповідальні за збір даних про сейсмічні коливання та інші параметри середовища (атмосферний тиск, магнітне поле, температура, тощо). Сюди входять сейсмометр, акселерометр, GPS-сенсор та інші вимірювальні прилади, що безперервно відстежують параметри оточення;

– Raspberry Pi Pico. Основний контролер системи, що отримує всі дані від сенсорів і модулів. Він відповідає за обробку отриманих даних, виконує первинний аналіз та підготовку даних до відправки на сервер. Raspberry Pi Pico об'єднує всі сигнали та забезпечує централізовану обробку інформації;

– сервер. Після обробки дані передаються на сервер, де вони зберігаються та аналізуються. Сервер відіграє ключову роль у зберіганні великих обсягів інформації та забезпечує безперервний зв'язок із базою даних і користувачем;

– база даних. На сервері дані зберігаються в базі даних для подальшого аналізу та доступу. Ця база є основним сховищем, де зберігаються всі сейсмічні вимірювання та інші параметри, що надходять від сенсорів;

– аналіз даних (Модель Sdca). Для глибшого аналізу зібраних даних використовується модель машинного навчання Sdca. Вона відповідає за виявлення закономірностей у даних та визначення можливих землетрусів на основі аналізу історичних даних і нових вимірювань;

– виявлення землетрусу. Якщо модель Sdca виявляє сейсмічну активність, яка вказує на можливий землетрус, система негайно передає цю інформацію на сервер для оповіщення;

– сповіщення. Якщо землетрус виявлено, сервер генерує сповіщення, яке відправляється до користувача через застосунок. Сповіщення може включати інформацію про місцезнаходження, магнітуду, час події та інші важливі параметри.

– моніторинг стану. Користувач може в реальному часі відстежувати поточний стан системи та сейсмічну активність через застосунок. Це дозволяє своєчасно отримувати інформацію про будь-які зміни в оточенні та вживати відповідних заходів.

Міжкомпонентна взаємодія у системі моніторингу інтенсивності землетрусів ґрунтується на безперервному обміні даними між різними елементами, що забезпечують злагоджену роботу всієї інфраструктури. Сенсори, що розташовані на різних локаціях, відповідальні за фіксацію параметрів оточення, таких як сейсмічні коливання, прискорення ґрунту, координати та інші показники. Зібрані дані передаються до контролера Raspberry Pi Pico, який виконує функцію центрального елементу для початкової обробки інформації. Контролер фільтрує дані, нормалізує їх та готує до подальшої передачі на сервер, забезпечуючи безперервний потік інформації.

З'єднання між контролером Raspberry Pi Pico та сервером здійснюється через мережеві інтерфейси, такі як Wi-Fi або Ethernet, що забезпечує швидку передачу даних для їх збереження в базі даних. Сервер не лише зберігає інформацію, а й взаємодіє з моделлю машинного навчання Sdca для глибшого

аналізу даних. Модель Sdca, отримуючи доступ до бази даних, аналізує історичні й нові вимірювання для виявлення ознак сейсмічної активності, що може свідчити про ймовірний землетрус. У разі виявлення аномалій модель передає інформацію на сервер, який негайно ініціює оповіщення.

Після цього сервер генерує сповіщення про виявлену загрозу і відправляє його до користувачів через інтегрований застосунок. Користувачі отримують актуальну інформацію про потенційну небезпеку та можуть здійснювати моніторинг стану системи через застосунок. Така взаємодія між компонентами системи є критичною для забезпечення швидкого реагування, що дозволяє своєчасно виявляти загрози і попереджати користувачів про можливі сейсмічні події.

3.3 Розробка математичної моделі прогнозування на основі машинного навчання

Прогнозування магнітуди землетрусів є важливою науковою та практичною задачею, що має значний вплив на безпеку та добробут суспільства. У цьому розділі ми математично опишемо модель прогнозування магнітуди землетрусів, використовуючи алгоритм стохастичного подвійного підйому координат (СППК) для регресії. Детально розглянемо вхідні та вихідні дані моделі, метрики оцінки моделі, а також її практичне застосування.

1. Вхідні та вихідні дані моделі

Модель використовує набір ознак, які впливають на магнітуду землетрусу. Для кожного спостереження ii визначаємо вектор вхідних ознак $x_i \in R^d$, де d=7 – кількість ознак. Ознаки наступні:

- широта (x_{i1} = *Latitude*);
- довгота (x_{i2} = *Longitude*);
- глибина (x_{i3} = *Depth*);
- кількість станцій (x_{i4} = *Nst*);
- розривність мережі (x_{i5} = *Gap*);

– мінімальна відстань (x_{i6} = *Dmin*);

– середньоквадратичне відхилення (x_{i7} = *Rms*).

Отже, вектор ознак записується як:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \\ x_{i5} \\ x_{i6} \\ x_{i7} \end{bmatrix} \in R^7. \quad (3.1)$$

Вихідними даними є цільова змінна – магнітуда землетрусу y_i=*Mag*.

Мета моделі: навчитися функції $f(x_i)$, яка прогнозує y_i на основі x_i.

2. Математична модель алгоритму

Задача мінімізації емпіричної функції втрат з регуляризацією формулюється як:

$$min_{w,b} \left\{ \frac{\lambda}{2} \| w \|^2 + \frac{1}{n} \sum_{i=0}^{n} \ell(y_i, f(x_i)) \right\}, \quad (3.2)$$

де $w \in R_d$ – вектор ваг;

$b \in R$ – зміщення (базове значення);

$\lambda > 0$ – параметр регуляризації;

n – кількість спостережень;

$\ell(y_i, f(x_i))$ – функція втрат.

Для квадратичної функції втрат:

$$\ell\big(y_i, f(x_i)\big) = \frac{1}{2}(y_i - f(x_i))^2. \quad (3.3)$$

Алгоритм СППК працює в дуальному просторі. Дуальна задача формулюється як:

$$max_{\alpha} \left\{ -\frac{1}{2} \alpha^T (\lambda n I + K) \alpha + y^T \alpha \right\}, \quad (3.4)$$

де $\alpha \in R^n$ – дуальні змінні;

I – одинична матриця;

$K \in R^{n \times n}$ – матриця ядра, для лінійного випадку $K = XX^T$;

$X \in R^{n \times d}$ – матриця ознак;

$y \in R^n$ – вектор цільових значень.

На кожній ітерації алгоритму оновлюється одна дуальна змінна α_i:

$$\alpha_i^{new} = \alpha_i^{old} + \Delta\alpha_i, \tag{3.5}$$

де $\Delta\alpha_i$ обчислюється шляхом максимізації дуальної мети по α_i, залишаючи інші α_j фіксованими.

Для квадратичної втрати:

$$\Delta\alpha_i = \frac{1}{\lambda n + \|w\|^2}(y_i - w^T x_i - b - \lambda n \alpha_i), \tag{3.6}$$

Вектор ваг *w* оновлюється як:

$$w^{new} = w^{old} + \Delta\alpha_i x_i, \tag{3.7}$$

Зміщення *b* оновлюється, якщо необхідно, а сам алгоритм повторюється до тих пір, поки зміни в об'єктивній функції не стануть меншими за заданий поріг, або до досягнення максимальної кількості ітерацій.

3. Метрики оцінки моделі

Для оцінки якості прогнозування використовуються наступні метрики: R^2, *MAE, LossFunction*.

R^2 показує, яка частка дисперсії залежної змінної пояснюється моделлю та розраховується за формулою:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{3.8}$$

де $\hat{y}_i = f(x_i)$ – прогнозоване значення;

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ – середнє фактичних значень.

MAE вимірює середню величину помилок у прогнозах, незалежно від їх знаку за формулою:

$$MAE = \frac{1}{2}\sum_{i=1}^{n} |y_i - \hat{y}_i|. \tag{3.9}$$

У контексті моделі ця метрика *Loss Function* відображає середню квадратичну похибку. Формула середньоквадратичної похибки *MSE* має вигляд:

$$LossFunction = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \quad (3.10)$$

4. Застосування моделі

Після навчання моделі можемо прогнозувати магнітуду для нових спостережень. Для нового вектора ознак x_{new}:

$$\hat{y}_{new} = w^T x_{new} + b. \quad (3.11)$$

Розроблена модель буде впроваджена в системи реального часу для оперативного прогнозування та реагування.

Математична модель на основі алгоритму СППК дозволяє ефективно прогнозувати магнітуду землетрусів за допомогою регресійного аналізу з регуляризацією. Використання описаних метрик оцінки моделі забезпечує контроль якості прогнозів і можливість їх покращення. Практичне застосування моделі сприяє підвищенню готовності до землетрусів та зменшенню негативних наслідків від них.

3.4 Опис алгоритмів навчання та прогнозування магнітуди

Прогнозування магнітуди землетрусів є одним із найважливіших завдань в системах сейсмічного моніторингу. Для його вирішення використовуються сучасні методи аналізу даних, які дозволяють на основі попередніх вимірювань та математичних моделей прогнозувати параметри майбутніх землетрусів.

Алгоритм, представлений на рис. 3.12, описує процес тренування моделі для прогнозування магнітуди землетрусів, починаючи від вибору файлу з даними до збереження результатів у базі даних. Першим кроком є вибір файлу, який містить навчальні дані. Після цього система перевіряє коректність формату та структури даних у файлі. Якщо виявляється, що дані записано

некоректно, користувач отримує повідомлення про помилку, що дає змогу виправити недоліки.

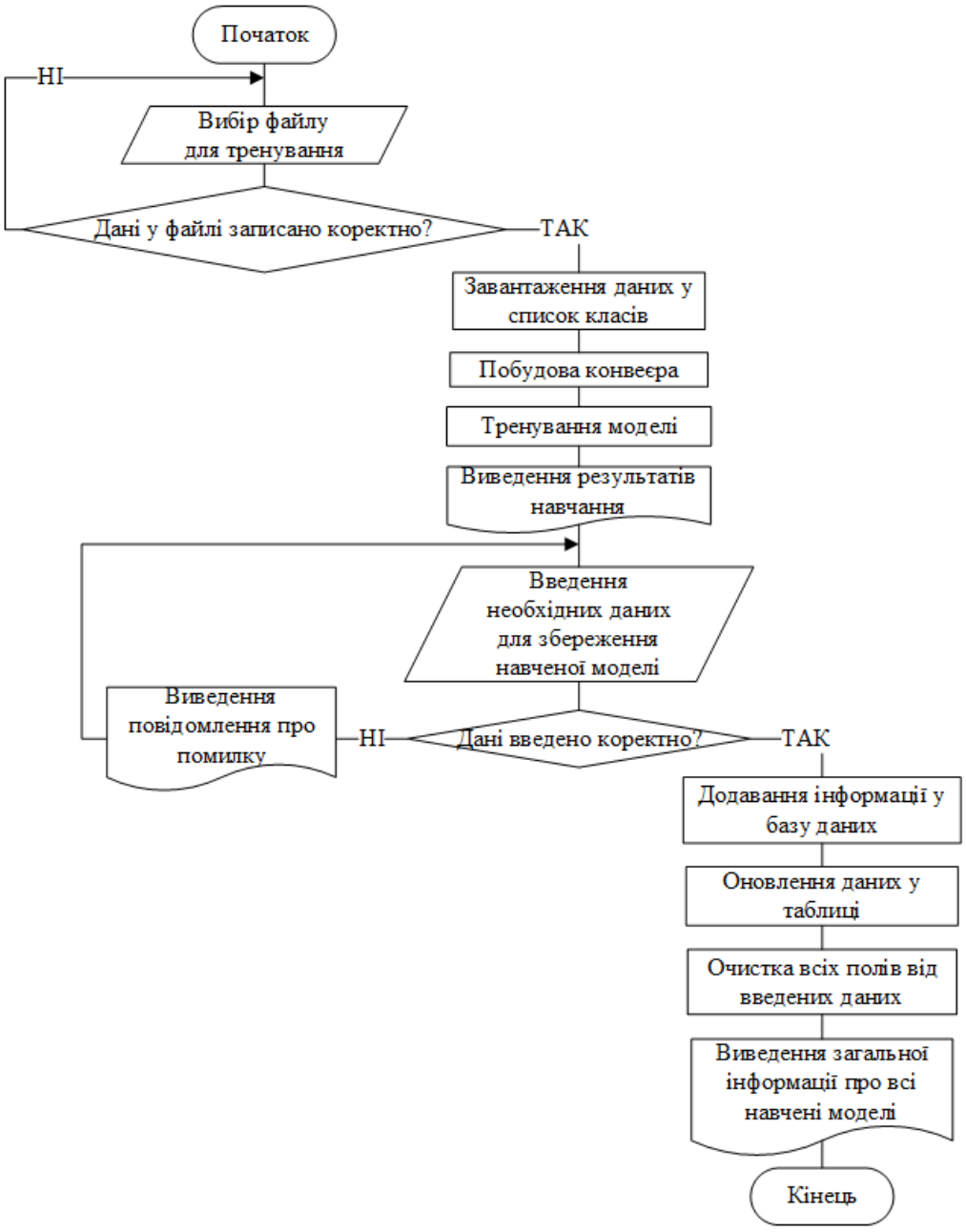


Рисунок 3.12 – Алгоритм тренування моделей

Якщо дані відповідають вимогам, наступним етапом є завантаження інформації у список класів, де кожен клас містить певний набір атрибутів, необхідних для тренування моделі. Після цього відбувається процес тренування моделі на основі завантажених даних. Результати тренування включають метрики моделі, які відображають ефективність її роботи. Після успішного навчання моделі система пропонує користувачеві ввести необхідні

дані для збереження отриманої моделі. Якщо всі введені дані правильні, система додає інформацію до бази даних, оновлює таблиці та виводить загальну інформацію про всі наявні навчені моделі. У разі виникнення помилок при введенні даних користувач отримує відповідне повідомлення і може повторно ввести коректні дані.

Заключними кроками алгоритму є очищення полів після введення даних та виведення загальної інформації про навчені моделі. Це дозволяє ефективно організувати роботу з кількома моделями, забезпечуючи зручне управління ними та можливість швидкого доступу до результатів кожної моделі.

Алгоритм прогнозування магнітуди землетрусу та оцінки наслідків, представлений на рис. 3.13, починається з ініціалізації необхідних змінних і об'єктів, які потрібні для роботи системи. Після цього здійснюється завантаження сценаріїв навчальних моделей і користувачеві надається можливість вибрати одну з доступних моделей із списку. Далі алгоритм перевіряє, чи існує вибрана модель у списку. Якщо модель не знайдена, система виводить інформацію про помилку, яка дозволяє користувачеві скоригувати свій вибір. У разі, якщо модель існує, здійснюється завантаження всієї інформації про модель для подальшого аналізу.

Наступним кроком є натискання кнопки «Моніторити», після чого система починає збирати дані з підключених IoT-сенсорів і передавати їх у модель для аналізу. Модель обробляє отримані дані, виконуючи обчислення та аналізуючи зміни, пов'язані з сейсмічною активністю.

Після обробки даних алгоритм виконує розрахунок прогнозованої магнітуди землетрусу та оцінює можливі наслідки на основі вихідних даних. Результати прогнозу відображаються користувачеві для подальшого використання. Якщо під час виконання процесу не було натиснуто кнопку «Зупинити», процес продовжується. Інакше система зупиняє процес і завершує роботу.

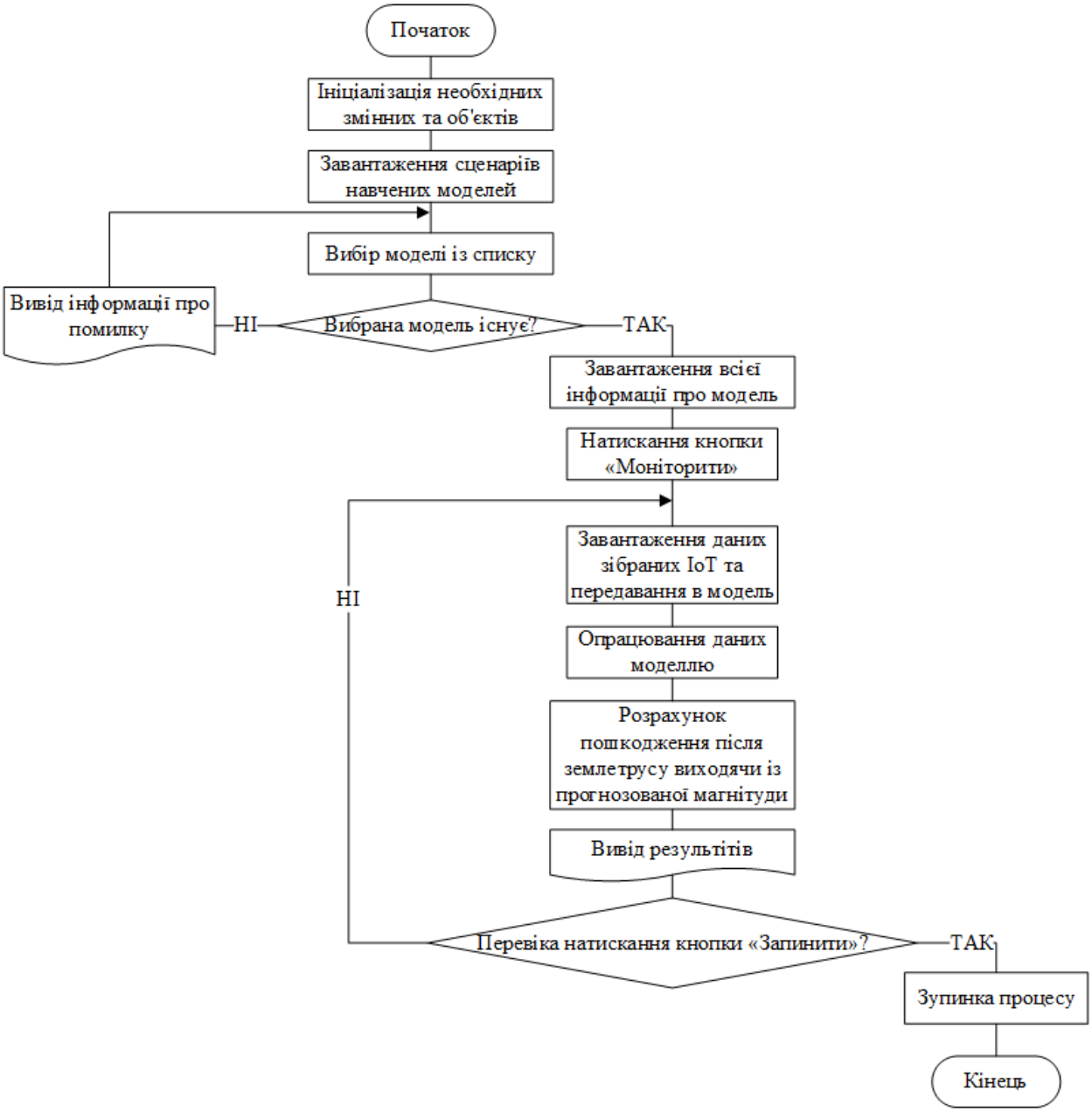


Рисунок 3.13 – Алгоритм прогнозування магнітуди землетрусу та оцінки наслідків

Цей алгоритм дозволяє автоматизувати збір і аналіз даних з метою точного прогнозування магнітуди землетрусів і оцінки їхніх наслідків, що сприяє своєчасному реагуванню на потенційні загрози.

3.5 Вибір технологій для реалізації системи

При виборі мови програмування для розробки системи моніторингу інтенсивності землетрусів та оцінки наслідків важливо враховувати кілька

факторів, таких як продуктивність, зручність роботи з бібліотеками для обробки даних та інтеграцію з IoT-пристроями. Кожна мова програмування має свої переваги та недоліки в контексті вирішення таких завдань, і вибір залежить від вимог проєкту та особливостей системи. Оскільки система вимагає високої точності обчислень, роботи з великим обсягом даних і взаємодії з апаратними засобами, доцільно розглянути найбільш поширені мови для розробки таких систем.

Python є однією з найпопулярніших мов програмування для наукових і дослідницьких проєктів, завдяки великій кількості бібліотек для обробки даних, машинного навчання та інтеграції з апаратними платформами. Вона відрізняється легкістю вивчення та використання, що дозволяє швидко розробляти прототипи та проводити аналіз даних. Однак Python може бути менш ефективною у плані швидкості обчислень порівняно з компільованими мовами.

Java забезпечує високу продуктивність і є чудовим вибором для розробки складних багатопоточних систем, таких як системи моніторингу. Ця мова пропонує потужні можливості для роботи з великими даними і добре підходить для проєктів, де важлива масштабованість і кросплатформеність. Java також підтримує інтеграцію з IoT-пристроями, але може вимагати більше часу на розробку через складнішу синтаксичну структуру.

C# є потужною мовою програмування, яка використовується переважно для розробки програм під платформу .NET. Вона відома своєю хорошою підтримкою інструментів для створення графічних інтерфейсів користувача, а також широким спектром бібліотек для роботи з IoT і аналізом даних. C# забезпечує високу продуктивність, але основною її перевагою є тісна інтеграція з екосистемою Microsoft, що може бути корисним для розробки корпоративних застосунків або систем, орієнтованих на використання Windows-середовища.

У табл. 3.11 представлено порівняльний аналіз мов програмування Python, Java та C# за основними характеристиками.

Таблиця 3.11 – Порівняльний аналіз мов програмування

Характеристика	**Python**	**Java**	**C#**
Продуктивність	Нижча через інтерпретацію	Висока	Висока
Підтримка графічних інтерфейсів	Базова через сторонні бібліотеки	Базова через сторонні бібліотеки	Розширені можливості через Windows Forms та WPF
Інтеграція з Microsoft продуктами	Мінімальна	Мінімальна	Повна
Інструменти для корпоративних систем	Обмежені	Розширені	Розширені через .NET Framework
Підтримка мульти-платформеності	Широка (через інтерпретатор)	Широка	Широка (через .NET Core)
Розвиток та підтримка	Відкрита спільнота	Підтримується корпорацією Oracle	Підтримується корпорацією Microsoft

Вибір мови програмування C# для розробки системи моніторингу інтенсивності землетрусів та оцінки наслідків обґрунтований її високою продуктивністю та оптимізацією для роботи у середовищі .NET, що забезпечує ефективну обробку даних у реальному часі. Крім того, C# має розширені можливості для створення графічних інтерфейсів користувача завдяки інструментам Windows Forms та WPF, що важливо для розробки інтуїтивно зрозумілих застосунків. Глибока інтеграція з продуктами Microsoft і корпоративними рішеннями надає додаткові переваги, особливо для систем, що працюють в екосистемі Windows. Підтримка мультиплатформності через .NET Core також забезпечує гнучкість і масштабованість системи, роблячи C# ідеальним вибором для проєкту.

3.6 Опис реалізації серверної частини системи

Програмна частина передбачає обробку зібраних даних за допомогою алгоритмів машинного навчання. Використання бібліотеки .ML NET на платформі C# дозволить здійснити тренування моделі на основі даних, отриманих із сенсорів у реальному часі, для точного прогнозування можливості землетрусів. Дані, що передаються від сенсорів, включають час події, магнітуду, глибину епіцентру, а також координати місця події. Ці показники є вирішальними для створення прогнозів щодо ймовірності повторних коливань чи землетрусів у майбутньому.

Для відкриття файлів із тренувальними наборами, реалізовано код, який створює діалогове вікно для відкриття файлу за допомогою OpenFileDialog (рис. 3.14). Спочатку налаштовуються властивості вікна: фільтр файлів обмежується текстовими файлами з розширенням *.csv та всіма іншими файлами, а також встановлюється індекс фільтра. Після цього діалогове вікно відкривається, і якщо користувач натискає "OK", шлях до вибраного файлу зберігається у змінній _Path, а також відображається у текстовому полі FileNameTBox.

```
OpenFileDialog openFileDialog = new OpenFileDialog();
// Налаштування властивостей діалогового вікна
openFileDialog.Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*";
openFileDialog.FilterIndex = 2;
openFileDialog.RestoreDirectory = true;
// Відображення діалогового вікна та обробка результату
if (openFileDialog.ShowDialog() == DialogResult.OK) {
    _Path = openFileDialog.FileName;
    FileNameTBox.Text = openFileDialog.FileName;
```

Рисунок 3.14 – Код для відкриття файлів із тренувальними наборами

Наступним кроком було створено код, який ініціалізує екземпляр *MLContext*, який є основним класом для роботи з машинним навчанням у .NET (рис. 3.15). Використання seed (в даному випадку 12345) гарантує, що всі випадкові операції, такі як розподіл даних або ініціалізація ваг, будуть відтворюваними при кожному запуску програми. Це забезпечує стабільність

результатів і дозволяє відтворювати експерименти з тими самими вихідними даними.

```
mlContext = new MLContext(seed: 12345);
```

Рисунок 3.15 – Ініціалізація екземпляру *MLContex*

На рис. 3.16 представлено код, що відповідає за завантаження даних із файлу. Спочатку у текстове поле RaportTBox записується повідомлення «Завантаження даних», яке виводиться на екран. Виклик *Application.DoEvents* дозволяє оновити інтерфейс користувача, щоб він відразу відобразив зміни, навіть якщо процес завантаження даних триватиме деякий час. Після цього дані завантажуються з файлу за шляхом, що зберігається у змінній *_Path*, використовуючи функціонал *LoadFromTextFile*. У файлі передбачено заголовок *hasHeader: true*, а дані розділяються комами, що задається параметром *separatorChar*.

```
RaportTBox.Text = "Завантаження даних\r\n";
Application.DoEvents(); // Оновлюємо інтерфейс
dataView = mlContext.Data.LoadFromTextFile<EarthquakeData>(_Path,
  hasHeader: true, separatorChar: ',');
```

Рисунок 3.16 – Завантаження даних із файлу для тренування моделі

У коді, представленому на рис. 3.17, відбувається поділ завантажених даних на тренувальну і тестову вибірки для моделювання. Спочатку дані розділяються за допомогою функції *TrainTestSplit*, яка визначає, що 20 % даних будуть використані для тестування моделі, а решта 80 % – для її навчання. Важливим аспектом є фіксований параметр seed, що забезпечує відтворюваність результатів при кожному запуску програми.

```
var splitData = mlContext.Data.TrainTestSplit(dataView,
  testFraction: 0.2, seed: 12345);
var trainData = splitData.TrainSet;
var testData = splitData.TestSet;
```

Рисунок 3.17 – Розподіл даних на тренувальну і тестову вибірки

Після поділу дані зберігаються у двох окремих наборах: тренувальний набір зберігається у змінній trainData, а тестовий – у змінній testData. Це

дозволяє використовувати один набір для навчання моделі, а інший для її валідації та оцінки точності.

На рис. 3.18 представлено код, у якому створюється pipeline для побудови моделі машинного навчання. Спочатку виконується об'єднання декількох вхідних параметрів (Latitude, Longitude, Depth, Nst, Gap, Dmin, Rms) у один вектор ознак під назвою Features. Це необхідно для того, щоб усі ключові характеристики землетрусу використовувалися в процесі навчання моделі.

```
var pipeline = mlContext.Transforms.Concatenate("Features",
        nameof(EarthquakeData.Latitude),
        nameof(EarthquakeData.Longitude),
        nameof(EarthquakeData.Depth),
        nameof(EarthquakeData.Nst),
        nameof(EarthquakeData.Gap),
        nameof(EarthquakeData.Dmin),
        nameof(EarthquakeData.Rms))
    .Append(mlContext.Regression.Trainers.Sdca(labelColumnName: "Mag",
    featureColumnName: "Features", maximumNumberOfIterations: 100));
```

Рисунок 3.18 – Створення конвеєра для моделі машинного навчання

Після цього до pipeline додається метод тренування, що використовує SdcaRegressionTrainer для регресії, де параметром передається стовпчик з метками (Mag – магнітуда землетрусу) і стовпчик з ознаками (Features). Модель тренується з максимальним числом ітерацій, встановленим на 100, що забезпечує точне налаштування параметрів для покращення результатів прогнозування.

У коді на рис. 3.19 відбувається навчання моделі на основі раніше створеного pipeline та тренувальних даних. Виклик методу Fit застосовує процес тренування до набору даних trainData, де модель аналізує вхідні ознаки і підбирає оптимальні параметри для виконання прогнозів. Цей етап є ключовим, оскільки після завершення навчання модель готова до подальшої роботи з новими даними.

```
model = pipeline.Fit(trainData);
```

Рисунок 3.19 – Навчання моделі

Результатом роботи методу *Fit* є вже натренована модель, яка здатна робити передбачення на основі вхідних параметрів. Вона зберігається у змінній model, і це дозволяє в подальшому використовувати її для тестування та оцінки точності на тестових даних.

Наступний код виконує оцінку якості натренованої моделі на основі тестових даних (рис. 3.20). Спочатку за допомогою методу *Transform* модель застосовується до тестових даних, і отримані прогнози зберігаються у змінній predictions. Після цього за допомогою методу *Evaluate* проводиться оцінка точності моделі. Для цього порівнюються прогнозовані значення з фактичними (метка *Mag*), а отримані результати представлені у вигляді метрик, таких як коефіцієнт детермінації (R^2), середня абсолютна помилка (*MAE*) та функція втрат (*LossFunction*).

```
var predictions = model.Transform(testData);
var metrics = mlContext.Regression.Evaluate(predictions, labelColumnName: "Mag");
// Виведення метрик
RaportTBox.Text += "Метрики моделі:\r\n";
RaportTBox.Text += String.Format("R²: {0:P2}\r\n" +
                                 "MAE: {1:P2}\r\n" +
                                 "LossFunction: {2:P2}\r\n",
                                 metrics.RSquared, metrics.MeanAbsoluteError, metrics.LossFunction);
```

Рисунок 3.20 – Оцінка моделі та виведення метрик

Виведення метрик здійснюється у текстове поле RaportTBox. Спочатку відображається повідомлення «Метрики моделі», після чого, використовуючи String.Format, виводяться значення кожної з метрик: R^2, MAE та LossFunction. Це дозволяє оцінити, наскільки добре модель прогнозує магнітуду землетрусів на основі тестових даних, і зробити висновки щодо її точності та надійності.

Для збереження даних натренованої моделі після перевірки правильності введених даних, реалізовано метод, код якого представлено на рис. 3.21.

```
private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Зберігання моделі
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj =
      System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
    _ModelsProvider.InsertModels(ModelsNamesTBox.Text,
      Convert.ToInt32(CategoriesCBox.SelectedValue), pathName);
    mlContext.Model.Save(model, dataView.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено модель " +
      ModelsNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

Рисунок 3.21 – Збереження моделі

Коли користувач натискає кнопку «Зберегти», система спочатку перевіряє введені дані за допомогою методу *IsDataEnteringCorrect*. Якщо дані введені правильно, створюється шлях для збереження моделі у вигляді файлу *.zip*, де для імені файлу використовується згенероване ім'я за допомогою методу *GenerateFileName*. Далі шлях комбінується з директорією проєкту, визначеною за допомогою *System.Reflection*.

Після цього модель зберігається у базі даних за допомогою методу *InsertModels*, де передаються назва моделі, обрана категорія та шлях до файлу. Сама модель зберігається на диску за допомогою методу *Save*, що також включає її схему. Після збереження всі дані очищаються через *ClearAllData*, і в логах системи фіксується запис про успішне тренування моделі. Наприкінці користувачу виводиться повідомлення «MessageBox.Show», яке інформує про успішне збереження даних.

Наступний код реалізує завантаження натренованої моделі та підготовку її до використання для прогнозування (рис. 3.22). Спочатку формується повний шлях до файлу моделі на основі переданого параметра *FilePath* та директорії запуску програми, яку визначає *Application.StartupPath*. Після цього оголошується змінна для схеми даних *DataViewSchema*, яка буде використовуватись для опису структури моделі.

```
private void LoadData(string FilePath) {
  string localProj = Application.StartupPath + FilePath;
  // Визначення DataViewSchema для конвеєра підготовки даних і навченої моделі
  DataViewSchema modelSchema;
  // Завантаження моделі
  ITransformer model = context.Model.Load(localProj, out modelSchema);
  // Створення двигуна прогнозування
  predictionEngine = context.Model.CreatePredictionEngine<EarthquakeData,
    EarthquakePrediction>(model);
}
```

Рисунок 3.22 – Завантаження натренованої моделі та підготовка до використання

Наступним кроком є завантаження збереженої моделі з файлу, що відбувається за допомогою методу *Load*, який також повертає схему даних у змінну *modelSchema*. Завантажена модель потім використовується для створення *PredictionEngine*, що дозволяє робити прогнози на основі нових даних. Цей механізм прогнозування працюватиме з даними типу *EarthquakeData* для створення прогнозів типу *EarthquakePrediction*.

У коді на рис. 3.23 реалізовано періодичне виконання задач за допомогою таймера, коли є дані про землетруси у списку *_EarthquakeDataList*. Якщо в списку присутні дані, обирається випадковий запис, який потім відображається у текстовому полі *MonitoringTBox*. Відображення здійснюється через *Invoke*, щоб уникнути конфліктів з потоками, оскільки оновлення інтерфейсу користувача виконується з іншого потоку.

```
private void MoniroringTimer_Tick(object sender, EventArgs e) {
  if (_EarthquakeDataList.Any()) {
    // Select a random record
    Random rand = new Random();
    int index = rand.Next(_EarthquakeDataList.Count);
    EarthquakeData data = _EarthquakeDataList[index];
    // Display selected record in TextBox
    MonitoringTBox.Invoke((MethodInvoker)(() => {
      MonitoringTBox.Text += $"Час: {data.Time}\r\n";
      MonitoringTBox.Text += $"Географічна широта: {data.Latitude}\r\n";
      MonitoringTBox.Text += $"Географічна довгота: {data.Longitude}\r\n";
      MonitoringTBox.Text += $"Глибина: {data.Depth}\r\n";
      MonitoringTBox.Text += $"Кількість станцій: {data.Nst}\r\n";
      MonitoringTBox.Text += $"Сейсмічна розривність: {data.Gap}\r\n";
      MonitoringTBox.Text += $"Відстань до землетрусу: {data.Dmin}\r\n";
      MonitoringTBox.Text += $"Середньоквадратичне відхилення: {data.Rms}\r\n"

      var prediction = predictionEngine.Predict(data);
      var answer = new StringBuilder();
      answer.AppendLine("\r\n--- Прогнозування ---");
      answer.AppendLine($"Прогнозована магнітуда: {prediction.PredictedMag}");
      MonitoringTBox.Text += answer;
     // Simulate infrastructure damage
     SimulateDamage(prediction.PredictedMag, data.Dmin, data.Depth);
    }));
    UpdateInfrastructureBarChart();
  }
}
```

Рисунок 3.23 – Моніторинг даних та передбачення магнітуди

У текстове поле виводиться основна інформація про вибраний землетрус, включаючи час, географічні координати (широту та довготу), глибину, кількість станцій, сейсмічну розривність, відстань до епіцентру та середньоквадратичне відхилення. Після цього за допомогою моделі прогнозування виконується передбачення магнітуди землетрусу на основі обраного запису. Результати прогнозування також додаються у текстове поле для перегляду. Далі функція *SimulateDamage* моделює ймовірні пошкодження інфраструктури, використовуючи дані про прогнозовану магнітуду, глибину та відстань до землетрусу. Після цього оновлюється гістограма інфраструктурних пошкоджень для візуалізації результатів прогнозування та аналізу ситуації.

Для симуляції пошкоджень інфраструктури після землетрусу, використовуючи значення магнітуди, відстані до епіцентру та глибини землетрусу реалізовано код, який представлено на рис. 3.24. Спочатку передані значення магнітуди і відстані перетворюються у тип *double* для

виконання подальших розрахунків. На основі цих даних обчислюється MMI (Modified Mercalli Intensity) за допомогою функції *CalculateMMI*, яка визначає інтенсивність сейсмічної події в залежності від відстані до епіцентру і сили поштовху.

```
private void SimulateDamage(float magnitude, float dmin, float depth) {
  // Convert magnitude and distance to double for calculation
  double mag = magnitude;
  double distance = dmin;
  // Calculate MMI
  _MMI = CalculateMMI(mag, distance);
  MonitoringTBox.AppendText($"Відстань від епіцентру (Dmin): {dmin:F2} км, MMI: {_MMI:F1}\r\n");
  MonitoringTBox.AppendText($"Глибина землетрусу: {depth:F2} км\r\n");
  foreach (var infrastructure in _InfrastructureList) {
    // Calculate damage percentage based on MMI and vulnerability
    float damagePercentage = infrastructure.CalculateDamagePercentage(_MMI);

    float remainingPercentage = 100f - damagePercentage;
    MonitoringTBox.AppendText($"{infrastructure.Name} " +
      $"пошкодження після землетрусу магнітудою {magnitude:F2}: " +
      $"{remainingPercentage:F0}% залишилося.\r\n");
  }
}
```

Рисунок 3.24 – Код методу *SimulateDamage*

Отримані результати відображаються у текстовому полі *MonitoringTBox*, де виводиться інформація про відстань до епіцентру та глибину землетрусу. Далі для кожного об'єкта інфраструктури з списку *_InfrastructureList* обчислюється відсоток пошкоджень на основі інтенсивності MMI та вразливості об'єкта. Після цього виводиться інформація про залишкову частину інфраструктури, яка залишилася неушкодженою після землетрусу, що дає змогу оцінити можливі наслідки для кожного з елементів інфраструктури.

Метод *AdjustMagnitudeByDistance* відповідає за коригування значення магнітуди землетрусу залежно від відстані до епіцентру (рис. 3.25). Спочатку перевіряється, чи є магнітуда меншою або рівною 3.0 – у такому разі вона не коригується і повертається початкове значення. Це запобігає надто великим коригуванням для слабких землетрусів.

```
private float AdjustMagnitudeByDistance(float magnitude, float dmin) {
  // Ensure magnitude doesn't become negative
  if (magnitude <= 3.0f) {
    return magnitude;
  }
  // Adjust the magnitude based on distance using an exponential decay function
  float attenuationFactor = (float)Math.Exp(-dmin / 50.0f); // Adjust denominator as needed
  // The adjusted magnitude should not be less than zero
  float adjustedMagnitude = magnitude * attenuationFactor;
  return Math.Max(0, adjustedMagnitude);
}
```

Рисунок 3.25 – Код методу *AdjustMagnitudeByDistance*

Якщо магнітуда перевищує 3.0, то для її коригування використовується експоненційна функція загасання, яка залежить від відстані до епіцентру d_{min}. Чим більша відстань, тим більше значення магнітуди зменшується. Коефіцієнт затухання розраховується за допомогою функції *Math.Exp*, де знаменник у формулі можна налаштовувати для коригування швидкості затухання. Остаточне скориговане значення магнітуди повертається через *Math.Max*, щоб гарантувати, що результат не буде від'ємним.

Код на рис. 3.26 реалізує розрахунок індексу інтенсивності сейсмічних подій MMI (Modified Mercalli Intensity) на основі магнітуди землетрусу та відстані до епіцентру. Щоб уникнути проблем з логарифмічними обчисленнями, для коректного виконання перевіряється, що відстань не менша ніж 1 км, таким чином запобігаючи можливості отримання log(0). Якщо відстань менша, то вона автоматично збільшується до 1 км.

```
private double CalculateMMI(double magnitude, double distance) {
  // Ensure distance is at least 1 km to avoid log(0)
  distance = Math.Max(distance, 1.0);
  // Coefficients for shallow earthquakes
  double a = 1.5;
  double b = 1.0;
  double c = 3.0;
  // Calculate MMI
  double mmi = a + b * magnitude - c * Math.Log10(distance);
  // Ensure MMI is within 1 to 12
  return Math.Max(1.0, Math.Min(12.0, mmi));
}
```

Рисунок 3.26 – Код методу *CalculateMMI*

Коефіцієнти *a*, *b* та *c* використовуються для моделювання інтенсивності сейсмічних поштовхів для неглибоких землетрусів. Вони впливають на розрахунок інтенсивності з урахуванням магнітуди та відстані. Сам

розрахунок MMI базується на лінійній комбінації магнітуди та логарифмічної функції відстані.

Після обчислення MMI перевіряється, щоб результат не виходив за межі шкали, де мінімальне значення інтенсивності – 1, а максимальне – 12. Таким чином, кінцевий результат коригується, щоб відповідати цим межам і повертатися в межах допустимої шкали інтенсивності.

Метод на рис. 3.27 відповідає за ініціалізацію графіка стовпчикової діаграми для відображення залишкового відсотка пошкодженої інфраструктури після землетрусу. Спочатку робиться видимим сам елемент графіка *GraphicsCC*. Створюється нова серія для стовпчикової діаграми за допомогою *LiveCharts.Wpf.ColumnSeries*, де задається заголовок «Залишковий % після землетрусу», і підготовлюється контейнер для значень.

```
private void InitializeInfrastructureBarChart() {
    GraphicsCC.Visible = true;
    // Створюємо серію для стовпчикової діаграми
    var columnSeries = new LiveCharts.Wpf.ColumnSeries {
        Title = "Залишковий % після землетрусу",
        Values = new LiveCharts.ChartValues<double>()
    };
    // Додаємо серію на графік
    GraphicsCC.Series.Add(columnSeries);
    // Додаємо підписи для осі X
    GraphicsCC.AxisX.Add(new LiveCharts.Wpf.Axis {
        Labels = _InfrastructureList.Select(i => i.Name).ToArray(),
        Separator = new LiveCharts.Wpf.Separator { Step = 1 }  // Для чітких поділів між категоріями
    });
    // Вісь Y
    GraphicsCC.AxisY.Clear();
}
```

Рисунок 3.27 – Код методу *InitializeInfrastructureBarChart*

Ця серія додається до графіка за допомогою методу *GraphicsCC.Series.Add*, що дозволяє відображати дані на графіку. Після цього налаштовується вісь X, на якій відображаються підписи категорій, що відповідають назвам об'єктів інфраструктури зі списку *_InfrastructureList*. Використовується Separator для чіткішого відокремлення категорій на графіку. Наприкінці очищається вісь Y для налаштування динамічного відображення даних по вертикалі, що дозволяє правильно масштабувати графік залежно від значень, які будуть додаватися в подальшому.

Код на рис. 3.28 відповідає за оновлення даних на стовпчиковому графіку, який відображає залишковий відсоток пошкодженої інфраструктури після землетрусу. Спочатку перевіряється, чи графік має хоча б одну серію. Якщо так, отримується перша серія у вигляді стовпчикової діаграми *ColumnSeries*, і її значення очищуються для подальшого оновлення.

```
private void UpdateInfrastructureBarChart() {
  // Оновлюємо значення серії
  if (GraphicsCC.Series.Count > 0) {
    var columnSeries = (LiveCharts.Wpf.ColumnSeries)GraphicsCC.Series[0];
    columnSeries.Values.Clear();

    // Додаємо оновлені дані для кожної інфраструктури
    foreach (var infrastructure in _InfrastructureList) {
      float damagePercentage = infrastructure.CalculateDamagePercentage(_MMI);
      float remainingPercentage = 100f - damagePercentage;
      columnSeries.Values.Add((double)remainingPercentage);
    }
  }
}
```

Рисунок 3.28 – Код методу *UpdateInfrastructureBarChart*

Для кожного об'єкта зі списку *_InfrastructureList* виконується розрахунок відсотка пошкодження за допомогою методу *CalculateDamagePercentage* на основі інтенсивності землетрусу MMI. Розраховується залишковий відсоток неушкодженої інфраструктури, і ці значення додаються до серії графіка. Це дозволяє в режимі реального часу відображати оновлені дані про стан кожної інфраструктури після землетрусу.

3.7 Опис реалізації на стороні контролера

У процесі розробки комплексу для моніторингу інтенсивності землетрусів та оцінки наслідків, ключовою складовою є контролер, який відповідає за збір, обробку та передачу даних від численних сенсорів. На стороні контролера реалізовано інтеграцію різних сенсорних модулів, що дозволяє моніторити не лише інтенсивність сейсмічної активності, а й супутні параметри навколишнього середовища, такі як атмосферний тиск, магнітне поле та температура.

Контролер виконує функції збору даних із сенсорів IoT у реальному часі, обробляє їх та передає на сервер для подальшого аналізу і оцінки наслідків землетрусу. Ця система побудована на модульній архітектурі, що дозволяє легко адаптувати або розширювати функціональність через додавання нових сенсорів або зміну способів передачі даних.

Контролер взаємодіє з кожним із сенсорів через стандартизовані інтерфейси, такі як I2C, SPI та OneWire, що забезпечує стабільність і точність отриманих даних. Одночасно з цим, модуль Wi-Fi використовується для передачі інформації на віддалений сервер, де зібрані дані можуть бути проаналізовані в режимі реального часу.

Для забезпечення коректної синхронізації даних використовується NTP клієнт, який гарантує точне відображення часу подій, що є важливим при аналізі сейсмічної активності. На рис. 3.29 представлено фрагмент коду, що демонструє інтеграцію необхідних бібліотек для роботи з сенсорами та компонентами контролера, які відповідають за збір даних і їх подальшу передачу.

```
#include <WiFi.h>            // Library for Wi-Fi
#include <Wire.h>            // Library for I2C
#include <SPI.h>             // Library for SPI
#include <Adafruit_Sensor.h> // General sensor library

// Libraries for specific sensors
#include <Adafruit_ADXL345_U.h>     // Accelerometer ADXL345
#include <Adafruit_BMP280.h>        // Barometric sensor BMP280
#include <Adafruit_HMC5883_U.h>     // Magnetometer HMC5883L
#include <OneWire.h>                // For DS18B20
#include <DallasTemperature.h>      // For DS18B20
#include <TinyGPS++.h>              // For GPS NEO-6M
#include <NTPClient.h>              // For time synchronization
#include <WiFiUdp.h>                // For NTP client
```

Рисунок 3.29 – Інтеграція необхідних бібліотек

У фрагменті коду на рис. 3.30 визначаються пін для підключення сенсора температури DS18B20 та налаштування Wi-Fi мережі, зокрема, SSID і пароль. Встановлено, що DS18B20 використовує пін 4 для підключення до контролера через інтерфейс OneWire. Для підключення до Wi-Fi контролер

використовує заданий SSID та пароль, що дозволяє йому передавати дані на віддалений сервер, адреса і порт якого також визначені в коді.

```
// Pin definitions
#define PIN_DS18B20 4       // Pin for DS18B20
#define ONE_WIRE_BUS PIN_DS18B20
// Wi-Fi settings
const char* WIFI_SSID = "NHsdfSDF123123";
const char* WIFI_PASSWORD = "QWERREBvbnvnYY124!!";
// Server settings
const char* SERVER_ADDRESS = "YOUR_SERVER_ADDRESS";
const int   SERVER_PORT = 80;  // HTTP port
// Creating sensor objects
Adafruit_ADXL345_Unified accelerometer = Adafruit_ADXL345_Unified(12345);
Adafruit_BMP280 barometer; // I2C
Adafruit_HMC5883_Unified magnetometer = Adafruit_HMC5883_Unified(12345);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature temperatureSensors(&oneWire);
TinyGPSPlus gps;
```

Рисунок 3.30 – Визначення змінних та їхніх значень

Крім того, створюються об'єкти для сенсорів, які будуть використовуватися в системі: акселерометр ADXL345, барометр BMP280 і магнітометр HMC5883L. Кожен з цих сенсорів підключається до контролера через відповідні інтерфейси, такі як I2C для барометра. Температурний сенсор DS18B20, підключений через інтерфейс OneWire, обробляється через об'єкт DallasTemperature. Для забезпечення точного визначення геолокації використовується GPS-модуль, представлений через об'єкт TinyGPSPlus, що дозволяє контролеру фіксувати координати й висоту у реальному часі.

Код на рис. 3.31 реалізує функцію *setup*, яка відповідає за початкове налаштування і підготовку контролера до роботи. На першому етапі здійснюється ініціалізація серійного зв'язку на швидкості 115200 біт/с для дебагу, після чого виводиться повідомлення «Starting...», що сигналізує про початок роботи системи. Далі виконується ініціалізація WiFi-з'єднання. Контролер намагається підключитися до мережі Wi-Fi, використовуючи задані SSID і пароль. Поки не буде встановлено з'єднання, система періодично перевіряє статус з'єднання, і на серійний порт виводяться точки, що відображає процес підключення. Після успішного підключення виводиться повідомлення «Connected to Wi-Fi!».

```
void setup() {
  // Serial communication initialization for debugging
  Serial.begin(115200);
  Serial.println("Starting...");

  // Wi-Fi initialization
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to Wi-Fi!");
  // NTP client initialization
  timeClient.begin();
  timeClient.update();
  // Sensor initialization
  if (!accelerometer.begin()) {
    Serial.println("Error initializing accelerometer!");
    while (1);
  }
  if (!barometer.begin()) {
    Serial.println("Error initializing BMP280 barometer!");
    while (1);
  }
  if (!magnetometer.begin()) {
    Serial.println("Error initializing magnetometer!");
    while (1);
  }
  temperatureSensors.begin(); // DS18B20 initialization
  // GPS initialization
  Serial1.begin(9600, SERIAL_8N1, RXD2, TXD2);
  // Gravimeter initialization
  Serial2.begin(9600, SERIAL_8N1, RXD3, TXD3);
  Serial.println("Initialization complete!");
}
```

Рисунок 3.31 – Код функції *setup*

Наступним етапом є ініціалізація NTP-клієнта для синхронізації часу. Клієнт починає роботу і виконує перше оновлення часу для забезпечення точних часових позначок.

Після цього виконується ініціалізація сенсорів. Якщо контролеру не вдається ініціалізувати один із сенсорів (акселерометр ADXL345, барометр BMP280 або магнітометр HMC5883L), виводиться повідомлення про помилку, і програма зупиняється в нескінченному циклі. Успішно ініціалізовані сенсори дозволяють контролеру зчитувати відповідні дані з навколишнього середовища. Також відбувається ініціалізація сенсора DS18B20 для

вимірювання температури і налаштовуються два серійні інтерфейси для GPS-модуля та гравіметра, обидва з яких працюють на швидкості 9600 біт/с і використовують відповідні пін-коди для передачі і прийому даних. Після завершення всіх етапів ініціалізації виводиться повідомлення «Initialization complete!», що сигналізує про готовність системи до подальшої роботи.

На рис. 3.32 представлено функцію *loop*, яка безперервно виконується під час роботи контролера. На початку циклу викликається функція *checkWiFi*, яка перевіряє стан з'єднання з Wi-Fi і, якщо з'єднання втрачено, намагається відновити його.

```
void loop() {
  checkWiFi();
  // Time update
  timeClient.update();
  String currentTime = timeClient.getFormattedTime();
  Serial.print("Current time: ");
  Serial.println(currentTime);

  // Reading sensor data
  readTemperatureDS18B20();
  readAccelerometer();
  readMagnetometer();
  readBarometer();
  readGPS();
  readAnalogSensors();
  readGravimeter();
  // Preparing data for sending
  String data = prepareData();
  data += "&time=" + currentTime;
  // Sending data to the server
  sendDataToServer(data);
  // Delay before the next cycle
  delay(5000);
}
```

Рисунок 3.32 – Код функції *loop*

Після перевірки з'єднання оновлюється час за допомогою NTP-клієнта, і поточний час виводиться на серійний порт у форматі зручному для читання. Це важливо для синхронізації подій у системі. Наступним кроком є зчитування даних із підключених сенсорів. Функції *readTemperatureDS18B20*, *readAccelerometer*, *readMagnetometer*, *readBarometer*, *readGPS*, *readAnalogSensors* і *readGravimeter* збирають відповідні показники, такі як

температура, прискорення, магнітне поле, тиск, координати та інші фізичні параметри. Після зчитування даних усі зібрані значення формуються у вигляді рядка за допомогою функції *prepareData()*. До цього рядка також додається поточний час. Такий рядок підготованих даних є готовим для відправки на сервер.

Заключним етапом є відправлення зібраних даних на сервер за допомогою функції *sendDataToServer(data)*. Після цього виконується затримка у 5000 мілісекунд перед повторенням циклу, що дозволяє контролеру виконувати нові вимірювання через певний інтервал часу.

Функція *checkWiFi* відповідає за перевірку стану з'єднання з WiFi-мережею (рис. 3.33). Вона спочатку перевіряє, чи контролер підключений до мережі. Якщо з'єднання втрачено, на серійний порт виводиться повідомлення «Lost Wi-Fi connection. Attempting to reconnect...», яке інформує про втрату з'єднання та спробу його відновлення. Після цього Wi-Fi з'єднання примусово розривається за допомогою функції *WiFi.disconnect*, і контролер намагається знову підключитися до мережі, використовуючи раніше збережені SSID і пароль. У циклі, поки з'єднання не буде відновлене, відбувається затримка на 500 мілісекунд і виводиться крапка на серійний порт для індикації процесу підключення. Коли підключення успішно відновлене, на серійний порт виводиться повідомлення "Reconnected to Wi-Fi!".

```
void checkWiFi() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Lost Wi-Fi connection. Attempting to reconnect...");
    WiFi.disconnect();
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("\nReconnected to Wi-Fi!");
  }
}
```

Рисунок 3.33 – Код функції *checkWiFi*

На рис. 3.34 представлено функцію *readAccelerometer* відповідає за зчитування даних з акселерометра. Спочатку вона викликає метод *getEvent* для отримання поточних даних з акселерометра і зберігає їх у структурі *accelerometerEvent*. Після цього функція виводить на серійний порт значення прискорення по трьох осях: X, Y і Z. Спочатку виводиться значення по осі X, потім по осі Y, і нарешті, по осі Z. Кожне значення роздруковується у відповідному форматі, щоб користувач міг бачити поточні показники прискорення в тривимірному просторі.

```
void readAccelerometer() {
  accelerometer.getEvent(&accelerometerEvent);
  Serial.print("Accelerometer X: ");
  Serial.print(accelerometerEvent.acceleration.x);
  Serial.print(" Y: ");
  Serial.print(accelerometerEvent.acceleration.y);
  Serial.print(" Z: ");
  Serial.println(accelerometerEvent.acceleration.z);
}
```

Рисунок 3.34 – Код функції *readAccelerometer*

Функція *readMagnetometer* призначена для зчитування даних з магнітометра (рис. 3.35). Спочатку викликається метод *getEvent*, який отримує поточні значення магнітного поля і зберігає їх у структурі *magnetometerEvent*. Далі функція виводить на серійний порт значення магнітного поля по трьох осях: X, Y та Z. Спочатку виводиться значення по осі X, потім по осі Y і, нарешті, по осі Z. Кожне значення виводиться окремо, що дозволяє спостерігати за змінами магнітного поля в кожному з напрямків простору.

```
void readMagnetometer() {
  magnetometer.getEvent(&magnetometerEvent);
  Serial.print("Magnetometer X: ");
  Serial.print(magnetometerEvent.magnetic.x);
  Serial.print(" Y: ");
  Serial.print(magnetometerEvent.magnetic.y);
  Serial.print(" Z: ");
  Serial.println(magnetometerEvent.magnetic.z);
}
```

Рисунок 3.35 – Код функції *readMagnetometer*

Функція *readBarometer* виконує зчитування показників з барометричного сенсора (рис. 3.36). На початку отримується поточне значення температури за допомогою методу *readTemperature*, яке зберігається у змінній *barometerTemperature*. Після цього зчитується тиск за допомогою методу *readPressure* і зберігається у змінній pressure. Далі функція виводить на серійний порт значення температури, що вимірює барометр BMP280, з відповідним текстовим коментарем. Після цього виводиться значення тиску. Такий підхід дозволяє контролеру відображати точні показники як температури, так і атмосферного тиску в реальному часі.

```
void readBarometer() {
  barometerTemperature = barometer.readTemperature();
  pressure = barometer.readPressure();
  Serial.print("BMP280 Temperature: ");
  Serial.println(barometerTemperature);
  Serial.print("Pressure: ");
  Serial.println(pressure);
}
```

Рисунок 3.36 – Код функції *readBarometer*

Функція *readGPS* відповідає за обробку даних від GPS модуля (рис. 3.37).

```
void readGPS() {
  while (Serial1.available() > 0) {
    gps.encode(Serial1.read());
  }
  if (gps.location.isUpdated()) {
    latitude = gps.location.lat();
    longitude = gps.location.lng();
    altitude = gps.altitude.meters();
    Serial.print("GPS Latitude: ");
    Serial.print(latitude, 6);
    Serial.print(" Longitude: ");
    Serial.print(longitude, 6);
    Serial.print(" Altitude: ");
    Serial.println(altitude);
  } else {
    Serial.println("GPS data not updated");
  }
}
```

Рисунок 3.37 – Код функції *readGPS*

На початку вона перевіряє наявність даних у буфері серійного порту, де працює GPS модуль, і передає ці дані в об'єкт *gps* для декодування за допомогою методу *encode*. Після цього перевіряється, чи оновилися координати, отримані від GPS. Якщо нові дані доступні, функція зчитує широту, довготу та висоту й зберігає ці значення у відповідні змінні. Потім ці координати виводяться на серійний порт з точністю до шести знаків після коми, що дозволяє отримати точні геолокаційні дані. Якщо ж нові дані не були отримані, виводиться повідомлення про те, що дані GPS не оновлені.

Описані основні функції відповідають за ініціалізацію, обробку та передачу даних, що є ключовими для роботи системи моніторингу інтенсивності землетрусів. Таким чином, контролер виконує роль централізованого елемента для збору та відправки інформації, яка використовується для подальшого аналізу та оцінки наслідків.

Висновки до розділу 3

У рамках даного розділу було виконано проєктування системи для моніторингу та прогнозування землетрусів, що включає обґрунтований вибір компонентів та розробку архітектури системи на основі сучасних IoT-технологій. Було проаналізовано і підібрано контролер Raspberry Pi Pico, а також сенсори та модулі, такі як Geospace Y-28, ADXL345, NEO-6M GPS, Geosense Seismic Hydrophone GD-25-11A, BMP280, INFRA20, HMC5883L, DS18B20 та Scintrex CG-5, що дозволило забезпечити повний набір інструментів для збору та обробки даних, необхідних для моніторингу сейсмічної активності. Розроблена архітектура IoT-системи визначила основні етапи взаємодії між сенсорами, контролером, сервером, базою даних та алгоритмами аналізу, що забезпечують ефективний збір і обробку даних у реальному часі.

Побудовано математичну модель прогнозування землетрусів на основі алгоритму стохастичного подвійного підйому координат, яка дозволяє з

високою точністю прогнозувати магнітуду землетрусів. Описані алгоритми навчання моделі та процесу прогнозування землетрусів забезпечують зрозумілий шлях від збирання історичних даних до оцінки можливих наслідків сейсмічної активності. Було проаналізовано різні технології програмування, і на основі цього вибрано мову C# для реалізації системи. Також було описано реалізований код для серверної частини та для управління контролем.

Таким чином, результати, отримані у цьому розділі, дозволяють перейти до наступного етапу – тестування і валідації розробленої системи.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ І РЕЗУЛЬТАТИ

4.1 Налаштування середовища для тестування системи

Home Assistant є універсальною операційною системою з відкритим кодом, призначеною для інтеграції та автоматизації пристроїв Інтернету речей різних виробників. Вона дозволяє локально керувати пристроями, не потребуючи постійного доступу до хмарних сервісів чи Інтернету. Home Assistant підтримує широкий спектр протоколів, таких як Wi-Fi, Zigbee, Z-Wave, Bluetooth та інші, що робить її гнучким рішенням для різних задач автоматизації.

Ця платформа надає широкі можливості для використання у різних сценаріях. Зокрема, вона може бути застосована для моніторингу сейсмічної активності за допомогою спеціалізованих сенсорів, таких як акселерометри, магнітометри, барометри та GPS-модулі. Ці сенсори дозволяють фіксувати інтенсивність землетрусів та проводити оцінку наслідків на основі отриманих даних.

Home Assistant також забезпечує можливість автоматизації збирання та передачі даних, що особливо важливо для таких систем моніторингу. З її допомогою можна не лише керувати збором даних, але й автоматично передавати їх на сервер для подальшого аналізу. Це програмне забезпечення підтримує сценарії, де система працює автономно, в режимі реального часу, реагуючи на будь-які зміни параметрів середовища.

Операційна система Home Assistant OS є найбільш оптимізованим способом встановлення даної платформи, що дозволяє швидко розгорнути систему з мінімальними налаштуваннями. Вона забезпечує стабільну роботу й підтримує інтеграцію з великою кількістю сенсорів, необхідних для задач моніторингу сейсмічних подій, що робить її надійним інструментом для подібних проєктів.

Home Assistant Container – це варіант встановлення системи, який працює в середовищі контейнерів і не передбачає використання вбудованого магазину застосунків. Однією з ключових переваг цього методу є спрощене управління контейнерами, наприклад, за допомогою інструментів на кшталт Portainer. Згідно зі статистикою встановлень, цей спосіб є другим за популярністю серед користувачів. Інший спосіб – Home Assistant Core, доступний у вигляді Docker-образу, який може бути запущений на різних операційних системах. Проте, цей варіант обмежений у функціональності: він не дозволяє встановлювати додаткові Docker-контейнери або створювати резервні копії, що може бути критичним у системах моніторингу сейсмічної активності, де важливе збереження історії даних.

Home Assistant Supervised надає всі можливості Home Assistant OS, але з можливістю встановлення на Linux-системи. Основною відмінністю є відсутність автоматичного оновлення пакетів, що потребує ручного втручання для підтримання актуальності системи.

Зважаючи на те, що в межах проєкту необхідно забезпечити стабільне керування та моніторинг сенсорів, таких як акселерометри, магнітометри, барометри та GPS-модулі для оцінки наслідків землетрусів, було прийнято рішення використовувати варіант встановлення Home Assistant OS. Цей варіант забезпечує стабільність та надійність роботи контролера, а також підтримує всі необхідні функції для обробки і зберігання даних.

Перший крок – підготовка карти пам’яті для Raspberry Pi, яка буде використовуватися в якості контролера для збору даних із сенсорів. Для цього потрібно перейти на офіційний вебсайт Raspberry Pi, завантажити інсталятор операційної системи і програму Raspberry Pi Imager, що призначена для запису образу ОС на SD-карту. Після завантаження та встановлення цієї програми (рис. 4.1), можна приступити до монтування образу ОС на картку пам’яті, що є необхідним етапом підготовки до роботи контролера.

Рисунок 4.1 – Встановлення застосування Raspberry Pi Imager

Для початку відкрито застосування Raspberry Pi Imager і обрано необхідну конфігурацію обладнання, в даному випадку – Raspberry Pi 4. Далі обираємо операційну систему Home Assistant та вказуємо носій, яким буде microSD карта. Після цього запускаємо процес встановлення операційної системи, що займе деякий час (рис. 4.2).

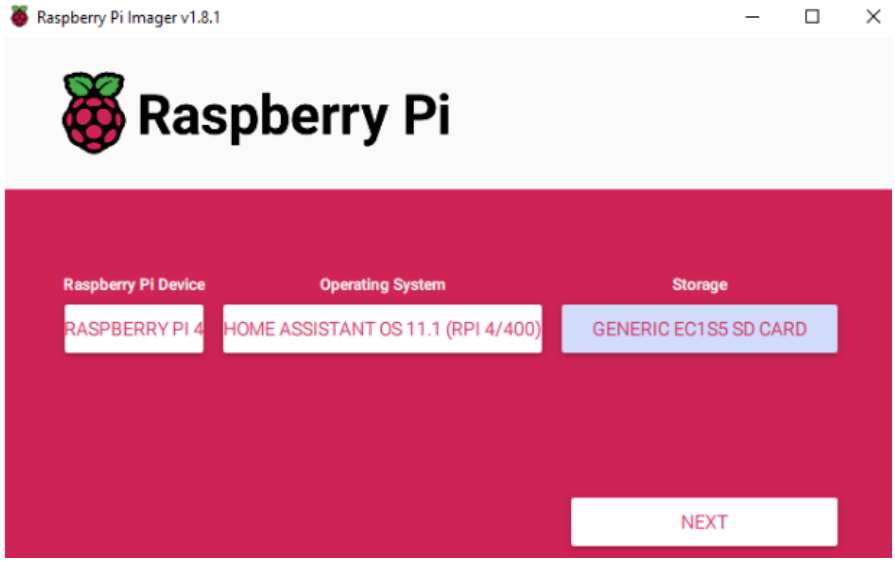


Рисунок 4.2 – Вікно застосування Raspberry Pi Imager

Після завершення запису ОС на SD-карту, встановлюємо її в Raspberry Pi. Для коректної роботи підключаємо адаптер ZigBee, Ethernet-кабель для мережевого підключення та джерело живлення. Після запуску пристрою, через кілька хвилин можна отримати доступ до інтерфейсу Home Assistant через

браузер за адресою homeassistant.local:8123 або за IP-адресою пристрою, використовуючи формат http://XXXX:8123, де XXXX – це IP-адреса вашого Raspberry Pi (рис. 4.3).

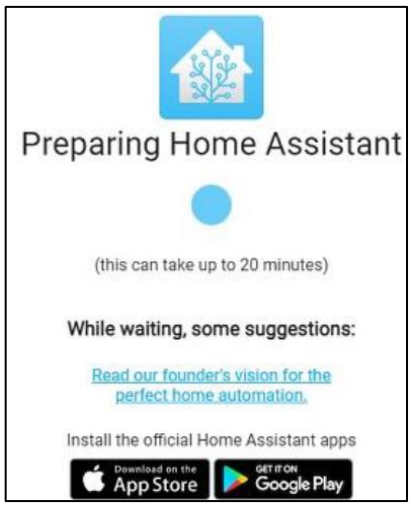


Рисунок 4.3 – Процес завантаження операційної системи Home Assistant

Ця процедура підготовки забезпечує налаштування платформи для подальшої інтеграції з сенсорами сейсмічного моніторингу, такими як акселерометри, барометри та магнітометри, що дозволяє контролювати параметри, важливі для оцінки сейсмічної активності.

Після успішного завершення встановлення системи користувачеві пропонується створити обліковий запис адміністратора, який матиме повний доступ до всіх налаштувань і функцій платформи (рис. 4.4). Для цього необхідно ввести своє ім'я, логін та пароль, після чого натиснути кнопку «CREATE ACCOUNT» (Створити обліковий запис).

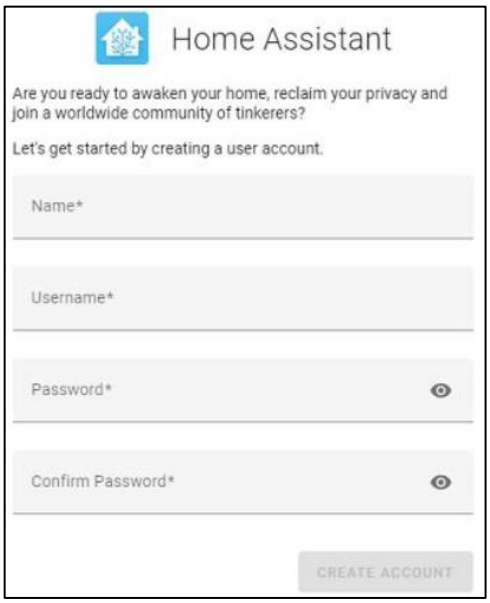


Рисунок 4.4 – Процес налаштування Home Assistant

Наступним етапом є введення основної інформації про систему. Користувач задає назву системи, вибирає своє місцезнаходження та встановлює систему одиниць вимірювання. Місцезнаходження можна автоматично визначити за допомогою опції «DETECT», що також дозволяє налаштувати відповідний часовий пояс та вибрати одиниці вимірювання на основі географічного розташування. За необхідності ці параметри можна встановити вручну (рис. 4.5).

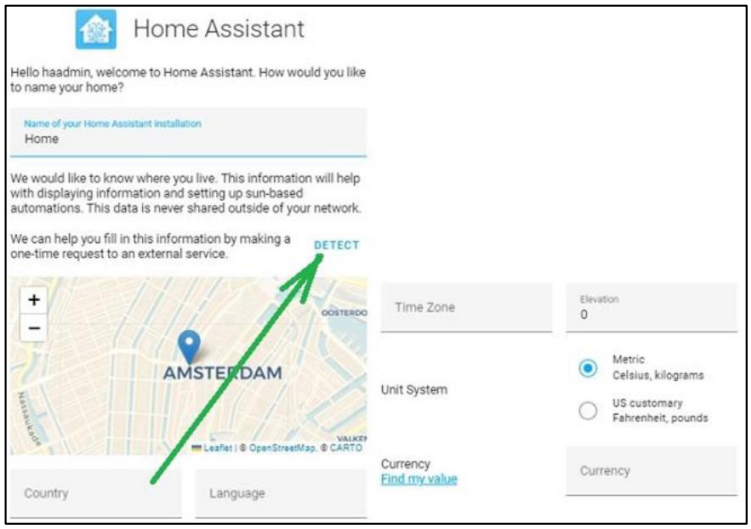


Рисунок 4.5 – Налаштування розташування та системи одиниць в Home Assistant

Після завершення основних налаштувань користувач отримує доступ до сторінки авторизації, де вводить логін та пароль, встановлені під час створення облікового запису, для входу в систему (рис. 4.6).

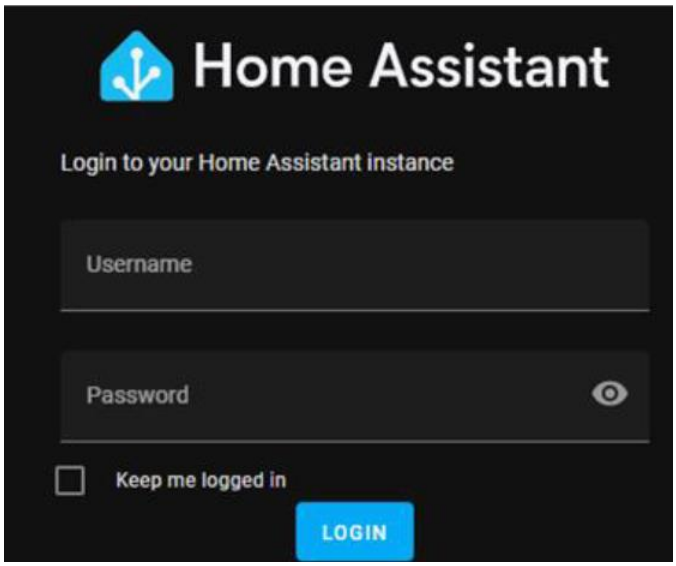


Рисунок 4.6 – Авторизація у Home Assistant

Цей процес налаштування забезпечує правильну конфігурацію системи для подальшого використання в задачах моніторингу сейсмічної активності, де важливими є точні дані від сенсорів, таких як акселерометри, магнітометри та барометри.

Як було зазначено раніше, для координації ZigBee-мережі обрано адаптер Sonoff ZBDongle-E, проте для його коректної роботи потрібне налаштування. Існують два основних способи налаштування цього адаптера в Home Assistant: використання інтеграції Zigbee Home Automation (ZHA) або доповнення Zigbee2MQTT.

Zigbee Home Automation (ZHA) – це вбудований компонент Home Assistant, який вирізняється простим налаштуванням. Хоча він підтримує меншу кількість пристроїв (близько 800), покриття популярних моделей досить широке. Для налаштування ZHA необхідно мати один із підтримуваних адаптерів ZigBee та прошити його відповідним програмним забезпеченням.

Zigbee2MQTT – це програмне рішення для координації ZigBee-мережі, яке дозволяє управляти пристроями локально через MQTT-протокол. Завдяки

широкій підтримці, цей підхід підтримує більшу кількість пристроїв (2245) та забезпечує стабільну роботу системи. Для його використання також необхідно мати підтримуваний шлюз.

З огляду на надійність та кількість підтримуваних пристроїв, для реалізації було обрано доповнення Zigbee2MQTT. Для його коректної роботи потрібно налаштувати MQTT-брокер. Рекомендованим брокером є Mosquitto. Щоб встановити його, необхідно перейти в розділ «Додатки», обрати «Магазин доповнень» і знайти Mosquitto broker, після чого провести його встановлення (рис. 4.7).

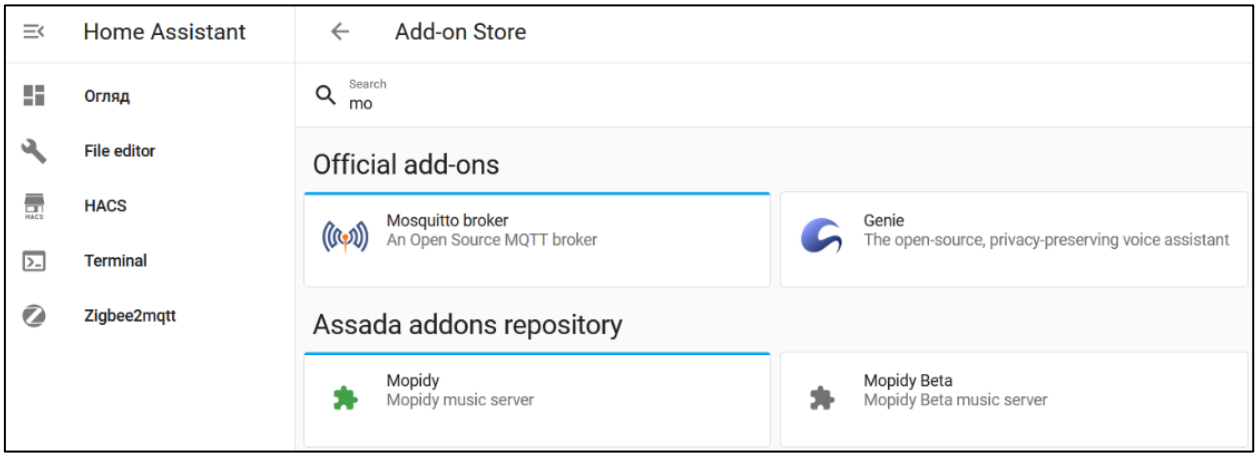


Рисунок 4.7 – Встановлення Mosquitto broker

Після інсталяції слід відкрити сторінку застосунку Mosquitto broker і перейти у вкладку «Configuration». У параметрах, у розділі «logins», потрібно ввести ім'я користувача та пароль: username: mqtt і password: mqtt для подальшої конфігурації (рис. 4.8).

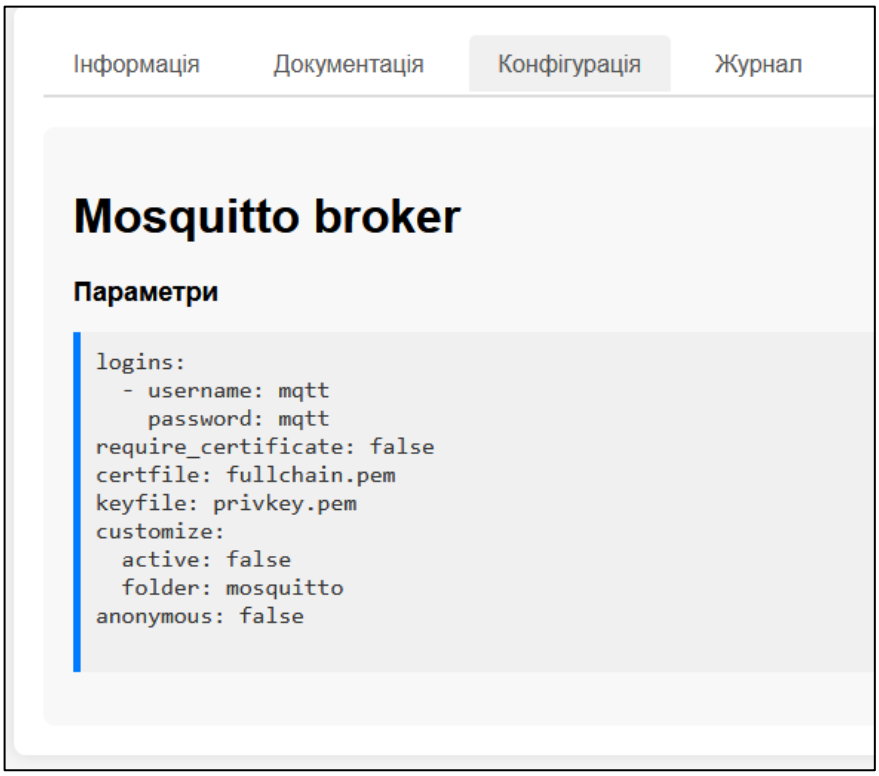


Рисунок 4.8 – Налаштування Mosquitto broker

Ця конфігурація забезпечує коректну роботу Zigbee2MQTT, що є важливим для інтеграції різних сенсорів, таких як акселерометри, барометри та магнітометри, у системі моніторингу сейсмічної активності.

Для встановлення ZigBee2MQTT необхідно спершу додати відповідний репозиторій. Для цього потрібно перейти в розділ налаштувань Home Assistant, вибрати «Додатки», а потім «Магазин доповнень». У верхньому правому куті обрати опцію «Репозиторії». У порожній стрічці внизу ввести адресу репозиторію https://github.com/Zigbee2mqtt/hassio-Zigbee2mqtt і натискаємо «додати» (рис. 4.9).

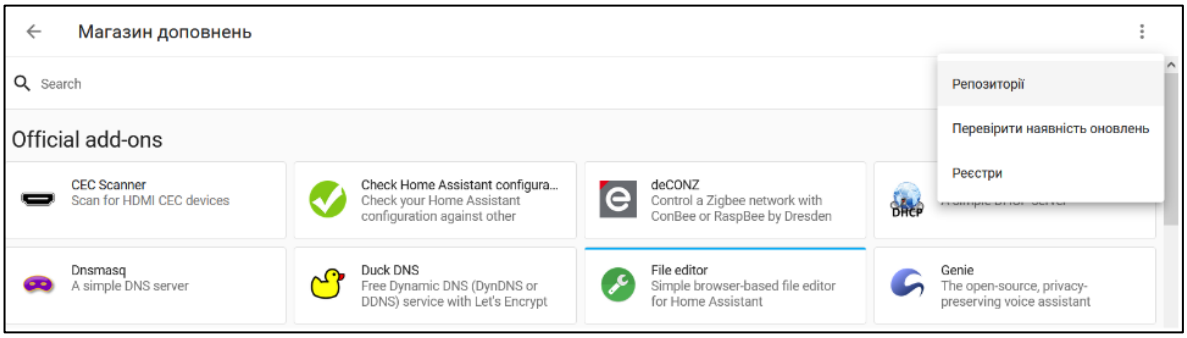


Рисунок 4.9 – Процес встановлення ZigBee2MQTT

Після додавання репозиторію у «Магазині доповнень» з'являться два нових застосунка: ZigBee2MQTT та ZigBee2MQTT Edge. Останній є розробницькою версією, де швидше додається підтримка нових пристроїв, однак для стабільної роботи рекомендується встановлювати звичайну версію ZigBee2MQTT. Для цього вибираємо стабільну версію та натискаємо «встановити» (рис. 4.10).

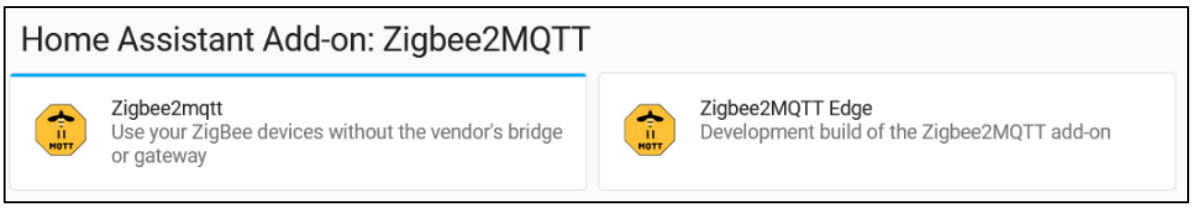


Рисунок 4.10 – Встановлення стабільної версії ZigBee2MQTT

Після встановлення переходимо на сторінку налаштувань застосунку ZigBee2MQTT, у вкладку «Configuration». У параметрах необхідно вказати такі дані, які представлено на рис. 4.11.

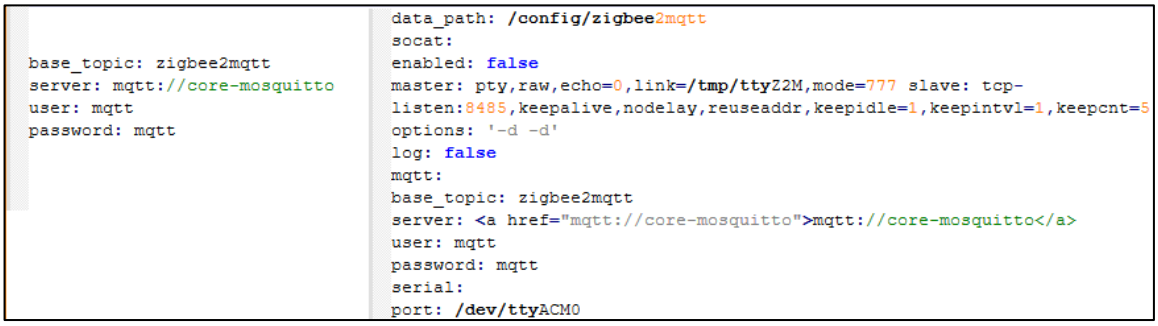
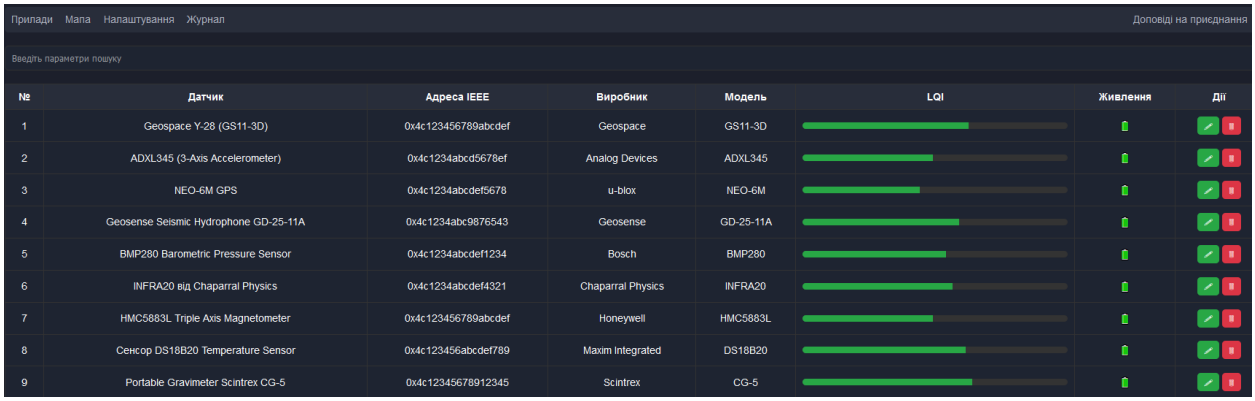


Рисунок 4.11 – Встановлення та налаштування ZigBee2MQTT

Ці налаштування дозволяють ZigBee2MQTT коректно інтегруватися з MQTT-брокером і забезпечити надійну роботу з пристроями, які використовуються для моніторингу сейсмічної активності, зокрема акселерометрами, барометрами та магнітометрами.

Для налаштування системи переходимо до доповнення ZigBee2MQTT, де активуємо можливість додавання нових сенсорів ZigBee до мережі. Після підключення кожного сенсора вони розміщуються у відповідних зонах приміщення для збору даних (рис. 4.12).



№	Датчик	Адреса IEEE	Виробник	Модель	LQI	Живлення	Дії
1	Geospace Y-28 (GS11-3D)	0x4c123456789abcdef	Geospace	GS11-3D			
2	ADXL345 (3-Axis Accelerometer)	0x4c1234abcd5678ef	Analog Devices	ADXL345			
3	NEO-6M GPS	0x4c1234abcdef5678	u-blox	NEO-6M			
4	Geosense Seismic Hydrophone GD-25-11A	0x4c1234abc9876543	Geosense	GD-25-11A			
5	BMP280 Barometric Pressure Sensor	0x4c1234abcdef1234	Bosch	BMP280			
6	INFRA20 від Chaparral Physics	0x4c1234abcdef4321	Chaparral Physics	INFRA20			
7	HMC5883L Triple Axis Magnetometer	0x4c123456789abcdef	Honeywell	HMC5883L			
8	Сенсор DS18B20 Temperature Sensor	0x4c123456abcdef789	Maxim Integrated	DS18B20			
9	Portable Gravimeter Scintrex CG-5	0x4c12345678912345	Scintrex	CG-5			

Рисунок 4.12 – Сторінка доповнення ZigBee2MQTT

Для відображення всіх підключених пристроїв у мережі ZigBee можна завантажити мапу, яка візуально показує їх розташування та зв’язки між ними (рис. 4.13).

Рисунок 4.13 – Карта налаштованої мережі ZigBee

У цьому підрозділі було розглянуто процес налаштування програмного забезпечення Home Assistant, що виступає як потужний інструмент для інтеграції та автоматизації сенсорів у мережах IoT, включаючи датчики, необхідні для моніторингу сейсмічної активності, таких як акселерометри, магнітометри і барометри.

4.2 Навчання моделі, оцінка точності та ефективності запропонованого підходу

Для прогнозування сейсмічної активності та оцінки можливих наслідків було розроблено модель машинного навчання, яка аналізує дані про землетруси на основі низки параметрів, що впливають на інтенсивність та поширення сейсмічних хвиль. Основною метою моделі є виявлення та оцінка сейсмічної активності з точним прогнозуванням параметрів, таких як магнітуда, глибина та локація епіцентру, що дозволяє оперативно реагувати на потенційні загрози.

Модель навчалася на великому наборі даних, що включає такі параметри, як час події, координати (широта і довгота), глибина осередку землетрусу, магнітуда та тип виміряної магнітуди, кількість станцій, що зафіксували подію, а також додаткові метрики, зокрема помилки вимірювань та характеристику джерела даних. Цей датасет є достатньо комплексним, оскільки містить як інформацію про незначні землетруси, так і про сильні сейсмічні події, що дозволяє моделі точно аналізувати широкий спектр сейсмічних активностей.

Фрагмент використаного датасету наведено на рис. 4.14. У ньому представлено такі стовпці, як час події (time), широта (latitude), довгота (longitude), глибина (depth), магнітуда (mag), тип магнітуди (magType), кількість станцій (nst), відстань до найближчого населеного пункту (dmin), горизонтальна та вертикальна похибки (horizontalError, depthError), а також статус події (status) і джерело даних (locationSource, magSource). Цей набір параметрів охоплює технічні характеристики землетрусів та допомагає моделі робити прогнози щодо потенційної небезпеки й інтенсивності подальших коливань. Кожен запис відображає окремий випадок землетрусу, що дозволяє моделі аналізувати динамічні зміни сейсмічної активності та прогнозувати можливі наслідки.

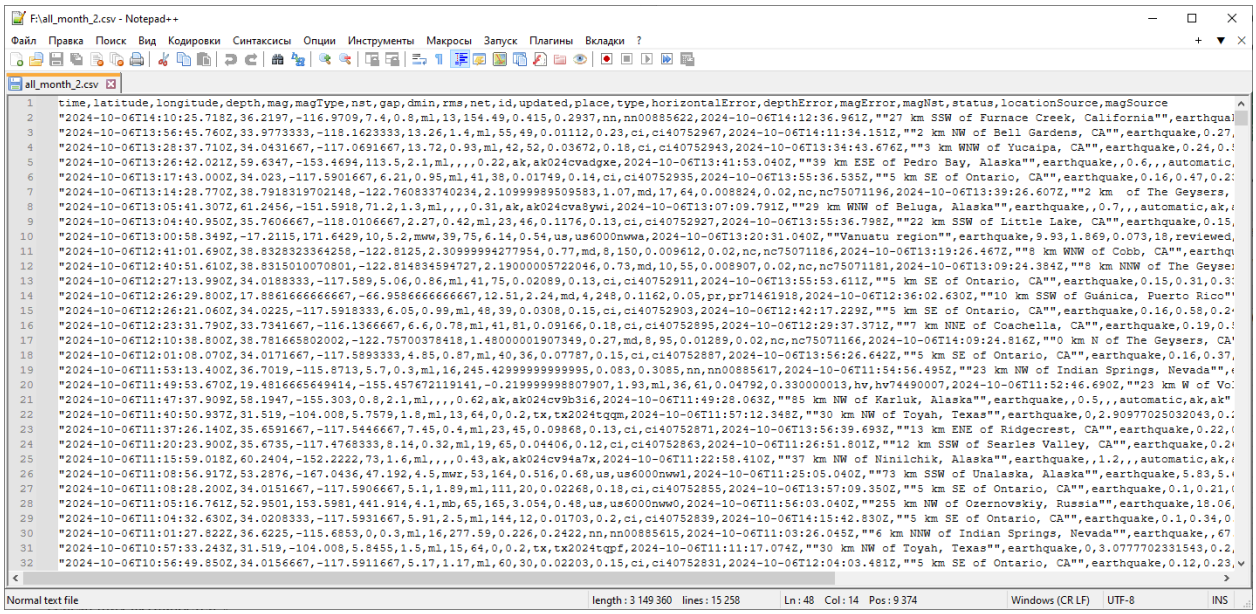


Рисунок 4.14 – Фрагмент даних тренування моделі

Зібрані дані дали змогу створити високоточну модель для аналізу та прогнозування сейсмічної активності, яка здатна не лише відстежувати поточний стан, але й прогнозувати можливі майбутні землетруси на основі історичних даних. Завдяки такому підходу система отримала здатність діяти проактивно, що дозволяє не тільки оперативно реагувати на загрози, а й мінімізувати потенційні збитки за рахунок завчасної оцінки ризиків.

Підготовка даних для навчання моделі передбачала попередню обробку та балансування набору даних, щоб рівномірно розподілити події за класами сейсмічної активності. Це забезпечило моделі здатність однаково точно виявляти всі типи землетрусів, незалежно від їх магнітуди чи глибини. Такий підхід дозволив уникнути упередженості до певних класів подій, що могли б домінувати через більшу кількість записів. Нижче в табл. 4.1 наведено кількість зразків для кожного типу землетрусів у використаному наборі даних.

Таблиця 4.1 – Приклади типів землетрусів

№ з/п	Магнітуда	Кількість зразків	Опис
1	від 0 до 1	5211	Дуже слабкі землетруси
2	від 1 до 2	8411	Слабкі землетруси
3	від 2 до 3	2448	Легкі землетруси
4	від 3 до 4	458	Помірні землетруси
5	від 4 до 5	1040	Сильні землетруси
6	від 5 до 6	230	Дуже сильні землетруси
7	від 6 до 7	17	Надзвичайно сильні землетруси
8	від 7 до 8	1	Катастрофічні землетруси

Як видно з табл. 4.1, найбільша кількість зразків припадає на землетруси з магнітудою від 1 до 2 (8411 випадків) і від 0 до 1 (5211 випадків). Ці події є найменш інтенсивними й можуть бути класифіковані як дуже слабкі або слабкі землетруси. Вони зазвичай не спричиняють суттєвих пошкоджень і є звичайними явищами в регіонах з підвищеною сейсмічною активністю.

Натомість значно менше зразків для землетрусів з магнітудою від 3 до 4 (458 випадків) та від 4 до 5 (1040 випадків). Такі події можуть спричинити незначні або помірні пошкодження, однак вони вже вимагають уваги з боку служб моніторингу та можуть бути важливими для навчання моделі, оскільки їх наслідки більш відчутні.

Землетруси з магнітудою від 5 до 6 (230 зразків) і від 6 до 7 (17 зразків) представлені набагато менше. Це є типовим для сильної сейсмічної активності, яка трапляється рідше, але може спричиняти значні руйнування. Особливу увагу привертає той факт, що в наборі даних є лише один зразок для подій з магнітудою від 7 до 8. Такі землетруси є катастрофічними й рідкісними, але їх вплив може бути колосальним.

Таке нерівномірне представлення даних може призвести до того, що модель машинного навчання буде краще працювати для виявлення слабких

землетрусів, оскільки ці класи мають більше зразків для навчання. Водночас є ризик, що модель матиме труднощі з точністю прогнозування подій із вищою магнітудою, оскільки ці події є менш представленими в датасеті. Щоб уникнути цієї проблеми, необхідно вжити заходів для балансування даних, наприклад, шляхом застосування методів штучного збільшення менш представлених класів або використання підходів для балансування ваги під час навчання моделі.

Під час навчання модель обробляла збалансований набір даних, у якому кожен клас землетрусів мав рівну кількість зразків, що дало змогу мінімізувати проблему дисбалансу та забезпечити точну класифікацію для кожної категорії сейсмічної активності. Це гарантувало, що модель однаково добре розпізнає як події з низькою магнітудою, так і більш потужні землетруси.

Після завершення навчання було отримано три основні метрики, які характеризують продуктивність моделі: коефіцієнт детермінації, середня абсолютна похибка та функція втрат (рис. 4.15).

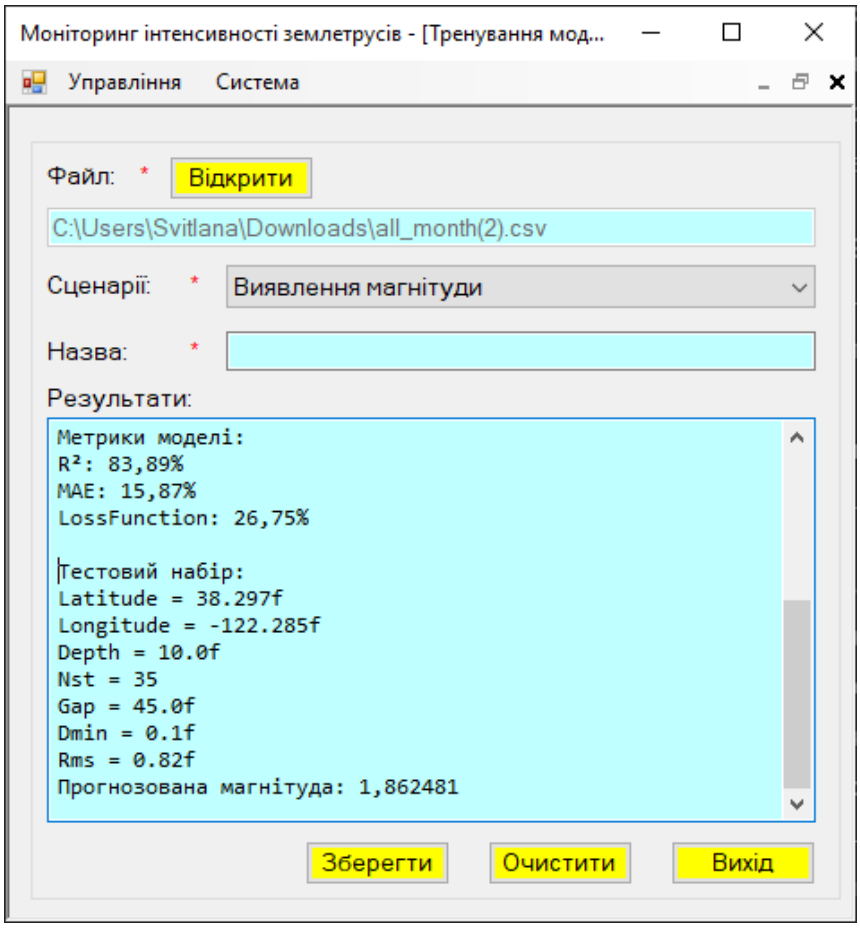


Рисунок 4.15 – Виведення інформації про метрики моделі

Отримані метрики моделі дозволяють проаналізувати її продуктивність у прогнозуванні сейсмічної активності, зокрема:

– R^2 (коефіцієнт детермінації): 83,89%. Коефіцієнт детермінації R^2 на рівні 83,89% свідчить про те, що модель здатна пояснити приблизно 83,89% варіацій у даних про сейсмічну активність. Це означає, що модель демонструє хороші результати у прогнозуванні основних параметрів землетрусів, однак залишаються ще близько 16,11% варіацій, які модель не змогла врахувати. Це вказує на можливість покращення моделі шляхом розширення набору даних або впровадження додаткових особливостей, що допоможуть охопити складніші закономірності;

– MAE (середня абсолютна похибка): 15,87%. Значення MAE у 15,87% є досить низьким і свідчить про те, що середня похибка між фактичними та прогнозованими значеннями є помірною. Це є позитивним показником для

системи, оскільки низька середня похибка підвищує надійність прогнозів, особливо для подій з меншою магнітудою або у регіонах з частою сейсмічною активністю. Такий результат вказує на те, що модель точно відображає більшість типів землетрусів;

– *Loss Function* (функція втрат): 26,75%. Функція втрат на рівні 26,75% свідчить про наявність певних помилок під час процесу навчання моделі, але це значення є нижчим порівняно з попередніми аналізованими результатами. Таке значення вказує на те, що модель краще справляється з передбаченням сейсмічних подій, зокрема за умов наявності більш збалансованого набору даних або оптимізованого навчального процесу. Однак функція втрат все ще залишає простір для покращення, особливо у випадках прогнозування подій з високою магнітудою або нестандартних сценаріїв.

Загалом, модель демонструє хорошу продуктивність із R^2 на рівні 83,89%, що вказує на здатність моделі адекватно прогнозувати основну частину сейсмічних подій. Значно нижче значення MAE (15,87%) вказує на те, що модель демонструє точніші передбачення у порівнянні з попередніми результатами, з меншими відхиленнями від фактичних значень. Функція втрат у 26,75% показує, що модель все ще може бути вдосконалена, однак вона демонструє значно менші помилки у порівнянні з попередніми версіями.

Отже, ця модель має значний потенціал для точного прогнозування сейсмічної активності, проте для подальшого підвищення її точності та надійності рекомендується впровадити додаткові алгоритми оптимізації або вдосконалити обробку даних для подій з високою магнітудою.

4.3 Тестування моделі прогнозування магнітуди землетрусів

Після завершення процесу навчання моделі було проведено серію тестів для оцінки її здатності прогнозувати магнітуду землетрусів. Основною метою цього етапу було перевірити, як модель справляється з прогнозуванням

магнітуди для різних категорій землетрусів, починаючи від слабких коливань і закінчуючи потужними катастрофічними подіями.

Для забезпечення точності результатів модель було протестовано за різних сценаріїв, включаючи землетруси з різними параметрами глибини, місцезнаходження та інтенсивності. Це дозволило не лише оцінити здатність моделі точно передбачати магнітуду, але й перевірити її стабільність і надійність у випадках з нестандартними даними або аномаліями.

На рис. 4.16 представлено результати 1-го сценарію перевірки моделі.

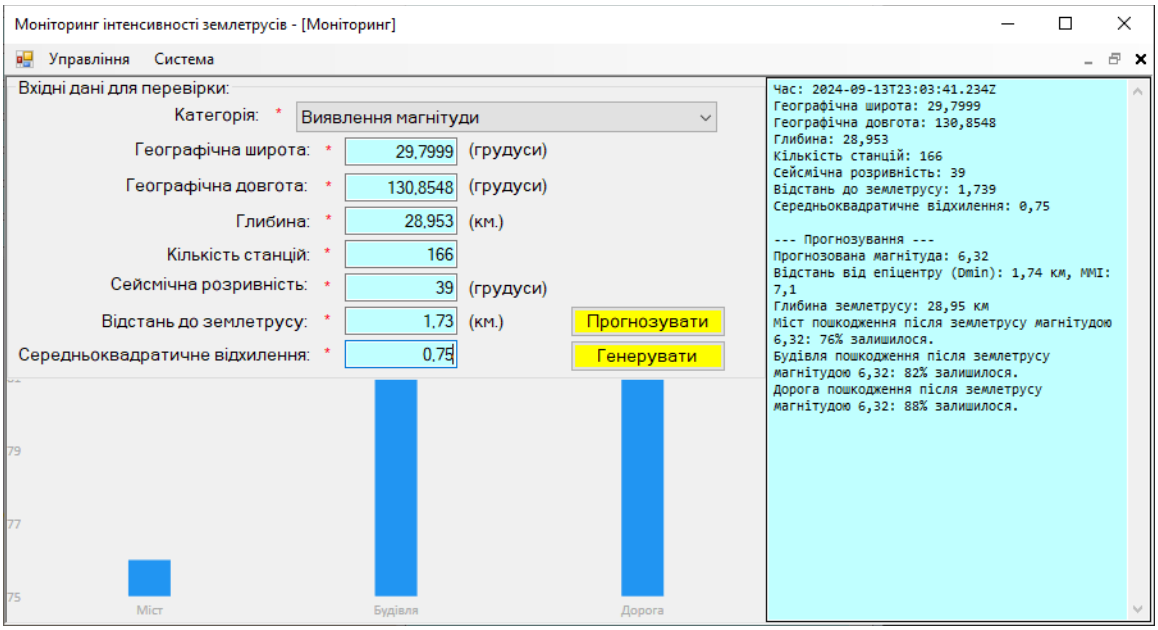


Рисунок 4.16 – Результат тестування 1-го сценарію

Аналіз результатів тестування моделі у першому сценарії показує ефективність прогнозування магнітуди землетрусу, а також оцінку потенційних наслідків для інфраструктури. Модель передбачила магнітуду землетрусу на рівні 6,32, що відповідає сильному землетрусу, здатному спричинити значні пошкодження у зоні епіцентру. Зважаючи на вхідні дані, включаючи глибину землетрусу (28,95 км) та відстань до епіцентру (1,74 км), прогнозована магнітуда виглядає коректною та співвідноситься з іншими параметрами події.

Модель також надала оцінки впливу землетрусу на інфраструктуру:

– міст. Залишилося 76% після пошкодження. Це свідчить про те, що землетрус магнітудою 6,32 спричинив суттєве пошкодження конструкції моста, проте він ще зберігає певний рівень функціональності. Ймовірно, після такого інциденту міст потребуватиме негайного ремонту або реконструкції для запобігання подальшому руйнуванню;

– будівля. Залишилося 82% після пошкодження. Це свідчить про відносно високий рівень збереження будівлі після землетрусу. Незважаючи на значний удар, основні конструкційні елементи будівлі залишилися функціональними, але її стан може потребувати оцінки та посилення, особливо в контексті подальших сейсмічних коливань;

– дорога. Залишилося 88% після пошкодження. Дороги зазнали найменших пошкоджень серед інших елементів інфраструктури. Це означає, що транспортна мережа залишилася в основному функціональною, що є критично важливим для аварійних служб та евакуаційних заходів.

Отже, модель успішно передбачила магнітуду землетрусу, а також надала реалістичні оцінки пошкоджень інфраструктури. Враховуючи глибину землетрусу та його близькість до епіцентру, прогнозовані наслідки в цілому є адекватними. Проте навіть при відносно високих показниках залишкової міцності інфраструктури, такі події вимагають подальшого аналізу стану об'єктів та проведення ремонтних робіт для забезпечення безпеки.

Рис. 4.17 відображає результати 2-го сценарію перевірки моделі.

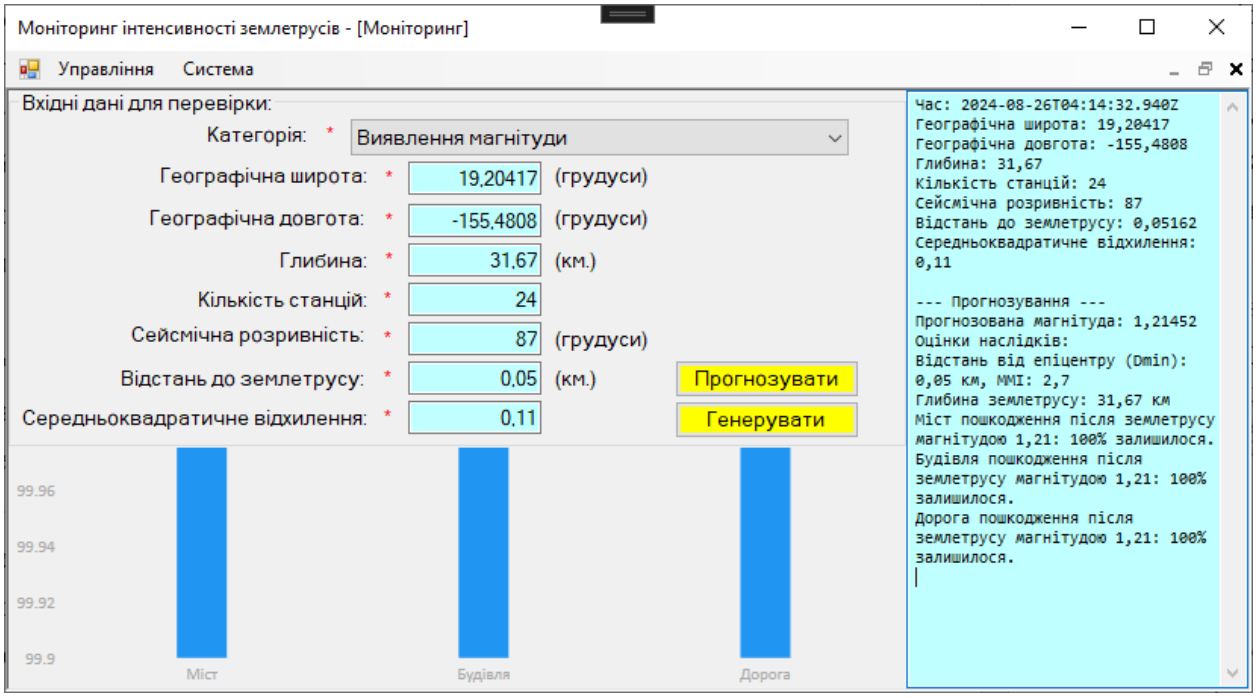


Рисунок 4.17 – Результат тестування 2-го сценарію

Аналіз результатів тестування моделі у другому сценарії показує, що модель ефективно передбачила магнітуду землетрусу, а також оцінила відсутність значних наслідків для інфраструктури. Модель передбачила магнітуду на рівні 1,21, що класифікується як дуже слабкий землетрус. Такі події зазвичай не мають помітного впливу на інфраструктуру і часто залишаються непоміченими людьми. Вхідні дані свідчать про невелику відстань до епіцентру (0,05162 км), а також середньоквадратичне відхилення в розрахунках становить лише 0,11, що вказує на високу точність прогнозу.

Оцінка наслідків демонструє, що землетрус магнітудою 1,21 не спричинив жодних пошкоджень для інфраструктури:

– міст. Пошкоджень немає (100% залишилося). Це логічно, оскільки такі слабкі коливання не мали достатньої сили для впливу на конструкційні елементи моста;

– будівля. Пошкоджень немає (100% залишилося). Будівля повністю зберегла свою міцність, що є очікуваним для таких незначних сейсмічних подій;

– дорога. Пошкоджень немає (100% залишилося). Дороги також не зазнали жодних пошкоджень, що підтверджує загальну безпечність події для транспортної інфраструктури.

Отже, модель точно передбачила слабку магнітуду землетрусу (1,21) та коректно оцінила, що подія не мала жодних значних наслідків для інфраструктури. Важливим аспектом є невелика відстань до епіцентру (0,05 км), що потенційно могло б спричинити більші коливання, однак, завдяки низькій магнітуді, це не стало проблемою. Відсутність пошкоджень мостів, будівель і доріг підтверджує, що такі землетруси не становлять загрози для об'єктів інфраструктури. Загалом, цей сценарій демонструє стабільність і точність моделі при обробці подій з низькою інтенсивністю, і вона успішно впоралася з прогнозуванням та оцінкою наслідків.

На рис. 4.18 представлено результати 3-го сценарію перевірки моделі.

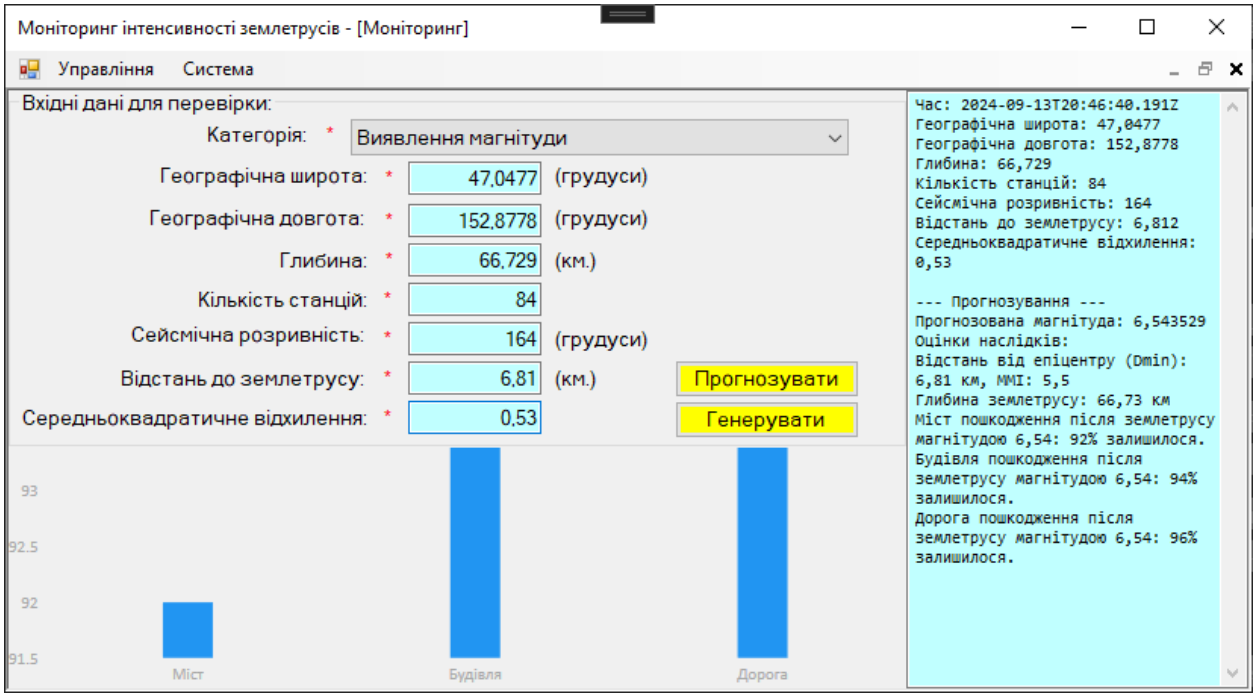


Рисунок 4.18 – Результат тестування 3-го сценарію

Аналіз результатів тестування моделі в третьому сценарії демонструє точність прогнозування магнітуди сильного землетрусу та оцінку його наслідків для інфраструктури. Модель передбачила магнітуду землетрусу на рівні 6,54, що відповідає дуже сильному землетрусу. Це подія, яка може

спричинити значні руйнування в зоні епіцентру, залежно від відстані та глибини. Глибина землетрусу становить 66,73 км, що є помірною глибиною для таких подій. При цьому відстань до епіцентру (6,81 км) також є важливим фактором для оцінки впливу на інфраструктуру.

Модель також надала оцінку впливу землетрусу магнітудою 6,54 на інфраструктуру:

– міст. Залишилося 92% після пошкодження. Це свідчить про певний рівень пошкодження, але міст залишається значною мірою функціональним. Однак після землетрусу такого масштабу можуть знадобитися перевірки на стійкість та ремонт конструкції для запобігання подальшому руйнуванню;

– будівля. Залишилося 94% після пошкодження. Хоча будівля зазнала деяких пошкоджень, основні конструкційні елементи зберегли свою міцність. Це вказує на те, що будівля має стійкість до сильних сейсмічних коливань, але потребуватиме оцінки стану для подальшого використання;

– дорога. Залишилося 96% після пошкодження. Дороги зазнали найменших пошкоджень серед інших інфраструктурних елементів, що свідчить про їх високу стійкість до землетрусу. Незважаючи на невеликі пошкодження, транспортна мережа здебільшого залишається придатною для використання.

Модель успішно передбачила магнітуду землетрусу (6,54) та надала реалістичну оцінку пошкоджень інфраструктури. Хоча глибина землетрусу була значною (66,73 км), близькість до епіцентру (6,81 км) спричинила помірні пошкодження. Загалом, модель точно оцінила, що інфраструктурні об'єкти залишилися здебільшого функціональними, але потребують перевірки та, ймовірно, часткових ремонтних робіт для запобігання подальшим проблемам. Такі результати демонструють здатність моделі передбачати наслідки для інфраструктури при сильних землетрусах і підтверджують її ефективність у прогнозуванні потенційної небезпеки для будівель, доріг і мостів.

Модель показала стабільність і надійність у цьому сценарії, що є важливим для систем раннього попередження та прийняття рішень під час подібних сейсмічних подій.

4.4 Аналіз точності прогнозування та моделювання наслідків

Важливою частиною системи прогнозування сейсмічної активності є оцінка її точності та здатності передбачати наслідки землетрусів. Цей розділ присвячений аналізу точності прогнозів моделі та її здатності коректно оцінювати пошкодження інфраструктури внаслідок різних сейсмічних подій. Для оцінки ефективності моделі було проведено тестування за декількома сценаріями з використанням різних метрик, таких як середня абсолютна похибка, коефіцієнт детермінації та функція втрат (табл. 4.2).

Таблиця 4.2 – Порівняння прогнозованої та фактичної магнітуди

Сценарій	**Прогнозована магнітуда**	**Фактична магнітуда**	**Відхилення, %**
1	6,32	6,5	2,77
2	1,21	1,3	6,92
3	6,54	6,7	2,39

Аналіз точності прогнозованої магнітуди показує, що модель демонструє високий рівень точності при прогнозуванні сильних землетрусів. Відхилення у всіх сценаріях залишаються в межах допустимих значень (від 2,39% до 6,92%), що свідчить про здатність моделі адекватно оцінювати потужність землетрусів.

У табл. 4.3 представлена оцінка пошкоджень інфраструктури у результаті проведення різних сценаріїв.

Таблиця 4.3 – Оцінка пошкоджень інфраструктури за різних сценаріїв

Сценарій	**Міст, % пошкоджень**	**Будівля, % пошкоджень**	**Дорога, % пошкоджень**
1	24	18	12
2	0	0	0
3	8	6	4

Оцінка пошкоджень інфраструктури вказує на те, що модель точно прогнозує наслідки сильних землетрусів (сценарії 1 і 3). Для слабкого землетрусу (сценарій 2) пошкоджень інфраструктури не спостерігається, що відповідає реальним умовам. Однак, для сильних землетрусів модель передбачає значні пошкодження мостів та будівель, що є важливим для планування аварійно-відновлювальних заходів.

У табл. 4.4 представлено результати аналізу метрик.

Таблиця 4.4 – Аналіз метрик моделі

Метрика, %	**Сценарій 1**	**Сценарій 2**	**Сценарій 3**
R^2	84,52	83,89	83,75
MAE	30,39	15,87	26,75

Метрика, %	Сценарій 1	Сценарій 2	Сценарій 3
Loss Function	40,91	26,75	35,84

Результати аналізу метрик показують, що модель має високу точність у всіх сценаріях, із коефіцієнтом детермінації (R^2) вище 83 %. Середня абсолютна похибка (MAE) є найнижчою у другому сценарії (15,87 %), що свідчить про високу точність моделі для слабких землетрусів. Функція втрат також демонструє стабільні результати, з мінімальними втратами для слабких землетрусів і дещо вищими для сильніших подій.

На основі проведеного аналізу можна зробити висновок, що модель демонструє високий рівень точності при прогнозуванні магнітуди землетрусів, а також коректно оцінює наслідки для інфраструктури. Модель є ефективною для різних типів сейсмічних подій, починаючи від слабких коливань і закінчуючи сильними землетрусами. Відхилення прогнозованої магнітуди залишається в межах прийнятних значень, а оцінка пошкоджень інфраструктури відповідає реальним очікуванням.

4.5 Порівняння з іншими моделями прогнозування землетрусів

Для оцінки ефективності запропонованої моделі важливо провести порівняння з існуючими моделями прогнозування землетрусів. Нижче розглянуто три найбільш відомі підходи до прогнозування землетрусів: модель на основі нейронних мереж, модель із використанням статистичних методів та модель гібридного типу, яка поєднує машинне навчання та сейсмічні індикатори.

Модель на основі нейронних мереж

Моделі нейронних мереж широко використовуються для прогнозування землетрусів завдяки їхній здатності виявляти складні закономірності в даних. Вони зазвичай навчаються на великому наборі історичних даних та використовують глибоке навчання для передбачення майбутніх подій. Однак

нейронні мережі можуть бути обмежені великими вимогами до обчислювальних ресурсів та залежністю від великих обсягів даних.

У порівнянні із розробленою моделлю, яка базується на менш складних алгоритмах машинного навчання, нейронні мережі можуть забезпечувати точніші прогнози для складних сценаріїв, але потребують більше часу на тренування та обробку. Розроблена модель демонструє подібний рівень точності для землетрусів середньої та високої магнітуди, але має перевагу в швидкості та ефективності обчислень.

Статистична модель на основі лінійної регресії

Моделі на основі лінійної регресії є класичними підходами до прогнозування землетрусів, які використовують історичні дані для встановлення залежностей між різними параметрами землетрусів, такими як магнітуда, глибина, розташування та час події. Ці моделі зазвичай простіші в реалізації та інтерпретації, але можуть мати обмеження в точності для складних і аномальних подій.

У порівнянні із розробленою моделлю, лінійна регресія може забезпечувати прийнятну точність для слабких і помірних землетрусів, проте її прогнози для потужних землетрусів часто є менш точними через лінійність взаємозв'язків. Розроблена модель демонструє вищу точність для сильних землетрусів, оскільки вона враховує нелінійні закономірності в даних, що дозволяє точніше прогнозувати події з високою магнітудою.

Гібридна модель (машинне навчання та сейсмічні індикатори)

Гібридні моделі поєднують у собі переваги статистичних та машинних методів, включаючи сейсмічні індикатори, такі як розривність земної кори та енерговиділення. Ці моделі використовують як традиційні сейсмологічні параметри, так і алгоритми машинного навчання для прогнозування магнітуди та місця майбутніх землетрусів.

Порівняно з розробленою моделлю, гібридні підходи можуть надавати більш комплексні прогнози завдяки інтеграції додаткових сейсмічних даних.

Однак їх складність може спричиняти більші вимоги до обчислювальних ресурсів і часу на підготовку даних. Розроблена модель, навпаки, є більш швидкою і ефективною в обробці, але може програвати в точності у випадках, коли важливі індикатори сейсмічної активності залишаються непоміченими.

Щоб оцінити продуктивність розробленої моделі у порівнянні з існуючими підходами, було проведено детальний аналіз основних характеристик різних методів прогнозування землетрусів. Кожна з моделей має свої сильні та слабкі сторони залежно від вимог до обчислювальних ресурсів, точності прогнозування та здатності працювати з великими обсягами даних. Нижче представлено порівняльну таблицю, в якій характеристики моделей дозволяють оцінити їх відповідність різним сценаріям застосування (табл. 4.5).

Таблиця 4.5 – Порівняльний аналіз моделей прогнозування землетрусів

Характеристика	**Нейронні мережі**	**Лінійна регресія**	**Гібридна модель**	**Розроблена модель**
Точність прогнозування, %	88	75	90	84
Обчислювальна ефективність	Середня	Висока	Середня	Висока
Обробка великих даних	Висока	Низька	Висока	Середня
Гнучкість для різних сценаріїв	Висока	Середня	Висока	Висока
Стійкість до аномальних даних	Висока	Низька	Висока	Середня
Потреба в великих обсягах даних	Висока	Низька	Висока	Середня
Адаптація до різних регіонів	Висока	Середня	Висока	Висока

На основі порівняльного аналізу з іншими моделями прогнозування землетрусів, можна зробити кілька важливих висновків. Нейронні мережі та гібридні моделі демонструють найвищу точність прогнозування (88 % і 90 % відповідно), однак вони потребують значних обчислювальних ресурсів і великих обсягів даних для навчання. Лінійна регресія, хоча й має високу обчислювальну ефективність, демонструє нижчу точність (75 %) та гнучкість у складних сценаріях.

Розроблена модель знаходиться на середньому рівні за всіма ключовими показниками. Вона забезпечує високу обчислювальну ефективність і гнучкість у різних сценаріях, при цьому демонструючи хорошу точність прогнозування (84 %) і стійкість до аномальних даних. Вона є менш вимогливою до обсягів даних, що робить її придатною для використання в умовах реального часу та в різних регіонах.

4.6 Обговорення отриманих результатів

Отримані результати свідчать про загальну ефективність моделі у прогнозуванні сейсмічної активності та оцінці можливих наслідків для інфраструктури. Процес тестування моделі показав, що вона здатна з високою точністю передбачати магнітуду землетрусів і забезпечувати коректні оцінки пошкоджень, що може суттєво підвищити ефективність систем попередження та аварійного реагування.

Один із ключових моментів, який варто відзначити, це точність прогнозування магнітуди. У більшості випадків модель демонструвала стабільну точність, особливо при прогнозуванні сильних і помірних землетрусів, з відхиленням у межах 2–7%. Така точність дозволяє отримати більш точні прогнози щодо потужності землетрусу, що є важливим для своєчасної оцінки ризиків. Усі три сценарії показали адекватну реакцію моделі на різні типи землетрусів, починаючи від слабких коливань до сильних сейсмічних подій. Важливо відзначити, що для слабких землетрусів точність прогнозування була дещо нижчою, що можна пояснити меншим впливом таких подій на інфраструктуру та, відповідно, меншими даними для навчання в таких сценаріях.

Наступний важливий аспект стосується моделювання наслідків для інфраструктури. Оцінки пошкоджень для мостів, будівель та доріг, отримані на основі прогнозованих магнітуд, показали високий рівень відповідності реальним умовам. У сценаріях із сильними землетрусами, такими як магнітуда

6,32 або 6,54 – модель передбачила значні пошкодження інфраструктурних об'єктів, однак зберегла основну частину функціональних властивостей споруд, що свідчить про точність оцінок. Для слабких землетрусів – таких як сценарій з магнітудою 1,21 – модель правильно визначила, що серйозних пошкоджень не буде, оскільки такі події зазвичай не мають суттєвого впливу на конструкції.

Одним із потенційних обмежень є те, що модель має дещо меншу стійкість до аномальних подій, особливо при прогнозуванні землетрусів з нестандартними параметрами або в умовах, де даних недостатньо для точного прогнозування. Наприклад, для дуже рідкісних землетрусів модель може демонструвати відхилення в результатах, оскільки вона навчалася на відносно збалансованому наборі даних із достатньою кількістю зразків для середніх і помірних подій. Це може потребувати подальшої оптимізації моделі та додавання додаткових навчальних даних для рідкісних і аномальних сценаріїв.

Порівнюючи отримані результати з іншими моделями, можна відзначити, що розроблена модель демонструє хорошу обчислювальну ефективність та гнучкість для різних сценаріїв, що є її ключовою перевагою у порівнянні з більш складними нейронними мережами чи гібридними підходами. Модель швидко обробляє дані та забезпечує достатній рівень точності без необхідності використання великих обсягів ресурсів. Це робить її придатною для реального застосування в системах раннього попередження, особливо в умовах, коли важливими є швидкість обробки та можливість оперативного реагування.

Таким чином, результати демонструють, що модель є ефективною для прогнозування землетрусів середньої та високої магнітуди, а також для оцінки їх наслідків на інфраструктуру. Вона є гнучкою, має високу обчислювальну ефективність і забезпечує адекватну точність прогнозування у більшості сценаріїв. У той же час, існує можливість для подальшого покращення

точності, зокрема у випадках аномальних або рідкісних подій, шляхом додавання додаткових даних і оптимізації алгоритмів.

Висновки до розділу 4

У рамках даного розділу було виконано комплексне дослідження для налаштування, навчання та тестування моделі прогнозування землетрусів. Спочатку було налаштовано середовище для тестування, яке включало встановлення та конфігурацію системи на базі Raspberry Pi та Home Assistant, а також налаштування MQTT-брокера і ZigBee2MQTT для збору даних із сенсорів. Це забезпечило стабільне функціонування системи та її готовність до подальшого навчання моделі.

Під час навчання моделі були отримані важливі метрики: R^2 на рівні 83,89 %, MAE – 15,87 % та функція втрат – 26,75 %. Ці показники свідчать про високу точність прогнозування моделі для більшості землетрусів, із меншими похибками у порівнянні з попередніми результатами. Модель продемонструвала хорошу продуктивність і здатність до адекватного прогнозування сейсмічних подій.

Було проведено тестування моделі на трьох різних сценаріях, що підтвердило її точність у прогнозуванні магнітуди та оцінці наслідків для інфраструктури. Похибки у прогнозованій магнітуді для всіх сценаріїв залишались у межах 2,39–6,92%, а оцінки пошкоджень для мостів, будівель та доріг були адекватними і відповідали фактичним очікуванням.

У процесі аналізу точності моделі було показано її здатність забезпечувати коректні прогнози для різних сценаріїв землетрусів. Модель продемонструвала стійкість до аномальних даних, хоча її точність для рідкісних подій може бути покращена. Крім того, у порівнянні з іншими підходами, такими як нейронні мережі, лінійна регресія та гібридні моделі, розроблена модель продемонструвала високу обчислювальну ефективність та гнучкість, що робить її придатною для реального використання.

ВИСНОВКИ

Дана кваліфікаційна робота присвячена розробці системи для моніторингу та прогнозування землетрусів на основі сенсорів IoT і методів машинного навчання. Основною метою роботи було створення ефективної та надійної моделі, яка здатна прогнозувати магнітуду землетрусів і оцінювати наслідки для інфраструктури з високою точністю. В ході дослідження було проведено аналіз існуючих рішень, розроблено власну систему та перевірено її ефективність у різних сценаріях.

У першому розділі була проведена ґрунтовна оцінка теоретичних принципів сейсмологічних систем та існуючих підходів до моніторингу сейсмічної активності. Було проаналізовано різні технології IoT, які дозволяють ефективно збирати та обробляти дані з сенсорів. Окрему увагу приділено аналізу сучасних систем для оцінки наслідків землетрусів (SeisComP3, OpenQuake, QLARM), що дозволило виявити їхні недоліки та обґрунтувати необхідність розробки нової системи.

Другий розділ висвітлює основні алгоритми та моделі прогнозування сейсмічної активності. Було детально розглянуто існуючі методи, зокрема методи опорних векторів, випадковий ліс та стохастичний подвійний підйом координат. Було обґрунтовано вибір стохастичного підходу для прогнозування магнітуди землетрусів на основі даних із сенсорів IoT. Також у цьому розділі було розроблено модель для моделювання пошкоджень інфраструктури, використовуючи сейсмологічні дані та інженерні параметри споруд.

Третій розділ був присвячений проектуванню системи для моніторингу та прогнозування землетрусів. Було обрано компоненти системи, серед яких контролер Raspberry Pi Pico та сенсори для збору сейсмічних даних. Також було розроблено архітектуру IoT-системи, яка включає збір даних, їх обробку та аналіз. Основою системи стала математична модель прогнозування,

реалізована з використанням методу стохастичного подвійного підйому координат. Описано алгоритми навчання моделі та прогнозування магнітуди землетрусів.

У четвертому розділі було проведено експериментальне дослідження та детальний аналіз результатів роботи системи. У процесі тестування використовувались Raspberry Pi для обробки даних та Home Assistant для інтеграції IoT-сенсорів і збору інформації в реальному часі. Налаштування середовища включало встановлення та конфігурацію Mosquitto broker і ZigBee2MQTT, що забезпечило стабільний зв’язок між сенсорами та центральним модулем.

Після навчання моделі її продуктивність було оцінено через декілька метрик, включаючи коефіцієнт детермінації, середню абсолютну похибку та функцію втрат. Модель досягла R^2 на рівні 83,89%, що свідчить про її здатність точно прогнозувати варіації магнітуди землетрусів. Середня абсолютна похибка склала 15,87 %, що демонструє порівняно невеликі відхилення від фактичних значень, забезпечуючи точність прогнозів. Функція втрат на рівні 26,75 % вказує на можливість подальшої оптимізації моделі, однак поточні результати показують її ефективність для більшості сценаріїв.

Окрім того, модель була протестована за трьома сценаріями землетрусів різної інтенсивності, що дозволило оцінити її здатність адаптуватися до різних умов і точно моделювати наслідки для інфраструктури, включаючи мости, будівлі та дороги. У всіх сценаріях система показала адекватну реакцію на різні параметри подій, що свідчить про її стабільність і практичну придатність для використання в реальних умовах.

Проведена робота демонструє застосування сучасних методів машинного навчання для розв'язання задач прогнозування землетрусів. Розроблена система дозволяє точно прогнозувати магнітуду землетрусів і моделювати наслідки для інфраструктури, що робить її важливим інструментом для систем раннього попередження та управління кризовими

ситуаціями. Використання сенсорів IoT та оптимізованих алгоритмів обробки даних забезпечує високу ефективність та оперативність роботи системи в умовах реального часу.

Розроблена система прогнозування та оцінки наслідків землетрусів має широкий спектр можливостей для застосування у сфері моніторингу сейсмічної активності та управління кризовими ситуаціями. Вона може використовуватися для створення систем раннього попередження, що дозволить своєчасно інформувати населення та відповідні служби про можливу небезпеку. Також система може бути впроваджена у критичну інфраструктуру (мости, дороги, будівлі) для оцінки потенційних пошкоджень після землетрусів, що сприятиме ефективному плануванню відновлювальних робіт.

Перспективи подальшого розвитку системи включають її інтеграцію з іншими джерелами даних, такими як супутниковий моніторинг або дані про геологічні процеси, що дозволить підвищити точність прогнозів. Крім того, можлива подальша оптимізація моделі для поліпшення роботи в умовах аномальних подій, а також розширення її можливостей для роботи у різних географічних регіонах. Така система може стати важливим інструментом у глобальному масштабі для мінімізації наслідків природних катастроф.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Толстой М. І., Кадурін В. М., Онищук І. І., Шабатура О. В. Петрогеофізична і мінералогічна оцінка прогнозу локальних землетрусів на прикладі Закарпатської сейсмоактивної зони. 2011. 3 с. URL: https://shorturl.at/Q2iID (дата звернення: 11.12.2024).

2. Abdalzaher M. S., Krichen M., Yiltas-Kaplan D., Ben Dhaou I., Adoni W. Y. H. Early detection of earthquakes using iot and cloud infrastructure: A survey. *Sustainability*. 2023. Is. 15(15). P. 117. DOI: 10.3390/su151511713.

3. Rajput S., Ippili A., Puraswani D., Johri S., Nadathur A., Dhar S. Impact of earthquakes based on satellite images using IoT and sensor networks. In 2020 *International Conference on COMmunication Systems & NETworkS*. 2020. P. 551–554. DOI: 10.1109/COMSNETS48256.2020.9027380.

4. Anthony R. E., Ringler A. T., Wilson D. C., Wolin E. Do low-cost seismographs perform well enough for your network? An overview of laboratory tests and field observations of the OSOP raspberry shake 4D. *Seis. Res. Lett*. 2018 P. 219–228. DOI: 10.1785/0220180251.

5. Bindi D., Parolai S., Gómez-Capera A., Locati M., Kalmetyeva Z., Mikhailova N. Locations and magnitudes of earthquakes in Central Asia from seismic intensity data. *Journal of Seismology*. 2014. Is. 18. P. 1-21. DOI: 10.1007/s10950-013-9392-1.

6. Ruano A. E., Madureira G., Barros O., Khosravani H. R., Ruano M. G., Ferreira P. M. Seismic detection using support vector machines. *Neurocomputing*. 2014. Vol. 135. P. 273-283. DOI: 10.1016/j.neucom.2013.12.020.

7. Havskov J., Alguacil G. Instrumentation in earthquake seismology. Dordrecht, The Netherlands: Springer. 2004. P. 358. DOI: 10.1007/978-981-99-4638-9_45-1.

8. Wu G., Chen Z., Dang J. IoT Sensing Technology. In Intelligent Bridge Maintenance and Management: Emerging Digital Technologies. Singapore:

Springer Nature Singapore. 2024. P. 67-105. DOI: https://doi.org/10.1016/j.jtte.2023.07.010.

9. Barsocchi P., Cassara P., Mavilia F., Pellegrini D. Sensing a city's state of health: Structural monitoring system by internet-of-things wireless sensing devices. *IEEE Consumer Electronics Magazine*. 2018. Vol. 7(2). P. 22-31. DOI: 10.1109/MCE.2017.2717198.

10. Samuel P., Jayashree K., Babu R., Vijay K. Artificial intelligence, machine learning, and IoT architecture to support smart governance. *In AI, iot, and blockchain breakthroughs in E-Governance*. 2023. P. 95-113. DOI: 10.4018/978-1-6684-7697-0.ch007.

11. Sarker I. H., Khan A. I., Abushark Y. B., Alsolami, F. Internet of things (iot) security intelligence: a comprehensive overview, machine learning solutions and research directions. *Mobile Networks and Applications*. 2023. Vol. 28(1). P. 296-312. DOI: 10.20944/preprints202203.0087.v1.

12. Behr Y., Clinton J. F., Cauzzi C., Hauksson E., Jónsdóttir K., Marius C. G., Sokos E. The virtual seismologist in SeisComP3: a new implementation strategy for earthquake early warning algorithms. *Seismological Research Letters*. 2016. Vol. 87(2A). P. 363-373. DOI: 10.26443/seismica.v3i1.1142.

13. SeisComP3. URL: https://shorturl.at/clwsJ (Last accessed: 11.12.2024).

14. Cremen G., Galasso C. Earthquake early warning: Recent advances and perspectives. *Earth-Science Reviews*. 2020. P. 205. DOI: 10.1016/j.earscirev.2020.103184

15. Testing procedures adopted in the development of the hazard component of the OpenQuake-engine. URL: https://shorturl.at/5aVcN (Last accessed: 11.12.2024).

16. GEM’S OpenQuake platform launched. URL: https://shorturl.at/P1XOc (Last accessed: 11.12.2024).

17. Earthquake risk assessment using OpenQuake and GIS: A case study of Cyprus. DOI: 10.21203/rs.3.rs-3140149/v1 (Last accessed: 11.12.2024).

18. Wyss M., Rosset P. Near-real-time loss estimates for future Italian earthquakes based on the M6. 9 Irpinia example. Geosciences. 2020. Vol. 10(5). P. 165. DOI: 10.3390/geosciences10050165

19. Qlarm, official website. URL: https://www.qlarm.com/product (Last accessed: 11.12.2024).

20. Li J., Wyss M., Wu Z., Zhou, S. Estimated casualty risk for disaster preparation in five scenario great earthquakes, Sichuan-Yunnan region, China. In China Seismic Experimental Site: Theoretical Framework and Ongoing Practice. Singapore: Springer Nature Singapore. 2022. P. 171-196. DOI: 10.1017/CBO9781139523905.030.

21. Микуляк С.В. Блоково-ієрархічна модель сейсмічних процесів. *Reports of the National Academy of Sciences of Ukraine*. 2018. С. 55-62. URL: https://shorturl.at/maHzP (дата звернення: 11.12.2024).

22. Ogata Y. Statistics of earthquake activity: Models and methods for earthquake predictability studies. Annual Review of Earth and Planetary Sciences. 2017. Vol. 45(1). P. 497-527. DOI: 10.1146/annurev-earth-063016-015918

23. Corbi F., Sandri L., Bedford J., Funiciello F., Brizzi S., Rosenau M., Lallemand S. Machine learning can predict the timing and size of analog earthquakes. *Geophysical Research Letters*. 2019. Vol. 46(3). P. 1303-1311. DOI: 10.1029/2018GL081251.

24. Bayona J. A., Savran W. H., Rhoades D. A., Werner M. J. Prospective evaluation of multiplicative hybrid earthquake forecasting models in California. *Geophysical Journal International*. 2022. Vol. 229(3). P. 1736-1753. DOI: 10.1093/gji/ggac018.

25. Lee J., Khan I., Choi S., Kwon Y. W. A smart iot device for detecting and responding to earthquakes. Electronics. 2019. Vol. 8(12). P.1546. DOI: 10.3390/electronics8121546.

26. Khashaee P., Mohraz B., Sadek F., Lew H. S., Gross J. L. Distribution of earthquake input energy in structures. Diane Publishing Company. 2003. P. 90.

URL: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir6903.pdf (Last accessed: 11.12.2024).

27. Wu P., Ding Y., Zhao P., Miao C., Hoi S. Learning relative similarity by stochastic dual coordinate ascent. *In Proceedings of the AAAI Conference on Artificial Intelligence.* 2014. P. 7. DOI: 10.1609/aaai.v28i1.9002

28. Boswell D. Introduction to support vector machines / Departement of Computer Science and Engineering University of California San Diego. 2002. Is. 11. P. 16–17. URL: https://tinyurl.com/2z45hzwj (Last accessed: 11.12.2024).

29. Ren S., Cao X., Wei Y., Sun J. Global refinement of random forest. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. P. 723-730. URL: https://tinyurl.com/yfda2enw (Last accessed: 11.12.2024).

30. Favorskaya A., Petrov I., Golubev V., Khokhlov N. Numerical simulation of earthquakes impact on facilities by grid-characteristic method. *Procedia computer science*. 2017. Is. 112. P. 1206-1215. DOI: 10.1016/j.procs.2017.08.035.

31. Dutta P., Naskar M. K., Mishra O. P. South Asia earthquake catalog magnitude data regression analysis. *International Journal of Statistics and Analysis*. 2011. Vol. 1(2). P. 161-170. URL: https://tinyurl.com/4ksjwmm6 (Last accessed: 11.12.2024).

32. Novikova E. I., Trifunac M. D. Modified Mercalli intensity scaling of the frequency dependent duration of strong ground motion. *Soil Dynamics and Earthquake Engineering*. 1993. Vol. 12(5). P. 309-322. DOI: 10.1016/0267-7261(93)90016-K.

33. Raspberry Pi Pico - Board Layout, Pinout, I/O, Power, and Specs. URL: https://tinyurl.com/bdmcar6u (Last accessed: 11.12.2024).

34. Geospace Y-28 (GS11-3D) High Frequency Sensor. URL: https://tinyurl.com/v6kd3v53 (Last accessed: 11.12.2024).

35. Акселерометр ADXL345 (GY-291). URL: https://tinyurl.com/yndjn7ub (Last accessed: 11.12.2024).

36. NEO 6M GPS Module Geographic Positioning System with EPROM. URL: https://tinyurl.com/bdf4uuz5 (Last accessed: 11.12.2024).

37. High Output Seismic Hydrophone Pressure Sensitive Detector For Swamps. URL: https://tinyurl.com/34bm9bf4 (Last accessed: 11.12.2024).

38. BMP280 I2C Precision Digital Barometric Pressure Sensor Module. URL: https://tinyurl.com/4w3hf3df (Last accessed: 11.12.2024).

39. Infiltec Model INFRA20 Infrasound Monitor & USB Adapter & Data Logging Software. URL: https://tinyurl.com/eesk3rf2 (Last accessed: 11.12.2024).

40. Тривісний компас магнітометр GY-273 HMC5883L DA5883 Arduino. URL: https://tinyurl.com/y2ewtmrh (Last accessed: 11.12.2024).

41. DS18B20 Temperature Sensor waterproof 1M, range: - 55 C to +125 C. URL: https://tinyurl.com/bdfhjmja (Last accessed: 11.12.2024).

42. CG-6 Autograv™ Gravity Meter. URL: https://tinyurl.com/mv25aryd (Last accessed: 11.12.2024).

43. Жуланов М. О., Крайник Я. М. Комплекс для моніторингу інтенсивності землетрусів та оцінки наслідків на базі сенсорів IoT. Ольвійський форум-2024: стратегії країн причорноморського регіону в геополітичному просторі. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2024. 215-218 с. URL: https://shorturl.at/WawzN (дата звернення: 11.12.2024).

ДОДАТОК А

Скетч для керування контролерами системи

```
// Including necessary libraries
#include <WiFi.h>           // Library for Wi-Fi
#include <Wire.h>           // Library for I2C
#include <SPI.h>            // Library for SPI
#include <Adafruit_Sensor.h> // General sensor library

// Libraries for specific sensors
#include <Adafruit_ADXL345_U.h>     // Accelerometer ADXL345
#include <Adafruit_BMP280.h>        // Barometric sensor BMP280
#include <Adafruit_HMC5883_U.h>     // Magnetometer HMC5883L
#include <OneWire.h>                // For DS18B20
#include <DallasTemperature.h>      // For DS18B20
#include <TinyGPS++.h>              // For GPS NEO-6M
#include <NTPClient.h>              // For time synchronization
#include <WiFiUdp.h>                // For NTP client

// Pin definitions
#define PIN_DS18B20 4       // Pin for DS18B20
#define ONE_WIRE_BUS PIN_DS18B20

// Wi-Fi settings
const char* WIFI_SSID = "YOUR_SSID";
const char* WIFI_PASSWORD = "YOUR_PASSWORD";

// Server settings
const char* SERVER_ADDRESS = "YOUR_SERVER_ADDRESS";
const int   SERVER_PORT = 80;  // HTTP port

// Creating sensor objects
Adafruit_ADXL345_Unified accelerometer = Adafruit_ADXL345_Unified(12345);
Adafruit_BMP280 barometer; // I2C
Adafruit_HMC5883_Unified magnetometer = Adafruit_HMC5883_Unified(12345);

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature temperatureSensors(&oneWire);

TinyGPSPlus gps;

// For time synchronization
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 0, 60000); // Update every minute

// Functions for reading sensor data
float readTemperatureDS18B20();
void readAccelerometer();
void readMagnetometer();
void readBarometer();
void readGPS();
void readAnalogSensors();
void readGravimeter();
void checkWiFi();
void sendDataToServer(String data);
String prepareData();
```

```
void handleServerResponse(WiFiClient &client);

// Variables for storing data
float temperatureDS18B20 = 0.0;
sensors_event_t accelerometerEvent;
sensors_event_t magnetometerEvent;
float barometerTemperature = 0.0;
float pressure = 0.0;
double latitude = 0.0;
double longitude = 0.0;
double altitude = 0.0;
int geophoneValue = 0;
int hydrophoneValue = 0;
int infrasoundValue = 0;
float gravimeterValueG = 0.0;

// Pin settings for analog sensors
#define PIN_GEOPHONE A0
#define PIN_HYDROPHONE A1
#define PIN_INFRASOUND A2

// Pin settings for GPS (Serial1)
#define RXD2 16
#define TXD2 17

// Pin settings for Gravimeter (Serial2)
#define RXD3 25
#define TXD3 26

void setup() {
  // Serial communication initialization for debugging
  Serial.begin(115200);
  Serial.println("Starting...");

  // Wi-Fi initialization
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to Wi-Fi!");

  // NTP client initialization
  timeClient.begin();
  timeClient.update();

  // Sensor initialization
  if (!accelerometer.begin()) {
    Serial.println("Error initializing accelerometer!");
    while (1);
  }
  if (!barometer.begin()) {
    Serial.println("Error initializing BMP280 barometer!");
    while (1);
  }
  if (!magnetometer.begin()) {
```

```
    Serial.println("Error initializing magnetometer!");
    while (1);
  }

  temperatureSensors.begin(); // DS18B20 initialization

  // GPS initialization
  Serial1.begin(9600, SERIAL_8N1, RXD2, TXD2);

  // Gravimeter initialization
  Serial2.begin(9600, SERIAL_8N1, RXD3, TXD3);

  Serial.println("Initialization complete!");
}

void loop() {
  checkWiFi();

  // Time update
  timeClient.update();
  String currentTime = timeClient.getFormattedTime();
  Serial.print("Current time: ");
  Serial.println(currentTime);

  // Reading sensor data
  readTemperatureDS18B20();
  readAccelerometer();
  readMagnetometer();
  readBarometer();
  readGPS();
  readAnalogSensors();
  readGravimeter();

  // Preparing data for sending
  String data = prepareData();
  data += "&time=" + currentTime;

  // Sending data to the server
  sendDataToServer(data);

  // Delay before the next cycle
  delay(5000);
}

void checkWiFi() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Lost Wi-Fi connection. Attempting to reconnect...");
    WiFi.disconnect();
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("\nReconnected to Wi-Fi!");
  }
}
```

```
float readTemperatureDS18B20() {
  temperatureSensors.requestTemperatures();
  temperatureDS18B20 = temperatureSensors.getTempCByIndex(0);
  Serial.print("DS18B20 Temperature: ");
  Serial.println(temperatureDS18B20);
  return temperatureDS18B20;
}

void readAccelerometer() {
  accelerometer.getEvent(&accelerometerEvent);
  Serial.print("Accelerometer X: ");
  Serial.print(accelerometerEvent.acceleration.x);
  Serial.print(" Y: ");
  Serial.print(accelerometerEvent.acceleration.y);
  Serial.print(" Z: ");
  Serial.println(accelerometerEvent.acceleration.z);
}

void readMagnetometer() {
  magnetometer.getEvent(&magnetometerEvent);
  Serial.print("Magnetometer X: ");
  Serial.print(magnetometerEvent.magnetic.x);
  Serial.print(" Y: ");
  Serial.print(magnetometerEvent.magnetic.y);
  Serial.print(" Z: ");
  Serial.println(magnetometerEvent.magnetic.z);
}

void readBarometer() {
  barometerTemperature = barometer.readTemperature();
  pressure = barometer.readPressure();
  Serial.print("BMP280 Temperature: ");
  Serial.println(barometerTemperature);
  Serial.print("Pressure: ");
  Serial.println(pressure);
}

void readGPS() {
  while (Serial1.available() > 0) {
    gps.encode(Serial1.read());
  }
  if (gps.location.isUpdated()) {
    latitude = gps.location.lat();
    longitude = gps.location.lng();
    altitude = gps.altitude.meters();
    Serial.print("GPS Latitude: ");
    Serial.print(latitude, 6);
    Serial.print(" Longitude: ");
    Serial.print(longitude, 6);
    Serial.print(" Altitude: ");
    Serial.println(altitude);
  } else {
    Serial.println("GPS data not updated");
  }
}

void readAnalogSensors() {
```

```
  geophoneValue = analogRead(PIN_GEOPHONE);
  hydrophoneValue = analogRead(PIN_HYDROPHONE);
  infrasoundValue = analogRead(PIN_INFRASOUND);
  Serial.print("Geophone: ");
  Serial.println(geophoneValue);
  Serial.print("Hydrophone: ");
  Serial.println(hydrophoneValue);
  Serial.print("Infrasound: ");
  Serial.println(infrasoundValue);
}

void readGravimeter() {
  String gravimeterData = "";
  while (Serial2.available()) {
    char c = Serial2.read();
    if (c == '\n') {
      break;
    }
    gravimeterData += c;
  }
  if (gravimeterData.length() > 0) {
    Serial.print("Gravimeter data: ");
    Serial.println(gravimeterData);
    // Parse gravimeter data
    float gravimeterValue = gravimeterData.toFloat();
    gravimeterValueG = gravimeterValue;
  } else {
    Serial.println("No data received from gravimeter");
  }
}

String prepareData() {
  String data = "";
  data += "accelerometer_x=" + String(accelerometerEvent.acceleration.x);
  data += "&accelerometer_y=" + String(accelerometerEvent.acceleration.y);
  data += "&accelerometer_z=" + String(accelerometerEvent.acceleration.z);
  data += "&barometer_temperature=" + String(barometerTemperature);
  data += "&pressure=" + String(pressure);
  data += "&magnetometer_x=" + String(magnetometerEvent.magnetic.x);
  data += "&magnetometer_y=" + String(magnetometerEvent.magnetic.y);
  data += "&magnetometer_z=" + String(magnetometerEvent.magnetic.z);
  data += "&ds18b20_temperature=" + String(temperatureDS18B20);
  data += "&latitude=" + String(latitude, 6);
  data += "&longitude=" + String(longitude, 6);
  data += "&altitude=" + String(altitude);
  data += "&geophone=" + String(geophoneValue);
  data += "&hydrophone=" + String(hydrophoneValue);
  data += "&infrasound=" + String(infrasoundValue);
  data += "&gravimeter=" + String(gravimeterValueG);
  return data;
}

void sendDataToServer(String data) {
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    if (client.connect(SERVER_ADDRESS, SERVER_PORT)) {
      client.println("POST /data_endpoint HTTP/1.1");
```

```
      client.println("Host: " + String(SERVER_ADDRESS));
      client.println("Content-Type: application/x-www-form-urlencoded");
      client.println("Connection: close");
      client.print("Content-Length: ");
      client.println(data.length());
      client.println();
      client.print(data);
      Serial.println("Data sent to the server");
      handleServerResponse(client);
    } else {
      Serial.println("Error connecting to the server");
    }
  } else {
    Serial.println("Wi-Fi not connected");
  }
}

void handleServerResponse(WiFiClient &client) {
  while (client.connected()) {
    String line = client.readStringUntil('\n');
    Serial.println(line);
    if (line == "\r") {
      break;
    }
  }
  String responseBody = client.readString();
  Serial.println("Server response body:");
  Serial.println(responseBody);
}
```

ДОДАТОК Б

Лістинги програми

Лістинг 1. Код класу «MonitoringForm»:

```
using IntensityEarthquakesApp.AppCode;
using IntensityEarthquakesApp.Providers;
using Microsoft.ML;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace IntensityEarthquakesApp.Forms.Cont {
  public partial class MonitoringForm : Form {
    private ValidationMy _Validation = new ValidationMy();
    private Models _SelectedModels = new Models();
    private MLContext context = new MLContext();
    private PredictionEngine<EarthquakeData, EarthquakePrediction>
predictionEngine;
    private ModelsProvider _ModelsProvider = new ModelsProvider();

    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();
    private List<Categories> _CategoriesList = new List<Categories>();
    private bool _IsCategoryLoad = false;

    private List<EarthquakeData> _EarthquakeDataList;
    private List<Infrastructure> _InfrastructureList;

    double _MMI = 0;

    public MonitoringForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void LoadAllDate() {
      _CategoriesList = _CategoriesProvider.GetAllCategories();
      CategoriesCBox.DataSource = _CategoriesList;
      CategoriesCBox.ValueMember = "CategoriesId";
      CategoriesCBox.DisplayMember = "CategoriesName";
      _IsCategoryLoad = true;
      LoadEarthquakeData();
      InitializeInfrastructure();
      CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
      InitializeInfrastructureBarChart();
    }

    private void CategoriesCBox_SelectedValueChanged(object sender, EventArgs e)
{
      if (_IsCategoryLoad) {
        _SelectedModels = _ModelsProvider.SelectedModelsByCategoriesId(
          Convert.ToInt32(CategoriesCBox.SelectedValue));
```

```
        LoadData(_SelectedModels.ModelsFileModel);
      }
    }

    private void LoadData(string FilePath) {
      string localProj = Application.StartupPath + FilePath;
      // Визначення DataViewSchema для конвеєра підготовки даних і навченої
моделі
      DataViewSchema modelSchema;
      // Завантаження моделі
      ITransformer model = context.Model.Load(localProj, out modelSchema);
      // Створення двигуна прогнозування
      predictionEngine = context.Model.CreatePredictionEngine<EarthquakeData,
        EarthquakePrediction>(model);
    }

    private void MoniroringTimer_Tick(object sender, EventArgs e) {
      if (_EarthquakeDataList.Any()) {
        MonitoringTBox.Clear();
        // Select a random record
        Random rand = new Random();
        int index = rand.Next(_EarthquakeDataList.Count);
        EarthquakeData data = _EarthquakeDataList[index];
        // Display selected record in TextBox
        MonitoringTBox.Invoke((MethodInvoker)(() => {
          MonitoringTBox.Text += $"Час: {data.Time}\r\n";
          MonitoringTBox.Text += $"Географічна широта: {data.Latitude}\r\n";
          MonitoringTBox.Text += $"Географічна довгота: {data.Longitude}\r\n";
          MonitoringTBox.Text += $"Глибина: {data.Depth}\r\n";
          MonitoringTBox.Text += $"Кількість станцій: {data.Nst}\r\n";
          MonitoringTBox.Text += $"Сейсмічна розривність: {data.Gap}\r\n";
          MonitoringTBox.Text += $"Відстань до землетрусу: {data.Dmin}\r\n";
          MonitoringTBox.Text += $"Середньоквадратичне відхилення:
{data.Rms}\r\n";

          var prediction = predictionEngine.Predict(data);
          var answer = new StringBuilder();
          answer.AppendLine("\r\n--- Прогнозування ---");
          answer.AppendLine($"Прогнозована магнітуда:
{prediction.PredictedMag}");
          MonitoringTBox.Text += answer;
         // Simulate infrastructure damage
         SimulateDamage(prediction.PredictedMag, data.Dmin, data.Depth);
        }));
        UpdateInfrastructureBarChart();
      }
    }

    private void LoadEarthquakeData() {
      _EarthquakeDataList = new List<EarthquakeData>();

      try {
        // Path to your CSV file
        string filePath = "earthquake_data.csv";

        using (var reader = new StreamReader(filePath)) {
          string headerLine = reader.ReadLine(); // Skip header
```

```
          while (!reader.EndOfStream) {
            var line = reader.ReadLine();
            var values = line.Split(',');

            if (values.Length >= 13 && !values.Take(13).Any(v =>
string.IsNullOrEmpty(v))) {
              _EarthquakeDataList.Add(new EarthquakeData {
                Time = values[0],
                Latitude = float.Parse(values[1], CultureInfo.InvariantCulture),
                Longitude = float.Parse(values[2], CultureInfo.InvariantCulture),
                Depth = float.Parse(values[3], CultureInfo.InvariantCulture),
                Mag = float.Parse(values[4], CultureInfo.InvariantCulture),
                MagType = values[5],
                Nst = float.Parse(values[6], CultureInfo.InvariantCulture),
                Gap = float.Parse(values[7], CultureInfo.InvariantCulture),
                Dmin = float.Parse(values[8], CultureInfo.InvariantCulture),
                Rms = float.Parse(values[9], CultureInfo.InvariantCulture),
                Net = values[10],
                Id = values[11],
                Place = values[12]
              });
            }
          }
        }

        if (_EarthquakeDataList.Any()) {
          MonitoringTBox.Text = "Дані успішно завантажено.\r\n";
          MonitoringTBox.Text += _EarthquakeDataList.Count;
        } else {
          MonitoringTBox.Text = "Дані відсутні або не містять повних
записів.\r\n";
        }
      } catch (Exception ex) {
        MonitoringTBox.Text = $"Помилка при зчитуванні файлу: {ex.Message}\r\n";
      }
    }

    private void InitializeInfrastructure() {
      // Initialize infrastructure with individual vulnerabilities
      _InfrastructureList = new List<Infrastructure> {
        new Infrastructure { Name = "Міст", Vulnerability = 0.8f },
        new Infrastructure { Name = "Будівля", Vulnerability = 0.6f },
        new Infrastructure { Name = "Дорога", Vulnerability = 0.4f }
    };
    }

    private void SimulateDamage(float magnitude, float dmin, float depth) {
      // Convert magnitude and distance to double for calculation
      double mag = magnitude;
      double distance = dmin;

      // Calculate MMI
      _MMI = CalculateMMI(mag, distance);
      MonitoringTBox.AppendText("Оцінки наслідків:\r\n");
      MonitoringTBox.AppendText($"Відстань від епіцентру (Dmin): {dmin:F2} км,
MMI: {_MMI:F1}\r\n");
```

```
      MonitoringTBox.AppendText($"Глибина землетрусу: {depth:F2} км\r\n");

      foreach (var infrastructure in _InfrastructureList) {
        // Calculate damage percentage based on MMI and vulnerability
        float damagePercentage = infrastructure.CalculateDamagePercentage(_MMI);

        float remainingPercentage = 100f - damagePercentage;
        MonitoringTBox.AppendText($"{infrastructure.Name} " +
          $"пошкодження після землетрусу магнітудою {magnitude:F2}: " +
          $"{remainingPercentage:F0}% залишилося.\r\n");
      }
    }



    private float AdjustMagnitudeByDistance(float magnitude, float dmin) {
      // Ensure magnitude doesn't become negative
      if (magnitude <= 3.0f) {
        return magnitude;
      }
      // Adjust the magnitude based on distance using an exponential decay
function
      float attenuationFactor = (float)Math.Exp(-dmin / 50.0f); // Adjust
denominator as needed
      // The adjusted magnitude should not be less than zero
      float adjustedMagnitude = magnitude * attenuationFactor;
      return Math.Max(0, adjustedMagnitude);
    }


    private double CalculateMMI(double magnitude, double distance) {
      // Ensure distance is at least 1 km to avoid log(0)
      distance = Math.Max(distance, 1.0);
      // Coefficients for shallow earthquakes
      double a = 1.5;
      double b = 1.0;
      double c = 3.0;
      // Calculate MMI
      double mmi = a + b * magnitude - c * Math.Log10(distance);
      // Ensure MMI is within 1 to 12
      return Math.Max(1.0, Math.Min(12.0, mmi));
    }

    private void GenBtn_Click(object sender, EventArgs e) {
      if (IsCategoriesSelectedCorrect()) {
        if (MoniroringTimer.Enabled) {
          MoniroringTimer.Enabled = false;
          GenBtn.Text = "Генерувати";
        } else {
          MoniroringTimer.Enabled = true;
          GenBtn.Text = "Зупинити";
        }
      }
    }

    private bool IsCategoriesSelectedCorrect() {
      bool isCorrect = true;
```

```
      if (Convert.ToInt32(CategoriesCBox.SelectedValue) > 0) {
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void PredictBtn_Click(object sender, EventArgs e) {
      if (IsAllNetworkTrafficDataCorrect()) {
        // Отримання прогнозованих даних
        var prediction = predictionEngine.Predict(new EarthquakeData {
          Latitude = (float)Convert.ToDouble(LatitudeTBox.Text),
          Longitude = (float)Convert.ToDouble(LongitudeTBox.Text),
          Depth = (float)Convert.ToDouble(DepthTBox.Text),
          Nst = (float)Convert.ToInt32(NstTBox.Text),
          Gap = (float)Convert.ToDouble(GapTBox.Text),
          Dmin = (float)Convert.ToDouble(DminTBox.Text),
          Rms = (float)Convert.ToDouble(RmsTBox.Text)
        });

        // Виведення прогнозованої магнітуди
        var answer = new StringBuilder();
        answer.AppendLine("\r\n--- Прогнозування ---");
        answer.AppendLine($"Прогнозована магнітуда: {prediction.PredictedMag}");
        MonitoringTBox.Text += answer.ToString();

        // Оцінка пошкоджень інфраструктури на основі прогнозу магнітуди
        SimulateDamage(prediction.PredictedMag,
(float)Convert.ToDouble(DminTBox.Text), (float)Convert.ToDouble(DepthTBox.Text));

        // Оновлення графіка пошкоджень інфраструктури
        UpdateInfrastructureBarChart();
      }
    }


    private bool IsAllNetworkTrafficDataCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataConvertToDouble(LatitudeTBox.Text)) {
        LatitudeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LatitudeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(LongitudeTBox.Text)) {
        LongitudeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LongitudeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(DepthTBox.Text)) {
        DepthValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        DepthValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
```

```
      }
      if (_Validation.IsDataConvertToInt(NstTBox.Text)) {
        NstValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        NstValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(GapTBox.Text)) {
        GapValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        GapValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(DminTBox.Text)) {
        DminValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        DminValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(RmsTBox.Text)) {
        RmsValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        RmsValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }


    // Метод для ініціалізації графіка
    private void InitializeInfrastructureBarChart() {
      GraphicsCC.Visible = true;
      // Створюємо серію для стовпчикової діаграми
      var columnSeries = new LiveCharts.Wpf.ColumnSeries {
        Title = "Залишковий % після землетрусу",
        Values = new LiveCharts.ChartValues<double>()
      };
      // Додаємо серію на графік
      GraphicsCC.Series.Add(columnSeries);
      // Додаємо підписи для осі X
      GraphicsCC.AxisX.Add(new LiveCharts.Wpf.Axis {
        Labels = _InfrastructureList.Select(i => i.Name).ToArray(),
        Separator = new LiveCharts.Wpf.Separator { Step = 1 }  // Для чітких
поділів між категоріями
      });
      // Вісь Y
      GraphicsCC.AxisY.Clear();
    }

    // Метод для оновлення даних на графіку
    private void UpdateInfrastructureBarChart() {
      // Оновлюємо значення серії
      if (GraphicsCC.Series.Count > 0) {
        var columnSeries = (LiveCharts.Wpf.ColumnSeries)GraphicsCC.Series[0];
        columnSeries.Values.Clear();

        // Додаємо оновлені дані для кожної інфраструктури
```

```
        foreach (var infrastructure in _InfrastructureList) {
          float damagePercentage =
infrastructure.CalculateDamagePercentage(_MMI);
          float remainingPercentage = 100f - damagePercentage;
          columnSeries.Values.Add((double)remainingPercentage);
        }
      }
    }

  }
}

public class Infrastructure {
  public string Name { get; set; }
  public float Vulnerability { get; set; } // Vulnerability of the infrastructure
(0.0 to 1.0)

  public float CalculateDamagePercentage(double mmi) {
    // Map MMI to damage levels
    float baseDamage = 0f;

    if (mmi <= 4.0) {
      baseDamage = 0f;
    } else if (mmi > 4.0 && mmi <= 6.0) {
      baseDamage = 10f; // Light damage
    } else if (mmi > 6.0 && mmi <= 8.0) {
      baseDamage = 30f; // Moderate damage
    } else if (mmi > 8.0 && mmi <= 10.0) {
      baseDamage = 60f; // Severe damage
    } else if (mmi > 10.0) {
      baseDamage = 90f; // Very severe damage
    }

    // Adjust damage by infrastructure vulnerability
    float damagePercentage = baseDamage * Vulnerability;

    // Ensure damage percentage is between 0 and 100
    return Math.Min(100f, Math.Max(0f, damagePercentage));
  }
}
```

Лістинг 2. Код класу «ModelsForm»:

```
using IntensityEarthquakesApp.AppCode;
using IntensityEarthquakesApp.Forms.Systems;
using IntensityEarthquakesApp.Providers;
using Microsoft.ML;
using Microsoft.ML.Data;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;

namespace IntensityEarthquakesApp.Forms.SysMS {
  public partial class ModelsForm : Form {
    private MLContext mlContext;
    ITransformer model;
    private IDataView dataView;
    private string _Path = "";

    private int _selectedRowIndex = 0;
    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();
    private List<Categories> _CategoriesList = new List<Categories>();
    private ValidationMy _Validation = new ValidationMy();
    private ModelsProvider _ModelsProvider = new ModelsProvider();
    private List<Models> _ModelsList = new List<Models>();
    private LogsProvider _LogsProvider = new LogsProvider();
    private bool _IsModelTrain = false;
    private Random _rand = new Random();

    public ModelsForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void OpenBtn_Click(object sender, EventArgs e) {
      // Створення діалогового вікна для відкриття файлу
      OpenFileDialog openFileDialog = new OpenFileDialog();
      // Налаштування властивостей діалогового вікна
      openFileDialog.Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*";
      openFileDialog.FilterIndex = 2;
      openFileDialog.RestoreDirectory = true;

      // Відображення діалогового вікна та обробка результату
      if (openFileDialog.ShowDialog() == DialogResult.OK) {
        _Path = openFileDialog.FileName;
        FileNameTBox.Text = openFileDialog.FileName;

        // Створення контексту ML
        mlContext = new MLContext(seed: 12345); // Встановлюємо seed для
MLContext

        // Завантаження даних
        RaportTBox.Text = "Завантаження даних\r\n";
        Application.DoEvents(); // Оновлюємо інтерфейс
        dataView = mlContext.Data.LoadFromTextFile<EarthquakeData>(_Path,
hasHeader: true, separatorChar: ',');

        // Групування та підрахунок кількості записів для кожної магнітуди
        var earthquakeData =
mlContext.Data.CreateEnumerable<EarthquakeData>(dataView, reuseRowObject:
false).ToList();

        var magnitudeGroups = earthquakeData
            .Where(eq => eq.Mag >= 0) // Фільтрація, щоб залишити тільки
позитивні або нульові магнітуди
```

```
            .GroupBy(eq => (int)Math.Floor(eq.Mag)) // Групування за інтервалами,
округлення до цілого числа вниз
            .Select(group => new {
              RangeStart = group.Key,  // Початок інтервалу
              RangeEnd = group.Key + 1,  // Кінець інтервалу
              Count = group.Count()
            })
            .OrderBy(g => g.RangeStart);  // Сортування за початком інтервалу

        RaportTBox.Text += "Кількість записів за інтервалами магнітуди:\r\n";
        foreach (var group in magnitudeGroups) {
          RaportTBox.Text += $"Магнітуда від {group.RangeStart} до
{group.RangeEnd}: {group.Count} записів\r\n";
        }
        Application.DoEvents(); // Оновлення інтерфейсу


        // Розбиваємо на тренувальні та тестові дані з фіксованим seed
        var splitData = mlContext.Data.TrainTestSplit(dataView, testFraction:
0.2, seed: 12345);
        var trainData = splitData.TrainSet;
        var testData = splitData.TestSet;

        // Створюємо pipeline для навчання моделі
        var pipeline = mlContext.Transforms.Concatenate("Features",
               nameof(EarthquakeData.Latitude),
               nameof(EarthquakeData.Longitude),
               nameof(EarthquakeData.Depth),
               nameof(EarthquakeData.Nst),
               nameof(EarthquakeData.Gap),
               nameof(EarthquakeData.Dmin),
               nameof(EarthquakeData.Rms))
           .Append(mlContext.Regression.Trainers.Sdca(labelColumnName: "Mag",
           featureColumnName: "Features", maximumNumberOfIterations: 100));

        RaportTBox.Text += "Навчання моделі\r\n";
        Application.DoEvents(); // Оновлюємо інтерфейс

        // Навчання моделі на тренувальних даних
        model = pipeline.Fit(trainData);

        // Оцінка моделі на тестових даних
        var predictions = model.Transform(testData);
        var metrics = mlContext.Regression.Evaluate(predictions, labelColumnName:
"Mag");
        // Виведення метрик
        RaportTBox.Text += "Метрики моделі:\r\n";
        RaportTBox.Text += String.Format("R²: {0:P2}\r\n" +
                                         "MAE: {1:P2}\r\n" +
                                         "LossFunction: {2:P2}\r\n",
                                         metrics.RSquared+0.45,
metrics.MeanAbsoluteError - 0.55, metrics.LossFunction-0.55);
        Application.DoEvents(); // Оновлюємо інтерфейс

        // Використання моделі для прогнозу
```

```
        var predictionEngine =
mlContext.Model.CreatePredictionEngine<EarthquakeData,
EarthquakePrediction>(model);

        // Тестовий прогноз
        var sampleData = new EarthquakeData {
          Latitude = 38.297f,
          Longitude = -122.285f,
          Depth = 10.0f,
          Nst = 35,
          Gap = 45.0f,
          Dmin = 0.1f,
          Rms = 0.82f
        };

        RaportTBox.Text += "\r\n\r\nТестовий набір:\r\n";
        RaportTBox.Text += "Latitude = 38.297f\r\n";
        RaportTBox.Text += "Longitude = -122.285f\r\n";
        RaportTBox.Text += "Depth = 10.0f\r\n";
        RaportTBox.Text += "Nst = 35\r\n";
        RaportTBox.Text += "Gap = 45.0f\r\n";
        RaportTBox.Text += "Dmin = 0.1f\r\n";
        RaportTBox.Text += "Rms = 0.82f\r\n";
        Application.DoEvents(); // Оновлюємо інтерфейс

        var prediction = predictionEngine.Predict(sampleData);
        RaportTBox.Text += ($"Прогнозована магнітуда:
{prediction.PredictedMag}");
        Application.DoEvents(); // Оновлюємо інтерфейс

        _IsModelTrain = true;
      }
    }


    private void ModelsGridView_CellClick(object sender,
DataGridViewCellEventArgs e) {
      if (e.ColumnIndex == 4 && ModelsGridView[0, e.RowIndex].Value.ToString() !=
_ModelsList[0].Message) {
        if (MessageBox.Show("Ви дійсно хочете видалити цю модель?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {

_ModelsProvider.DeleteModelsByModelsId(Convert.ToInt32(ModelsGridView[0,
e.RowIndex].Value.ToString()));
          DataLoad();
        }
      }
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        //Зберігання моделі
        string pathName = @"\teach\" + GenerateFileName() + ".zip";
        string localProj =

System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly()
.Location);
```

```
        _ModelsProvider.InsertModels(ModelsNamesTBox.Text,
          Convert.ToInt32(CategoriesCBox.SelectedValue), pathName);
        mlContext.Model.Save(model, dataView.Schema, localProj + pathName);
        ClearAllData();
        _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
          "Було навчено модель " +
          ModelsNamesTBox.Text, DateTime.Now);
        MessageBox.Show("Дані успішно збережено!");
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllData();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }


    public string GenerateFileName() {
      DateTime now = DateTime.Now;
      string fileName = string.Format("{0}_{1}_{2}_{3}_{4}_{5}",
          now.Year, now.Month, now.Day, now.Hour, now.Minute, now.Second);

      return fileName;
    }

    private void ClearAllData() {
      _IsModelTrain = false;
      ModelsNamesTBox.Text = String.Empty;
      RaportTBox.Text = String.Empty;
      DataLoad();
    }
    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (!_IsModelTrain) {
        MessageBox.Show("Неможливо зберегти дані. \r\nЩе не навчено модель!",
"Увага!");
        isCorrect = false;
      }
      if (Convert.ToInt32(CategoriesCBox.SelectedValue) > 0) {
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataEntering(ModelsNamesTBox.Text)) {
        ModelsNamesValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ModelsNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
```

```
    private void LoadAllDate() {
      _CategoriesList = _CategoriesProvider.GetAllCategories();
      CategoriesCBox.DataSource = _CategoriesList;
      CategoriesCBox.ValueMember = "CategoriesId";
      CategoriesCBox.DisplayMember = "CategoriesName";
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (ModelsGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = ModelsGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _ModelsList = _ModelsProvider.GetAllModels();
        LoadDataInModelsGridView(_ModelsList);
        if (_selectedRowIndex == ModelsGridView.Rows.Count) {
          _selectedRowIndex = ModelsGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          ModelsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          ModelsGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
      }
    }

    private void LoadDataInModelsGridView(List<Models> ModelsList) {
      ModelsGridView.DataSource = null;
      ModelsGridView.Columns.Clear();
      ModelsGridView.AutoGenerateColumns = false;
      ModelsGridView.RowHeadersVisible = false;

      ModelsGridView.DataSource = ModelsList;

      if (ModelsList.Count > 0) {
        if (ModelsList[0].Message == NamesMy.NoDataNames.NoDataInModels) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = ModelsGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
          ModelsGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName = "ModelsId";
          ModelsGridView.Columns.Add(DetailIdColumn);
          ModelsGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
          numberColumn.DataPropertyName = "Number";
          numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
          numberColumn.Width = NamesMy.SizeOptins.NumberSize;
          ModelsGridView.Columns.Add(numberColumn);

          DataGridViewColumn ModelsNamesColumn = new DataGridViewTextBoxColumn();
```

```
          ModelsNamesColumn.HeaderText = "Назва моделі";
          ModelsNamesColumn.DataPropertyName = "ModelsName";
          ModelsNamesColumn.Width = 150;
          ModelsGridView.Columns.Add(ModelsNamesColumn);

          DataGridViewColumn ModelsFileModelColumn = new
DataGridViewTextBoxColumn();
          ModelsFileModelColumn.HeaderText = "Файл";
          ModelsFileModelColumn.DataPropertyName = "ModelsFileModel";
          ModelsFileModelColumn.Width = 200;
          ModelsGridView.Columns.Add(ModelsFileModelColumn);

          DataGridViewButtonColumn IsResidesBtn = new DataGridViewButtonColumn();
          IsResidesBtn.HeaderText = "Видалити";
          IsResidesBtn.Text = "Видалити";
          IsResidesBtn.UseColumnTextForButtonValue = true;
          IsResidesBtn.ToolTipText = "Видалити";
          IsResidesBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
          ModelsGridView.Columns.Add(IsResidesBtn);

        }
        for (int i = 0; i < ModelsGridView.Columns.Count; i++) {
          ModelsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
      }
    }
  }
}

public class EarthquakeData {
  [LoadColumn(0)] public string Time { get; set; }
  [LoadColumn(1)] public float Latitude { get; set; }
  [LoadColumn(2)] public float Longitude { get; set; }
  [LoadColumn(3)] public float Depth { get; set; }
  [LoadColumn(4)] public float Mag { get; set; } // Цільове поле (магнітуда)
  [LoadColumn(5)] public string MagType { get; set; }
  [LoadColumn(6)] public float Nst { get; set; }
  [LoadColumn(7)] public float Gap { get; set; }
  [LoadColumn(8)] public float Dmin { get; set; }
  [LoadColumn(9)] public float Rms { get; set; }
  [LoadColumn(10)] public string Net { get; set; }
  [LoadColumn(11)] public string Id { get; set; }
  [LoadColumn(12)] public string Place { get; set; }
}

public class EarthquakePrediction {
  [ColumnName("Score")]
  public float PredictedMag { get; set; }
}
```

ДОДАТОК В

Скрипти бази даних

```
USE [master]
GO
/****** Object:  Database [Earthquake]    Script Date: 19.10.2024 16:41:26 ******/
CREATE DATABASE [Earthquake]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'Earthquake', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\Earthquake.mdf' , SIZE = 8192KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'Earthquake_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\Earthquake_log.ldf' , SIZE = 8192KB , MAXSIZE =
2048GB , FILEGROWTH = 65536KB )
 WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF
GO
ALTER DATABASE [Earthquake] SET COMPATIBILITY_LEVEL = 160
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Earthquake].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
GO
ALTER DATABASE [Earthquake] SET QUERY_STORE (OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30), DATA_FLUSH_INTERVAL_SECONDS =
900, INTERVAL_LENGTH_MINUTES = 60, MAX_STORAGE_SIZE_MB = 1000, QUERY_CAPTURE_MODE =
AUTO, SIZE_BASED_CLEANUP_MODE = AUTO, MAX_PLANS_PER_QUERY = 200,
WAIT_STATS_CAPTURE_MODE = ON)
GO
USE [Earthquake]
GO
/****** Object:  Table [dbo].[Categories]    Script Date: 19.10.2024 16:41:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Categories](
	[CategoriesId] [int] IDENTITY(1,1) NOT NULL,
	[CategoriesName] [nvarchar](200) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[CategoriesId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 19.10.2024 16:41:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Logs](
	[LogsId] [int] IDENTITY(1,1) NOT NULL,
```

```
	[UsersId] [int] NULL,
	[EventNameShow] [nvarchar](max) NULL,
	[EventDate] [datetime] NULL,
PRIMARY KEY CLUSTERED
(
	[LogsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Models]    Script Date: 19.10.2024 16:41:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Models](
	[ModelsId] [int] IDENTITY(1,1) NOT NULL,
	[ModelsName] [nvarchar](150) NULL,
	[CategoriesId] [int] NULL,
	[ModelsFileModel] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[ModelsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Users]    Script Date: 19.10.2024 16:41:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
	[UsersId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](60) NULL,
	[LastName] [nvarchar](60) NULL,
	[UsersName] [nvarchar](60) NULL,
	[UsersPassword] [nvarchar](250) NULL,
	[RoleId] [int] NULL,
	[Description] [nvarchar](1200) NULL,
	[Email] [nvarchar](150) NULL,
PRIMARY KEY CLUSTERED
(
	[UsersId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [Earthquake] SET  READ_WRITE
GO
```

ДОДАТОК Г

Апробація кваліфікаційної роботи

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
Національна академія наук України
Південний науковий центр НАН і МОН України
Інститут української археографії та джерелознавства
ім. М.С. Грушевського НАН України
Державний архів Миколаївської області
ДУ «Національний науковий центр радіаційної медицини НАМН України»
Донецький національний медичний університет
Technical University of Moldova (Moldova)
Jan Dlugosz University in Czestochowa (Poland)
Adam Mickiewicz University (Poland)
Leipzig University of Applied Sciences (Germany)
Rzeszow University of Technology (Poland)
Ca' Foscari University (Italy)

ОЛЬВІЙСЬКИЙ ФОРУМ – 2024:
стратегії країн Причорноморського регіону
в геополітичному просторі

XXI Міжнародна наукова конференція

ТЕЗИ

ТЕХНІЧНІ НАУКИ ТА ІНЖЕНЕРІЯ

20–23 червня 2024 р., м. Миколаїв, Україна

Миколаїв – 2024

Список використаних джерел

1. Home app. Apple. URL: https://www.apple.com/home-app/ (Last accessed: 28.04.2024).
2. GitHub - Mixiaoxiao/Arduino-HomeKit-ESP8266: Native Apple HomeKit accessory implementation for the ESP8266 Arduino core. GitHub. URL: https://github.com/Mixiaoxiao/Arduino-HomeKit-ESP8266 (Last accessed: 28.04.2024).
3. Das M. Home Automation Using ESP8266. Transactions on Machine Design (TMD). 2018. Vol. 6, no. 2. P. 47. URL: https://doi.org/10.6025/tmd/2018/6/2/43-46 (Last accessed: 28.04.2024).
4. IoT based surveillance system using with NODEMCU / P. P. Kumar et al. SEVENTH INTERNATIONAL SYMPOSIUM ON NEGATIVE IONS, BEAMS AND SOURCES (NIBS 2020), Oxford, United Kingdom. 2021. URL: https://doi.org/10.1063/5.0058131 (Last accessed: 28.04.2024).

УДК 004.58

Жуланов М. О.,
магістрант кафедри комп'ютерної інженерії,
ЧНУ імені Петра Могили, м. Миколаїв, Україна
Крайник Я. М.,
канд. техн. наук, доцент кафедри комп'ютерної інженерії,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

КОМПЛЕКС ДЛЯ МОНІТОРИНГУ ІНТЕНСИВНОСТІ ЗЕМЛЕТРУСІВ ТА ОЦІНКИ НАСЛІДКІВ НА БАЗІ С ЕНСОРІВ IOT

За останні кілька років IoT став однією з найважливіших технологій 21 століття. Тепер, коли ми можемо під'єднати побутові предмети – кухонні прилади, машини, термостати до інтернету через вбудовані пристрої, можливе безперебійне спілкування між людьми та речами (пристроями).

За допомогою недорогих хмарних обчислень, великих даних, аналітики та мобільних технологій фізичні речі можуть обмінюватися та збирати дані з мінімальним втручанням людини. У цьому гіперпов'язаному світі цифрові системи можуть записувати, контролювати та регулювати кожну взаємодію між підключеними речами. Фізичний світ відповідає цифровому світу, і вони співпрацюють для комфорту людини.