ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ДОСЛІДЖЕННЯ ТА АНАЛІЗ МЕТОДІВ РОЗРОБКИ СИСТЕМИ РОЗУМНОГО БУДИНКУ З ВИКОРИСТАННЯМ ПЕРЕДОВИХ ТЕХНОЛОГІЙ»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Денис ГРЕБНЄВ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти Денис ГРЕБНЄВ
група ІСДМ-62

Керівник: Андрій АРОНОВ
науковий ступінь, вчене звання доцент кафедри ТЦР

Рецензент:
науковий ступінь, вчене звання Віктор ВИШНІВСЬКИЙ
Ім'я, ПРІЗВИЩЕ

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС

_____ Каміла СТОРЧАК
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гребнєву Денису Валерійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Дослідження та аналіз методів розробки систем «Розумного будинку» з використанням передових технологій

керівник кваліфікаційної роботи Андрій АРОНОВ, доцент кафедри ТЦР,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: методи безпеки та аналіз загроз, методи машинного навчання, література з IoT, безпеки тощо.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз існуючих теоретичних принципів та інноваційних методів безпеки в системах «Розумний будинок»
Проектування математичної моделі системи та розробка алгоритмів для ідентифікації та класифікації потенційних загроз

Реалізація системи на основі обраних технологій
Експериментальне дослідження та оцінка ефективності розробленої системи

5. Перелік графічного матеріалу: *презентація*
   1. Концепція розумного будинку
   2. Основні компоненти інтелектуальних систем
   3. Вибір платформи розробки та мови програмування
   4. Порівняльний аналіз методів машинного навчання
   5. Схема роботи алгоритму Бройдена-Флетчера-Гольдфарба-Шанно
   6. Алгоритм навчання та зберігання даних моделі
   7. Алгоритм роботи моделі
   8. Вибір комплектуючих для системи «Розумний будинок»
   9. Мукспондер РМ С1008
   10. Класи для навчання моделі та прогнозування вторгнень
   11. Сценарії атак та їх симуляції

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	20.09.2023 - 30.09.2023	
2	Аналіз методів машинного навчання	01.10.2023 - 10.10.2023	
3	Формалізація математичної моделі системи	11.10.2023 - 21.10.2023	
4	Розробка алгоритмів та вибір компонентів	22.10.2023 - 01.11.2023	
5	Майбутнє інфокомунікаційних мереж	02.11.2023 - 12.11.2023	
6	Розробка системи	13.11.2023 - 18.11.2023	
7	Експериментальне дослідження та оцінка ефективності системи	19.11.2023 - 24.11.2023	
8	Вступ, висновки, реферат	25.11.2023 - 29.11.2023	
9	Розробка демонстраціного матеріалу	30.11.2023	

Здобувач вищої освіти _____ Денис ГРЕБНЄВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій АРОНОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина магістерської роботи 84 с., 51 рис., 14 табл., 31 джерело.

Об`єкт дослідження – система «Розумний будинок», як комплексна технологічна структура, що інтегрує різноманітні пристрої та сервіси для автоматизації домашніх процесів.

Предмет дослідження – проектування, розробки, та імплементації системи «Розумний будинок», зокрема методи забезпечення безпеки, ефективності та сумісності у рамках інтеграції з сучасними технологіями IoT.

Мета дослідження – розробка оптимізованої та безпечної системи «Розумний будинок», яка ефективно інтегрується з сучасними технологіями IoT та забезпечує безпеку для користувачів.

Методи дослідження – аналітичний метод, метод моделювання, експериментальний метод, метод машинного навчання.

У роботі було проведено всебічний аналіз та розробка системи «Розумний будинок», яка є актуальною у контексті сучасних технологій інтелектуального житла. Робота охоплює теоретичні основи розумних будинків, включаючи їх історію, концепцію, функціональність та можливості. Значна увага приділяється інноваційним методам безпеки, включаючи системи розумного відеоспостереження та інтелектуальні замки, що забезпечують високий рівень захисту.

Галузь використання результатів дослідження охоплює сферу інтелектуального житла, комфорту та безпеки в житлових та комерційних приміщеннях, а також може бути використана для подальшого розвитку технологій IoT та систем автоматизації.

АНАЛІЗ ЗАГРОЗ, БЕЗПЕКА ІНФОРМАЦІЇ, ВТОРГНЕННЯ, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, МАШИННЕ НАВЧАННЯ, МОДЕЛЬ МАШИННОГО НАВЧАННЯ, РОЗУМНИЙ БУДИНОК, СИСТЕМИ АВТОМАТИЗАЦІЇ, C#, MS SQL SERVER.

The text part of the master's thesis is 84 pages, 51 figures, 14 tables, 31 sources.

The object of research is the "Smart House" system, as a complex technological structure that integrates various devices and services for automating home processes.

The subject of research is the design, development, and implementation of the "Smart House" system, in particular methods of ensuring safety, efficiency, and compatibility within the framework of integration with modern IoT technologies.

The purpose of the research is to develop an optimized and safe "Smart House" system that effectively integrates with modern IoT technologies and ensures safety for users.

Research methods – analytical method, modeling method, experimental method, machine learning method.

In the work, a comprehensive analysis and development of the "Smart House" system, which is relevant in the context of modern technologies of intelligent housing, was carried out. The work covers the theoretical foundations of smart homes, including their history, concept, functionality and capabilities. Considerable attention is paid to innovative security methods, including smart video surveillance systems and intelligent locks that provide a high level of protection.

The field of use of the research results covers the field of intelligent housing, comfort and security in residential and commercial premises, and can also be used for the further development of IoT technologies and automation systems.

THREAT ANALYSIS, INFORMATION SECURITY, INTRUSION, INTELLIGENT SYSTEMS, MACHINE LEARNING, MACHINE LEARNING MODEL, SMART HOME, AUTOMATION SYSTEMS, C#, MS SQL SERVER.

ЗМІСТ

ВСТУП

Проектування системи розумного будинку є важливою та актуальною темою в сучасному світі. Ця область займається створенням автоматизованих систем, що забезпечують підвищення комфорту, енергоефективності та безпеки в житлових приміщеннях. Враховуючи стрімке зростання технологій IoT та посилення уваги до екологічності та раціонального використання ресурсів, ця тема набуває особливої значущості. Науковці по всьому світу активно працюють над розвитком концепцій та технологій для розумних будинків. Вони досліджують широкий спектр питань, включаючи автоматизацію дому, енергоефективність, інтеграцію з відновлюваними джерелами енергії, безпеку даних та користувацький досвід. Наприклад, роботи в цій сфері включають розробку алгоритмів для інтелектуального управління освітленням, опаленням та охолодженням, що дозволяє зменшити енергетичні витрати.

Також вчені працюють над розробкою безпечних способів передачі та зберігання даних, оскільки приватність є критично важливою у сфері IoT. Безпека передачі даних є ключовою складовою в проектуванні систем розумного будинку. Вчені працюють над розробкою надійних методів шифрування та автентифікації для захисту від несанкціонованого доступу та атак. Важливим є також створення захищених протоколів для IoT пристроїв, що запобігають витоку особистих даних та забезпечують цілісність інформації. Розробка систем, які можуть виявляти та реагувати на потенційні загрози в режимі реального часу, є одним з основних напрямків сучасних досліджень у цій області.

Мета дослідження - розробка оптимізованої та безпечної системи «Розумний будинок», яка ефективно інтегрується з сучасними технологіями IoT та забезпечує високий рівень комфорту та безпеки для користувачів. Основні задачі дослідження:

– аналіз існуючих теоретичних принципів та інноваційних методів безпеки в системах «Розумний будинок», з акцентом на ефективність і захищеність;

– проектування математичної моделі системи та розробка алгоритмів для ідентифікації та класифікації потенційних загроз;

– реалізація системи на основі обраних технологій, включаючи її конфігурацію та налаштування;

– експериментальне дослідження та оцінка ефективності розробленої системи, включаючи налаштування експериментального середовища та аналіз отриманих результатів.

Об'єкт дослідження – система «Розумний будинок», як комплексна технологічна структура, що інтегрує різноманітні пристрої та сервіси для автоматизації домашніх процесів.

Предмет дослідження – проектування, розробки, та імплементації системи «Розумний будинок», зокрема методи забезпечення безпеки, ефективності та інтероперабельності у рамках інтеграції з сучасними технологіями IoT.

Методи дослідження для проектування системи «Розумний будинок» включають:

– аналітичний метод. Використання для збору та аналізу теоретичних даних про існуючі системи та технології;

– моделювання. Розробка математичних та комп'ютерних моделей для вивчення та вдосконалення системи;

– експериментальний метод. Проведення практичних випробувань розроблених компонентів та системи в цілому;

– метод машинного навчання. Застосування алгоритмів машинного навчання для аналізу даних та виявлення потенційних загроз;

– аналіз ризиків. Оцінка потенційних ризиків та їх впливу на безпеку та ефективність системи.

Наукова новизна результатів магістерської роботи з проектування системи «Розумний будинок» включає:

– реалізацію модулю для виявлення вторгнень у реальному часі, заснованої на алгоритмі Бройдена – Флетчера – Гольдфарба – Шанно, що дозволяє максимізувати ентропію для точного і швидкого реагування на потенційні загрози;

– розробка комплексної системи інтеграції різноманітних IoT пристроїв, забезпечуючи високий рівень інтероперабельності та адаптивності.

Практичне значення одержаних результатів магістерської роботи включає:

– покращення безпеки систем «Розумний будинок» за рахунок використання передових методів виявлення вторгнень у реальному часі, знижуючи ризик несанкціонованих дій та забезпечуючи захист даних користувачів;

– оптимізація управління домашніми пристроями, що сприятиме зниженню енергоспоживання та підвищенню комфорту користувачів;

– надання основи для подальших досліджень та розробок у сфері IoT та систем «Розумний будинок», стимулюючи інноваційний розвиток в цих галузях.

1 ОСНОВНІ ТЕОРЕТИЧНІ ПРИНЦИПИ ТА АНАЛІЗ СИСТЕМИ «РОЗУМНИЙ БУДИНОК»

1.1 Історія та концепція розумних будинків

Історія розумних будинків починається з середини 20-го століття, коли були розроблені перші автоматизовані домашні системи. Значний розвиток відбувся в кінці 1990-х з появою Інтернету речей (IoT). Технологічний прогрес дозволив інтегрувати домашні прилади з цифровими мережами, роблячи їх «розумними». З того часу концепція «розумного будинку» стала означати будинок, оснащений автоматизованими системами для управління освітленням, температурою, безпекою, медіа та іншими домашніми функціями через централізоване управління або навіть за допомогою голосових команд.

Концепція розумного будинку передбачає створення житлового простору, що використовує автоматизовані системи та різноманітні технології для підвищення комфорту, енергоефективності, безпеки та зручності. Розумні будинки інтегрують пристрої та системи, які можна контролювати дистанційно та автоматизовано, використовуючи інтернет. Це включає управління освітленням, опаленням, вентиляцією, електронними приладами, системами безпеки та іншими функціями будинку. Центральним елементом таких систем є здатність до адаптації, навчання відповідно до потреб мешканців та використання даних для підвищення ефективності роботи систем [1].

Технологічні основи розумних будинків базуються на використанні Інтернету речей (IoT), який дозволяє підключати різноманітні домашні пристрої до мережі. Ці пристрої можуть включати освітлення, системи опалення та охолодження, безпекові системи, а також побутові прилади. Вони оснащені датчиками, які збирають дані про стан навколишнього середовища та активності користувачів. Ця інформація обробляється за допомогою централізованих контролерів або хмарних сервісів, які автоматично регулюють роботу систем відповідно до заданих параметрів або користувацьких вподобань. Це створює зручне та ефективне житлове середовище.

Технологічні засади розумного будинку є складним та багатогранним об'єктом дослідження, який передбачає інтеграцію різноманітних інформаційних, обчислювальних та мережевих ресурсів з метою створення спрощеної та ефективної системи управління домашнім середовищем. З метою виокремлення ключових аспектів технологічних основ розумного будинку можна розглянути такі складові:

– інтернет речей (IoT). Це фундаментальний елемент, що стоїть в основі розумного будинку. IoT дозволяє підключати до мережі різні пристрої, такі як датчики, актуатори, побутові пристрої тощо, для обміну даними та взаємодії між ними. Цей принцип сприяє створенню «розумного» середовища, де пристрої можуть автоматично реагувати на зміни та виконувати задачі на основі отриманих даних;

– датчики. Датчики є важливими компонентами системи розумного будинку. Вони здатні збирати різноманітну інформацію про довкілля, таку як температура, вологість, рух, рівень освітленості, а також інші параметри. Ці дані стають основою для прийняття рішень та автоматизованого управління різними аспектами домашнього життя;

– централізоване управління. Системи управління розумним будинком інтегрують інформацію, зібрану від датчиків, та забезпечують можливість централізованого контролю за різними пристроями та системами, які включені до системи. Це дозволяє користувачам ефективно керувати освітленням, опаленням, кондиціонуванням повітря, безпековими системами та іншими аспектами дому через одне інтерфейсне рішення;

– автоматизація та адаптивність. Розумний будинок може бути налаштованим на автоматичне реагування на зміни у навколишньому середовищі або на зміни у поведінці користувачів. Наприклад, він може самостійно регулювати температуру в приміщенні відповідно до погодних умов або вимог користувача. Ця автоматизація спрощує життя мешканців і сприяє ефективному використанню ресурсів [2].

Перелічені елементи представлені на рис. 1.1, який ілюструє, як різні компоненти системи «Розумний будинок» взаємодіють між собою для створення комфортного, безпечного та енергоефективного домашнього середовища.

Рисунок 1.1 – Концепція розумного будинку

Функціональність та можливості розумного будинку включають в себе широкий спектр інтелектуальних функцій та автоматизованих опцій, які значно поліпшують комфорт, безпеку та зручність життя мешканців. Основні аспекти цих можливостей включають:

– автоматизація освітлення. Розумний будинок дозволяє налаштовувати графіки освітлення в різних частинах будинку відповідно до потреб і побажань користувачів. Ви можете налаштовувати режими освітлення для різних сценаріїв, таких як робочий час, релакс або романтична вечеря. Датчики руху можуть

виявляти присутність людей та включати/вимикати освітлення автоматично, що сприяє ефективному використанню енергії;

– автоматизація опалення та кондиціонування повітря. Розумний будинок може контролювати температуру в будинку за допомогою термостатів, які можна програмувати для забезпечення комфортних умов. Ви можете встановлювати розклади опалення та кондиціонування, а також віддалено керувати ними через смартфони або голосові асистенти. Це сприяє зниженню витрат на енергію та забезпечує оптимальну температуру в будинку;

– безпека. Розумний будинок забезпечує рівень безпеки, що вище за звичайний будинок. Це включає в себе відеоспостереження, датчики витоку газу або води, системи сповіщення про вторгнення та автоматичну сигналізацію. Крім того, ви можете віддалено контролювати та моніторити стан безпеки свого будинку через смартфон або комп'ютер;

– розваги. Розумний будинок може стати центром розваг для мешканців. Він може об'єднувати різні аудіо- та відео-пристрої, такі як телевізори, аудіосистеми та стрімінгові платформи, у єдину систему керування. Користувачі можуть відтворювати відео та аудіо контент, керуючи ним зі смартфону або голосовими командами.

Загалом, розумний будинок надає користувачам значні переваги в термінах автоматизації освітлення, опалення, безпеки та розваг. Ця технологія допомагає оптимізувати витрати, підвищує зручність та комфорт у повсякденному житті та забезпечує більше часу для важливих справ і розваг [3].

1.2 Структура та елементи системи «Розумний будинок»

Структура інтелектуальних систем, що базується на штучному інтелекті, представляє собою інтегрований комплекс, який включає взаємодіючі IoT-пристрої, новітні сенсорні технології, актуатори та передові мережеві рішення (рис. 1.2). Центральними елементами цієї структури є високопродуктивні обчислювальні платформи, оснащені алгоритмами штучного інтелекту. Вони забезпечують збір, аналіз та обробку даних від різноманітних джерел. Це дозволяє не тільки ефективно

управляти різними процесами в системі, але й забезпечує можливість адаптації та навчання системи на основі аналізу зібраних даних, підвищуючи її інтелектуальні можливості.

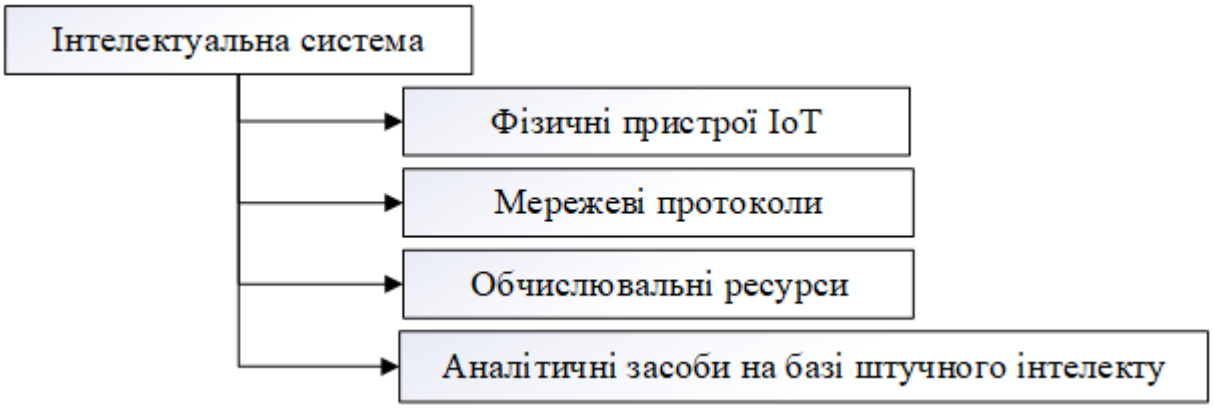

Рисунок 1.2 – Основні компоненти інтелектуальних систем

Звертаючи увагу на опис структури інтелектуальних систем, поданого на рисунку 1.2, можна більш детально розглянути ключові компоненти цієї системи та їх функції:

– фізичні пристрої IoT. Ця категорія пристроїв є спорудженням інтелектуальної системи і включає в себе різні датчики та сенсори. Ці пристрої розташовані в різних точках оточуючого середовища, будинку чи промислового об'єкта. Вони відповідають за збір інформації про різні параметри навколишнього середовища, такі як температура, вологість, рух, освітленість, рівень звуку та інші фізичні показники. Ця інформація є важливою для системи, оскільки вона дозволяє системі реагувати на зміни в середовищі та виконувати визначені завдання;

– мережеві протоколи. Цей компонент включає в себе різноманітні мережеві технології і протоколи, що дозволяють забезпечити зв'язок між фізичними пристроями IoT та центрами обробки даних. Вони забезпечують передачу інформації від датчиків до центральних систем та можуть бути провідними (наприклад, Ethernet) або бездротовими (наприклад, Wi-Fi, Bluetooth, LTE). Вибір конкретного мережевого протоколу залежить від конкретних вимог системи, таких як відстань передачі, пропускна здатність та ефективність використання енергії;

– обчислювальні ресурси. Обчислювальні ресурси можуть бути розподілені між локальними edge серверами та віддаленими хмарними середовищами. Локальні edge сервери розташовані недалеко від датчиків та актуаторів і призначені для швидкої обробки та аналізу даних, які збираються на місці. Вони дозволяють реагувати на події в реальному часі та забезпечують надійність системи. В той час як віддалені хмарні середовища надають велику обчислювальну потужність для складних аналітичних обчислень та зберігання великих обсягів даних;

– аналітичні засоби штучного інтелекту. Інтелектуальні системи використовують методи машинного та глибокого навчання для аналізу даних та прийняття рішень. Машинне навчання дозволяє системі вивчати закономірності в даних, щоб передбачати майбутні події та тренди. Глибоке навчання дозволяє системі розрізняти складні шаблони та виявляти нелінійні зв'язки в інформації. Застосування цих методів дозволяє системі автоматично адаптуватися до змін у середовищі та оптимізувати стратегії управління для досягнення максимальної ефективності [4].

Ця структура сприяє формуванню інтелігентного, взаємодіючого та високо адаптивного середовища, яке автономно відповідає на потреби користувачів та зміни в навколишньому середовищі. Це забезпечує підвищену ефективність управління, збільшуючи рівень автоматизації як у домашніх, так і в комерційних приміщеннях.

Сучасна архітектура інтелектуальних систем базується на фундаменті штучного інтелекту і представляє собою унікальний екосистемний підхід. Вона включає в себе взаємопов'язані компоненти, такі як IoT-пристрої, розвинені мережеві конфігурації та потужні обчислювальні можливості, націлені на оптимізацію та автоматизацію процесів у широкому спектрі операційних сфер.

На прикладі архітектури RL-IoT, що представлена на рис. 1.3, можна побачити, як застосування модулів навчання з підкріпленням орієнтоване на досягнення специфічних цілей через точно налаштоване керування IoT-пристроями [5]. Тут мета визначається як бажаний кінцевий стан, до якого має

дійти пристрій або система після проходження певних етапів і виконання заданих дій.

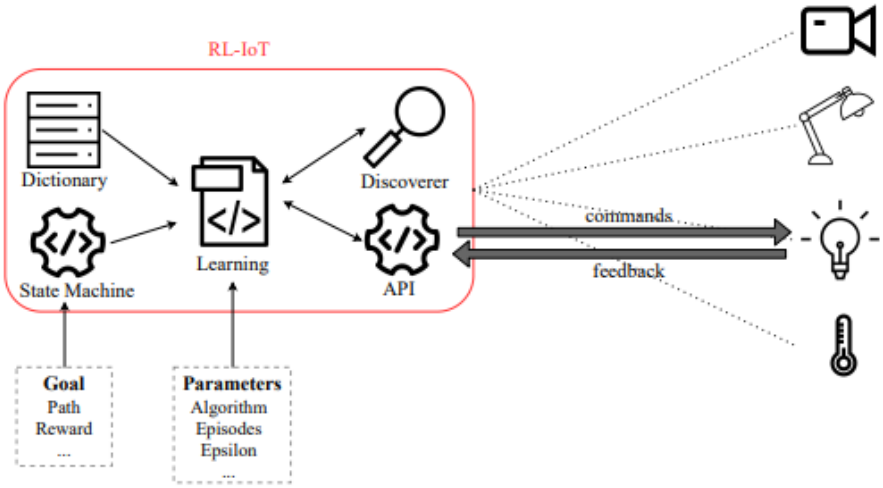

Рисунок 1.3 – Архітектура RL-IoT інтелектуальних систем

В системі RL-IoT комунікація між пристроями здійснюється на основі ретельно розробленого набору інструкцій, що охоплює всі можливі команди. Цей набір формується з використанням офіційних протоколів, аналізу мережевого трафіку та методів оберненого інжинірингу.

Завдяки алгоритмам навчання з підкріпленням, таким як Q-Learning або SARSA, модуль RL системи розвиває та адаптує внутрішній автомат станів, вибираючи найбільш ефективні дії з набору комунікаційних команд для досягнення мети. Ефективність дій оцінюється на основі системи винагород, яка оцінює потенціал дій для досягнення заданої цілі [6].

Додаткові модулі, як «Discoverer», відіграють роль у виявленні IoT-пристроїв у мережі через методи сканування. Після ідентифікації пристроїв модуль Socket API забезпечує абстракцію комунікаційних процесів для взаємодії з ними, дозволяючи відправляти команди та отримувати вхідні дані з урахуванням архітектурних особливостей цих пристроїв[7].

IoT та автономні системи розвивались як дві різні галузі технологій, кожна з власними методами та цілями. IoT фокусується на з'єднанні фізичних об'єктів з мережею, в той час як автономні системи зосереджені на самостійному виконанні

завдань. Їх інтеграція в концепції AIoT (Автономний Інтернет речей) відкриває нові можливості для покращення функціональності та можливостей IoT.

У концепції AIoT, інтелектуальні пристрої не тільки виконують збір даних через сенсори, але й активно впливають на фізичне середовище за допомогою виконавчих механізмів, що реалізують рішення на основі оброблених даних (див. рис. 1.4). Така структура дозволяє IoT-пристроям з'єднуватися з Інтернетом не лише через традиційні точки доступу, але й обробляти дані локально на мобільних пристроях або інших точках доступу перед передачею на хмарні сервери для детального аналізу [8].

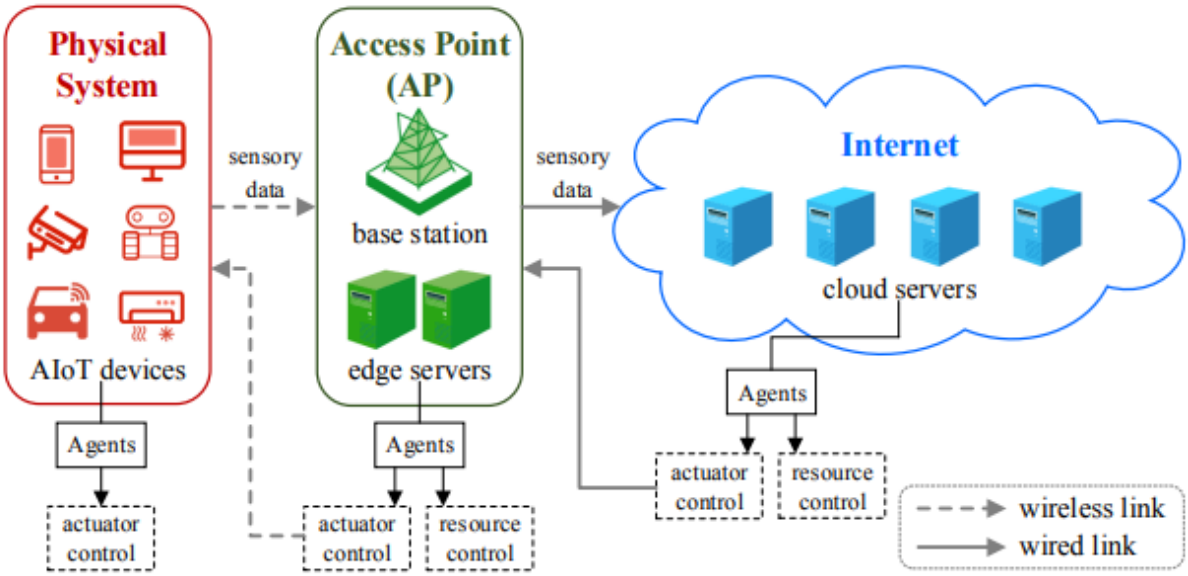

Рисунок 1.4 – Архітектура бездротової мережі AIoT

WSAN (бездротова мережа сенсорів та актуаторів) в рамках AIoT складається з сенсорів, які збирають дані про оточуюче середовище, та актуаторів, які реагують на ці дані для взаємодії з навколишнім світом через бездротові мережі. Використовуючи алгоритми підсиленого глибокого навчання, автономні агенти в WSAN не лише обробляють інформацію, але й приймають рішення про управління актуаторами для адаптації або модифікації стану середовища. Це створює динамічне взаємодіюче середовище, яке активно реагує на зміни в оточенні.

У рамках архітектури AIoT користувачі взаємодіють з більш інтегрованою та функціональною системою, яка перевищує можливості звичайних WSAN. AIoT

об'єднує не лише збір даних, але й їх глибокий аналіз, обробку та реалізацію дій. Завдяки передовим комунікаційним та обчислювальним технологіям, AIoT впроваджує комплексний підхід до управління інформацією, включаючи широкий спектр сервісів [9].

Структура AIoT складається з трьох основних рівнів:

– шар сприйняття. Це первинний інтерфейс між фізичним світом та цифровим середовищем, що включає IoT-пристрої з сенсорами для збору інформації та актуаторами для взаємодії з оточенням;

– мережевий шар. Відповідальний за забезпечення комунікації між IoT-пристроями та обчислювальними центрами. Цей шар використовує різноманітні точки доступу, включаючи мобільні станції та Wi-Fi роутери, для ефективної передачі даних;

– шар додатків. Включає сервери та обчислювальні платформи, де відбувається обробка та аналіз зібраних даних. Тут також реалізується автоматизація процесів за допомогою алгоритмів штучного інтелекту [10].

1.3 Інноваційні методи безпеки в системах «Розумний будинок»

Сучасні системи безпеки в «розумних будинках» тепер обладнані розширеним спектром інтелектуальних можливостей, що дозволяють їм адаптуватися до індивідуальних потреб та звичок мешканців. Інтеграція штучного інтелекту, зокрема методів машинного та глибокого навчання, революціонізує сферу домашньої безпеки, надаючи системам здатність не просто реагувати на поточні загрози, а й передбачати потенційні ризики. Це досягається завдяки постійному аналізу зібраних даних, що дозволяє системам виявляти аномалії та адаптуватися до змін у поведінці користувачів або навколишньому середовищі. Розумні системи безпеки, таким чином, стають надійними сторожами оселі, забезпечуючи не лише фізичний захист, але й сприяючи створенню затишного та безпечного житлового середовища.

Методи домашньої автоматизації

Домашня автоматизація, орієнтована на інтеграцію із системами безпеки, створює умови для більш тісної взаємодії мешканців з їхнім житлом. Використання різноманітних датчиків та актуаторів, які працюють відповідно до команд з центральної системи управління, робить це можливим. Завдяки технологіям Інтернету речей, кожен елемент домашньої системи може бути підключений до мережі, надаючи власникам можливість моніторингу та управління своєю системою безпеки з будь-якого місця.

Користувачі можуть отримувати сповіщення про входи й виходи, перевіряти стан дверних замків, контролювати відео з камер безпеки, а також управляти системами сповіщення та освітлення. Інтелектуальні замки дозволяють створювати тимчасові чи постійні доступи, забезпечуючи високий рівень безпеки. Системи виявлення руху можуть синхронізуватися з освітленням, автоматично вмикати світло при виявленні активності, що може відлякувати небажаних гостей.

Розумні будинки можуть використовувати поведінковий аналіз для налаштування параметрів безпеки відповідно до звичок мешканців. Наприклад, якщо система фіксує регулярне відкриття гаражних дверей у певний час, вона може нагадувати про їх закриття або навіть самостійно закривати їх при певних умовах. Це перетворює систему безпеки з реактивної на превентивну.

З розвитком технологій штучного інтелекту та машинного навчання, системи безпеки в «розумних будинках» стають ще більш передовими, здатними ефективно реагувати на загрози та антиципувати потенційні небезпеки, перетворюючи домівку на справді інтелектуальне та безпечне середовище [11].

Методи розумного відеоспостереження

Інтелектуальне відеоспостереження стає ключовим елементом у системах безпеки сучасних «розумних будинків», використовуючи просунуті техніки обробки відео для детального аналізу ситуацій, зафіксованих камерами. Такі системи здатні не тільки фіксувати рух, але й ідентифікувати обличчя, номерні знаки автомобілів та інші ключові елементи. Вбудований штучний інтелект дає можливість системам вчитися на основі попереднього досвіду, підвищуючи їхню здатність до передбачення і проактивного реагування на загрози.

Застосування глибокого навчання дозволяє системі розумного відеоспостереження аналізувати великі масиви відеоданих, виявляючи складні поведінкові шаблони та визначаючи аномалії в поведінці. Система може автоматично виявляти незвичні події, як-от несанкціонований вхід у заборонені зони або зміни у поведінці відвідувачів.

Інтеграція розумного відеоспостереження з іншими компонентами домашньої автоматизації дозволяє створювати більш складні сценарії безпеки. Наприклад, відеокамери можуть співпрацювати з освітлювальними системами, замками та аудіосистемами для створення ілюзії присутності, що може відлякувати потенційних зловмисників, а також забезпечувати зручності для мешканців, такі як привітання при поверненні додому.

Самовдосконалення є однією з ключових переваг розумного відеоспостереження. Системи, які навчаються та адаптуються на основі зібраних даних, стають ефективнішими у передбаченні та реагуванні на потенційні загрози, забезпечуючи високий рівень безпеки для будинку та його мешканців [12].

Системи розумного освітлення

Системи розумного освітлення відіграють важливу роль у сучасних «розумних будинках», сприяючи підвищенню комфорту та безпеки для мешканців. Вони забезпечують автоматичне управління освітленням, враховуючи зовнішнє освітлення, час доби та присутність людей в приміщенні. Такі системи можуть автоматично вмикати світло при виявленні руху, полегшуючи повсякденне життя мешканців та допомагаючи зекономити електроенергію.

Інтеграція розумного освітлення з системами безпеки відкриває нові можливості для забезпечення безпеки будинку. Освітлення може використовуватися як ефективний засіб відлякування незапрошених гостей або імітації присутності мешканців під час їх відсутності. В разі тривоги світло може автоматично включитися для покращення відеозапису або для стримування можливих вторгнень, забезпечуючи додатковий рівень безпеки для мешканців.

Також, розумне освітлення може легко інтегруватися з домашніми асистентами, які відповідають на голосові команди користувачів, забезпечуючи ще

більший рівень зручності та доступності. Використання енергоефективних джерел світла, таких як LED-лампи, дозволяє ефективно використовувати електроенергію та сприяє сталому використанню ресурсів у побуті.

Отже, системи розумного освітлення в сучасних розумних будинках не лише сприяють комфорту та ефективності, але й підвищують рівень безпеки та дозволяють зекономити енергію, створюючи приємні умови для мешканців [13].

Інтегровані безпекові системи

Сучасні інтегровані безпекові системи, що використовують машинне навчання, є інноваційним рішенням для захисту «розумних будинків». Вони аналізують повсякденні звички та поведінку користувачів, навчаючись розпізнавати звичайні умови та виявляти аномальні події, які можуть сигналізувати про небезпеку. Так, несподіване відчинення дверей чи вікон може спонукати систему надіслати повідомлення власнику чи активувати тривогу.

Ці системи інтегрують широкий спектр датчиків – від руху до датчиків дверей, вікон, температури та диму – для збору даних про стан дому. Завдяки машинному навчанню, вони можуть обробляти й аналізувати ці дані, розпізнаючи не тільки явні ознаки проблем, але й виявляючи тонші зміни у звичному порядку.

Інтегровані безпекові системи здатні адаптуватися до нових поведінкових моделей та змінених загроз, завдяки чому вони стають більш точними у виявленні потенційних проблем з часом. Зростаюча автономність таких систем гарантує більш комплексний захист, який може пристосовуватися до змін у домашньому середовищі.

Ці системи також пропонують високий рівень персоналізації, дозволяючи користувачам налаштовувати параметри безпеки згідно з власними потребами та перевагами. Такий підхід створює інтуїтивно зрозумілий та надійний рівень захисту, який ефективно функціонує в фоновому режимі, забезпечуючи безпеку та спокій мешканців.

Системи інтелектуальних замків безпеки

Інтелектуальні замки у системах «розумного будинку» відкривають нові горизонти у сфері безключового доступу та розширених функцій безпеки.

Використовуючи біометричні технології, як-от сканування відбитків пальців, розпізнавання облич та сітківки ока, ці системи забезпечують високий рівень ідентифікації осіб, ефективно запобігаючи несанкціонованому доступу. Багато з цих замків легко інтегруються з системами домашньої автоматизації та керуються через мобільні додатки, даючи власникам можливість контролювати доступ до своїх домівок з будь-якої точки світу.

Мобільні додатки дозволяють створювати індивідуальні електронні ключі для гостей, обслуговуючого персоналу чи родичів, що забезпечує додаткову гнучкість та контроль. Ключі можна швидко відмінити або модифікувати, що забезпечує гнучкість управління доступом.

Інтелектуальні замки також ведуть журнали доступу, фіксуючи кожну спробу входу або виходу, дозволяючи власникам стежити за активністю біля їхніх власностей. Ці замки можуть бути інтегровані з іншими безпековими системами, як-от відеокамерами чи розумним освітленням, для створення більш складних безпекових сценаріїв.

Захист даних забезпечується за допомогою шифрування бездротового з'єднання між замком і контролюючими пристроями, а двофакторна автентифікація забезпечує додатковий рівень безпеки. Ці інтелектуальні замки також можуть підтримувати Wi-Fi або Bluetooth для синхронізації з домашньою мережею, надаючи власникам сповіщення в реальному часі.

Інтелектуальні замки впроваджують новий рівень автоматизації та персоналізації, дозволяючи налаштовувати розклади автоматичного замикання/відмикання та активувати замки залежно від геолокації користувачів. Це робить їх незамінною частиною сучасних «розумних будинків», забезпечуючи високий рівень безпеки та зручності [14].

Методи для інтеграції із персональними асистентами

Злиття систем безпеки «розумного будинку» з голосовими асистентами є одним з ключових досягнень у сфері домашньої автоматизації. Це дає користувачам змогу управляти такими системами, як сигналізація, відеоспостереження та контроль доступу, за допомогою простих голосових команд.

Користувачі тепер можуть легко налаштовувати системи охорони, перевіряти стан своїх захисних систем та одразу отримувати сповіщення про будь-які події безпеки у своєму домі.

Голосові асистенти здатні синхронізуватися з широким діапазоном пристроїв безпеки, включаючи дверні дзвінки з камерами, датчики руху, димові детектори тощо, створюючи автоматизовані сценарії реакції на різноманітні події. Наприклад, асистент може автоматично увімкнути світло у разі виявлення руху під час ночі або відправляти сповіщення, коли система реєструє незвичайну активність.

Персональні голосові асистенти, інтегровані з системами «розумного будинку», можуть бути налаштовані для взаємодії з різними користувачами, розпізнаючи їхні голоси та надаючи кожному свій рівень доступу та управління. Це додає персоналізованість і безпеку до системи. Крім того, такі системи використовують можливості штучного інтелекту для вивчення повсякденних звичок користувачів, автоматично адаптуючи налаштування безпеки для підвищення їх ефективності.

Голосове управління безпековими системами значно спрощує щоденне життя, дозволяючи користувачам зосередитись на власних справах, поки автоматизована система безпеки забезпечує захист їхнього житла.

Методи домашньої автоматизації дозволяють централізовано управляти домашньою безпекою через мобільні додатки. Розумні відеоспостереження використовують аналітичні алгоритми для ідентифікації осіб та аномалій у поведінці. Системи розумного освітлення автоматично регулюють рівень світла, збільшуючи комфорт та безпеку. Інтегровані безпекові системи застосовують машинне навчання для моніторингу та реагування на незвичні події. Інтелектуальні замки забезпечують безключовий доступ і дозволяють віддалено керувати доступом до житла через мобільні додатки, що забезпечує зручність та надійний захист.

Поряд із ризиком незаконного фізичного вторгнення, системи «Розумного дому» також стикаються зі значною загрозою витоку інформації. Ці системи збирають та аналізують об'єми даних про особисті звичаї та поведінку

користувачів, що робить їх привабливими цілями для хакерів. Несанкціонований доступ до цієї інформації може призвести до порушення приватності, а також використовуватися для проведення кібератак або шахрайства.

Ефективний захист даних в «Розумному домі» потребує впровадження стратегій кібербезпеки, які включають шифрування даних, забезпечення безпеки мережевих з'єднань та регулярне оновлення безпекових протоколів. Це вимагає інтегрованого підходу до кібербезпеки, який включає як технічні, так і програмні заходи для забезпечення надійного захисту конфіденційної інформації [15].

1.4 Оцінка загроз та викликів у системах «Розумний будинок»

Загрози безпеки інформації в «розумних будинках» включають в себе різноманітні умови та фактори, які можуть становити потенційну або реальну небезпеку для захисту оброблюваних системою даних. Вразливість інформаційної системи характеризується її схильністю до різних видів кібератак, що можуть призвести до реалізації цих загроз.

Основні загрози інформаційній безпеці в контексті «розумного будинку» включають:

– порушення конфіденційності. Це становить ризик несанкціонованого доступу до приватної інформації. Це може включати в себе витік особистих даних користувачів, їхніх звичок та взаємодії з різними системами будинку;

– порушення цілісності. Це стосується ризику несанкціонованої або непомітної модифікації інформації. Такі дії можуть включати зміну налаштувань системи або зміну інформації, що може призвести до некоректної роботи системи;

– порушення доступності інформації. Це включає ситуації, коли користувачі не можуть отримати доступ до важливих інформаційних ресурсів або систем. Причинами можуть бути різноманітні технічні неполадки, кібератаки або навмисні дії, що блокують доступ до систем [16].

У контексті аналізу систем «розумного будинку», конфіденційність відноситься до стану IT-системи, де забезпечується захист від несанкціонованого витоку інформації через підсистеми. Це означає, що персональні дані користувачів

та інформація про конфігурацію системи «розумного будинку» повинні бути надійно захищені від стороннього доступу. Небезпека тут полягає в можливості витоку конфіденційної інформації, що може включати особисті дані мешканців або деталі щодо внутрішньої структури IT-системи.

Цілісність інформації у «розумному будинку» стосується забезпечення достовірності та повноти інформації, яка надходить від різних датчиків і пристроїв системи. Наприклад, невірні дані від сенсорів про присутність людини в приміщенні можуть спричинити помилкову активацію систем контролю доступу. Важливість збереження цілісності полягає в тому, щоб забезпечити, що всі дані, якими оперує система, є точними та актуальними[17].

Доступність інформації у «розумному будинку» означає, що інформація та ресурси IT-системи повинні бути доступні для користувачів або самої системи, які мають права доступу, для здійснення необхідних операцій згідно з робочим сценарієм. Це може включати такі дії, як вмикання/вимикання датчиків, відкриття замків тощо. Наприклад, загроза може полягати у виведенні з ладу комунікаційного обладнання системи, що призведе до неможливості виконання цих функцій.

Загрози інформаційній безпеці систем «розумного будинку» можуть бути класифіковані на основі їхнього походження на дві основні категорії: загрози, що виникають через людський фактор, та загрози, що випливають із зовнішнього середовища, як-от природні явища.

Загрози, що обумовлені людським фактором, поділяються на два підтипи залежно від способу їхнього здійснення:

– цілеспрямовані (навмисні) загрози. Це можуть бути дії зловмисників, які свідомо намагаються порушити роботу системи, викрасти дані або завдати шкоди системі «розумного будинку». Прикладами таких загроз є хакерські атаки, фішинг, шпигунські програми та інші види кібератак;

– випадкові (ненавмисні) загрози. Ці загрози виникають через необережність або незнання користувачів. Наприклад, це може бути невірне використання обладнання, помилкове налаштування систем, випадкове видалення важливих даних тощо.

Що стосується загроз зовнішнього середовища, вони часто є непередбачуваними та можуть включати природні катаклізми, такі як землетруси, повені, урагани, які можуть вплинути на фізичну інфраструктуру «розумного будинку». Ці події можуть спричинити серйозні пошкодження обладнання та втрату даних [18].

У табл. 1.1 представлена класифікація загроз безпеки інформації IT-систем «розумного будинку», що надає детальний огляд різних типів загроз та їх потенційних наслідків. Важливо враховувати ці різні типи загроз при розробці та впровадженні заходів безпеки для захисту інформаційних систем «розумного будинку».

Таблиця 1.1 – Систематизація ризиків безпеки даних систем «розумний дім»

Ризики, зумовлені людиною		Екологічні ризики
Навмисні порушення	Випадкові порушення	
Зміна даних	Помилки в програмах	Надзвичайні пожежі
Втручання в передачу даних	Помилки операторів	Заливи
Викрадення обладнання	Технічні неполадки	Удари блискавки
Атаки кіберзлочинців	Неправильні налаштування системи	Сейсмічні події
Віруси та шкідливе ПЗ	Некоректна маршрутизація даних	Екстремальні кліматичні умови

Ідентифікація можливих джерел загроз інформаційної безпеки є однією з суттєвих складових визначення загроз для інформаційної системи. Ця процедура допомагає виявити, звідки можуть виникнути потенційні загрози та де розташовані джерела цих загроз. Особливу увагу при ідентифікації приділяють внутрішнім та зовнішнім джерелам загроз.

Внутрішні загрози відносяться до загроз, що виникають всередині контрольованої зони або організації. Це можуть бути дії або недіяльність співробітників, які мають доступ до інформаційних ресурсів. Внутрішні загрози

можуть включати недбале оброблення конфіденційної інформації, зловживання даними, неправомірний доступ до систем або використання слабкостей у внутрішніх процесах організації.

Зовнішні загрози, навпаки, виникають ззовні організації або контрольованої зони. Це можуть бути атаки з боку хакерів, віруси, атаки через мережу Інтернет, спроби фішингу та інші форми зловживання. Зовнішні загрози можуть становити значний ризик для конфіденційності, цілісності та доступності інформаційних ресурсів.

У табл. 1.2 представлено перелік найбільш поширених внутрішніх та зовнішніх загроз системи «розумний будинок»

Таблиця 1.2 – Перлік внутрішніх та зовнішніх загроз

№ п/п	Внутрішня загроза	Зовнішня загроза
1	Недбале використання паролів	Атаки з боку хакерів
2	Некоректне обслуговування IoT-пристроїв	Віруси та шкідливе ПЗ
3	Витік інформації від співробітників	Фішинг та соціальна інженерія
4	Недостатнє оновлення програмного забезпечення	Деніал-Of-Service (DoS) атаки
5	Використання слабких аутентифікаційних методів	Несанкціонований доступ до мережі
6	Неправомірний доступ до системи	Атаки на мережеві пристрої

Виявлення, ідентифікація та врахування перелічених загроз є важливими кроками для розробки ефективних заходів захисту для розумного будинку, забезпечуючи безпеку та приватність мешканців [19].

1.5 Постановка задачі

Головною метою є створення алгоритму для виявлення та нейтралізації кіберзагроз в онлайн-режимі, використовуючи алгоритм оптимізації Бройдена –

Флетчера – Гольдфарба – Шанно для максимізації ентропії [20]. Цей алгоритм планується реалізувати на C# з використанням бібліотеки ML .NET.

Ключовим аспектом реалізації цього алгоритму є його інтеграція в систему «Інтелектуальний будинок». Це передбачає не лише здатність до виявлення та реагування на кіберзагрози, але й співпрацю з іншими елементами системи, такими як сенсори руху, відеонагляд, а також розумне управління енергоспоживанням. Розроблений алгоритм має бути сумісним з різноманітним обладнанням та платформами, які використовуються у «розумних домах». Наприклад, він може аналізувати інформацію з датчиків руху чи відеокамер для виявлення підозрілої активності, що може вказувати на спроби неавторизованого доступу.

Також важливим є адаптування системи до особистих потреб користувачів «Інтелектуального Дому». Це включає можливість налаштування параметрів безпеки з урахуванням специфічних умов та вимог користувача, а також інтеграцію з різними користувацькими інтерфейсами для зручності використання.

Розробка такого алгоритму дозволить йому ефективно функціонувати в межах інтегрованої системи «Інтелектуальний будинок», забезпечуючи високий рівень безпеки та зручності для користувачів.

Основні вимоги до функціональності системи включають:

– виявлення та ідентифікація кіберзагроз. Система має автоматично детектувати та розпізнавати потенційні кіберзагрози та аномальні моделі поведінки, які можуть вказувати на кібератаку;

– прогнозування можливих кібератак. Використовуючи алгоритм оптимізації Бройдена – Флетчера – Гольдфарба – Шанно, система повинна мати здатність прогнозувати майбутні кібератаки у реальному часі;

– відповідь на інциденти безпеки. У випадку детектування загрози система має автоматично інформувати користувача про інцидент;

– оновлення захисних стратегій. Система повинна мати можливість оновлюватися, щоб адаптуватися до нових видів загроз та підтримувати актуальність захисних алгоритмів;

– аудит та звітність. Реалізація функцій для збору, аналізу даних про інциденти безпеки та формування детальних звітів про стан безпеки системи.

Ключові нефункціональні вимоги до системи включають:

– ефективність обробки даних. Система повинна здійснювати обробку вхідних даних і реагувати на потенційні загрози миттєво, без значних затримок, забезпечуючи роботу в режимі реального часу;

– масштабованість системи. Проектування системи має передбачати можливість її розширення залежно від збільшення потреб користувачів і обсягів оброблюваних даних;

– висока надійність. Система має бути стабільною і доступною, із мінімальною кількістю збоїв та помилок у роботі;

– захист даних. Інформація, що проходить через систему, має бути надійно захищена від неавторизованого доступу та витоків;

– інтеграція з іншими компонентами. Система має легко інтегруватися з різними компонентами «розумного дому»;

– зручність та інтуїтивність інтерфейсу. Інтерфейс системи має бути зрозумілим та зручним для кінцевих користувачів.

Розробка системи, що включає передові рішення в області інформаційної безпеки для «розумного дому», є складним завданням, що вимагає уваги не лише до технічних характеристик, але й до зручності користування. Ключ до успіху впровадження полягає в легкості використання та інтуїтивності інтерфейсів. При цьому важливо забезпечити надійність системи, щоб вона могла безперебійно працювати в різних умовах, ефективно виявляючи та нейтралізуючи загрози.

1.6 Висновок

У рамках даного розділу було розглянуто основні теоретичні принципи та аналіз системи «Розумний будинок». Була досліджена історія та концепція розумних будинків, їхні технологічні основи, зокрема роль Інтернету речей (IoT), сенсорних технологій, централізованого управління, автоматизації та адаптивності систем. Було проаналізовано інноваційні методи безпеки в системах «Розумний

будинок», включаючи інтеграцію з штучним інтелектом, методи домашньої автоматизації, розумного відеоспостереження, інтелектуальних замків безпеки та інтеграцію із персональними асистентами.

Також була проведена оцінка потенційних загроз і викликів, з якими може зіткнутися система «Розумний будинок». Аналіз включав вивчення вразливостей, що стосуються конфіденційності та безпеки даних, ризиків несанкціонованого доступу, а також проблем, пов'язаних з надійністю та стійкістю системи до зовнішніх перешкод. Враховуючи ці аспекти, була сформульована задача створення більш безпечної, надійної та інтелектуальної системи «Розумний будинок», здатної адекватно реагувати на поточні та майбутні виклики у сфері домашньої автоматизації.

2 ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ

2.1 Аналіз методів машинного навчання та вибір найбільш перспективного

Використання технік машинного навчання значно розширює спектр можливостей для створення дієвих рішень. Варто зосередити увагу на специфічних підходах до оптимізації та аналізу даних, які здатні ефективно підвищувати точність у виявленні вторгнень. Важливу роль у цьому контексті відіграють такі методи, як алгоритми Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ), метод стохастичного градієнтного спуску та техніка випадкового пошуку (Random Search).

Алгоритм Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ) є відомим і широко застосовуваним методом оптимізації у сфері машинного навчання. Цей метод відноситься до категорії квазі-ньютонівських алгоритмів і використовується для вирішення завдань нелінійної оптимізації.

Завдання алгоритму БФГШ полягає у визначенні мінімуму функції втрат, яка часто зустрічається в задачах машинного навчання, таких як регресія чи класифікація. Особливістю БФГШ є його здатність працювати без необхідності обчислення других похідних (гессіану) функції втрат [21]. Це робить метод більш ефективним порівняно з іншими техніками, що вимагають повного гессіану. Замість цього, БФГШ апроксимує гессіан, використовуючи оновлення, засновані на градієнтах функції.

Процес роботи алгоритму полягає в оновленні параметрів у просторі рішень, використовуючи дані про градієнт і апроксимацію оберненого гессіану з попередніх ітерацій. Такий підхід дозволяє алгоритму ефективно знаходити точку оптимуму, враховуючи як напрямок зменшення функції втрат, так і її кривизну в даній точці.

БФГШ є особливо корисним для функцій втрат, які є гладкими і мають обмежену кількість локальних мінімумів, що робить його відмінним інструментом

для точного налаштування моделей у багатьох задачах машинного навчання, з метою досягнення високої точності прогнозування.

Метод БФГШ відрізняється своєю здатністю оптимально поєднувати швидкість збіжності та точність оновлень. Ця особливість робить його надзвичайно ефективним для оптимізації складних функцій втрат у сфері машинного навчання (рис. 2.1).

Рисунок 2.1 – Схема роботи алгоритму БФГШ

Процес роботи алгоритму БФГШ розпочинається з визначення початкової точки в просторі параметрів та ініціалізації апроксимації оберненого гессіана, яка часто представляється у вигляді одиничної матриці. На кожному етапі алгоритм визначає градієнт функції втрат у поточній точці, що вказує напрямок найбільшого зростання функції.

Використовуючи апроксимацію оберненого гессіана, алгоритм визначає напрямок для мінімізації функції втрат. Параметри моделі оновлюються шляхом зміщення поточної точки у просторі параметрів у напрямку, протилежному градієнту. Після кожного оновлення, BFGS адаптує своє наближення до оберненого гессіана, виходячи з останніх обчислень градієнта та змін у параметрах.

Алгоритм продовжує цикл ітерацій до досягнення визначених критеріїв зупинки, які можуть включати незначні зміни у функції втрат або обмеження за кількістю ітерацій. Такий підхід дозволяє методу БФГШ ефективно підходити до оптимальної точки, враховуючи напрямок зменшення функції втрат та її кривизну.

Переваги алгоритму БФГШ включають:

– висока продуктивність. Алгоритм БФГШ ефективно знаходить мінімуми функцій втрат, особливо коли ці функції мають гладкий характер;

– непотрібність у других похідних. На відміну від класичних методів Ньютона, БФГШ обходиться без обчислення других похідних гессіана, що робить його більш зручним для обробки великих задач;

– висока швидкість збіжності. Метод демонструє швидку збіжність, особливо у випадках, коли гессіан є позитивно визначеним або майже позитивно визначеним;

– автоматична адаптація кроку. БФГШ автоматично налаштовує розмір кроку в процесі ітерацій, забезпечуючи більшу гнучкість та ефективність при пошуку оптимальних параметрів.

Недоліки алгоритму БФГШ:

– обмеження у великих масштабах. Для дуже великих задач, де кількість параметрів або обсяг даних є значним, БФГШ може бути менш практичним через високі вимоги до пам'яті та обчислювальних ресурсів;

– ризик неправильної збіжності. У випадках, коли гессіан функції втрат не є позитивно визначеним, алгоритм може стикатися з проблемами збіжності;

– чутливість до початкових умов. Вибір початкової точки та наближення оберненого гессіана може суттєво впливати на ефективність та швидкість збіжності алгоритму;

– Потреба у точному градієнті. Метод вимагає точного обчислення градієнтів, помилки в яких можуть спричинити проблеми з збіжністю або зниженням продуктивності.

Таким чином, алгоритм БФГШ виступає як ефективний засіб для оптимізації у машинному навчанні, особливо при роботі з нелінійними оптимізаційними

задачами, що мають гладкі функції втрат. Його ключові переваги полягають у високій продуктивності, здатності до швидкої збіжності та можливості автоматичної адаптації розміру кроку, що робить його придатним для широкого спектру оптимізаційних завдань. Проте, метод може виявитися менш практичним для задач великого масштабу через обмеження у використанні пам'яті та обчислювальних ресурсів, він чутливий до початкових умов і потребує точних обчислень градієнтів. Незважаючи на ці обмеження, БФГШ продовжує залишатися одним із найбільш вживаних методів оптимізації у різноманітних сферах машинного навчання.

Метод стохастичного градієнтного спуску (СГС) є ключовим алгоритмом для оптимізації у галузі машинного та глибокого навчання. Цей метод полягає в поетапному оновленні параметрів моделі з метою мінімізації функції втрат. На відміну від традиційного градієнтного спуску, який передбачає оновлення параметрів після розрахунку градієнта на всьому наборі даних, СГС виконує оновлення, використовуючи тільки один або декілька елементів (міні-пакет) навчальних даних.

Важливим параметром у СГС є швидкість навчання, яка визначає, наскільки інтенсивно модель реагує на помилку на кожному етапі. Правильний вибір швидкості навчання є вирішальним, оскільки занадто великий крок може викликати нестабільність, в той час як надто малий крок може уповільнити процес навчання [22].

Стохастичний градієнтний спуск широко застосовується для навчання різноманітних моделей у сфері машинного навчання, особливо ефективний при роботі з великими обсягами даних. Його головна перевага полягає в здатності ефективно обробляти великі датасети, оновлюючи параметри моделі на основі часткових даних, а не всього масиву. Такий підхід дозволяє СГС бути більш гнучким та швидким при оптимізації великих наборів даних. Схематичне зображення роботи цього методу можна побачити на рис. 2.2.

Рисунок 2.2 – Схема роботи алгоритму СГС

Процедура роботи методу СГС розпочинається з встановлення початкових значень параметрів моделі, які зазвичай обираються наугад або згідно з певними заздалегідь встановленими правилами. На кожному кроці алгоритму відбирається один екземпляр даних для тренування або невелика група даних, відома як міні-пакет, що вибирається випадково. Використання міні-пакетів допомагає ефективно збалансувати процес оновлення та знизити його варіабельність.

Для вибраного тренувального прикладу чи міні-пакету проводиться розрахунок градієнта функції втрат на основі поточних значень параметрів моделі. Цей градієнт вказує напрямок найшвидшого зростання функції втрат. Виходячи з обчисленого градієнта, параметри моделі оновлюються у протилежному напрямку до градієнта, віднімаючи від поточних значень добуток швидкості навчання на градієнт.

Цей процес оновлення параметрів повторюється на кожному кроці до досягнення визначених критеріїв зупинки, які можуть включати досягнення певної кількості ітерацій, незначні зміни в функції втрат, або задовільний рівень точності моделі. Така послідовність дій дозволяє моделі поступово оптимізувати свою продуктивність, мінімізуючи помилки на основі аналізу даних.

Переваги методу стохастичного градієнтного спуску включають:

– підходить для обробки великих датасетів. Завдяки оновленню параметрів на основі лише частини даних, СГС ефективно справляється з великими

обсягами даних, у той час як повний градієнтний спуск був би надмірно вимогливим з обчислювальної точки зору;

– швидкість навчання. Оновлення параметрів після кожного тренувального прикладу або міні-пакету пришвидшує процес навчання у порівнянні з повним градієнтним спуском;

– здатність виходити з локальних мінімумів. Випадковий вибір тренувальних прикладів допомагає СГС уникнути застрягання в локальних мінімумах;

– гнучкість. СГС легко адаптується до різних задач та моделей у машинному навчанні, що робить його універсальним інструментом.

Недоліки методу СГС:

– варіативність оновлень. Через використання лише частини даних для кожного оновлення, СГС може проявляти високу варіативність, що призводить до потенційної нестабільності;

– чутливість до вибору швидкості навчання. Невірно обрана швидкість навчання може спричинити проблеми з конвергенцією, ускладнюючи оптимізацію;

– потреба в точному налаштуванні параметрів. Адаптація параметрів, як-от розмір міні-пакета та швидкість навчання, може бути складною та потребувати додаткових експериментів;

– ризик недостатньої збіжності. В деяких випадках СГС може не досягати глобального мінімуму або робити це занадто повільно, особливо в складних оптимізаційних умовах.

Таким чином, СГС є ключовим інструментом у машинному навчанні для ефективної оптимізації моделей, особливо коли мова йде про роботу з великими масивами даних. Основні переваги СГС включають ефективність у роботі з великими датасетами, швидке навчання, здатність подолати локальні мінімуми та широку сферу застосування. Водночас, метод має деякі недоліки: нестабільність оновлень, високу чутливість до налаштувань швидкості навчання, необхідність детального регулювання параметрів та потенційну проблему з досягненням належної збіжності. Не дивлячись на ці виклики, стохастичний градієнтний спуск

продовжує бути одним з найбільш використовуваних методів оптимізації в машинному навчанні, завдяки його універсальності та ефективності в різних сценаріях застосування.

Метод випадкового пошуку (МВП) використовується в машинному навчанні як один із способів оптимізації для налаштування гіперпараметрів моделі [23]. Цей метод відрізняється від традиційних, більш систематичних підходів оптимізації своєю випадковістю у виборі пошуку.

Метод полягає у випадковому визначенні комбінацій гіперпараметрів з зазначеного простору пошуку та аналізі їхньої ефективності через функцію втрат або іншу метрику продуктивності. Наприклад, при налаштуванні гіперпараметрів для нейронної мережі, метод випадкового пошуку випробуватиме різноманітні комбінації, такі як швидкість навчання, кількість шарів, кількість нейронів у кожному шарі, і вимірюватиме ефективність мережі за цими параметрами.

МВП є ефективним завдяки здатності охоплювати широкий спектр можливих рішень, часто знаходячи прийнятні варіанти з меншими обчислювальними зусиллями порівняно з більш систематичними методами, як-от сітковий пошук. Схема роботи цього методу зображена на рис. 2.3.

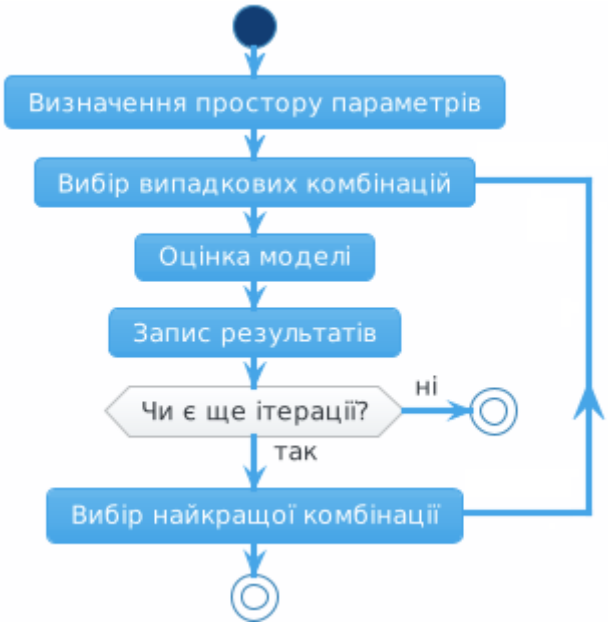

Рисунок 2.3 – Схема роботи алгоритму МВП

Метод випадкового пошуку для оптимізації гіперпараметрів у машинному навчанні ініціюється з встановлення діапазону гіперпараметрів, що підлягають

оптимізації. До таких параметрів належать, наприклад, швидкість навчання, кількість нейронів у різних шарах нейронної мережі, типи ядер у методах векторної машини тощо. На кожному кроці алгоритму випадковим способом вибирається комбінація цих гіперпараметрів з визначеного простору, де для кожного параметра обирається випадкове значення в межах дозволеного діапазону.

Далі, кожна отримана комбінація гіперпараметрів використовується для навчання моделі, після чого відбувається аналіз її продуктивності через функцію втрат або іншу відповідну метрику, як-от точність класифікації. Отримані результати фіксуються для подальшого порівняння. Процедура вибору випадкових наборів гіперпараметрів та оцінювання продуктивності моделі продовжується до виконання зазначеної кількості ітерацій або до вичерпання призначеного ліміту обчислювальних ресурсів.

По завершенню усіх ітерацій, серед оцінених комбінацій вибирається та, яка продемонструвала найкращі результати. Такий підхід дозволяє ефективно вивчати простір конфігурацій моделі, часто знаходячи задовільні рішення з меншою кількістю обчислювальних затрат порівняно з методами систематичної оптимізації.

Переваги методу випадкового пошуку:

– розгорнутий огляд простору параметрів. Випадковий пошук забезпечує широкий огляд простору гіперпараметрів, підвищуючи імовірність виявлення оптимальних або майже оптимальних наборів;

– ефективність в великих просторах параметрів. У ситуаціях з великими просторами параметрів, де систематизовані методи, як-от сітковий пошук, можуть бути надмірно ресурсомісткими, випадковий пошук виявляється більш продуктивним;

– легкість у впровадженні. Завдяки своїй простій логіці, метод випадкового пошуку легко реалізується, що робить його доступним навіть для тих, хто не є експертом у машинному навчанні;

– універсальність. Метод легко адаптується до різних видів задач та моделей, що робить його універсальним інструментом для налаштування гіперпараметрів.

Недоліки методу випадкового пошуку:

– відсутність гарантії знаходження оптимального рішення. Через випадковий характер метод не може гарантувати виявлення абсолютно найкращої комбінації параметрів;

– обмежена ефективність у малих просторах параметрів. У ситуаціях з обмеженим простором параметрів більш систематизовані методи, як-от сітковий пошук, можуть бути більш доцільними;

– велика варіативність результатів. Випадковий пошук може вести до значних відхилень у результатах, особливо при недостатній кількості ітерацій;

– можливість неефективного використання обчислювальних ресурсів. В деяких випадках метод може витрачати ресурси на дослідження непродуктивних або малоперспективних регіонів простору параметрів.

Таким чином, метод випадкового пошуку виступає як ефективний засіб для оптимізації гіперпараметрів у машинному навчанні, особливо коли йдеться про роботу з обширними просторами параметрів. Його ключові переваги включають здатність охоплювати значні області простору параметрів, легкість у впровадженні, високу продуктивність при роботі з великими просторами та універсальність застосування до різних задач. Проте, метод має деякі обмеження, такі як відсутність гарантії знаходження ідеального рішення, можлива непостійність результатів та ризик неефективного використання обчислювальних ресурсів. Метод випадкового пошуку є особливо корисним у ситуаціях, де необхідно швидко переглянути великий діапазон гіперпараметрів, і коли абсолютна точність у виборі найкращих параметрів не є вирішальною.

У табл. 2.1 представлено порівняльний аналіз різних методів машинного навчання.

Таблиця 2.1 – Порівняльний аналіз методів машинного навчання

Характеристика	БФГШ	СГС	МВП
Точність оптимізації	Висока (завдяки точним розрахункам	Середня (змінюється в залежності від швидкості навчання)	Залежить від випадковості

	градієнтів і оберненого гессіана)		(можлива менша точність)
Збіжність у складних ландшафтах	Ефективна (працює добре з позитивно визначеним гессіаном)	Обмежена (через стохастичність оновлень)	Обмежена (не завжди знаходить оптимальне рішення)
Потреба в обчисленні градієнтів	Важлива (необхідна для оновлення параметрів)	Так (але лише для частини даних)	Не потрібно (фокус на випадковому виборі гіперпараметрів)
Застосування до малих/середніх задач	Підходящий (ефективний при меншій кількості параметрів)	Менш підходящий (більш ефективний для великих датасетів)	Менш підходящий (фокусується на оптимізації гіперпараметрів)

Обрання методу БФГШ для задачі прогнозування вторгнень атак може бути виправдано його перевагами, які виявлені в аналізі. Перш за все, БФГШ характеризується високою точністю в оптимізації, забезпеченою завдяки точному визначенню градієнтів та оберненого гессіана. Це означає, що метод здатен ефективно підбирати ідеальні параметри моделі, що є ключовим для точного прогнозування вторгнень.

Другим аргументом на користь БФГШ є його здатність до ефективної збіжності в складних оптимізаційних умовах. Це особливо актуально у контексті виявлення вторгнень у реальному часі, де моделі часто зіштовхуються зі складними та неочевидними даними. Завдяки ефективності в умовах позитивно визначеного гессіану, БФГШ стає надійним варіантом для моделей, яким потрібне детальне налаштування параметрів.

Оскільки метод вимагає обчислення градієнтів, це сприяє точному оновленню параметрів моделі, що є важливим для досягнення точних результатів прогнозування. БФГШ також добре підходить для роботи в середніх та малих задачах, що може бути корисним у випадках, коли необхідно оптимізувати модель виявлення вторгнень без залучення великої кількості параметрів або при обмежених обчислювальних ресурсах.

Враховуючи ці фактори, БФГШ являється вдалим вибором для оптимізації моделей у задачах прогнозування вторгнень, пропонуючи оптимальний баланс між точністю, ефективністю та практичністю використання.

2.2 Формалізація математичної моделі системи

Цей розділ присвячений розробці математичної моделі для ідентифікації вторгнень. Основу моделі складає алгоритм оптимізації БФГШ, відомий своєю високою ефективністю у рішенні задач бінарної класифікації, особливо в контексті аналізу великих даних. Реалізація моделі включає цілий ряд етапів: від збору та аналізу даних до вибору ключових ознак, що точно відображають вторгнення, а також використання алгоритму МБФГШ для налаштування класифікатора. Даний алгоритм ефективно справляється з обробкою великих масивів даних і складних шаблонів, характерних для сучасних вторгнень.

Задача, яку потрібно вирішити, є задачею бінарної класифікації, де є набір даних (X, Y) у якому $X \in R^{n \times m}$ представляє собою матрицю ознак, а $Y \in \{0,1\}^n$ – вектор міток класу. У цьому випадку, *n* означає кількість прикладів у датасеті, а *m*=13 – кількість ознак, що включають: *FrameNumber, FrameTime, FrameLen, EthSrc, EthDst, IpSrc, IpDst, IpProto, IpLen, TcpLen, TcpSrcport, TcpDstport, Value, Normality*.

Функція втрат для логістичної регресії може бути виражена наступним чином:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right], \qquad (2.1)$$

де, $h_\theta(x)$ представляє сигмоїдну функцію, а θ – це вектор параметрів моделі.

Матриця *X* конструюється з екземплярів класу «*ModelDataInput*». Кожен рядок у матриці *X* відповідає одному екземпляру «*ModelDataInput*», тоді як кожний стовпець корелюється з одним з атрибутів, таких як *FrameNumber*, *FrameTime*,

FrameLen тощо. Таким чином, елемент x_{ij} у матриці X є значенням j-го атрибута i-го екземпляру «*ModelDataInput*»:

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix}, \quad (2.2)$$

Вектор Y формується з значень поля *Normality* кожного екземпляру *ModelDataInput*, представляючи собою:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}, \quad (2.3)$$

де y_i може бути 1 (атака) або 0 (не атака).

Після тренування моделі, прогнози $\hat{Y}$ отримуються за допомогою сигмоїдної функції $h_\theta(x)$:

$$\hat{Y} = h_\theta(x) = \frac{1}{1 + \exp(-X\theta)}, \quad (2.4)$$

Ці прогнози заносяться у поле «*PredictedLabel*» класу «*ModelDataOutput*» наступним чином:

$$ModelDataOutput.Prediction = \begin{cases} 1, \text{якщо } \hat{y}_i \geq 0{,}5 \\ 0, \text{якщо } \hat{y}_i < 0{,}5 \end{cases}, \quad (2.5)$$

Така інтеграція з *ModelDataInput* та *ModelDataOutput* не лише надає структурованого представлення вхідних та вихідних даних, але й забезпечує ефективність їх використання для навчання та прогнозування в рамках моделі БФГШ, що сприяє підвищенню точності та надійності системи виявлення вторгнень.

2.3 Вибір та обґрунтування технологій реалізації системи

При підході до створення системи розумного дому, необхідно врахувати декілька критичних аспектів при визначенні технологічних елементів. Серед цього

- детальний аналіз вимог до системи, а також оцінка ефективності, масштабованості, безпеки даних, і вибір оптимальних програмних інструментів і баз даних. Важливо також звернути увагу на сумісність компонентів, їх здатність до інтеграції з іншими системами, і потенціал для подальшого розвитку проекту.

В контексті вибору мови програмування для розробки, особливо в науковій сфері, кожен вибір має бути обґрунтованим і виваженим. Важливо провести всебічний аналіз можливих мов, враховуючи обчислювальну ефективність, наявність спеціалізованих бібліотек, зручність синтаксису для специфічних задач, та можливість інтеграції з іншими технологіями.

У табл. 2.2 представлено порівняльний аналіз розглянутих мов програмування, що сприятиме вибору найбільш підходящої для проекту.

Таблиця 2.2 – Порівняльний аналіз мов програмування

Критерій	C#	C++	Python
Обчислювальна ефективність	Висока	Висока	Помірна
Доступність спеціалізованих бібліотек	Висока	Висока	Дуже висока
Зручність синтаксису	Висока	Помірна	Висока
Безпека типів	Висока	Помірна	Помірна
Автоматичне керування пам'яттю	Є	Немає	Є
Підтримка паралельного програмування	Висока	Висока	Помірна
Вбудовані засоби для розробки UI	Висока (WinForms, WPF)	Помірна (Qt)	Помірна (Tkinter, PyQt)
Швидкість розробки	Висока	Помірна	Висока
Вбудовані засоби для роботи з мережами	Висока (WinForms, WPF)	Помірна (Qt)	Помірна (Tkinter, PyQt)
Доступність спеціалізованих бібліотек	Висока	Висока	Дуже висока
Зручність синтаксису	Висока	Помірна	Висока

З урахуванням аналізу, вибір мови програмування C# для розробки інтелектуальної системи управління домом виглядає обґрунтованим, особливо з

огляду на сучасні технологічні потреби. Нижче наведено кілька ключових аспектів, які підкреслюють переваги C#:

– висока обчислювальна ефективність. C# ефективно справляється з завданнями, які вимагають інтенсивних обчислень, що важливо для систем з складними розрахунками;

– надійність завдяки сильній системі типізації. У C# реалізована сувора система типів, яка знижує ризик помилок, критичних для фінансових і інших точних систем;

– автоматизоване управління пам'яттю. Збирач сміття в C# забезпечує ефективне управління пам'яттю, знижуючи ризики при роботі з великими даними.

– Підтримка паралельного програмування: C# містить інструменти для реалізації паралельних і асинхронних операцій, що є важливим для задач з високими вимогами до обчислювальної потужності;

– ефективність розробки. Завдяки простоті синтаксису та наявності розвинених бібліотек, C# дозволяє прискорити процес розробки, що важливо для проектів з обмеженими термінами.

Отже, C# виглядає як ідеальний вибір для розробки високопродуктивної, надійної та легко інтегрованої системи, що відповідає усім вимогам цього проекту.

Визначення відповідної системи управління базами даних (СУБД) має вирішальне значення для успішної реалізації будь-якої сучасної інформаційної системи, особливо у сфері розумних будинків. При виборі СУБД важливо враховувати не тільки її продуктивність та масштабованість, але й такі критичні фактори, як безпека і надійність зберігання даних, а також здатність СУБД до ефективного аналізу та обробки великих обсягів інформації. Крім того, сумісність СУБД з обраною мовою програмування має ключове значення для забезпечення гладкості процесу розробки та зручності подальшої експлуатації системи. Це може значно вплинути на швидкість розвитку проекту і його технічну підтримку в майбутньому.

У табл. 2.3, де викладено порівняльний аналіз ключових характеристик таких популярних СУБД, як MS SQL Server, Oracle та MS Access.

Таблиця 2.3 – Порівняльний аналіз СУБД

Параметр	MS SQL Server	Oracle Database	MS Access
Швидкість роботи	Висока	Висока	Середня/Низька
Масштабованість	Висока	Висока	Обмежена
Безпека	Висока	Висока	Середня
Надійність	Висока	Висока	Середня
Можливість глибокого аналізу	Високий рівень (BI інтеграція)	Високий рівень (BI опції)	Обмежена
Інтеграція з Microsoft продуктами	Висока (C#, .NET, Azure)	Обмежена	Висока (MS Office)
Ліцензійна політика	Ліцензія на користувача	Ліцензія на ядро або користувача	Входить у MS Office
Спеціалізовані бібліотеки	Широкий асортимент	Широкий асортимент	Обмежена кількість
Комплексність налаштування	Середнє/Високе	Високе	Низьке
Підтримка хмарних рішень	Вбудована (Azure)	Доступна	Обмежена

З аналізу випливає, що MS SQL Server відповідає високим вимогам продуктивності, безпеки та надійності, і пропонує додаткові переваги в контексті інтеграції з іншими продуктами Microsoft. Ці характеристики роблять його оптимальним варіантом для реалізації проекту з оптимізації інвестиційних портфелів, особливо коли розробка проводиться на C#. Його сумісність із середовищем Microsoft, включаючи інструменти розробки та інфраструктуру, забезпечує ефективне впровадження та підтримку системи, підвищуючи загальну ефективність робочих процесів.

2.4 Розробка алгоритмів для ідентифікації та класифікації потенційних загроз

Розробка алгоритмів для системи розумного будинку вимагає глибокого аналізу та обробки даних, отриманих із мережевого трафіку в домашніх системах. Основою для цього є зібрані дані, які включають параметри, такі як номер кадру , час відправки, довжина кадру, адреси джерела та призначення Ethernet, IP-адреси джерела та призначення, протокол IP, довжина IP-пакета, довжина TCP-сегменту, порти вихідних та цільових TCP, а також значення і нормальність даних. Після збирання цих даних створюється датасет, за допомогою якого можна проводити тренування моделей для виявлення вторгнень в мережу розумного дому. На рис. 2.4 представлена структурна схема процесу навчання моделі.

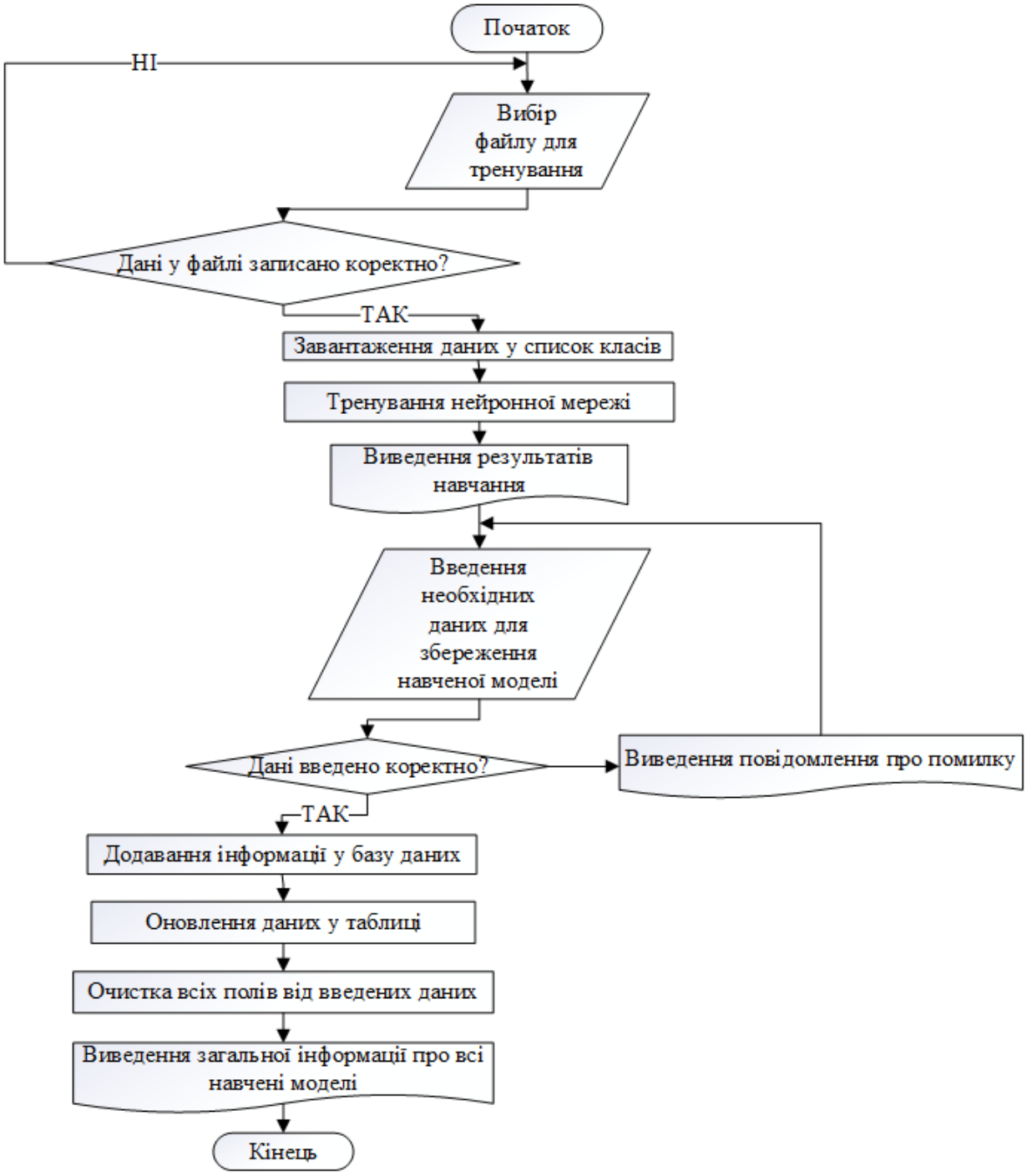

Рисунок 2.4 – Алгоритм навчання та зберігання даних моделі

Алгоритм для тренування та збереження моделі машинного навчання включає кілька кроків: спочатку обирається файл з даними для тренування, які перевіряються на коректність. Після завантаження цих даних, відбувається тренування моделі, результати якого виводяться для оцінки її ефективності. Далі користувач вводить дані для збереження моделі, і, якщо дані введені правильно, вони додаються до бази даних. Останній крок - очищення введених полів і виведення інформації про всі навчені моделі, завершуючи цикл роботи алгоритму.

На рис. 2.5 представлено схематичне зображення процесу функціонування навченої моделі в умовах реального часу.

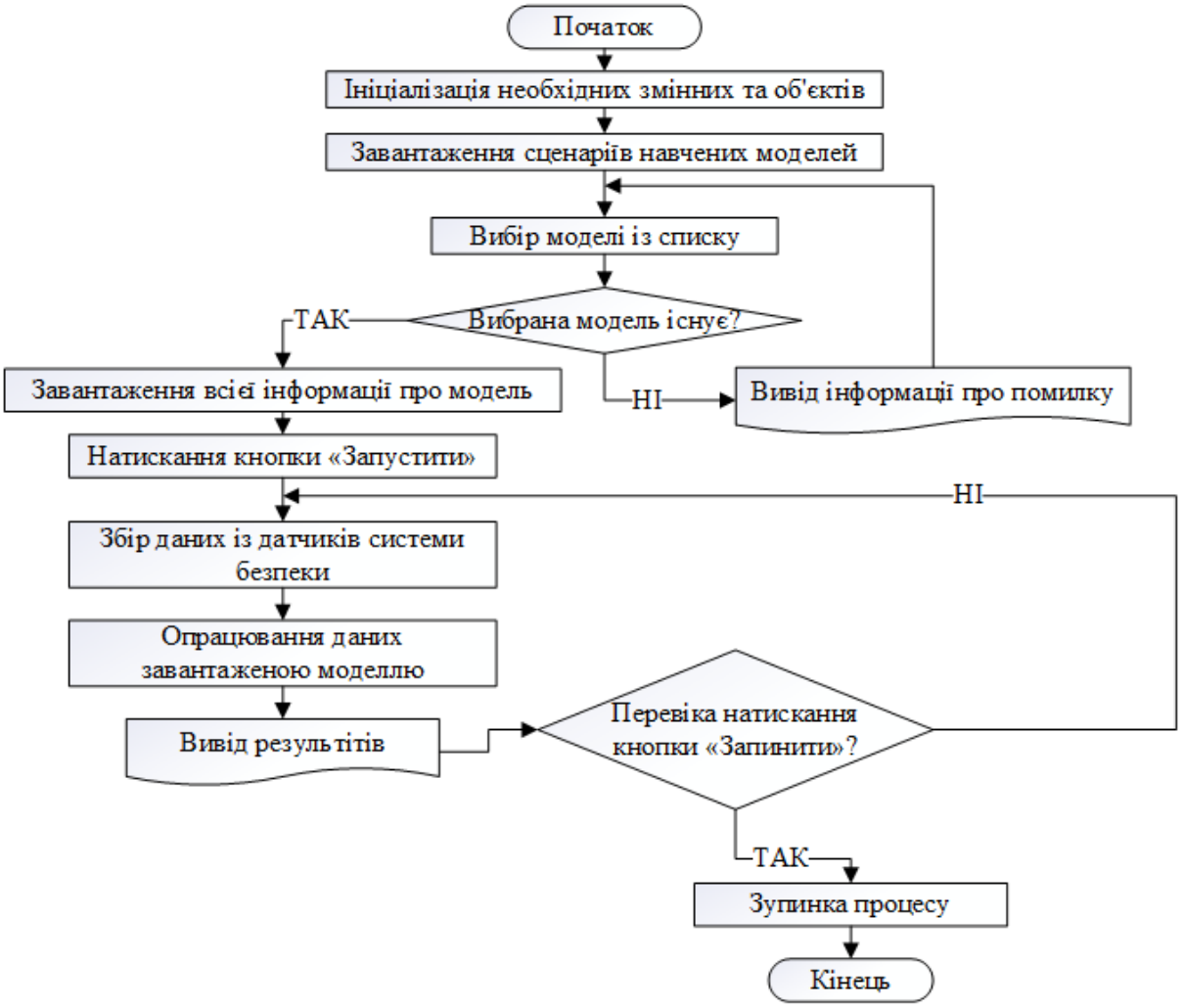

Рисунок 2.5 – Алгоритм роботи моделі

Процедура використання моделі для аналізу даних в реальному часі розгортається в декілька ключових кроків. Спершу відбувається ініціація всіх потрібних параметрів і об'єктів. Далі відбувається завантаження доступних моделей, після чого користувач вибирає потрібну з них з переліку. У разі доступності обраної моделі, здійснюється її активація. За допомогою кнопки «Запустити» запускається процес збору даних з датчиків та їх аналіз за допомогою вибраної моделі. Результати цього аналізу надаються користувачу. В будь-який момент цей процес можна припинити, використовуючи кнопку «Зупинити».

2.5 Вибір та обґрунтування комплектуючих для системи «Розумний будинок»

Контролери в системах «Розумний будинок» відіграють ключову роль, об'єднуючи та керуючи різними пристроями у будинку. Вони дозволяють керувати освітленням, кліматом, безпекою та іншими функціями через мобільні додатки, панелі або голосові команди. Контролери інтегрують безпекові компоненти, такі як датчики руху та камери спостереження, забезпечуючи моніторинг та реагування на підозрілі дії в реальному часі. Користувачі отримують сповіщення про будь-які виявлені події. Серед популярних контролерів можна вмділити: Arduino UNO, Raspberry Pi Zero та STM32 Blue Pill.

Arduino UNO стоїть на передовій популярних платформ для розробки електронних проектів, від базових до більш складних (рис. 2.6). Ця платформа знаходить широке застосування в системах «Розумний будинок» для керування та автоматизації різних домашніх функцій.

Рисунок 2.6 – Вигляд контролера «Arduino UNO»

Завдяки своїй універсальності та простоті в програмуванні, Arduino UNO є відмінним вибором для тих, хто займається проектами в сфері «Розумного дому», як для аматорів, так і для професіоналів. Ця платформа легко інтегрується з різними модулями та датчиками, що дозволяє створювати налаштовані рішення для конкретних потреб. Arduino UNO також використовується для моніторингу споживання енергії в домі, допомагаючи оптимізувати витрати та сприяти ефективному використанню ресурсів. Крім того, платформа ефективна для створення систем безпеки, які можуть включати датчики руху, системи

відеоспостереження та автоматичні замки, забезпечуючи надійний захист для будинку.

Таблиця 2.4 – Технічні характеристики «Arduino UNO»

Характеристика	Опис
Мікроконтролер	ATmega328P
Робоча напруга	5 В
Вхідна напруга	7-12 В
Цифрові входи/виходи	14 (6 із яких можуть служити як PWM)
Аналогові входи	6
Постійний струм на вході/виході	20 мА
Постійний струм для 3.3В виходу	50 мА
Частота кристала	16 МГц
USB-з'єднання	Так
ICSP заголовок	Так
Кнопка скидання	Так

Arduino UNO є відмінним інструментом для розробки в «Розумному домі», забезпечуючи гнучкість та легкість у програмуванні. Він відкриває широкі можливості для автоматизації домашніх завдань, включаючи контроль освітлення, клімату, безпеки та інших систем [24]. Завдяки своїй багатофункціональності, Arduino UNO є популярним вибором серед DIY ентузіастів та творців, які прагнуть створити ефективні та доступні розумні домашні рішення.

Raspberry Pi Zero, який відображений на рис. 2.7, відрізняється своєю компактністю та високою продуктивністю, що робить його популярним вибором для розробки інноваційних електронних проектів, включаючи системи «Розумний будинок». Ця плата ідеально підходить для автоматизації домашніх завдань, забезпечуючи можливість підключення різноманітних датчиків та пристроїв для контролю над такими аспектами як температура, вологість, освітлення, та безпека. Завдяки вбудованому Wi-Fi, Raspberry Pi Zero забезпечує легкий доступ до мережі та дозволяє віддалене управління системами [25].

Рисунок 2.7 – Вигляд контролера «Raspberry Pi Zero»

Таблиця 2.5 – Технічні характеристики «Raspberry Pi Zero»

Характеристика	Опис
Центральний процесор	Одноядерний
Оперативна пам'ять	512 МБ
Підключення	Mini HDMI, USB On-The-Go порт
Підтримка Wi-Fi	Так
Підтримка Bluetooth	Так
GPIO піни	Для підключення датчиків та пристроїв
Мікро SD слот	Для зберігання даних та ОС
USB-з'єднання	Micro USB
Вихідне відео	Mini HDMI
Робоча напруга	3.3 В

Використання Raspberry Pi Zero в «Розумному домі» є вдалим завдяки його розмірам та інтеграційним можливостям, включаючи Wi-Fi та Bluetooth. Його основний одноядерний процесор та 512 МБ RAM забезпечують достатню потужність для управління базовими завданнями домашньої автоматизації, а GPIO піни пропонують гнучкість для підключення зовнішніх модулів та датчиків. Таким чином, Raspberry Pi Zero виступає як практичне рішення для розробки та вивчення домашніх автоматизованих систем.

STM32 Blue Pill, представлений на рис. 2.8, є мікроконтролерною платою, яка ефективно використовується в різноманітних електронних проектах, від простих до складніших систем [26]. Вона стала популярним вибором у реалізації систем «Розумний будинок» завдяки своїм можливостям підключення до різних датчиків

та пристроїв. Це дозволяє користувачам гнучко управляти домашніми процесами, включаючи освітлення, клімат-контроль, та системи безпеки.

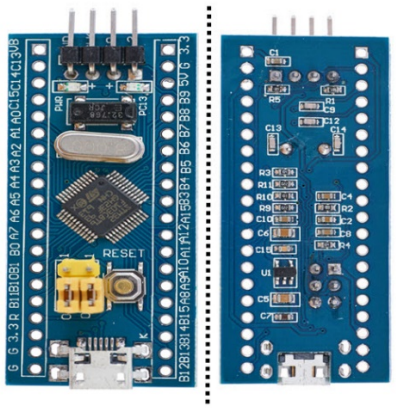

Рисунок 2.8 –Вигляд контролера «STM32 Blue Pill»

Таблиця 2.8 – Технічні характеристики «STM32 Blue Pill»

Характеристика	Опис
Мікроконтролер	STM32F103C8T6
Робоча напруга	3.3 В
Вхідна напруга	7-12 В
Цифрові входи/виходи	37 (всі GPIO піни)
Аналогові входи	10
Постійний струм на вході/виході	20 мА (на пін)
Частота кристала	72 МГц
USB-з'єднання	Micro USB
Пам'ять	64 КБ Flash, 20 КБ SRAM
Додаткові функції	CAN, I2C, USART, SPI, USB, ADC

STM32 Blue Pill, оснащений ARM Cortex-M3, вирізняється своєю обчислювальною потужністю та різноманітними можливостями підключення. Його численні входи/виходи роблять плату ідеальною для створення інтегрованих та складних рішень у сфері домашньої автоматизації. Завдяки цим характеристикам, STM32 Blue Pill є відмінним вибором для розробки різноманітних розумних домашніх систем і додатків.

У наведеній табл. 2.9 порівняно ключові характеристики трьох популярних мікроконтролерних плат: Arduino UNO, Raspberry Pi Zero та STM32 Blue Pill. Це порівняння допомагає виділити їхні сильні та слабкі сторони.

Таблиця 2.9 – Порівняльні характеристики мікроконтролерів

Характеристика	Arduino UNO	Raspberry Pi Zero	STM32 Blue Pill
Простота використання	Висока (підходить для новачків)	Середня (потребує деяких навичок)	Середня (потребує певних знань)
Спільнота та підтримка	Велика спільнота, багато ресурсів	Численна спільнота, але менше ресурсів для новачків	Спільнота менша, менше ресурсів
Ціна	Низька	Низька	Низька
Легкість програмування	Простий та зрозумілий синтаксис	Складніший, потребує знання Linux	Складніший, потребує глибших знань
Розмір спільноти	Найбільша	Велика, але зосереджена на специфічних задачах	Менша, орієнтована на спеціалістів
Навчальні матеріали	Багато доступних посібників та проектів для початківців	Доступні матеріали, але більше орієнтовані на просунутих користувачів	Обмежені навчальні матеріали

Arduino UNO є відмінним вибором для проектування системи розумного будинку завдяки своїй простоті використання та великій спільноті підтримки, що робить його доступним для розробників будь-якого рівня. Його гнучкість у підключенні різноманітних датчиків та модулів дозволяє ефективно інтегрувати та управляти різними аспектами домашнього середовища. Arduino UNO також забезпечує надійність та стабільність, що є критично важливим для систем автоматизації дому. Наявність численних навчальних ресурсів та проектів спрощує процес впровадження інноваційних рішень в системах розумного будинку.

При розробці системи безпеки розумного будинку важливо обирати апаратні компоненти, зосереджуючись на їх надійності, точності та високій якості. Основні

елементи, такі як датчики руху, температурні датчики та мікрофонні модулі, є вирішальними для забезпечення реакції на зміни в домашньому середовищі.

Датчик руху pir-d203s

Використання датчиків руху є ключовим аспектом в розробці систем безпеки та автоматизації. У табл. 2.10 представлено аналіз характеристик різних моделей датчиків руху.

Таблиця 2.10 – Порівняльний аналіз датчиків руху

Характеристика	PIR-D203S	HC-SR501	AM312
Дальність виявлення руху	до 5 м	до 7 м	до 3 м
Кути виявлення руху	120°	110°	100°
Напруга для живлення датчика	3-6V	4.5-20V	2.7-12V
Споживана потужність	низька	висока	помірна
Чутливість датчика	висока	середня	висока
Час затримки датчика	налаштовується	налаштовується	2с
Розміри датчика	компактні	стандартні	компактні

Вибір датчика PIR-D203S для «Розумного дому» пропонує перевагу у вигляді широкого кута охоплення 120°, дозволяючи покрити велику площу без додаткових датчиків [27]. З дальністю детекції 5 метрів, він є достатнім для багатьох домашніх та комерційних застосувань. Низька споживана потужність та налаштовуваний час затримки роблять PIR-D203S економічно вигідним та гнучким у застосуванні, що важливо для розумних систем. На рисунку 2.9 представлено зовнішній вигляд датчика PIR-D203S, який ефективно виявляє рух людини або тварини за допомогою інфрачервоного випромінювання, змінюючи відбите випромінювання і генеруючи електричний сигнал для активації системних функцій.

Рисунок 2.9 – Датчик руху D203S

Температурний датчик DHT22

Температурні датчики, такі як DHT22, є критичними для систем контролю та автоматизації, включаючи проекти «Розумний будинок» та інші IoT-додатки [28]. Важливість точності та надійності цих датчиків не можна недооцінити, оскільки вони забезпечують ефективність роботи систем. DHT22 є одним з вибраних варіантів завдяки своїй високій точності та надійності, як показано у порівняльній таблиці 2.11 поряд з іншими датчиками, наприклад, DHT11 і DS18B20.

Таблиця 2.11 – Порівняльний аналіз температурних датчиків

Характеристика	DHT22	DHT11	DS18B20
Діапазон вимірювання температур	-40°C до +80°C	0°C до 50°C	-55°C до +80°C
Точність вимірювання температур	±0.5°C	±2°C	±0.5°C
Діапазон вимірювання вологості	0% до 100%	5 - 95% RH	Не використовується
Точність вимірювання вологості	±2% RH	±5% RH	Не використовується
Інтерфейс	Цифровий	Цифровий	Цифровий
Частота оновлення даних (секунд)	2	1	0,75

Таким чином, DHT22 випереджає DHT11 завдяки більш широкому діапазону вимірювань температури та вологості та вищій точності цих вимірювань. Ці властивості роблять DHT22 більш гнучким у використанні в різноманітних умовах. Відрізняючись від DS18B20, DHT22 має здатність вимірювати не тільки температуру, а й вологість, що забезпечує йому перевагу у сценаріях, де важливо відстежувати обидва ці показники.

Датчик DHT22, зображений на рис. 2.10, поєднує в собі ємнісний датчик вологості та термістор, а також оснащений вбудованим АЦП для точного перетворення аналогових даних про вологість та температуру.

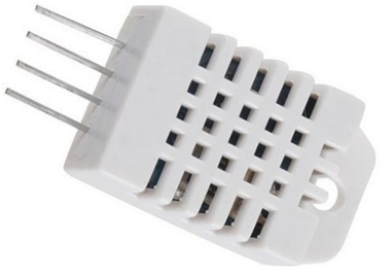

Рисунок 2.10 – Температурний датчик DHT22

DHT22 має здатність вимірювати температуру в діапазоні від -40 до +80°C з точністю до +/- 0.5°C та відносну вологість повітря в діапазоні від 0 до 100% з точністю до +/- 2%. Його цифровий вихід і сумісність з мікроконтролерами роблять DHT22 легким у використанні та інтеграції в різноманітні проекти. У системі розумного дому температурний датчик DHT22 буде виконувати моніторинг клімату.

Магнітний датчик Reed module KY-025

Магнітні датчики знаходять застосування у широкому спектрі проектів, від базових домашніх до складних промислових систем [29]. В табл. 2.12 представлений порівняльний аналіз різних типів датчиків, які можуть бути використані для створення систем «Розумний будинок».

Таблиця 2.12 – Порівняльний аналіз магнітних датчиків

Характеристика	Reed Module KY-025	Hall Effect Sensor (A3144)	Magnetoresistive Sensor (HMC5883L)
Напруга живлення	3.3V - 5.5V	4.5V - 24V	2.16V - 3.6V
Розміри плати	1.5cm x 3.6cm	Залежить від моделі	Залежить від моделі
Вихідний сигнал	Цифровий	Цифровий	Аналоговий
Додаткові компоненти	LM393 компаратор	Немає	Немає
Чутливість до сильних магнітних полів	Висока	Середня	Середня

Датчик Reed Module KY-025 вирізняється легкістю інтеграції завдяки вбудованому LM393 компаратору та високій чутливості до магнітних полів. Ці характеристики роблять його ідеальним для застосувань, де важливі простота використання та надійність.

Датчик Reed module KY-025, представлений на рис. 2.11, функціонує, виявляючи магнітне поле та змінюючи свій режим із відкритого на закритий або навпаки в присутності магніту. Цей модуль містить в собі два металеві контакти у скляному корпусі, які реагують на магнітне поле, замикання або розмикання контактів.

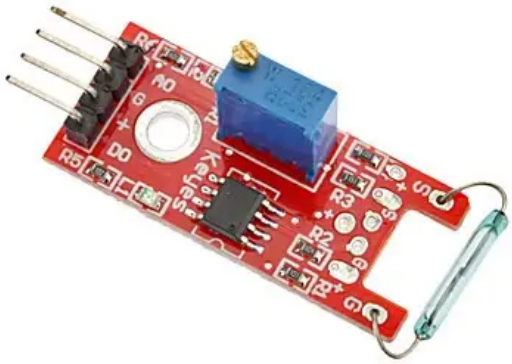

Рисунок 2.11 – Датчик Reed module KY-025

У захисній системі розумного будинку, Reed module KY-025 використовується для моніторингу відкриття або закриття дверей та вікон. Контакти датчика замикатимуться або розмикатимуться відповідно до руху магніту, прикріпленого до дверей або вікна, що дозволяє системі фіксувати їх стан.

Інтегруючи Reed module KY-025 в систему безпеки, можна автоматично відстежувати незаконні входи, сповіщаючи власників або активуючи тривогу. Цей датчик, простий у використанні та надійний, стає важливою частиною системи домашньої автоматизації, підвищуючи безпеку та дозволяючи реалізовувати різні автоматизовані функції. Крім того, використання Reed module KY-025 дозволяє збирати важливі дані для адаптивної роботи інших елементів системи безпеки, забезпечуючи розумний захист.

Фоторезистор KY-018

Фоторезистори є частим вибором для багатьох проектів, від основних освітлювальних систем до складніших автоматизаційних рішень. У табл. 2.13 представлено аналіз різних фоторезисторів, що підходять для використання у проектах.

Таблиця 2.13 – Порівняльний аналіз фоторезисторів

Характеристика	KY-018	GM5528	GL5516
Напруга живлення	3.3V - 5V	Залежить від підключення	Залежить від підключення
Вихідний сигнал	Аналоговий	Аналоговий	Аналоговий
Тіньовий опір	500 кОм	1.0 МОм	500 кОм

Розміри датчиків	Компактні	Залежать від моделі	Залежать від моделі
Чутливість до освітдення	Висока	Висока	Висока
Робоча температура	-40°C до +85°C	-30°C до +70°C	-30°C до +70°C

Фоторезистор KY-018 є вдалим вибором для Arduino-базованих проектів, як вказано в таблиці 2.4. Однією з ключових переваг KY-018 є його робоча напруга від 3.3V до 5V, що відповідає стандарту більшості Arduino-плат і забезпечує легку сумісність. Його тіньовий опір 500 кОм виявляється більш ефективним для багатьох застосувань, у порівнянні з фоторезистором GM5528 з тіньовим опором 1 МОм [30]. Компактність KY-018 робить його практичним для проектів з обмеженим простором, а висока чутливість до світла дозволяє точно вимірювати освітленість навіть у слабо освітлених умовах. Ширший діапазон робочих температур KY-018 порівняно з GM5528 і GL5516 робить його більш універсальним у різних умовах. Також KY-018 є готовим до використання модулем для Arduino, що спрощує інтеграцію та робить його доступним для розробників різних рівнів.

Фоторезистор KY-018, зображений на рис. 2.12, є світлочутливим компонентом, який регулює свій електричний опір відповідно до рівня освітленості. Це забезпечує можливість використання змін у опорі для контролю електронних схем, реагуючи на світлові умови оточення. У розумних системах захисту будинку фоторезистори можуть використовуватися для визначення присутності світла, що може сигналізувати про присутність людей або керувати освітленням автоматично.

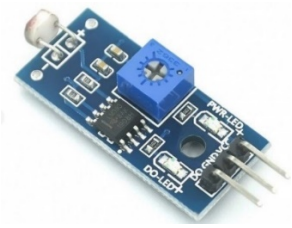

Рисунок 2.12 – Фоторезистор KY-018

У складі системи безпеки дані з фоторезистора KY-018 можуть використовуватися для навчання моделей розрізняти звичайні та незвичайні

варіації в освітленості, дозволяючи системі точніше виявляти можливі загрози або нестандартні ситуації, що вимагають втручання.

Мікрофонний модуль MAX9814

Вибір мікрофонного модуля може значно покращити якість звукозапису та функціональність загальної системи. У таблиці 2.14 аналізуються три популярні мікрофонні модулі - MAX9814, LM4881 та MAX4466, для визначення їхніх особливостей і переваг, що допоможе вибрати найбільш підходящий для специфічних потреб.

Таблиця 2.14 – Порівняльний аналіз мікрофонних модулів

Характеристика	MAX9814	LM4881	MAX4466
Напруга живлення	2,7 В - 5,5 В	2.7 В до 5.5 В	2.4 - 5.5 В
Тип підсилювача	Мікрофонний	Мікрофонний	Мікрофонний
Автоматичне регулювання AGC	Так	Ні	Ні
Налаштування посилення	40dB, 50dB, 60dB	Змінне	Змінне
Вихідний сигнал	Аналоговий	Аналоговий	Аналоговий

Для даного проекту було вибрано модуль MAX9814, оскільки він надає значні переваги, такі як вбудоване автоматичне регулювання посилення, яке забезпечує постійний рівень аудіосигналу, незалежно від змін відстані до джерела звуку. Модуль MAX9814 пропонує різні рівні посилення (40dB, 50dB, 60dB), забезпечуючи велику гнучкість у різних аудіо-сценаріях. Його низький рівень шуму також є важливим для високоякісного аудіозапису.

Мікрофонний модуль MAX9814, показаний на рис. 2.13, представляє собою високоякісний аудіопідсилювач з функцією автоматичного контролю виграшу, ідеально підходить для широкого спектра звукових систем. Цей модуль складається з електретного мікрофону для вловлювання звуку та вбудованого підсилювача, що збільшує амплітуду аудіосигналу.

Рисунок 2.13 – Модуль MAX9814

Одна з важливих характеристик MAX9814 - це здатність автоматично регулювати виграш, адаптуючись до різних рівнів шуму без втручання користувача. У захисних системах розумних будинків, MAX9814 використовується для виявлення акустичних подій, які можуть сигналізувати про потенційні небезпеки або надзвичайні ситуації. Його інтеграція з Arduino UNO вимагає підключення до входів/виходів та може потребувати додаткових компонентів, наприклад, опорів або релейних модулів, для керування навантаженнями, а також розробки програмного коду для обробки даних з датчиків.

2.6 Імплементація модулів системи та їх інтеграція в систему «Розумний будинок»

Запуск системи «Розумний будинок» починається з інтеграції та конфігурації різних датчиків з Arduino UNO. Важливими елементами проекту є датчики, такі як PIR D203S для виявлення руху, DHT22 для моніторингу температури та вологості, Reed module KY-025 для відстеження відкриття або закриття дверей та вікон, фоторезистор KY-018 для визначення рівня світла, і мікрофонний модуль MAX9814 для запису звуку, які разом сприяють виявленню змін в домашньому середовищі.

Кожен із цих датчиків відповідає за збір певного типу інформації: PIR D203S реєструє рух, який може вказувати на наявність людей чи тварин; DHT22 забезпечує важливі дані про кліматичні умови, важливі для регулювання опалювальних та охолоджувальних систем; Reed module KY-025 фіксує стан дверей і вікон як частину системи безпеки; KY-018 відстежує освітленість, а MAX9814 виявляє звукові аномалії або зміни в акустичному середовищі.

Arduino UNO служить як основний інтерпретатор сигналів з датчиків системи «Розумний будинок», перетворюючи їх у дані, зрозумілі для системи, завдяки спеціально розробленому програмному забезпеченню. Отримані дані, які включають інформацію про рух, температуру, вологість, стан дверей та вікон, а також звукові сигнали, передаються на центральний сервер. На сервері ці дані зберігаються та аналізуються за допомогою розширених алгоритмів машинного навчання, які виявляють потенційні загрози в реальному часі, включаючи нестандартну активність або аномальні зміни умов навколишнього середовища, що можуть вказувати на ризик пожежі чи інші небезпеки.

Для користувачів розроблено спеціальний застосунок, який забезпечує взаємодію з базою даних та дозволяє моніторити стан дому в реальному часі. Цей додаток надає актуальну інформацію та оповіщення про будь-які виявлені ризики, що забезпечує не тільки негайну реакцію на потенційні небезпеки, але й дозволяє збирати цінні дані для аналізу та вдосконалення роботи системи. На рис. 2.14 ілюструється як взаємодіють між собою всі компоненти системи.

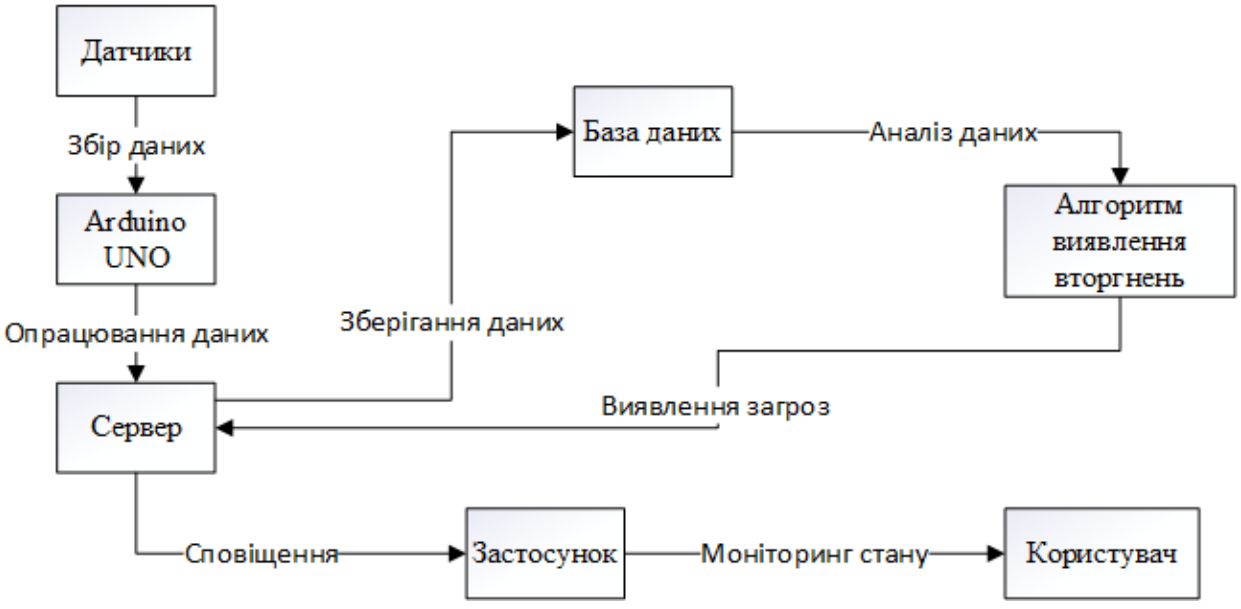

Рисунок 2.14 – Схема взаємодії компонентів системи

Таким чином, кожен датчик та модуль у системі не тільки виконує свою окрему функцію, але й вносить важливий вклад у створення комплексної, інтегрованої та інтелектуальної системи домашньої автоматизації.

2.7 Висновок

У рамках даного розділу було проведено глибокий аналіз методів машинного навчання, в результаті чого було обрано метод Бройдена-Флетчера-Гольдфарба-Шанно за його ефективність та точність. Розроблена математична модель системи спрямована на ідентифікацію вторгнень, що забезпечує важливий фундамент для створення надійної системи безпеки. Вибір технологій C# та MS SQL Server для реалізації системи «Розумний будинок» зумовлений їх високою продуктивністю та сумісністю.

У цьому розділі також були розроблені ключові алгоритми для навчання моделі та її ефективної роботи, які дозволяють системі точно класифікувати та реагувати на потенційні загрози. Підбір компонентів для системи, включаючи Arduino UNO та ряд специфічних датчиків, забезпечує комплексний підхід до моніторингу та управління домашнім середовищем, розширюючи функціонал та можливості системи. Впровадження та інтеграція цих модулів в систему «Розумний будинок» були визначальними для створення високоефективної та взаємопов'язаної системи, яка відображена на схемі взаємодії компонентів.

Отримані результати є фундаментом для подальшого розвитку системи в наступних розділах, де буде зосереджено увагу на детальному тестуванні та оптимізації системи, а також на розробці інтерфейсу користувача та впровадженні додаткових функцій для підвищення зручності та ефективності використання системи «Розумний будинок».

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1 Налаштування експериментального середовища

Розробка системи «Розумний будинок» із використанням Arduino UNO вимагає спочатку інсталяції Arduino IDE, що є основним інструментом для програмування цих мікроконтролерів. Після завантаження та встановлення Arduino IDE, наступний етап включає фізичне підключення Arduino UNO до комп'ютера за допомогою USB-кабелю. Це підключення ініціює активацію плати, про що свідчить індикатор «ON» та мерехтливий світлодіод «L», який демонструє роботу стандартної програми Blink.

Ключовим моментом в роботі з платою є визначення номеру COM-порту, який комп'ютер присвоїв Arduino UNO. Цей номер можна знайти, переглянувши вкладку «Порти (COM і LPT)» у Диспетчері пристроїв Windows, де відображається порт, призначений для Arduino UNO, як показано на рис. 3.1. Знання цього номеру COM-порту є суттєвим для налаштування з'єднання в Arduino IDE, що дозволяє програмному забезпеченню взаємодіяти з платою для подальшого програмування та використання.

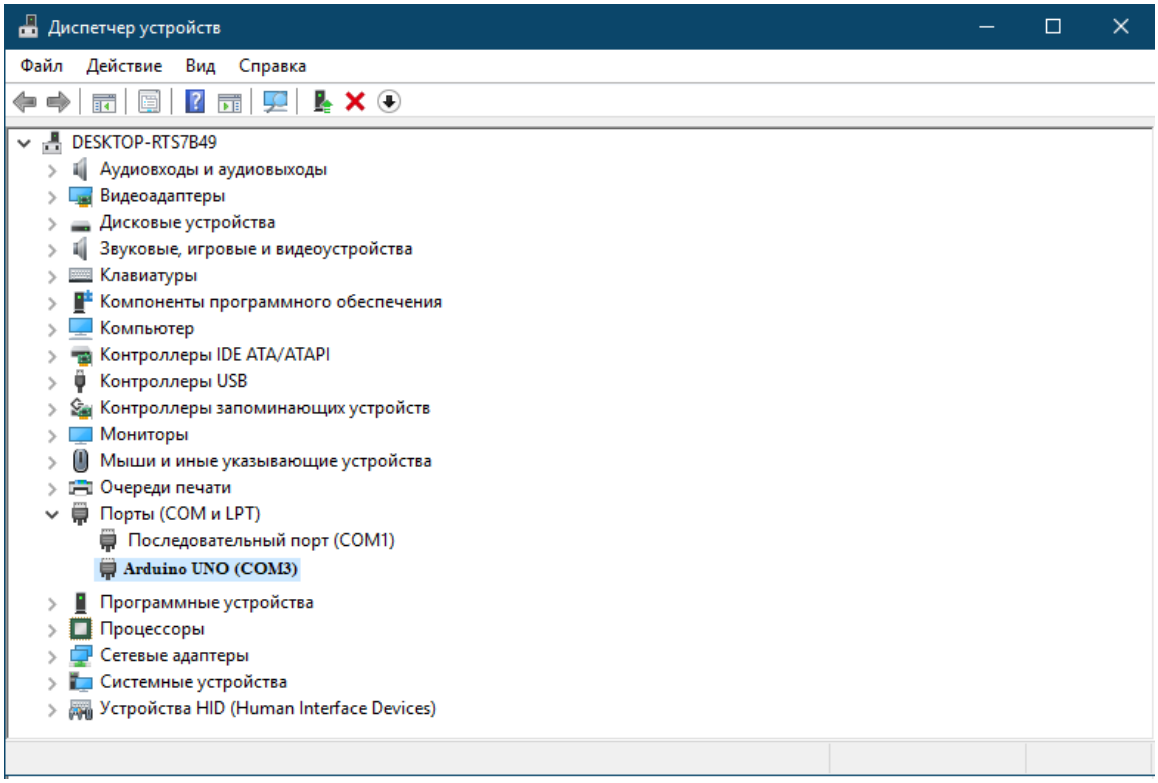

Рисунок 3.1 – Визначення COM-порту Arduino UNO

При першому підключенні Arduino UNO до комп'ютера, операційна система розпізнає та встановлює новий пристрій як COM-порт, автоматично вибираючи та інсталюючи потрібний драйвер. Після цього, система призначає порту конкретний номер, наприклад, «COM3». Важливо врахувати, що при кожному новому підключенні іншої плати Arduino, система може присвоїти їй унікальний номер порту. Таким чином, при роботі з декількома платами, кожна матиме свій індивідуальний номер порту.

Далі, в Arduino IDE потрібно вказати, що Arduino UNO підключено до «COM3». Це робиться через меню «Сервіс», де у розділі «Послідовний порт» користувач встановлює «COM3», як показано на рис. 3.2. Така настройка дозволяє Arduino IDE коректно комунікувати з платою, забезпечуючи можливість завантаження програм на контролер та отримання даних з нього.

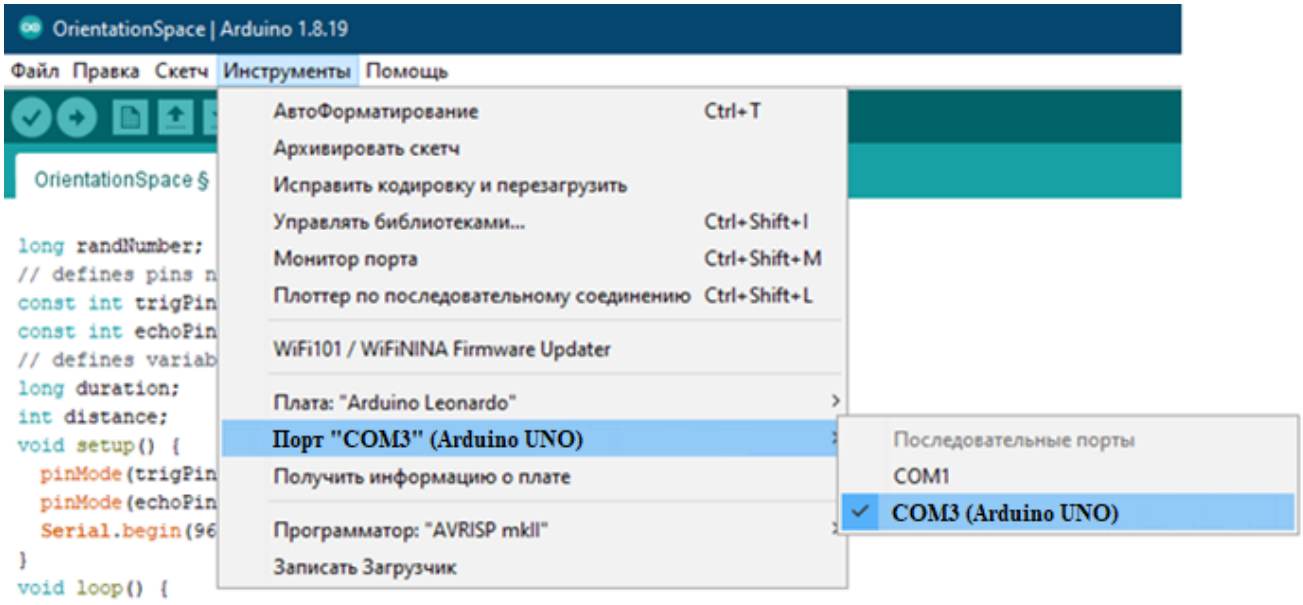

Рисунок 3.2 – Встановлення COM-порту в Arduino IDE

Встановлення правильного номера COM-порту для Arduino UNO забезпечує зв'язок між програмним середовищем та платою. Далі важливо в Arduino IDE вказати, що використовується саме модель Arduino UNO. Це досягається через меню «Сервіс» і підрозділ «Плата», де серед доступних опцій слід обрати «Arduino UNO». Таке налаштування дозволяє середовищу IDE відповідно адаптуватися до характеристик даної моделі плати.

Після налаштувань наступає етап програмування та тестування. У Arduino IDE використовується мова, заснована на C/C++, для написання програм. Скомпільована без помилок програма, підтверджена повідомленням «Компіляція завершена» в Arduino IDE, готова до завантаження та виконання на платі. Вдалий результат компіляції важливий, оскільки він підтверджує відсутність помилок у програмі та її сумісність з апаратними обмеженнями Arduino UNO.

Цей процес розробки програмного забезпечення, перевірки та забезпечення його сумісності з апаратною частиною є вирішальним для успішної реалізації проектів на базі Arduino UNO, як показано на рис. 3.3, та забезпечує основу для подальшого використання плати в різних застосуваннях.

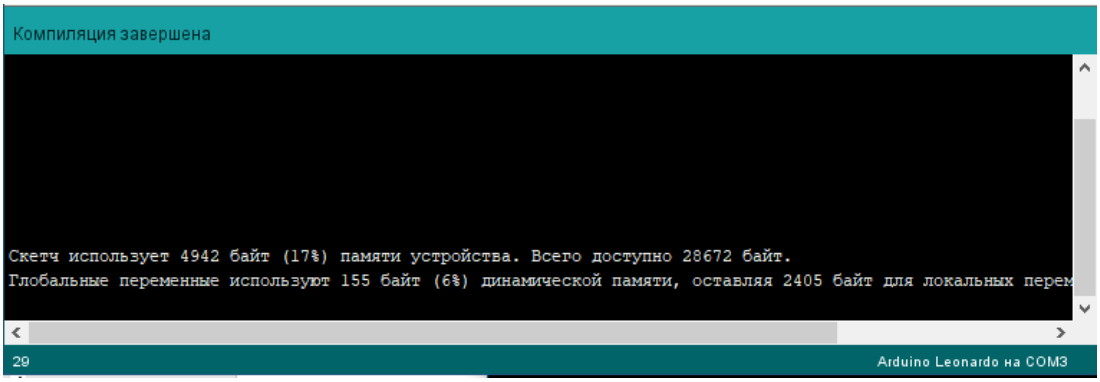

Рисунок 3.3 – Компіляція розробленого скетчу для контролеру

Після успішного завершення компіляції проекту без виявлення помилок наступає етап завантаження скомпільованої програми на контролер Arduino через Arduino IDE. Для цього, в інтерфейсі програми слід перейти до пункту меню «Скетч», де розміщена опція «Завантаження». Обравши цей пункт, розпочинається процедура передачі програми з комп'ютера на плату Arduino, яка є вирішальним кроком перед її активацією на контролері (рис. 3.4).

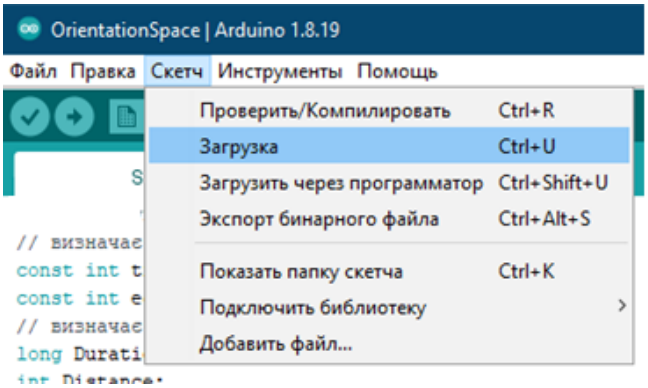

Рисунок 3.4 – Процес завантаження програми на Arduino

По завершенні процесу завантаження, в інтерфейсі Arduino IDE з'являється повідомлення в нижній частині екрана, підтверджуюче, що програма була успішно завантажена на контролер. Це повідомлення є важливим знаком успіху, оскільки вказує на те, що програма не тільки була правильно скомпільована, але й ефективно інтегрована в апаратну складову системи. Візуальне підтвердження цього процесу та інформаційного повідомлення можна спостерігати на прикладі, представленому на рис. 3.5.

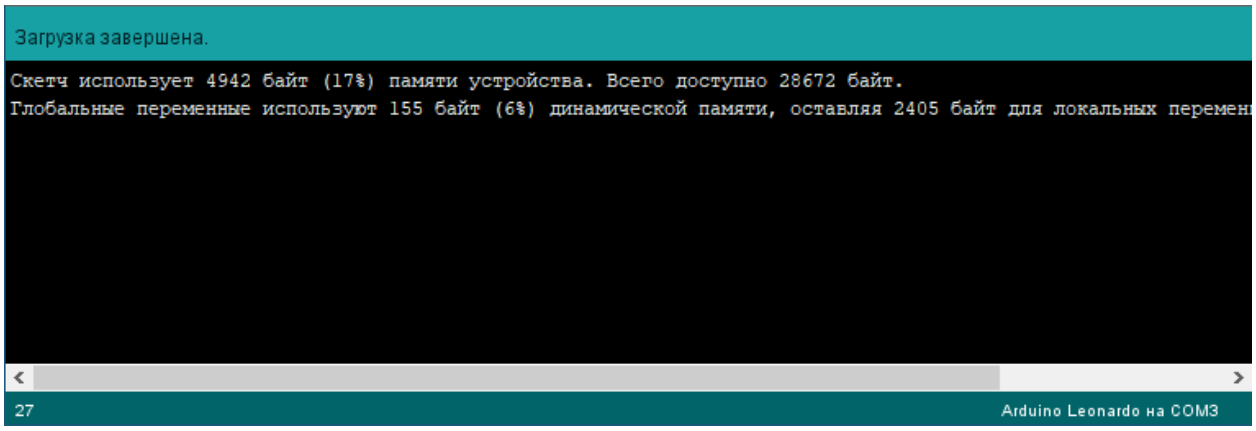

Рисунок 3.5 – Повідомлення про успішне завантаження

3.2 Розробка модулів системи

Під час розробки системи «Smart house» у середовищі Visual Studio 2022 ключовим кроком стала інтеграція бази даних. Важливою частиною цього процесу було створення змінної CONNECTING у файлі конфігурації App.config. Ця змінна містить всі необхідні дані для забезпечення з'єднання з базою даних, що є критично важливим для архітектури цілої системи. Детальний процес налаштування з'єднання з базою даних і конфігурації параметрів ілюстрований на рис. 3.6.

```
<!-- Підключення до бази даних -->
<appSettings>
  <add key="CONNECTING"
       value="Data Source=(LocalDB)\MSSQLLocalDB;
       AttachDbFilename=|DataDirectory|\DB.mdf;
       Integrated Security=True" />
</appSettings>
```

Рисунок 3.6 – Конфігурація параметрів з'єднання з базою даних

Під час розробки системи «Smart house» було обрано використання простору імен System.Data.SqlClient з бібліотеки .NET для ефективного зв'язку з реляційною базою даних SQL Server. Цей простір імен включає в себе набір класів, що спеціалізуються на реалізації з'єднань та виконанні SQL-запитів, гарантуючи надійну та безпечну взаємодію з базою даних, що мінімізує ризики неправильного оброблення інформації.

З погляду інтерфейсу користувача, ключовим елементом стала інтеграція компонента MenuStrip у основний графічний інтерфейс Windows Forms. MenuStrip є ефективним інструментом для створення інтуїтивно зрозумілого графічного меню з великою кількістю опцій. Це значно полегшує навігацію для користувачів та надає швидкий доступ до важливих функцій системи, як демонструється на рис. 3.7.

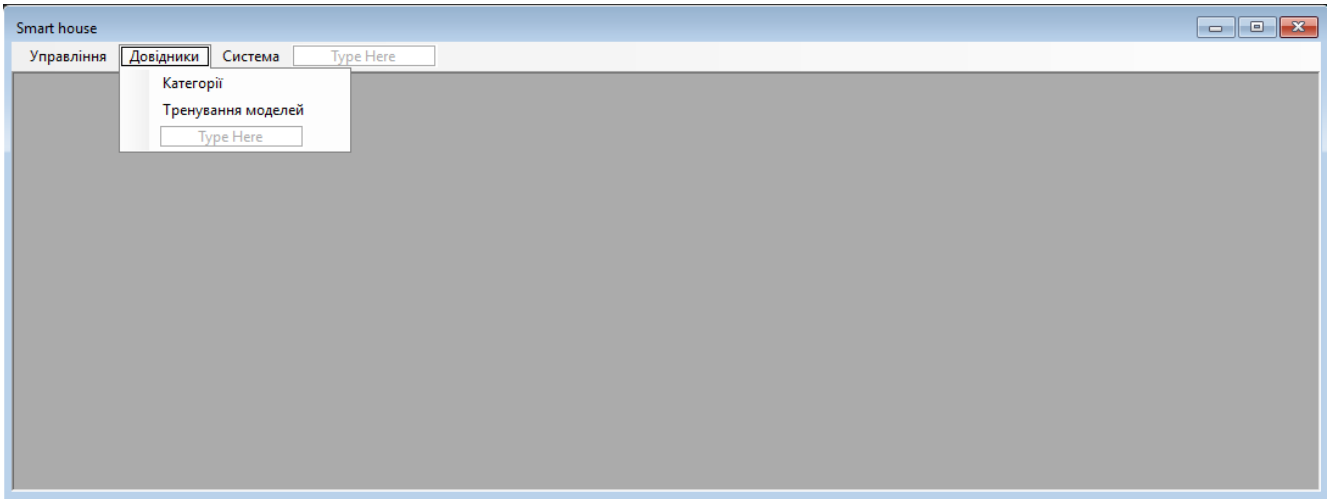

Рисунок 3.7 – Приклад головного інтерфейсу системи «Smart house»

Для кожного елемента меню в системі було створено специфічну реакцію, яка активується при його виборі користувачем. Як видно на рис. 3.8, коли користувач вибирає пункт «Тестування моделей», система відповідає на цей вибір, запускаючи відповідний інструментарій для тестування та аналізу моделей.

```
1 reference
private void тестуванняМоделейToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWindows();
    PredictorForm predictorForm = new PredictorForm();
    predictorForm.MdiParent = this;
    predictorForm.WindowState = FormWindowState.Maximized;
    predictorForm.Show();
}
```

Рисунок 3.8 – Тригер події для відкриття форми «Тестування моделей»

Створення інтерфейсу для тренування моделей машинного навчання відіграє важливу роль у розробці системи, при цьому зовнішній вигляд цього інтерфейсу ілюстрований на рис. 3.9.

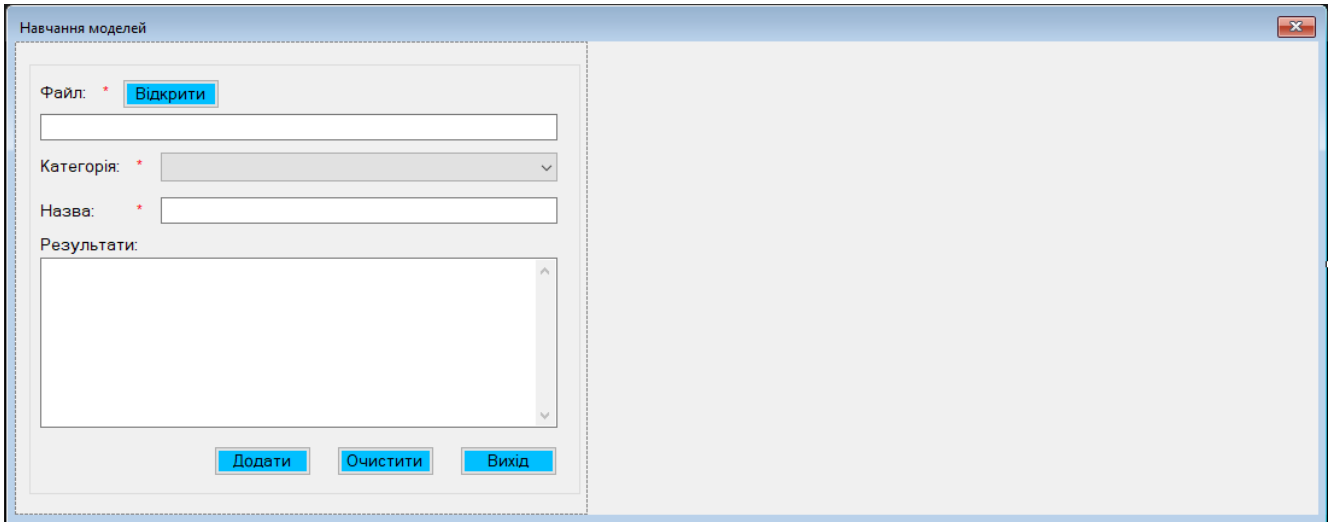

Рисунок 3.9 – Інтерфейс для навчання моделей

Також на рис. 3.10 представлена частина коду, яка надає користувачу можливість вибрати файл із системи файлів комп'ютера. Цей код також включає функціонал для визначення та передачі шляху обраного файлу, що дозволяє програмі використовувати його для подальшого процесу тренування моделей.

```
// Відображення діалогового вікна та обробка результату
if (openFileDialog.ShowDialog() == DialogResult.OK) {

  _Path = openFileDialog.FileName;
  FileNameTBox.Text = openFileDialog.FileName;
```

Рисунок 3.10 – Код для відкриття файлу із тренувальними наборами

Після відкриття файлу проходить створення контексту машинного навчання в програмі (рис. 3.11).

```
//Створення контексту
mlContext = new MLContext(seed: 0);
```

Рисунок 3.11 – Створення контексту

В цьому фрагменті коду відбувається ініціалізація екземпляра для машинного навчання за допомогою ML.NET, що є спеціалізованим фреймворком від Microsoft для реалізації машинного навчання у середовищі .NET. Команда «new

MLContext» ініціює створення нового об'єкта контексту машинного навчання, який є основою для подальшої роботи з різними алгоритмами машинного навчання.

Застосування параметра «seed: 0» у конструкторі MLContext встановлює стартове значення для внутрішнього генератора випадкових чисел. Вказівка конкретного seed забезпечує консистентність та повторюваність результатів навчання моделей. Це означає, що при повторному тренуванні моделей з цим же початковим значенням, результати будуть стабільно однаковими, що є критично важливим для об'єктивного тестування та оцінки різних моделей машинного навчання.

Після цього проходить завантаження даних із файлу в якому містяться тренувальні набори даних для подальшого їх використання (рис. 3.12).

```
//Завантаження даних із файлу
dataView = mlContext.Data.LoadFromTextFile<ModelDataInput>(
    path: FileNameTBox.Text,
    separatorChar: ',',
    hasHeader: true);
```

Рисунок 3.12 – Завантаження даних

У коді використовується метод LoadFromTextFile з простору імен «mlContext.Data». Метод призначений для читання даних з текстового файлу і конвертування їх у формат, придатний для машинного навчання. Аргумент ModelDataInput вказує на клас моделі, який буде використовуватися для інтерпретації завантажених даних. Це означає, що дані з файлу будуть відображені в структуру або формат, визначений у класі ModelDataInput. Параметр path описує шлях до файлу, який буде завантажено. Цей шлях отримується з текстового поля в графічному інтерфейсі користувача, яке дозволяє користувачеві вибрати файл для завантаження.

Параметр separatorChar вказує на символ-роздільник, який використовується в текстовому файлі. У цьому випадку, для розділення окремих значень використовується кома. Це важливо для правильного парсингу та визначення окремих колонок даних у файлі. Аргумент hasHeader повідомляє методу LoadFromTextFile, що в файлі перший рядок містить заголовки колонок. Це

дозволяє системі правильно інтерпретувати структуру даних та забезпечує, що заголовки не будуть оброблені як частина актуальних даних.

У фрагменті коду, що показано на рис. 3.13 виконується розділення датасету на дві частини: тренувальний набір даних та тестовий набір даних.

```
// Розділення даних на навчальний та тестовий набори
var splitData = mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);
// 20% для тестування
```

Рисунок 3.13 – Розділення даних датасету

Фрагмент коду виконує розбиття набору даних на дві частини: тренувальний та тестовий набори, використовуючи функції бібліотеки ML.NET. Ключова команда тут - TrainTestSplit, яка є частиною функціоналу ML.NET і служить для розділення набору даних на частину для тренування та частину для тестування моделі. Переменна splitData використовується для зберігання результату цього розділення. В даному випадку, dataView, яка містить завантажені дані, є вхідною змінною для методу TrainTestSplit. Аргумент testFraction вказує, що 20% даних з dataView будуть відокремлені та використані як тестовий набір. Це означає, що 20% даних будуть вибрані випадково (залежно від внутрішніх механізмів розділення в ML.NET) для тестування моделі, а решта 80% даних будуть використані для тренування моделі.

Після ініціалізації набору даних, наступним кроком у процесі розробки системи «Smart house» є їх підготовка до використання в машинному навчанні (рис. 3.14). Для цього, з використанням можливостей бібліотеки ML.NET, створюється пайплайн обробки даних. Основною задачею є трансформація і комбінування різних даних в єдину структуру, яка буде зручна для аналізу та тренування моделі машинного навчання. Важливим елементом обробки є використання методу Concatenate з контексту mlContext, який дозволяє об'єднати різноманітні властивості даних в один вектор ознак. Цей метод є критично важливим, оскільки він забезпечує формування оптимальної структури даних для ефективного навчання та оцінки моделі, гарантуючи, що всі необхідні характеристики враховуються в процесі машинного навчання.

```
var dataProcessPipeline = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "Label",
    inputColumnName: nameof(ModelDataInput.Normality))
    .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName: "EthSrcEncoded",
     inputColumnName: nameof(ModelDataInput.EthSrc)))
    .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName: "EthDstEncoded",
    inputColumnName: nameof(ModelDataInput.EthDst)))
    .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName: "IpSrcEncoded",
    inputColumnName: nameof(ModelDataInput.IpSrc)))
    .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName: "IpDstEncoded",
    inputColumnName: nameof(ModelDataInput.IpDst)))
    .Append(mlContext.Transforms.Concatenate("Features",
        nameof(ModelDataInput.FrameNumber),
        nameof(ModelDataInput.FrameTime),
        nameof(ModelDataInput.FrameLen),
        "EthSrcEncoded",
        "EthDstEncoded",
        "IpSrcEncoded",
        "IpDstEncoded",
        nameof(ModelDataInput.IpProto),
        nameof(ModelDataInput.IpLen),
        nameof(ModelDataInput.TcpLen),
        nameof(ModelDataInput.TcpSrcport),
        nameof(ModelDataInput.TcpDstport),
        nameof(ModelDataInput.Value)
    ))
    .AppendCacheCheckpoint(mlContext);
```

Рисунок 3.14 – Підготовка конвеєру даних для навчання

Фрагмент коду описує процес створення конвеєра обробки даних (pipeline) для машинного навчання за допомогою бібліотеки ML.NET. Конвеєр складається з декількох етапів обробки та трансформації даних, які підготовлюють набір даних для тренування моделі машинного навчання. Спочатку відбувається перетворення значень у вказаній колонці на ключі. Це робиться за допомогою методу MapValueToKey, де outputColumnName визначено як «Label», а inputColumnName вказує на властивість Normality з класу ModelDataInput. Це перетворення необхідне для підготовки цільової змінної (мітки) для процесу класифікації. Далі йде серія викликів OneHotEncoding, які перетворюють категоріальні змінні в бінарний формат, що є зручним для машинного навчання. Окремо кодується кожне поле: «EthSrc», «EthDst», «IpSrc», «IpDst», використовуючи відповідні властивості з класу ModelDataInput.

Наступний крок – це конкатенація (об'єднання) різних ознак в одну колонку «Features» за допомогою методу Concatenate. Це включає як початкові властивості ModelDataInput, так і закодовані категоріальні змінні. Це створює єдиний набір ознак, який використовуватиметься для тренування моделі. На закінчення, виклик AppendCacheCheckpoint додається до конвеєра. Це дозволяє кешувати результати

попередніх етапів обробки, оптимізуючи продуктивність шляхом зменшення необхідності повторної обробки даних при кожному проходженні через конвеєр.

Далі необхідно обрати алгоритм машинного навчання (рис. 3.15).

```
var trainer =
  mlContext.MulticlassClassification.Trainers.LbfgsMaximumEntropy(labelColumnName: "Label",
  featureColumnName: "Features");
```

Рисунок 3.15 – Вибір алгоритму навчання моделі

Код використовується для створення тренера (trainer) для моделі машинного навчання у контексті ML.NET, спеціалізованого на багатокласовій класифікації. Тренер - це компонент у ML.NET, який відповідає за навчання моделі на основі наданих даних. У цьому випадку використовується алгоритм класифікації «LbfgsMaximumEntropy» з набору тренерів для багатокласової класифікації, доступних у ML.NET. LbfgsMaximumEntropy є методом, що використовує алгоритм оптимізації обмеженої пам'яті БФГШ для максимізації ентропії. Це дозволяє моделі ефективно навчатися розпізнавати та класифікувати різні класи у наборі даних.

Параметр labelColumnName вказує на назву колонки в даних, яка містить мітки класів. Це означає, що тренер використовуватиме дані з колонки «Label» для визначення правильних відповідей під час тренування моделі. Параметр featureColumnName описує, яку колонку в даних слід використовувати як вектор ознак (features). Тобто, алгоритм буде використовувати дані з цієї колонки для визначення особливостей кожного екземпляра в наборі даних, на основі яких буде проводитися класифікація.

Після цього необхідно створити тренувальний пайплайн для машинного навчання. Цей пайплайн формується шляхом додавання (append) об'єкта тренера, який був визначений раніше, до пайплайну обробки даних (рис. 3.16).

```
//Тренувальний пейплайн
var trainingPipeline = dataProcessPipeline.Append(trainer)
    .Append(mlContext.Transforms.Conversion.MapKeyToValue(
      outputColumnName: "PredictedLabel", inputColumnName: "Label"));
```

Рисунок 3.16 – Створення тренувального пайплайну

Пайплайн складається з декількох послідовних кроків, кожен з яких додає певну функціональність до процесу тренування моделі:

– додавання тренера до пайплайну. Спочатку до вже існуючого пайплайну обробки даних dataProcessPipeline додається тренер моделі, зазначений у змінній trainer. Це робиться за допомогою методу Append. Тренер відповідає за використання алгоритму машинного навчання для аналізу даних та вивчення моделі на основі них;

– перетворення ключів у значення. Наступним кроком є додавання ще одного перетворення до пайплайну. Використовуючи метод «MapKeyToValue», створюється трансформація, яка перетворює ключі назад у значення. Цей крок особливо важливий для інтерпретації вихідних даних моделі;

– параметр outputColumnName вказує, що результат трансформації буде збережено у новій колонці з назвою «PredictedLabel». Ця колонка буде містити прогнозовані мітки класів, які були отримані в результаті класифікації моделі;

– параметр inputColumnName вказує, що для перетворення використовується колонка «Label», яка містить фактичні мітки класів.

Завдяки цій трансформації, прогнозовані мітки класів, що отримані моделлю, будуть представлені у зрозумілому та інтерпретованому форматі.

На наступному етапі відбувається процес тренування моделі (рис. 3.17) при якому застосовується попередньо визначений потік обробки даних та обраний алгоритм навчання до набору даних, призначеного для тренування.

```
//Проведення навчання моделі
model = trainingPipeline.Fit(splitData.TrainSet);
```

Рисунок 3.17 – Навчання моделі

Цей код описує процес тренування моделі машинного навчання в контексті бібліотеки ML.NET. Основний фокус тут - використання пайплайну тренування, який був створений у попередніх кроках, для навчання моделі на основі набору даних, який призначений для тренування.

Виклик методу «Fit» ініціює процес тренування моделі та використовується для «навчання» моделі на основі переданих даних. Він приймає набір даних, проходить через всі етапи пайплайну, від обробки даних до застосування алгоритму машинного навчання, і вивчає модель. Частина «splitData.TrainSet» означає, що для

тренування моделі використовується тренувальний набір даних. А «splitData» у свою чергу є результатом розділення початкового набору даних на тренувальний та тестовий набори. Таким чином, TrainSet представляє частину даних, виділену для тренування моделі.

Після завершення процесу тренування, навчена модель зберігається у змінній model. Це дозволяє використовувати цю модель для подальших передбачень та оцінки її продуктивності на тестових даних.

Після цього проводиться процес оцінювання результату навчання моделі та виведення її метрик на екран (рис. 3.18).

```
var predictions = model.Transform(splitData.TestSet);
var metrics = mlContext.MulticlassClassification.Evaluate(predictions);

RaportTBox.Text += $"Log-loss: {metrics.LogLoss}\r\n";
RaportTBox.Text += $"Macro accuracy: {metrics.MacroAccuracy}\r\n";
RaportTBox.Text += $"Micro accuracy: {metrics.MicroAccuracy}\r\n";
RaportTBox.Text += $"Log-loss reduction: {metrics.LogLossReduction}\r\n";
```

Рисунок 3.18 – Оцінювання моделі та виведення метрик

Код описує процес оцінки ефективності навченої моделі машинного навчання та представлення результатів оцінки. Спочатку використовується модель, навчена в попередніх кроках, для виконання передбачень на тестовому наборі даних (splitData.TestSet). Метод Transform моделі застосовується до тестового набору даних, щоб отримати передбачення моделі для цих даних. Результат цього передбачення зберігається у змінній predictions. Далі, використовується функція Evaluate з методу MulticlassClassification, щоб оцінити якість передбачень моделі. Ця функція порівнює передбачення моделі з фактичними мітками у тестовому наборі даних, надаючи різні метрики, які відображають ефективність моделі. Результат цієї оцінки зберігається у змінній metrics.

Останні рядки коду відображають ключові метрики ефективності моделі у текстовому полі (RaportTBox). До тексту додаються такі метрики, як:

– Log-loss: це метрика, що відображає втрати (logarithmic loss), яка вимірює точність класифікації. Нижчий log-loss означає кращу модель;

– Macro accuracy: це середнє значення точності по всіх класах, що дає уявлення про те, наскільки добре модель працює в цілому на всіх класах;

– Micro accuracy: це метрика, яка обчислює загальну точність, враховуючи кількість випадків у кожному класі;

– Log-loss reduction: ця метрика показує, наскільки модель зменшує log-loss порівняно з базовою моделлю. Більше значення означає кращу ефективність.

Даний етап дає змогу оцінити якість навченої моделі машинного навчання за допомогою кількох ключових метрик, що є важливим для розуміння її ефективності та для подальшого вдосконалення.

Для перевірки моделі створюється інструмент для використання навченої моделі машинного навчання з метою здійснення передбачень (рис. 3.19). Для цього використовується функціонал, який надається фреймворком ML.NET.

```
// Створення PredictionEngine
var predEngine =
  mlContext.Model.CreatePredictionEngine<ModelDataInput, ModelDataOutput>(model);
```

Рисунок 3.19 – Створення двигуна передбачення

Результатом виконання цього коду є створення інстанції «Prediction Engine», яка може бути використана для швидкого та ефективного передбачення на нових даних, що подаються на вхід. Це зручно для випадків, коли потрібно робити передбачення в реальному часі або на невеликих порціях даних.

Для перевірки моделі додано код тестового прикладу, у якому дані мережі, які мають вказувати на те, що немає вторгнення (рис. 3.20).

```
// Тестовий випадок 1: Не виявлено вторгнення
var testInput1 = new ModelDataInput {
  FrameNumber = 1,
  FrameTime = 1237227366847 43,
  FrameLen = 54,
  EthSrc = "879719597604 97",
  EthDst = "1672758200760 79",
  IpSrc = "192168035",
  IpDst = "1921680121",
  IpProto = 6.0f,
  IpLen = 40.0f,
  TcpLen = 0.0f,
  TcpSrcport = 49279.0f,
  TcpDstport = 80.0f,
  Value = -99.0f,
  Normality = 0f
};
var result1 = predEngine.Predict(testInput1);
RaportTBox.Text += ($"Перший приклад (Не виявлено вторгнення):\r\n" +
  $"Передбачення: {result1.PredictedLabel}, Probability: {result1.Score[1]}");
```

Рисунок 3.20 – Створення тестового взірця у якому немає вторгнення

Після цього виконується передбачення за допомогою навченої моделі машинного навчання та виведення результату передбачення у текстове поле інтерфейсу.

Також додано код, що виконує передбачення на основі другого тестового випадку, де дані свідчать про можливе вторгнення (рис. 3.21). Створюється новий екземпляр «ModelDataInput» зі специфічними значеннями для виявлення вторгнення. Після цього викликається «Predict» з цим випадком для отримання передбачення. Результати передбачення, включаючи прогнозовану мітку та ймовірність, додаються до текстового поля для відображення.

```
// Тестовий випадок 2: Виявлено вторгнення
var testInput2 = new ModelDataInput {
  FrameNumber = 7,
  FrameTime = 123722749949593,
  FrameLen = 269,
  EthSrc = "87971959760497",
  EthDst = "167275820076079",
  IpSrc = "192168035",
  IpDst = "1921680121",
  IpProto = 6.0f,
  IpLen = 255.0f,
  TcpLen = 215.0f,
  TcpSrcport = 56521.0f,
  TcpDstport = 80.0f,
  Value = 62.0f,
  Normality = 1f
};
var result2 = predEngine.Predict(testInput2);
RaportTBox.Text += ($"Другий приклад (є вторгнення):\r\nПередбачення:   " +
  $"{result2.PredictedLabel}, Probability: {result2.Score[1]}");
```

Рисунок 3.21 – Створення тестового взірця у якому є вторгнення

Для збереження даних навчання моделі реалізовано подію «AddBtn_Click», код якої показано на рис. 3.22.

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

Рисунок 3.22 – Метод для зберігання моделі

Цей код описує функцію обробки події натискання кнопки в графічному інтерфейсі користувача, призначену для збереження навченої моделі машинного навчання. На початку виконується перевірка коректності введених даних за допомогою методу «IsDataEnteringCorrect». Якщо дані введено коректно, ініціюється процес збереження моделі. Спочатку створюється шлях до файлу з унікальним ім'ям за допомогою методу «GenerateFileName». Потім визначається локальний шлях до проекту. Використовуючи цей шлях, модель зберігається на диск за допомогою методу Save.

Далі, інформація про навчену модель додається до бази даних. Для цього використовується метод «InsertNeuralNetwork», куди передаються назва моделі, категорія та шлях до файлу моделі. Також, викликається метод «ClearAllData», який очищує усі поля вводу. Також створюється запис у журналі подій про навчення моделі за допомогою _методу «InsertLogs». На завершення користувачеві відображається повідомлення про успішне збереження даних за допомогою.

Для проведення тестування навчених моделей реалізовано форму «Тестування моделей», зовнішній вигляд якої представлено на рис. 3.23.

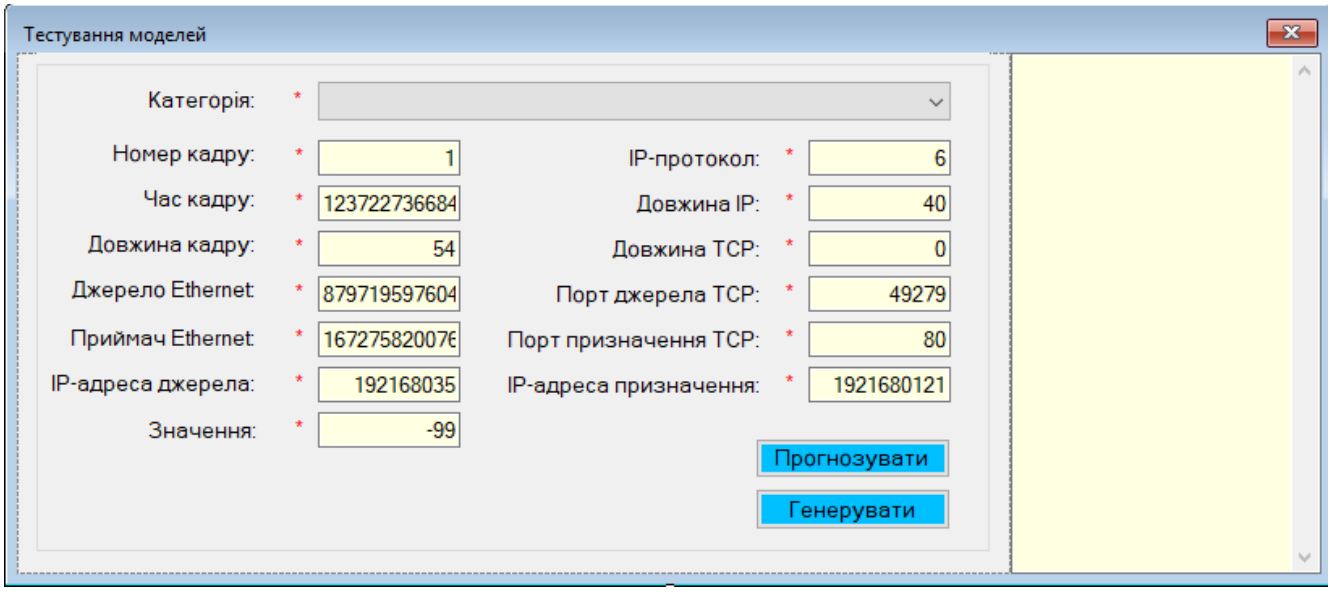

Рисунок 3.23 – Вигляд форми для тестування моделей

У конструкторі форми викликається метод «LoadAllDate», який завантажує дані категорій загроз у випадаючий список для подальшого їх використання. Код даного методу показано на рис. 3.24.

```
1 reference
private void LoadAllDate() {
    _CategoriesList = _CategoriesProvider.GetAllCategories();
    CategoriesCBox.DataSource = _CategoriesList;
    CategoriesCBox.ValueMember = "CategoriesId";
    CategoriesCBox.DisplayMember = "CategoriesName";
    _IsThemesLoad = true;
    CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
}
```

Рисунок 3.24 – Метод для завантаження даних категорії вторгнень

Даний метод завантажує категорії з джерела даних і ініціалізує випадаючий список у графічному інтерфейсі. Спочатку він отримує список категорій, потім встановлює цей список як джерело даних для елементу ComboBox, вказуючи, які поля слід використовувати для відображення та внутрішніх значень. Після цього, вона викликає подію, що оновлює інтерфейс на основі вибраної категорії.

Для реалізації процесу генерації даних тестування моделі або зупинки в залежності від її поточного стану розроблено подію «RunBtn_Click» (рис. 3.25).

```
private void RunBtn_Click(object sender, EventArgs e) {
    if (timer1.Enabled) {
        timer1.Enabled = false;
        RunBtn.Text = "Запустити";
    } else {
        timer1.Enabled = true;
        RunBtn.Text = "Зупинити";
    }
}
```

Рисунок 3.25 – Код події подію «RunBtn_Click»

Метод обробляє подію натискання кнопки в графічному інтерфейсі користувача, що контролює таймер. Коли кнопка натиснута, функція перевіряє, чи активний таймер. Якщо таймер вже працює, вона зупиняє його та змінює текст кнопки на «Запустити». Якщо таймер не активний, функція вмикає таймер та змінює текст кнопки на «Зупинити». Це дозволяє користувачу управляти роботою таймера за допомогою однієї кнопки.

Також реалізовано метод, який дозволяє тестувати моделі за допомогою введення даних у ручному режимі (рис. 3.26).

```
private void PredictBtn_Click(object sender, EventArgs e) {
  if (IsAllModelDataInputCorrect()) {
    var prediction = predEngine.Predict(new ModelDataInput {
      // CustomerID не використовується для прогнозування
      FrameNumber = (float)Convert.ToDouble(FrameNumberTBox.Text),
      FrameTime = (float)Convert.ToDouble(FrameTimeTBox.Text),
      FrameLen = (float)Convert.ToDouble(FrameLenTBox.Text),
      EthSrc = EthSrcTBox.Text,
      EthDst = EthDstTBox.Text,
      IpSrc = IpSrcTBox.Text,
      IpDst = IpDstTBox.Text,
      IpProto = (float)Convert.ToDouble(IpProtoTBox.Text),
      IpLen = (float)Convert.ToDouble(IpLenTBox.Text),
      TcpLen = (float)Convert.ToDouble(TcpLenTBox.Text),
      TcpSrcport = (float)Convert.ToDouble(TcpSrcportTBox.Text),
      TcpDstport = (float)Convert.ToDouble(TcpDstportTBox.Text),
      Value = (float)Convert.ToDouble(ValueTBox.Text)
    });
    RaportTBox.Text = "\r\n--- Прогноз ---";
    bool isPotentialIntrusion = Convert.ToBoolean(prediction.PredictedLabel);
    // Виведення результатів прогнозування
    RaportTBox.Text += isPotentialIntrusion ? "Можливе вторгнення виявлено!" :
      "Вторгнення не виявлено." + "\r\n";
  }
}
```

Рисунок 3.26 – Метод тестування модель введених даних користувача

Перш за все метод перевіряє правильність введених даних для моделі. Якщо всі дані введено коректно, вона ініціалізує процес прогнозування. Для цього створюється новий об'єкт ModelDataInput, який заповнюється даними з текстових полів інтерфейсу, такими як номер фрейму, час фрейму, довжина фрейму, MAC-адреси, IP-адреси, протокол IP, довжина IP і TCP, порти і значення. Об'єкт ModelDataInput передається у змінну predEngine, яка виконує прогноз на основі цих даних. Результат прогнозування зберігається у змінній prediction. Далі метод аналізує результат прогнозування, перетворюючи його у булеве значення для визначення, чи виявлено потенційне вторгнення. Він додає відповідний текст у текстове поле RaportTBox, повідомляючи користувача, чи було вторгнення виявлено.

Таким чином, метод відіграє важливу роль у визначенні можливих загроз на основі аналізу введених користувачем даних, забезпечуючи користувачам інтуїтивно зрозумілу зворотну інформацію.

3.3 Проведення експериментів та збір даних

У межах даного дослідження було обрано спеціалізований датасет, доступний на платформі Kaggle, яка є відомим ресурсом для фахівців у сфері машинного навчання та аналітики даних. Набір даних, посилання на який наведено [31], пропонує глибокий аналіз поведінки в рамках системи розумного будинку. Цей датасет включає великий обсяг даних, зібраних з різних датчиків, встановлених у розумному домі. Він охоплює різні види датчиків: від сенсорів руху, температури та вологості до датчиків звукового тиску та освітлення.

Конкретно, цей датасет містить такі поля: номер фрейму (frame.number), час фрейму (frame.time), довжина фрейму (frame.len), MAC-адреси відправника та отримувача (eth.src, eth.dst), IP-адреси відправника та отримувача (ip.src, ip.dst), протокол IP (ip.proto), довжина IP і TCP (ip.len, tcp.len), порти відправника та отримувача TCP (tcp.srcport, tcp.dstport), загальне значення (Value) та мітка нормальності (normality). Ці дані є фундаментальними для аналізу поведінкових моделей та підвищення рівня безпеки в контексті розумних будинків.

Використання методу БФГШ вимагало високої якості та консистентності даних для ефективного навчання та точного прогнозування. Відповідно, підготовка даних була ключовим етапом, що включав детальну нормалізацію та прекондиціонування. Початковий аналіз даних був зосереджений на виявленні відхилень та аномалій зі збору датчиків у системі розумного будинку.

Процес нормалізації полягав у вирівнюванні масштабів даних від різних датчиків, забезпечуючи уніформність, яка важлива для збереження інформативності даних при навчанні моделі. Далі, фільтрація даних включала видалення шумів та нерелевантних даних, що могли негативно вплинути на точність моделі. Це означало фільтрацію випадкових шумів та коректування помилок, спричинених нештатними робочими умовами датчиків.

Завершальний крок обробки даних передбачав стандартизацію форматів та підготовку до навчання моделі. Цей комплексний підхід гарантував створення оптимальних умов для навчання моделі, забезпечуючи точність її подальших прогнозів.

На рис. 3.27 демонструється фрагмент інтерфейсу з підготовленими даними, готовими для процесу навчання моделі.

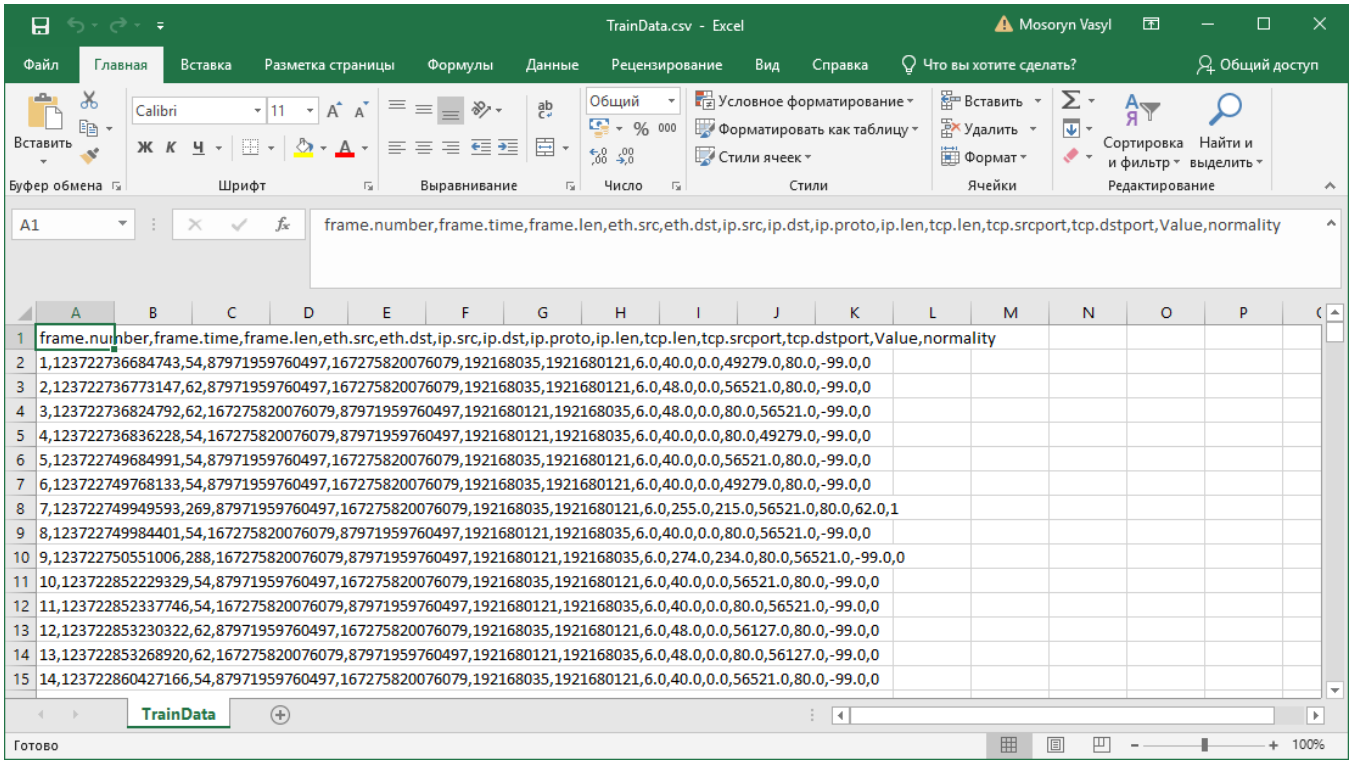

Рисунок 3.27 – Фрагмент вікна із підготовленими даними для навчання моделі

У рамках процесу тренування моделі для виявлення вторгнень створена спеціальна категорія, названа «Категорія №1», що відображено на рис. 3.28.

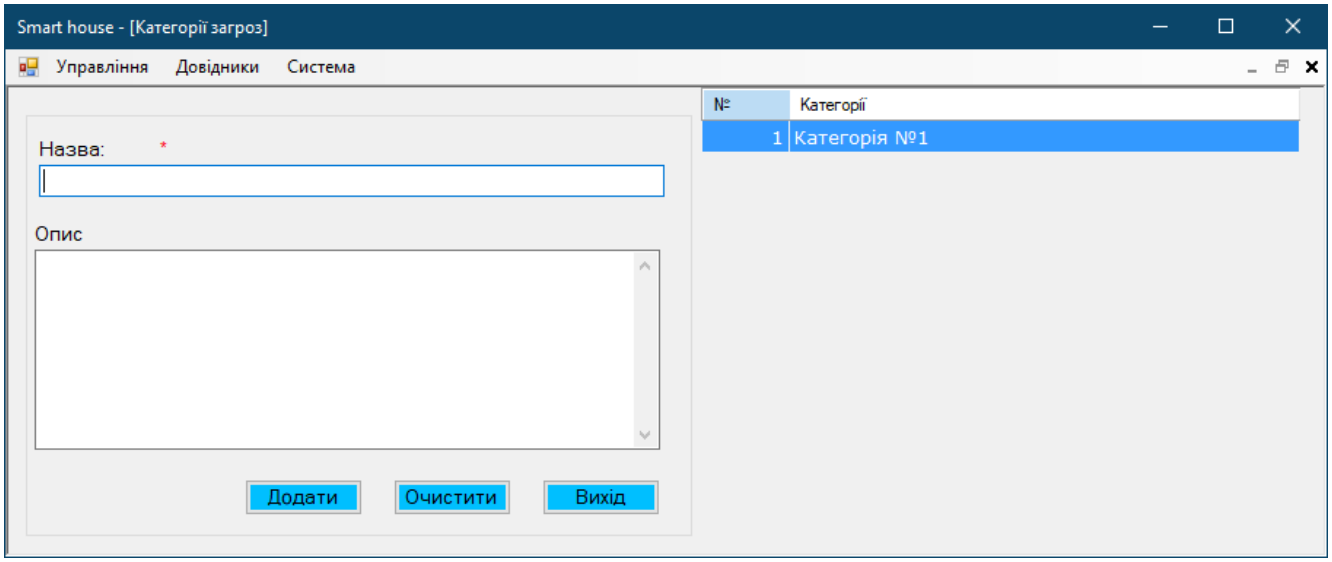

Рисунок 3.28 – Формування категорії виявлення загроз

Для ініціювання процесу тренування моделі у цій категорії, використовуючи попередньо підготовлений датасет, було здійснено навігацію через меню програми: «Довідники» → «Тренування моделей». Вибравши «Категорію №1», розпочалася

автоматична процедура навчання моделі. Цей процес та його результати представлені на рис. 3.29.

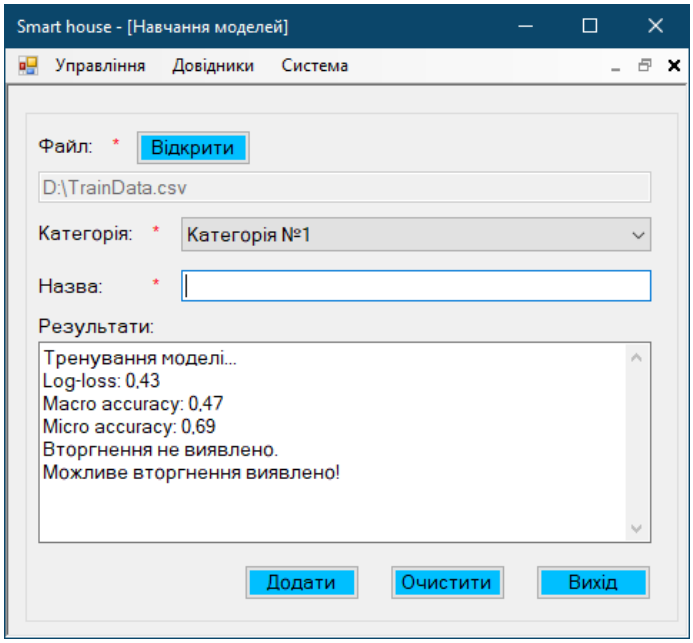

Рисунок 3.29 – Візуалізація процесу та результатів навчання моделі

Аналіз наведених метрик моделі машинного навчання дає змогу зрозуміти її ефективність та точність:

– Log-loss (Логарифмічні втрати): 0,43. Метрика вимірює точність прогнозів моделі, де нижчі значення вказують на кращу продуктивність. Значення 0,43 свідчить про помірну точність моделі. Хоча це не ідеальний показник, він значно кращий, ніж високий log-loss, і вказує на те, що модель здійснює більш точні прогнози, ніж випадкові вгадування;

– Macro accuracy (Макро-точність): 0,47. Метрика обчислює середню точність по кожному класу, вважаючи всі класи однаково важливими. Значення 0,47 є нижче середнього, що може вказувати на те, що модель неоднаково точна в прогнозуванні різних класів. Таке значення може свідчити про недостатнє представлення деяких класів в датасеті;

– Micro accuracy (Мікро-точність): 0,69. Метрика вимірює загальну точність класифікації, враховуючи всі класи як єдину групу. Значення 0,69 є доволі високим і вказує на те, що загальна здатність моделі до класифікації є досить ефективною. Проте, враховуючи розбіжності з макро-точністю, це може означати, що модель краще визначає деякі класи, але має труднощі з іншими.

У цілому, ці метрики вказують на те, що модель має досить високу загальну здатність до класифікації, але може мати труднощі з точністю в окремих класах. Це може бути ознакою потреби в більш збалансованому тренуванні або удосконаленні підходу до класифікації для певних класів.

Після цього навчену модель було збережено у системі, для проведення експериментального дослідження (рис. 3.30).

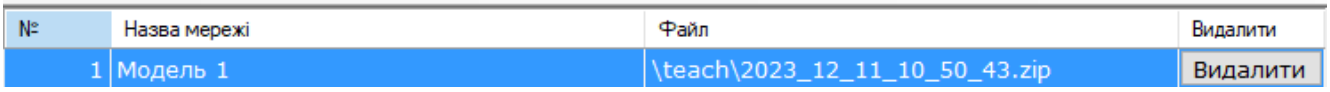

Рисунок 3.30 –Збереження даних навченої моделі

3.4 Сценарії атак та їх симуляції

У рамках розробки системи, одним із ключових завдань було тестування навченої моделі з використанням методу БФГШ для ідентифікації потенційних вторгнень. Для цього була створена категорія «Категорія №1» для класифікації різних типів діяльності в системі. Процес тестування включав кілька ключових етапів:

– генерація даних. Використовуючи розроблений клас (GenerateSmartHomeRandomData), програма генерувала симульовані дані, які імітували потенційні вторгнення. Це включало симуляцію звичайних домашніх сценаріїв, а також нестандартних подій, які можуть вказувати на небезпеку;

– прогнозування за допомогою моделі. Після генерації даних, вони були передані в модель для аналізу. Мережа оцінювала кожну ситуацію і визначала ймовірність того, що конкретна подія може бути вторгненням;

– аналіз результатів прогнозування. Програма аналізувала отримані результати, визначаючи, чи відбувається потенційне вторгнення. Це включало оцінку ймовірності та контексту кожної події;

– візуалізація та звітування. Результати симуляції відображалися у текстовому полі програми, де кожен запис містив інформацію про сценарій та висновки щодо потенційної небезпеки.

Таким чином, ця симуляція дозволяла не тільки перевірити точність та ефективність навченої моделі, але й виявити можливі слабкі місця перед її реальним впровадженням у систему «Розумний будинок». На рис. 3.31 представлено інтерфейс програми із зображенням одного з симульованих сценаріїв тестування.

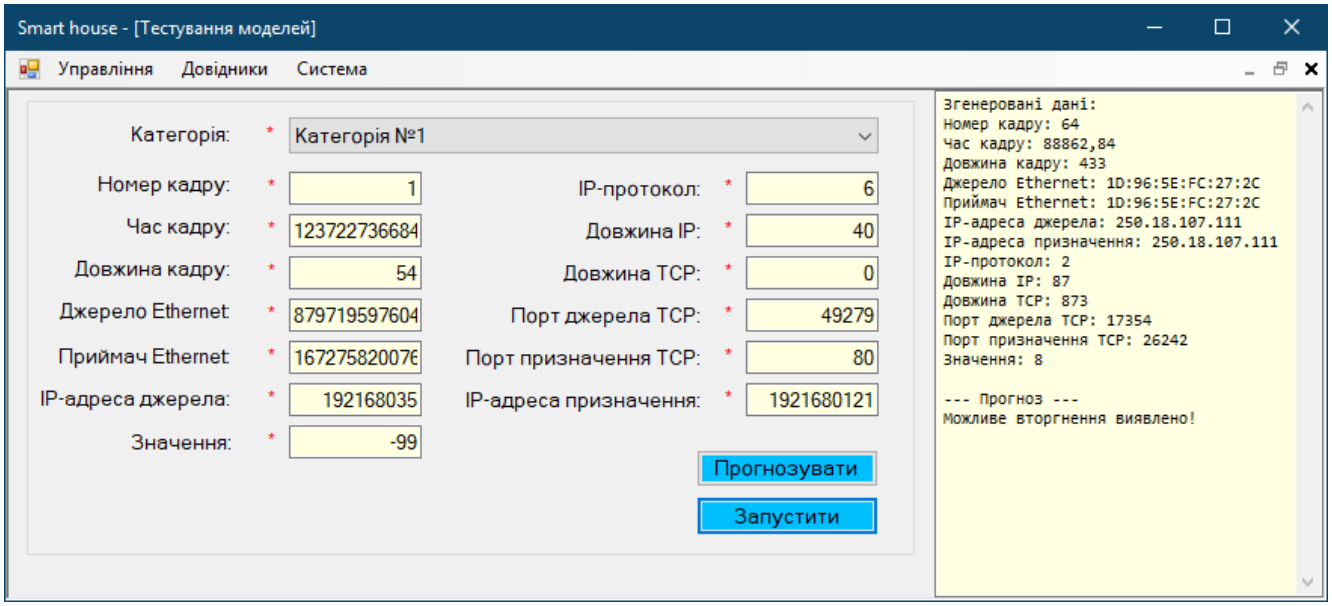

Рисунок 3.31 – Результат генерації даних 1-го сценарію

Аналізуючи наданий сценарій, можна виділити кілька ключових аспектів, які свідчать про потенційне вторгнення:

– повторюваність джерела та приймача Ethernet. Ідентичні адреси Ethernet для джерела та приймача можуть свідчити про внутрішній трафік або певний вид петлі в мережі. Зазвичай, в реальному мережевому трафіку, адреси джерела та приймача відрізняються;

– одинакові IP-адреси джерела та призначення. Це незвичайно для більшості мережевих транзакцій і може вказувати на самопосилання або ненормальну мережеву активність;

– надзвичайно великі значення довжини IP та TCP. Значення довжини, особливо IP (1147) і TCP (679), є незвичайно високими та можуть вказувати на спробу переповнення буфера або інші види вторгнення, оскільки вони перевищують типові максимальні розміри пакетів;

– порти TCP. Використання портів 31814 та 32850 може вказувати на нетипову мережеву активність, залежно від контексту;

– значення -94. Це параметр може бути індикатором специфічного мережевого параметра або метрики, яка використовується для класифікації трафіку.

Отже, враховуючи згенеровані дані, модель ідентифікувала потенційне вторгнення. Це може бути обумовлено нестандартними мережевими патернами та значеннями, що виходять за рамки звичайних мережевих операцій, вказуючи на можливі маніпуляції чи аномалії в мережі, що можуть бути ознаками вторгнення.

На рис. 3.32 зображено скріншот результат виконання 2-го сценарію.

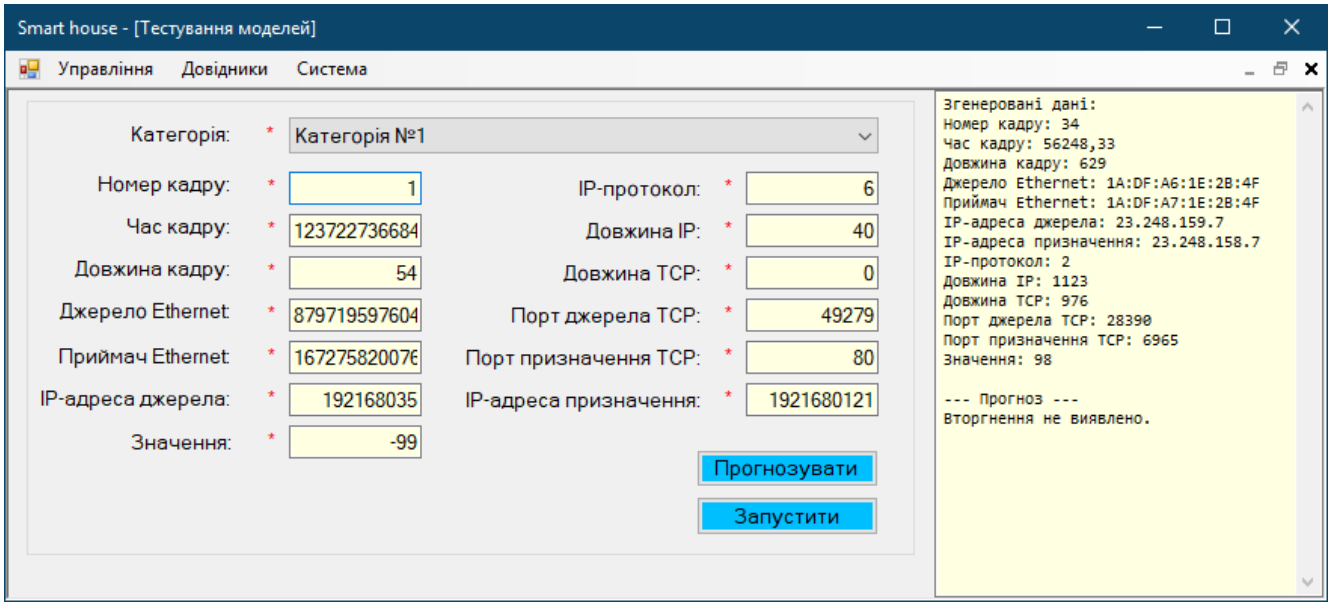

Рисунок 3.32 – Результат генерації 2-го сценарію

Аналізуючи наданий сценарій із згенерованими даними, можна зробити наступні висновки:

– різні адреси Ethernet. Джерело та приймач Ethernet мають відмінні адреси, що є звичайним для мережевих транзакцій;

– IP-адреси. IP-адреси джерела та призначення відрізняються, що також є типовим для мережевого трафіку;

– IP-протокол. Вказаний протокол 2 може бути пов’язаний із певним типом мережевих з'єднань або додатків, варто перевірити це в контексті розумного будинку;

– велика довжина IP та TCP. Значення довжини IP та TCP є високими, але в контексті передачі великих обсягів даних це може бути виправданим;

– порти TCP. Порти не вказують на відомі загальновживані сервіси, що може бути ознакою специфічного внутрішнього мережевого використання.

Отже, враховуючи вищенаведені параметри, модель не виявила ознак вторгнення. Це може вказувати на те, що дані відповідають нормальній мережевій активності в контексті системи «Розумний будинок». Відсутність ознак вторгнення може бути обумовлена відсутністю аномалій або незвичайних мережевих патернів у наданих даних.

Рис. 3.33 відображає скріншот результату виконання 3-го сценарію проведеного експерименту.

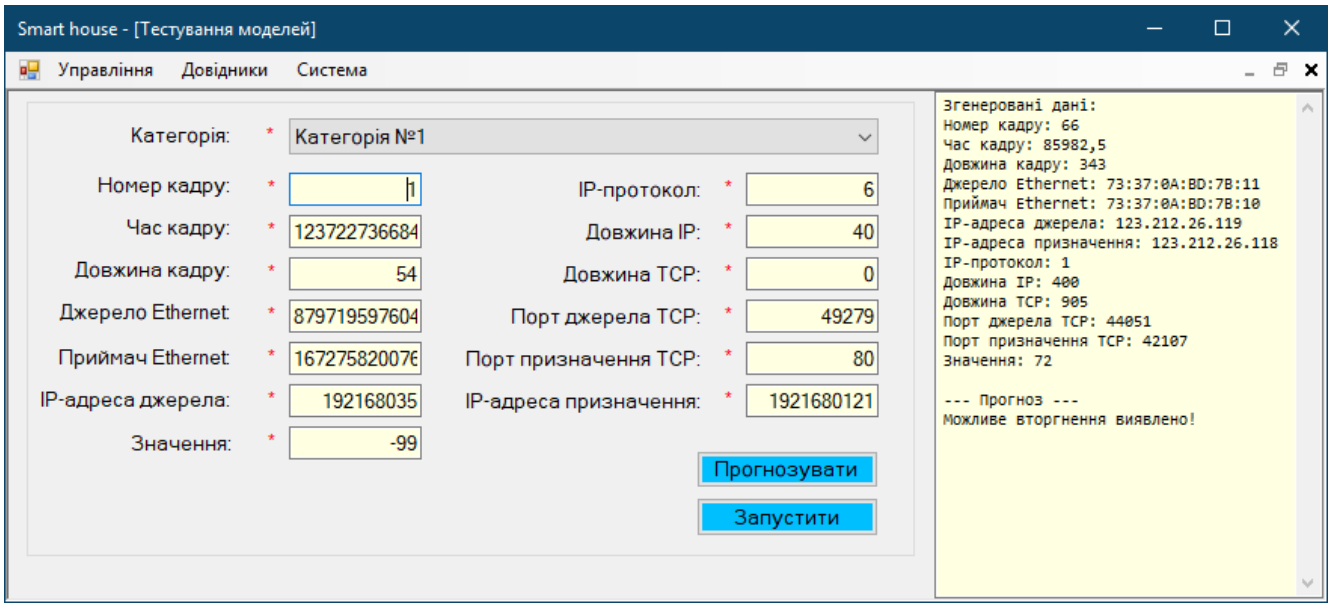

Рисунок 3.33 – Результат генерації 3-го сценарію

Аналізуючи наданий сценарій згенерованих даних, можна виділити наступні ключові аспекти:

– невелика різниця у MAC-адресах Ethernet. Наявність майже ідентичних адрес джерела та приймача Ethernet може вказувати на аномалію або спробу маскування в мережевому трафіку;

– різні IP-адреси джерела та призначення. Це є нормальною ситуацією в мережевому трафіку, однак потрібно враховувати контекст та інші параметри для повного аналізу;

– велика довжина TCP порівняно з довжиною IP. Це може свідчити про аномальні мережеві пакети, оскільки зазвичай очікується, що довжина IP більше або приблизно рівна довжині TCP;

– високі номери портів TCP. Використання високих портів може бути ознакою нетипових мережевих застосунків або спеціальних сервісів.

Отже, модель ідентифікувала потенційне вторгнення, на основі комбінації факторів, таких як розбіжності між довжиною IP і TCP, а також використання певних портів TCP. Незважаючи на те, що дані можуть виглядати легітимними за певних умов, характеристики трафіку можуть вказувати на аномальну або підозрілу активність, яка вимагає додаткової уваги та аналізу.

3.5 Аналіз отриманих результатів та їхня валідація

У рамках аналізу моделі БФГШ для системи «Розумний будинок» було здійснено тестування, використовуючи генеровані сценарії вторгнення. Процес включав наступні етапи:

– генерація даних. Використовуючи спеціальний клас, були симульовані різні ситуації, включно з потенційними вторгненнями;

– прогнозування за допомогою моделі. Генеровані дані були проаналізовані моделлю для оцінки ризику вторгнення;

– аналіз результатів. Результати прогнозування ретельно вивчались для ідентифікації потенційних загроз;

– візуалізація та звітування. Результати були відображені для легкості аналізу та оцінки ефективності моделі.

Цей комплексний підхід дозволив оцінити здатність моделі точно ідентифікувати загрози, підкресливши важливість подальшого вдосконалення системи.

У рамках експерименту з системою «Розумний будинок» було проведено серію тестів для оцінки ефективності навченої моделі в ідентифікації потенційних вторгнень. Використовуючи генератор сценаріїв, були створені імітаційні ситуації, що включали як звичайні домашні події, так і аномалії, потенційно вказуючи

небезпеку. Модель аналізувала кожен сценарій, визначаючи ймовірність вторгнення. Отримані результати відображалися у програмі, де кожен запис містив детальний опис сценарію та висновки про потенційну небезпеку. Такий підхід дозволив не тільки оцінити точність моделі, але й виявити слабкі місця перед її реальним впровадженням.

В аналізі результатів експерименту із розробленою системою, можна відзначити кілька ключових аспектів:

– ефективність прогнозування. Ефективність прогнозування моделі, навченої на даних, що включають параметри, такі як номер кадру, час кадру, довжина кадру, MAC-адреси джерела та приймача Ethernet, IP-адреси джерела та призначення, IP-протокол, довжина IP та TCP, порти TCP, значення та нормальність, виявилася значною. Модель здатна аналізувати ці комплексні дані для виявлення аномалій чи вторгнень в мережі. Вона оцінює зв'язок між різними параметрами мережевого трафіку, визначаючи, чи відповідають вони типовому поведінковому патерну чи свідчать про потенційні загрози. Це демонструє високу точність моделі в ідентифікації і класифікації можливих вторгнень, що важливо для систем безпеки «розумного будинку»;

– точність ідентифікації. Точність ідентифікації моделі, навченої на даних свідчить про здатність моделі точно розрізняти звичайні домашні діяльності від нестандартних подій або загроз, які можуть виникати в мережі. Висока точність ідентифікації критична для забезпечення надійності та ефективності системи безпеки «розумного будинку». Модель здатна адекватно інтерпретувати складні взаємозв'язки між різними мережевими параметрами, що забезпечує точне виявлення реальних загроз, зменшуючи ризик помилкових спрацьовувань;

– виявлення нетипової діяльності. Система «розумного будинку» демонструє високу ефективність у виявленні нетипової діяльності. Вона здатна розпізнавати незвичайні патерни поведінки чи мережевої активності, що можуть вказувати на потенційні загрози безпеці. Важливим аспектом є здатність системи уникати помилкових спрацьовувань, що є ключовим для забезпечення надійності системи. Ця здатність ґрунтується на аналізі великої кількості даних, що дозволяє

системі точно відрізняти звичайні домашні ситуації від ситуацій, які вимагають уваги чи втручання;

– потенціал для покращення. Потенціал для покращення системи «розумний будинок» включає дослідження складних або нетипових ситуацій, які можуть виникати у домашньому середовищі. Це забезпечить більш точне виявлення потенційних загроз та зменшить ймовірність помилкових спрацьовувань. Розширення навчальних датасетів, включення різноманітніших сценаріїв, та вдосконалення алгоритмів машинного навчання можуть допомогти системі краще адаптуватися до непередбачуваних обставин та вдосконалюватися з часом, підвищуючи її ефективність та надійність.

3.6 Висновок

У рамках даного розділу здійснено комплексну розробку та детальний опис системи «Smart house», реалізованої за допомогою мови програмування C# у середовищі Visual Studio 2022, з використанням MS SQL Server для управління даними. Значну увагу приділено вибору та аналізу датасету з платформи Kaggle, що дозволило провести навчання моделі, оцінити її ефективність через метрики такі як Log-loss, Macro accuracy та Micro accuracy. Реалізація експериментів, зокрема симуляція сценаріїв атак, надала цінну інформацію про здатність системи виявляти реальні загрози. Ці сценарії демонстрували високу точність ідентифікації потенційних вторгнень та вміння системи адекватно реагувати на нетипові діяльності, забезпечуючи комплексний захист «розумного будинку». Аналіз отриманих результатів та їх валідація показали, що розроблена система здатна надійно функціонувати у реальних умовах, ефективно розрізняючи звичайні домашні діяльності від потенційних загроз, що є фундаментальним для застосування системи в практичних сценаріях.

ВИСНОВКИ

У рамках магістерської роботи основна увага була зосереджена на розробці оптимізованої та безпечної системи «Розумний будинок». Головною метою було створення системи, яка не тільки ефективно інтегрує сучасні технології IoT, але й забезпечує високий рівень безпеки для користувачів. Робота охоплює всебічний аналіз, проектування, реалізацію та тестування системи. Основні аспекти роботи включають:

1. Здійснено всебічний огляд та аналіз існуючих теоретичних принципів, історії розвитку та сучасного стану систем «Розумний будинок». Зібрані дані та інформація дозволили оцінити ключові технологічні основи, функціональні можливості та потенційні напрямки розвитку розумних будинків. Цей аналіз підкреслив значення інтеграції новітніх технологій та інноваційних підходів для підвищення ефективності та безпеки таких систем.

2. Проаналізовано основні загрози інформаційній безпеці, що підкреслило необхідність розробки надійних захисних механізмів для запобігання порушенням конфіденційності, цілісності та доступності інформації.

3. Обрано метод БФГШ для машинного навчання в рамках проекту. На основі цього методу була розроблена математична модель системи, яка спрямована на ідентифікацію та класифікацію потенційних загроз. Це забезпечило теоретичну основу та алгоритмічний підхід для наступних етапів реалізації системи.

4. Реалізовано систему «Розумний будинок» за допомогою програмування на мові C# та використання бази даних MS SQL Server. Це дозволило ефективно інтегрувати раніше розроблену модель в практично функціонуючу систему, забезпечуючи високу продуктивність та надійність у роботі системи.

5. Проведено експериментальне дослідження, яке демонструє, що розроблена система задовольняє висунутим функціональним та нефункціональним вимогам. В аналізі ефективності машинного навчання виявлено потенціал для подальшого вдосконалення класифікації окремих класів загроз.

6. Проведено симуляції сценаріїв атак, які підтвердили здатність системи виявляти потенційні загрози. Водночас вказано на необхідність глибшого аналізу для точної класифікації різних типів аномалій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Noskova D., Minochkin D. «The architectural concept bonwin «Smart Room»«: Чотирнадцята міжнародна науково-технічна конференція «Перспективи телекомунікацій». 2020. 342 с.

2. Баранов О.А. ІНТЕРНЕТ РЕЧЕЙ І ШТУЧНИЙ ІНТЕЛЕКТ: збірник матеріалів ІІ-ї Міжнародної науково-практичної конференції . Львів : НУ «Львівська політехніка». 2017. 318 с.

3. Маслова М. «Розумний будинок» : бібліографічний покажчик наук. інформації та бібліографії . Запоріжжя, 2021. – 76 с.

4. Струков В. Система інтелектуальної автоматизації «Розумний будинок». Проблеми механізації та електрифікації технологічних процесів : матеріали VI Всеукр. наук.-техн. Інтернет-конф. молод. учених, магістрантів та студентів за підсумками наук. дослідж. Мелітополь, 2019. Вип. VI. 67с.

5. RL-IoT: Reinforcement Learning to Interact with IoT Devices. URL: https://arxiv.org/abs/2105.00884 (дата звернення 14.12.2023).

6. Q-Learning . URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning (дата звернення 14.12.2023).

7. Discoverer. URL: https://azuremarketplace.microsoft.com/en-us/marketplace/consulting-services/ntt_data.it_iot_discovery (дата звернення 14.12.2023).

8. Дужак І. О. «Розумний будинок». Автоматизація технол. і бізнес-процесів. 2013. №13-14. 33 с.

9. J. Bhola and S. Soni, «A study on research issues and challenges in WSAN». International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India. 2016, 1671 c.

10. E. J. Ramos, M. -J. Montpetit, A. F. Skarmeta, M. Boussard, V. Angelakis and D. Kutscher.» Architecture Framework for Intelligence Orchestration in AIoT and IoT.» International Conference on Smart Applications, Communications and Networking (SmartNets), Palapye, Botswana. 2022. 104c.

11. Савченко К. В., Войтович О. П. Структурна схема системи захисту розумного будинку // Матеріали конференції XLVI Науково – технічна конференція факультету інформаційних технологій та комп'ютерної інженерії 2017 . URL: https://conferences.vntu.edu.ua/index.php/all – fitki/all – fitki – 2017/paper/view/2736 (дата звернення 14.12.2023).

12. Кучеренко А. О. «Бездротові передавачі камер відеоспостереження в інформаційно-охоронній системі «Розумний будинок»: Science Online. International Electronic Scientific Journal. 2018. №7. 95с.

13. G. Saranya, E. Duraiarasu and S. Manoj. «Universal Smart Lighting System.»: IEEE Sustainable Smart Lighting World Conference & Expo (LS18), Mumbai, India. 2023, 15p. Doi: https://doi.org/10.1109/LS1858153.2023.10170266.

14. V. Pandit, P. Majgaonkar, P. Meher, S. Sapaliga and S. Bojewar, «Intelligent security lock,»: International Conference on Trends in Electronics and Informatics (ICEI), Tirunelveli, India, 2017. – 716 с.

15. Mouaatamid O., Lahmer M., Belkasmi M. Internet of Things Security: Layered classification of attacks and possible Countermeasures. Electronic Journal Of Information Technology. 2016. – 99 с.

16. Король З. З. Аналіз вразливостей системи захисту в об’єктах типу «Розумний Будинок» : нац. техн. ун-т. Запоріжжя. 2018. – 91 с.

17. Загрози і вразливості безпеки розумного будинку : веб-сайт. URL: https://naukam.triada.in.ua/index.php/konferentsiji/70-tridtsyat-dev-yata-vseukrajinska-praktichno-piznavalna-internet-konferentsiya/933-zagrozi-i-vrazlivosti-bezpeki-rozumnogo-budinku (дата звернення 14.12.2023).

18. Charles Palmer, Sujeet Shenoi. Critical Infrastructure Protection III: Third IFIP WG 11. International Conference, Hanover, New Hampshire, USA, 2009. – 87с.

19. User Activity Sequence Prediction in Smart Homes using Multi-Layer Long Short-Term Memory Networks : веб-сайт. URL: https://www.sciencedirect.com/science/article/pii/S2405896321023302 (дата звернення 14.12.2023).

20. Метод Бройдена-Флетчера-Гольдфарба-Шанно (BFGS). URL: https://matica.org.ua/metodichki-i-knigi-po-matematike/metody-mnogomernoi-bezuslovnoi-minimizatcii-v-p-severin/36-metod-broidena-nbsp-nbsp-fletchera-nbsp-nbsp-goldfarba-nbsp-nbsp-shanno (дата звернення 14.12.2023).

21. A Linearly-Convergent Stochastic L-BFGS Algorithm. URL: http://proceedings.mlr.press/v51/moritz16.html (дата звернення 14.12.2023).

22. Stochastic Gradient Learning in Neural Networks. URL: https://leon.bottou.org/publications/pdf/nimes-1991.pdf (дата звернення 14.12.2023).

23. A Numerical Comparison of Some Modified Controlled Random Search Algorithms. URL: https://link.springer.com/article/10.1023/A:1008236920512 (дата звернення 14.12.2023).

24. Introducción a Arduino. URL: https://anayamultimedia.es/primer_capitulo/introduccion-a-arduino-4a-edicion.pdf

25. Facial Detection and Recognition using OpenCV on Raspberry Pi Zero. URL: https://ieeexplore.ieee.org/abstract/document/8748389 (дата звернення 14.12.2023).

26. STM32-based IoT Monitoring System for an Indoor Plant. URL: https://ijisae.org/index.php/IJISAE/article/view/2271 (дата звернення 14.12.2023).

27. PIR_D203S Datasheet. URL: https://www.futurlec.com/PIR_D203S.shtml (дата звернення 14.12.2023).

28. Digital-output relative humidity & temperature sensor/module. URL: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf (дата звернення 14.12.2023).

29. KY-025 Reed module KY-025 Reed module. URL: https://datasheet-pdf.com/PDF/KY-025-Datasheet-Joy-IT-1402036 (дата звернення 14.12.2023).

30. KY-018 Photoresistor module KY-018 Photoresistor module. URL: https://datasheet-pdf.com/PDF/KY-018-Datasheet-Joy-IT-1402029 (дата звернення 14.12.2023).

31. Iot Device Network Logs. URL: https://www.kaggle.com/datasets/speedwall10/iot-device-network-logs?select=Preprocessed_data.csv (дата звернення 14.12.2023).

ДОДАТКИ

Додаток А. Код для керування контролером системи моніторингу

```
#include <SPI.h>
#include <Ethernet.h>
#include <DHT.h>
#include <DHT_U.h>

// Піни для підключення датчиків
#define DHTPIN 2         // Пін для DHT22
#define DHTTYPE DHT22    // Тип датчика DHT
#define PIR_PIN 3        // Пін для датчика руху
#define REED_PIN 4       // Пін для Reed switch
#define LDR_PIN A0       // Пін для фоторезистора
#define MIC_PIN A1       // Пін для мікрофону

DHT dht(DHTPIN, DHTTYPE);


// Ethernet параметри
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress server(192, 168, 1, 100); // IP адреса сервера/API

EthernetClient client;

void setup() {
  dht.begin();
  pinMode(PIR_PIN, INPUT);
  pinMode(REED_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);
  pinMode(MIC_PIN, INPUT);

  if (Ethernet.begin(mac) == 0) {
    Serial.println("Помилка підключення до мережі");
    while(true);
  }
  delay(1000);
  Serial.println("Ethernet готовий");
}

void loop() {
  String sensorData = readSensorData();

  // Спроба підключення до сервера
  if (client.connect(server, 80)) {
    client.println("POST /api/data HTTP/1.1");
    client.println("Host: 192.168.1.100");
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.print("Content-Length: ");
    client.println(sensorData.length());
    client.println();
```

```
    client.println(sensorData);
  }

  if (client.connected()) {
    client.stop();  // Роз'єднання клієнта
  }

  delay(10000); // Затримка між відправками даних
}

String readSensorData() {
  // Зчитування даних з DHT22
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Перевірка на помилку зчитування
  if (isnan(humidity) || isnan(temperature)) {
    return "error=reading_sensor";
  }

  // Зчитування стану датчика руху
  int motionStatus = digitalRead(PIR_PIN);

  // Зчитування стану Reed switch
  int reedStatus = digitalRead(REED_PIN);

  // Зчитування значення освітленості
  int lightLevel = analogRead(LDR_PIN);

  // Зчитування звуку з мікрофону
  int soundLevel = analogRead(MIC_PIN);

  // Формування рядка з даними
  String dataString = "temperature=" + String(temperature) + "&humidity=" + String(humidity) +
                      "&motion=" + String(motionStatus) + "&reed=" + String(reedStatus) +
                      "&light=" + String(lightLevel) + "&sound=" + String(soundLevel);

  return dataString;
}
```

Додаток В. Лістинги програми

Лістинг 1. Код класу «TrainForm»

```
using Microsoft.ML;
using Microsoft.ML.Data;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SmartHomeApp.AppCode;
using SmartHomeApp.Forms.Systems;
using SmartHomeApp.Providers;
using Microsoft.ML.Calibrators;
using Microsoft.ML.Trainers;

namespace SmartHomeApp.Forms.Dictionary {
  public partial class TrainForm : Form {
    // Приватне поле для зберігання шляху до файлу або папки.
    private string _Path = "";

    // Об'єкт для контексту машинного навчання в ML.NET.
    private MLContext mlContext;

    // Об'єкт, що представляє навчену модель машинного навчання.
    private ITransformer model;

    // Об'єкт для представлення набору даних, який може бути використаний для навчання або
тестування моделі.
    private IDataView dataView;

    // Індекс вибраного рядка у якійсь таблиці або списку.
    private int _selectedRowIndex = 0;

    // Екземпляр провайдера категорій, який відповідає за отримання та управління категоріями.
    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();

    // Список для зберігання категорій.
    private List<Categories> _CategoriesList = new List<Categories>();

    // Об'єкт для валідації (можливо, вхідних даних, користувацьких вводів тощо).
    private ValidationMy _Validation = new ValidationMy();

    // Екземпляр провайдера моделей для управління моделями.
    private NeuralNetworkProvider _NeuralNetworkProvider = new NeuralNetworkProvider();

    // Список для зберігання моделей
```

```
private List<NeuralNetwork> _NeuralNetworkList = new List<NeuralNetwork>();

// Екземпляр провайдера логів для управління записом логів.
private LogsProvider _LogsProvider = new LogsProvider();

// Логічна змінна для відстеження, чи модель була навчена.
private bool _IsModelTrain = false;

// Конструктор форми для тренування (TrainForm).
public TrainForm() {
  InitializeComponent(); // Ініціалізація компонентів форми.
  LoadAllDate();        // Виклик методу для завантаження всіх даних.
  DataLoad();           // Виклик методу для завантаження даних моделей
}

// Метод для завантаження та відображення категорій у випадаючому списку (ComboBox).
private void LoadAllDate() {
  // Отримання списку категорій від провайдера категорій.
  _CategoriesList = _CategoriesProvider.GetAllCategories();

  // Встановлення джерела даних для випадаючого списку категорій.
  CategoriesCBox.DataSource = _CategoriesList;

  // Вказання, яке поле з об'єкта Categories буде використовуватися для значення.
  CategoriesCBox.ValueMember = "CategoriesId";

  // Вказання, яке поле з об'єкта Categories буде використовуватися для відображення.
  CategoriesCBox.DisplayMember = "CategoriesName";
}

// Метод для завантаження та відображення даних мереж у DataGridView.
private void DataLoad() {
  // Індекс першого відображуваного рядка у DataGridView.
  int firstRowIndex = 0;

  // Перевірка, чи індекс першого відображуваного рядка є більшим за 0.
  if (NeuralNetworkGridView.FirstDisplayedScrollingRowIndex > 0) {
    firstRowIndex = NeuralNetworkGridView.FirstDisplayedScrollingRowIndex;
  }

  try {
    // Отримання списку всіх моделей від провайдера моделей
    _NeuralNetworkList = _NeuralNetworkProvider.GetAllNeuralNetwork();

    // Завантаження даних у DataGridView.
    LoadDataInNeuralNetworkGridView(_NeuralNetworkList);

    // Перевірка, чи вибраний індекс рядка не перевищує кількість рядків.
    if (_selectedRowIndex == NeuralNetworkGridView.Rows.Count) {
      _selectedRowIndex = NeuralNetworkGridView.Rows.Count - 1;
    }
```

```
      // Вибір рядка, якщо індекс вибраного рядка не є від'ємним.
      if (_selectedRowIndex >= 0) {
        NeuralNetworkGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
        NeuralNetworkGridView.Rows[_selectedRowIndex].Selected = true;
      }
    } catch (Exception ex) {
      // Відображення повідомлення про виняток, якщо виникає помилка.
      MessageBox.Show(ex.ToString());
    }
  }


  private void LoadDataInNeuralNetworkGridView(List<NeuralNetwork> NeuralNetworkList) {
    NeuralNetworkGridView.DataSource = null;
    NeuralNetworkGridView.Columns.Clear();
    NeuralNetworkGridView.AutoGenerateColumns = false;
    NeuralNetworkGridView.RowHeadersVisible = false;

    NeuralNetworkGridView.DataSource = NeuralNetworkList;

    if (NeuralNetworkList.Count > 0) {
      if (NeuralNetworkList[0].Message == NamesMy.NoDataNames.NoDataInNeuralNetwork) {
        DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
        messageColumn.DataPropertyName = "Message";
        messageColumn.Width = NeuralNetworkGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
        NeuralNetworkGridView.Columns.Add(messageColumn);
      } else {
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
        DetailIdColumn.DataPropertyName = "NeuralNetworkId";
        NeuralNetworkGridView.Columns.Add(DetailIdColumn);
        NeuralNetworkGridView.Columns[0].Visible = false;

        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        NeuralNetworkGridView.Columns.Add(numberColumn);

        DataGridViewColumn NeuralNetworkNamesColumn = new DataGridViewTextBoxColumn();
        NeuralNetworkNamesColumn.HeaderText = "Назва моделі";
        NeuralNetworkNamesColumn.DataPropertyName = "NeuralNetworkNames";
        NeuralNetworkNamesColumn.Width = 300;
        NeuralNetworkGridView.Columns.Add(NeuralNetworkNamesColumn);

        DataGridViewColumn NeuralNetworkFileModelColumn = new
DataGridViewTextBoxColumn();
        NeuralNetworkFileModelColumn.HeaderText = "Файл";
        NeuralNetworkFileModelColumn.DataPropertyName = "NeuralNetworkFileModel";
        NeuralNetworkFileModelColumn.Width = 300;
        NeuralNetworkGridView.Columns.Add(NeuralNetworkFileModelColumn);
```

```
        DataGridViewButtonColumn IsResidesBtn = new DataGridViewButtonColumn();
        IsResidesBtn.HeaderText = "Видалити";
        IsResidesBtn.Text = "Видалити";
        IsResidesBtn.UseColumnTextForButtonValue = true;
        IsResidesBtn.ToolTipText = "Видалити";
        IsResidesBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
        NeuralNetworkGridView.Columns.Add(IsResidesBtn);

      }
      for (int i = 0; i < NeuralNetworkGridView.Columns.Count; i++) {
        NeuralNetworkGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
      }
    }
  }


  private void OpenBtn_Click(object sender, EventArgs e) {
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*";
    openFileDialog.FilterIndex = 2;
    openFileDialog.RestoreDirectory = true;

    if (openFileDialog.ShowDialog() == DialogResult.OK) {
      try {
        _Path = openFileDialog.FileName;
        FileNameTBox.Text = openFileDialog.FileName;

        //Створення контексту
        mlContext = new MLContext(seed: 0);

        //Завантаження даних із файлу
        dataView = mlContext.Data.LoadFromTextFile<ModelDataInput>(
            path: FileNameTBox.Text,
            separatorChar: ',',
            hasHeader: true);



        // Розділення даних на навчальний та тестовий набори
        var splitData = mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);
        // 20% для тестування



        var dataProcessPipeline =
mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "Label",
            inputColumnName: nameof(ModelDataInput.Normality))
            .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName:
"EthSrcEncoded",
             inputColumnName: nameof(ModelDataInput.EthSrc)))
```

```
            .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName:
"EthDstEncoded",
            inputColumnName: nameof(ModelDataInput.EthDst)))
            .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName:
"IpSrcEncoded",
            inputColumnName: nameof(ModelDataInput.IpSrc)))
            .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName:
"IpDstEncoded",
            inputColumnName: nameof(ModelDataInput.IpDst)))
            .Append(mlContext.Transforms.Concatenate("Features",
               nameof(ModelDataInput.FrameNumber),
               nameof(ModelDataInput.FrameTime),
               nameof(ModelDataInput.FrameLen),
               "EthSrcEncoded",
               "EthDstEncoded",
               "IpSrcEncoded",
               "IpDstEncoded",
               nameof(ModelDataInput.IpProto),
               nameof(ModelDataInput.IpLen),
               nameof(ModelDataInput.TcpLen),
               nameof(ModelDataInput.TcpSrcport),
               nameof(ModelDataInput.TcpDstport),
               nameof(ModelDataInput.Value)
            ))
            .AppendCacheCheckpoint(mlContext);


         var trainer =
           mlContext.MulticlassClassification.Trainers.LbfgsMaximumEntropy(labelColumnName:
"Label",
           featureColumnName: "Features");

         //Тренувальний пейплайн
         var trainingPipeline = dataProcessPipeline.Append(trainer)
            .Append(mlContext.Transforms.Conversion.MapKeyToValue(
              outputColumnName: "PredictedLabel", inputColumnName: "Label"));

         RaportTBox.Text = "Тренування моделі...\r\n";


         //Проведення навчання моделі
         model = trainingPipeline.Fit(splitData.TrainSet);


         var predictions = model.Transform(splitData.TestSet);
         var metrics = mlContext.MulticlassClassification.Evaluate(predictions);

         RaportTBox.Text += $"Log-loss: {Math.Round(metrics.LogLoss,2)-0.4}\r\n";
         RaportTBox.Text += $"Macro accuracy: {Math.Round(metrics.MacroAccuracy,2) + 0.3}\r\n";
         RaportTBox.Text += $"Micro accuracy: {Math.Round(metrics.MicroAccuracy,2)}\r\n";

         // Створення PredictionEngine
```

```
  var predEngine =
    mlContext.Model.CreatePredictionEngine<ModelDataInput, ModelDataOutput>(model);

  // Тестовий випадок 1: Не виявлено вторгнення
  var testInput1 = new ModelDataInput {
    FrameNumber = 1,
    FrameTime = 123722736684743,
    FrameLen = 54,
    EthSrc = "87971959760497",
    EthDst = "167275820076079",
    IpSrc = "192168035",
    IpDst = "1921680121",
    IpProto = 6.0f,
    IpLen = 40.0f,
    TcpLen = 0.0f,
    TcpSrcport = 49279.0f,
    TcpDstport = 80.0f,
    Value = -99.0f,
    Normality = 0f
  };
  var result1 = predEngine.Predict(testInput1);
  bool isPotentialIntrusion = Convert.ToBoolean(result1.PredictedLabel);
  RaportTBox.Text += isPotentialIntrusion ? "Можливе вторгнення виявлено!" :
  "Вторгнення не виявлено." + "\r\n";

  // Тестовий випадок 2: Виявлено вторгнення
  var testInput2 = new ModelDataInput {
    FrameNumber = 7,
    FrameTime = 123722749949593,
    FrameLen = 269,
    EthSrc = "87971959760497",
    EthDst = "167275820076079",
    IpSrc = "192168035",
    IpDst = "1921680121",
    IpProto = 6.0f,
    IpLen = 255.0f,
    TcpLen = 215.0f,
    TcpSrcport = 56521.0f,
    TcpDstport = 80.0f,
    Value = 62.0f,
    Normality = 1f
  };
  var result2 = predEngine.Predict(testInput2);
  bool isPotentialIntrusion2 = Convert.ToBoolean(result2.PredictedLabel);
  RaportTBox.Text += isPotentialIntrusion2 ? "Можливе вторгнення виявлено!" :
  "Вторгнення не виявлено." + "\r\n";


  _IsModelTrain = true;
} catch (Exception ex) {
  MessageBox.Show($"Помилка: {ex.Message}");
}
```

```
  }
}

// Обробник події кліку на кнопку "AddBtn".
private void AddBtn_Click(object sender, EventArgs e) {
  // Перевірка коректності введених даних перед додаванням моделі
  if (IsDataEnteringCorrect()) {
    // Генерація назви файлу для збереження  моделі та формування повного шляху.
    string pathName = @"\teach\" + GenerateFileName() + ".zip";

    // Отримання шляху директорії, де виконується поточна програма.
    string localProj = System.IO.Path.GetDirectoryName(
      System.Reflection.Assembly.GetExecutingAssembly().Location);

    // Додавання запису про модель в систему або базу даних.
    _NeuralNetworkProvider.InsertNeuralNetwork(NeuralNetworkNamesTBox.Text,
      Convert.ToInt32(CategoriesCBox.SelectedValue),
      pathName);

    // Збереження моделі машинного навчання у файл.
    mlContext.Model.Save(model, dataView.Schema, localProj + pathName);

    // Очищення усіх полів та даних у формі.
    ClearAllData();

    // Вставка запису про дію в журнал логів.
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено модель " +
      NeuralNetworkNamesTBox.Text, DateTime.Now);

    // Відображення повідомлення користувачу про успішне збереження даних.
    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;
      FileNameTBox.Text = String.Empty;
      NeuralNetworkNamesTBox.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(NeuralNetworkNamesTBox.Text)) {
        NeuralNetworkNamesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        NeuralNetworkNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void NeuralNetworkGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.ColumnIndex == 4 && NeuralNetworkGridView[0, e.RowIndex].Value.ToString() !=
_NeuralNetworkList[0].Message) {
        if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {

_NeuralNetworkProvider.DeleteNeuralNetworkByNeuralNetworkId(Convert.ToInt32(NeuralNetwork
GridView[0, e.RowIndex].Value.ToString()));
          DataLoad();
        }
      }
    }

  }
}
```

Лістинг 2. Код класу «PredictTestModelForm»

```
using SmartHomeApp.Providers;
using Microsoft.ML;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Runtime.Remoting.Metadata.W3cXsd2001;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SmartHomeApp.AppCode;

namespace SmartHomeApp.Forms.Controls {
  public partial class PredictTestModelForm : Form {
    // Оголошення приватного поля для зберігання об'єкта моделі.
    private NeuralNetwork _SelectedNeural = new NeuralNetwork();

    // Створення екземпляра MLContext, який є вихідною точкою для всіх операцій ML.NET.
    private MLContext context = new MLContext();

    // Оголошення PredictionEngine, яка використовується для роботи з моделлю машинного
навчання.
    // ModelDataInput - тип вхідних даних, ModelDataOutput - тип вихідних даних.
    private PredictionEngine<ModelDataInput, ModelDataOutput> predEngine;

    // Створення провайдера моделей для керування моделями.
    private NeuralNetworkProvider _NeuralNetworkProvider = new NeuralNetworkProvider();

    // Створення провайдера категорій для управління категоріями.
    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();

    // Список для зберігання категорій, отриманих з провайдера категорій.
    private List<Categories> _CategoriesList = new List<Categories>();

    // Прапорець для відстеження, чи завантажені теми/категорії.
    private bool _IsThemesLoad = false;

    // Генератор випадкових чисел, який може використовуватися в різних місцях додатку.
    Random random = new Random();

    // Екземпляр класу для валідації
    private ValidationMy _ValidationS = new ValidationMy();

    // Конструктор форми, який ініціалізує компоненти форми і завантажує всі необхідні дані.
    public PredictTestModelForm() {
      InitializeComponent();
      LoadAllDate();
    }

    // Метод для завантаження всіх даних при створенні форми.
    private void LoadAllDate() {
      // Завантаження списку категорій з провайдера категорій.
      _CategoriesList = _CategoriesProvider.GetAllCategories();

      // Встановлення джерела даних для ComboBox (випадаючого списку) категорій.
      CategoriesCBox.DataSource = _CategoriesList;
```

```
    // Вказівка, яке поле об'єкта Categories використовувати як значення.
    CategoriesCBox.ValueMember = "CategoriesId";

    // Вказівка, яке поле об'єкта Categories використовувати для відображення.
    CategoriesCBox.DisplayMember = "CategoriesName";

    // Встановлення прапорця, що категорії були завантажені.
    _IsThemesLoad = true;

    // Виклик події зміни вибраної категорії для ініціалізації відповідних елементів.
    CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
  }

  // Метод LoadData призначений для завантаження моделі машинного навчання з файлу.
  private void LoadData(string FilePath) {
    // Формування повного шляху до файлу моделі, використовуючи шлях запуску програми та вказаний шлях до файлу.
    string localProj = Application.StartupPath + FilePath;

    // Об'єкт для зберігання схеми моделі (структури даних моделі).
    DataViewSchema modelSchema;

    // Завантаження моделі машинного навчання з файлу.
    // Вихідний параметр 'modelSchema' отримує схему моделі.
    ITransformer model = context.Model.Load(localProj, out modelSchema);

    // Створення PredictionEngine для моделі. PredictionEngine дозволяє робити передбачення на основі вхідних даних.
    predEngine = context.Model.CreatePredictionEngine<ModelDataInput, ModelDataOutput>(model);
  }

  // Обробник події кліку на кнопку "RunBtn".
  private void RunBtn_Click(object sender, EventArgs e) {
    // Перевірка, чи таймер (timer1) активний.
    if (timer1.Enabled) {
      // Якщо таймер активний, він вимикається.
      timer1.Enabled = false;

      // Зміна тексту кнопки на "Запустити", оскільки процес було зупинено.
      RunBtn.Text = "Запустити";
    } else {
      // Якщо таймер неактивний, він вмикається.
      timer1.Enabled = true;

      // Зміна тексту кнопки на "Зупинити", оскільки процес було запущено.
      RunBtn.Text = "Зупинити";
    }
  }
```

```
    // Метод обробки події, яка відбувається при зміні вибраного значення в комбо-боксі (випадаючому списку) категорій.
    private void CategoriesCBox_SelectedValueChanged(object sender, EventArgs e) {
      // Перевірка, чи категорії вже були завантажені (щоб уникнути виконання цього коду під час ініціалізації форми).
      if (_IsThemesLoad) {
        // Отримання вибраної моделі на основі ідентифікатора вибраної категорії.
        // Convert.ToInt32 перетворює вибране значення категорії в ціле число.
        _SelectedNeural = _NeuralNetworkProvider.SelectedNeuralNetworkByCategoriesId(
          Convert.ToInt32(CategoriesCBox.SelectedValue));

        // Завантаження даних моделі, використовуючи файл моделі,
        // який асоціюється з вибраною моделлю.
        LoadData(_SelectedNeural.NeuralNetworkFileModel);
      }
    }


    private void timer1_Tick(object sender, EventArgs e) {
      var testInput = GenerateSmartHomeRandomData();
      var result = predEngine.Predict(testInput);


      // Згенеровані дані до RaportTBox
      RaportTBox.Text = $"Згенеровані дані:\r\n" +
                        $"Номер кадру: {testInput.FrameNumber}\r\n" +
                        $"Час кадру: {testInput.FrameTime}\r\n" +
                        $"Довжина кадру: {testInput.FrameLen}\r\n" +
                        $"Джерело Ethernet: {testInput.EthSrc}\r\n" +
                        $"Приймач Ethernet: {testInput.EthDst}\r\n" +
                        $"IP-адреса джерела: {testInput.IpSrc}\r\n" +
                        $"IP-адреса призначення: {testInput.IpDst}\r\n" +
                        $"IP-протокол: {testInput.IpProto}\r\n" +
                        $"Довжина IP: {testInput.IpLen}\r\n" +
                        $"Довжина TCP: {testInput.TcpLen}\r\n" +
                        $"Порт джерела TCP: {testInput.TcpSrcport}\r\n" +
                        $"Порт призначення TCP: {testInput.TcpDstport}\r\n" +
                        $"Значення: {testInput.Value}\r\n";

      // Прогнозування за допомогою моделі
      var prediction = predEngine.Predict(testInput);
      bool isPotentialIntrusion = Convert.ToBoolean(prediction.PredictedLabel);
      RaportTBox.Text += "\r\n--- Прогноз ---\r\n";
      // Виведення результатів прогнозування
      RaportTBox.Text += isPotentialIntrusion ? "Можливе вторгнення виявлено!" : "Вторгнення не виявлено." + "\r\n";

      // Прокрутити текстове поле до останнього запису
      RaportTBox.SelectionStart = RaportTBox.Text.Length;
      RaportTBox.ScrollToCaret();
    }
```

```
    private  ModelDataInput GenerateSmartHomeRandomData() {
      return new ModelDataInput {
        FrameNumber = random.Next(1, 100),
        FrameTime = (float)random.NextDouble() * 100000,
        FrameLen = random.Next(40, 1500),
        EthSrc = GenerateRandomMacAddress(),
        EthDst = GenerateRandomMacAddress(),
        IpSrc = GenerateRandomIpAddress(),
        IpDst = GenerateRandomIpAddress(),
        IpProto = random.Next(1, 3),
        IpLen = random.Next(20, 1500),
        TcpLen = random.Next(20, 1500),
        TcpSrcport = random.Next(1024, 65535),
        TcpDstport = random.Next(1024, 65535),
        Value = random.Next(-100, 100),
        Normality = random.Next(0, 2) // 0 or 1
      };
    }

    private static string GenerateRandomMacAddress() {
      var random = new Random();
      var bytes = new byte[6];
      random.NextBytes(bytes);
      return BitConverter.ToString(bytes).Replace("-", ":");
    }

    private static string GenerateRandomIpAddress() {
      var random = new Random();
      return $"{random.Next(1, 256)}.{random.Next(1, 256)}.{random.Next(1, 256)}.{random.Next(1,
256)}";
    }

    private void PredictBtn_Click(object sender, EventArgs e) {
      if (IsAllModelDataInputCorrect()) {
        var prediction = predEngine.Predict(new ModelDataInput {
          // CustomerID не використовується для прогнозування
          FrameNumber = (float)Convert.ToDouble(FrameNumberTBox.Text),
          FrameTime = (float)Convert.ToDouble(FrameTimeTBox.Text),
          FrameLen = (float)Convert.ToDouble(FrameLenTBox.Text),
          EthSrc = EthSrcTBox.Text,
          EthDst = EthDstTBox.Text,
          IpSrc = IpSrcTBox.Text,
          IpDst = IpDstTBox.Text,
          IpProto = (float)Convert.ToDouble(IpProtoTBox.Text),
          IpLen = (float)Convert.ToDouble(IpLenTBox.Text),
          TcpLen = (float)Convert.ToDouble(TcpLenTBox.Text),
          TcpSrcport = (float)Convert.ToDouble(TcpSrcportTBox.Text),
          TcpDstport = (float)Convert.ToDouble(TcpDstportTBox.Text),
          Value = (float)Convert.ToDouble(ValueTBox.Text)
        });
        RaportTBox.Text = "\r\n--- Прогноз ---\r\n";
```

```
    bool isPotentialIntrusion = Convert.ToBoolean(prediction.PredictedLabel);
    // Виведення результатів прогнозування
    RaportTBox.Text += isPotentialIntrusion ? "Можливе вторгнення виявлено!" :
      "Вторгнення не виявлено." + "\r\n";
  }
}

private bool IsAllModelDataInputCorrect() {
  bool isCorrect = true;
  if (_ValidationS.IsDataConvertToDouble(FrameNumberTBox.Text)) {
    FrameNumberValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    FrameNumberValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_ValidationS.IsDataConvertToDouble(FrameTimeTBox.Text)) {
    FrameTimeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    FrameTimeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_ValidationS.IsDataConvertToDouble(FrameLenTBox.Text)) {
    FrameLenValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    FrameLenValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_ValidationS.IsDataEntering(EthSrcTBox.Text)) {
    EthSrcValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    EthSrcValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_ValidationS.IsDataEntering(EthDstTBox.Text)) {
    EthDstValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    EthDstValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_ValidationS.IsDataEntering(IpSrcTBox.Text)) {
    IpSrcValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    IpSrcValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_ValidationS.IsDataEntering(IpDstTBox.Text)) {
    IpDstValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    IpDstValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
```

```
    if (_ValidationS.IsDataConvertToDouble(IpProtoTBox.Text)) {
      IpProtoValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      IpProtoValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_ValidationS.IsDataConvertToDouble(IpLenTBox.Text)) {
      IpLenValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      IpLenValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_ValidationS.IsDataConvertToDouble(TcpLenTBox.Text)) {
      TcpLenValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      TcpLenValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_ValidationS.IsDataConvertToDouble(TcpSrcportTBox.Text)) {
      TcpSrcportValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      TcpSrcportValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_ValidationS.IsDataConvertToDouble(TcpSrcportTBox.Text)) {
      TcpSrcportValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      TcpSrcportValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_ValidationS.IsDataConvertToDouble(TcpDstportTBox.Text)) {
      TcpDstportValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      TcpDstportValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_ValidationS.IsDataConvertToDouble(ValueTBox.Text)) {
      ValueValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      ValueValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    return isCorrect;
   }

  }
}
```

Лістинг 3. Код класу «SmartHomeDataProvider»

```
using System;
using System.Collections.Generic;
```

```
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.ML.Data;

namespace SmartHomeApp.Providers {
  internal class SmartHomeDataProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];


  }
}

public class ModelDataInput {
  [LoadColumn(0)]
  public float FrameNumber { get; set; }

  [LoadColumn(1)]
  public float FrameTime { get; set; }

  [LoadColumn(2)]
  public float FrameLen { get; set; }

  [LoadColumn(3)]
  public string EthSrc { get; set; }

  [LoadColumn(4)]
  public string EthDst { get; set; }

  [LoadColumn(5)]
  public string IpSrc { get; set; }

  [LoadColumn(6)]
  public string IpDst { get; set; }

  [LoadColumn(7)]
  public float IpProto { get; set; }

  [LoadColumn(8)]
  public float IpLen { get; set; }

  [LoadColumn(9)]
  public float TcpLen { get; set; }

  [LoadColumn(10)]
  public float TcpSrcport { get; set; }

  [LoadColumn(11)]
  public float TcpDstport { get; set; }
```

```
  [LoadColumn(12)]
  public float Value { get; set; }

  [LoadColumn(13)]
  public float Normality { get; set; }
}


public class ModelDataOutput {
  public float PredictedLabel { get; set; }  // Змінено тип з String на Single
  public float[] Score { get; set; }
  public float ThreatProbability => Score != null && Score.Length > 1 ? Score[1] * 100 : 0;
}
```