Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувач кафедрою
кібербезпеки та захисту
інформації
_____Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційна робота
бакалавра
(назва освітнього рівня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Механізм оцінювання уразливості до кібератак систем аварійного електропостачання АЕС на базі штучного інтелекту»

Виконавець: студент IV курсу, групи КБ-43

_____ Рената ЗІПМАН
(підпис) (ім'я прізвище)

	Підпис	**Прізвище, ініціали**
Керівник		Микола БРАІЛОВСЬКИЙ
Нормоконтроль		Олександр ЛУКАШОВ

Київ 2025

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувач кафедрою
кібербезпеки
та захисту інформації
_____Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

студентові **КБ-43** (група) **Зіпман Ренаті Олегівні** (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Механізм оцінювання уразливості до кібератак систем аварійного електропостачання АЕС на базі штучного інтелекту

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Алгоритми штучного інтелекту, дані ICS з відкритих джерел, докумнетація по реалізації моделей штучного інтелекту та веб-додатків.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Кібербезпека в системах критичної інфраструктури, специфікація атак на системи критичної інфраструктури, підходи до захисту об’єктів критичної інфраструктури, обґрунтування необхідності проведення дослідження, обґрунтування вибору алгоритмів машинного навчання, розробка підходу до збирання і генерації даних, опис та специфікація даних для навчання, обґрунтування підходу до тестування моделей, проведення експериментів з виявлення атак, аналіз та порівняння результатів роботи алгоритмів, рекомендації для подальших досліджень.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Застосування сучасних алгоритмів штучного Інтелекту для класифікації кібератак на об’єкти критичної інфраструктури.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: «29» листопада 2024 року

Завдання видав	_____	Микола БРАІЛОВСЬКИЙ
	(підпис)	(ім’я, прізвище)
Завдання прийняв до виконання	_____	Рената ЗІПМАН
	(підпис)	(ім’я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	14.10.24-10.11.24	*Виконано*
2	Аналіз літератури	11.11.24-01.12.24	*Виконано*
3	Огляд джерел та аналіз проблеми	02.12.24-18.12.24	*Виконано*
4	Обґрунтування методики дослідження та вибору математичного апарату	19.12.24-25.01.25	*Виконано*
5	Реалізація моделей штучного інтелекту	26.01.25-23.02.25	*Виконано*
6	Аналіз експериментальних результатів	24.02.25-20.03.25	*Виконано*
7	Оформлення пояснювальної записки	21.03.25-18.04.25	*Виконано*
8	Підготовка до захисту	19.05.25-15.06.25	*Виконано*

Завдання видав _____ Микола БРАІЛОВСЬКИЙ
(підпис) (ім’я, прізвище)

Завдання прийняв до виконання _____ Рената ЗІПМАН
(підпис) (ім’я, прізвище)

Термін подання кваліфікаційної роботи до ЕК «13» червня 2025 року

РЕФЕРАТ

Кваліфікаційна робота на тему «Механізм оцінювання уразливості до кібер атак систем аварійного електропостачання АЕС на базі штучного інтелекту» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаних джерел та чотирьох додатків. Загальний обсяг роботи, без врахування додатків, складає 69 сторінок, робота містить 48 рисунків та 37 джерел посилання.

Метою роботи є розробка системи виявлення та класифікації кібератак на систему аварійного електропостачання АЕС з використанням алгоритмів машинного навчання.

Об'єкт дослідження є процес застосування класифікаційних алгоритмів штучного інтелекту для оцінки уразливості системи аварійного електропостачання АЕС до кібератак.

Предмет дослідження є система аварійного електропостачання атомної електростанції (АЕС) та її уразливість до кібератак.

Практичною цінністю отриманих результатів є можливість впровадження розробленої системи класифікації кібератак для підвищення надійності та безпеки систем аварійного електропостачання АЕС. Запропоноване рішення дозволяє своєчасно виявляти аномальні дії у критичних інформаційно-керуючих системах, мінімізуючи ризики порушення стабільної роботи обладнання в умоваї кібератак, оскільки складність та потужність шкідливого програмного забезпечення постійно зростає.

У результаті дослідження розроблено систему класифікації кібератак на основі моделі випадкового лісу, яка демонструє високу точність (90.33%) та збалансованість метрик для різних типів атак. Розроблено веб-додаток для аналізу даних системи аварійного електропостачання АЕС та виявлення потенційних кібератак в реальному часі. Рекомендується використовувати розроблену систему для підвищення рівня кібербезпеки систем аварійного електропостачання АЕС.

Ключові слова: кібербезпека, критична інфраструктура, атомна електростанція, система аварійного електропостачання, машинне навчання, випадковий ліс, класифікація кібератак, вебдодаток.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI – Artificial Intelligence, штучний інтелект, штучний інтелект;

DDoS – Distributed Denial of Service, розподілена атака на відмову в обслуговувані;

IDS – Intrusion Detection System, система виявлення вторгнень;

IPS – Intrusion Prevention System, система запобігання вторгнень;

IT – інформаційні технологіїї;

loT – Internet of Things, інтернет речей;

NIDS – Network Intrusion Detection System, мепрежева система виявлення вторгнень;

PLC – Programmable Logic Controller, програмований логічний контролер;

SCADA – Supervisory Control and Data Acquisition, система диспетчерського управління та збору даних;

ICS – Industrial Control Systems, промислові системи управління;

SIEM – Security Information and Event Management, система управління інформаційною безпекою;

ML – Machine Learning, машинне навчання;

APT – Advanced Persistent Threat, складна тривала загроза;

СКІ – системи критичної інфраструктури;

ЦОД – центр обробки даних;

АЕС – атомна електростанція;

LSTM – Long Short Term Memory, довга короткочасна пам’ять;

VPN – Virtual Private Network, віртуальна приватна мережа;

DNS – Domain Name System, система доменних імен;

API – Application Programming Interface, інтерфейс прикладного програмування.

ВСТУП

Актуальність. Досліджуючи специфіку функціонування сучасних систем критичної інфраструктури (СКІ), стає очевидним той факт, що абсолютна більшість активно використовують комп’ютерні та робототехнічні системи у процесі своєї діяльності. Така тенденція покликана, в першу чергу, необхідністю динамічної обробки великих об’єктів даних з метою подальшого аналізу останніх, що вимагає високого рівня точності та потужних обчислювальних ресурсів, якими володіють сучасні комп’ютерні системи. Атомні електростанції (АЕС), як об’єкти з одним з найбільших рівнів ризику, не є винятком, оскільки людський фактор може призвести до катастрофічних наслідків. Сучасні АЕС використовують широкий спектр технологічного обладнання та автоматизованих систем контролю якості всіх процесів, зокрема проміжних, такий рівень диференціації необхідний для досягнення задекларованих норм технічної безпеки. Однак, у свою чергу, активне використання великої кількості цифрових технологій, підвищує ризик й імовірність реалізації кібератак на модулі таких систем та їх підсистем. Такі кібератаки можуть мати різні цілі, однак враховуючи специфіку функціонування АЕС, стає зрозумілим, що довільна кіберфізична атака може призвести або до витоку важливої інформації, або до техногенної катастрофи.

АЕС є критичними об’єктами інфраструктури більшості країн світу, електроенергія, що виробляється на таких об’єктах займає значну частку усіх енергоресурсів країн, а також активно експортується. З огляду на це, будь-які зміни в нормальному процесі функціонування АЕС, з високою ймовірністю призведуть до таких наслідків як перебої в електропостачанні та шкода екології. Також одним із найбільш важливих і ключових модулів довільної АЕС є система аварійного електропостачання, що слугує буфером для унеможливлення знеструмлення даного об’єкта критичної інфраструктури, тобто цей модуль дозволяє мінімізувати ймовірність виникнення аварійних ситуацій на об’єкті. Саме коректна і безперебійна робота даної системи

дозволяє безпечно зупинити реактор та підтримувати його в безпечному стані. Підтримка кіберфізичної безпеки системи аварійного електропостачання є надзвичайно важливим аспектом.

Узагальнюючи, також варто зазначити, що зі зростанням обсягів даних, що обробляються і зберігаються на серверах об’єктів критичної інфраструктури, зросла кількість кібератак на інформаційні центи зберігання даних, що включають промислові системи управління (ICS) та системи диспетчерського управління та збору даних (SCADA). Саме отримання таких масивів даних, дозволяє зловмисникам отримувати критично важливу інформацію і підривати стабільність в регіоні функціонування АЕС. Це досягається як завдяки порушенню роботи обладнання, так й завдяки фізичним спотворенням інструкцій роботи модулів. Крім того, процес посилення захисту на об’єктах критичної інфраструктури, зокрема на АЕС, відбувається досить повільно і не включає в себе використання сучасних методів та підходів до захисту, шифрування та обробки даних, особливо великих розмірів, що нездатні зберігатись на стаціонарних комп’ютерах. Тому застосування сучасних алгоритмів машинного навчання є одним із ключових факторів, що дозволять мінімізувати ризики витоку даних, а також дозволять операторам відслідковувати потенційні кібератаки на всіх точках входу, особливо, на корпоративних мережах, що є найбільш уразливими до проведення атак.

Застосування сучасних методів штучного інтелекту є необхідністю, оскільки вони здатні ефективно й точно опрацьовувати великі масиви вхідних даних, серед яких можуть бути аномалії, що сигналізуватимуть про потенційну загрозу. Більш того, такий підхід дасть змогу не лише виявляти, а й запобігати виникненню атак на об’єкти критичної інфраструктури, оскільки такі моделі можуть бути імплементовані як підсистеми в усіх стандартних модулях АЕС, що дозволить децентралізувати аналітику даних та, відповідно, підвищить стійкість систем до різноманітних кіберфізичних атак.

Метою кваліфікаційної роботи є розробка модуля програмного забезпечення, використання алгоритмів штучного інтелекту для дослідження та оцінка уразливості системи аварійного електропостачання АЕС до кібератак.

Досягнення мети потребує розв'язання таких задач:

1. Проаналізувати сучасний стан кібербезпеки в системах критичної інфраструктури, зокрема в ядерній галузі та системах управління АЕС;
2. Розробити підхід до збирання та генерації даних для навчання моделей машинного навчання;
3. Розробити механізм оцінювання уразливості до кібератак систем аварійного електропостачання АЕС на базі штучного інтелекту.

Об'єкт дослідження: процес застосування класифікаційних алгоритмів штучного інтелекту для оцінки уразливості системи аварійного електропостачання АЕС до кібератак.

Предмет дослідження: система аварійного електропостачання атомної електростанції (АЕС) та її уразливість до кібератак.

Новизна: запропоновано підхід до виявлення кібератак на системи аварійного електропостачання атомних електростанцій, що базується на використанні алгоритмів машинного навчання. Новизна дослідження полягає у розробці спеціалізованог набору даних, що відображає поведінку систем диспетчерського управління та збору даних (Supervisory Control and Data Acquisition, SCADA) в умовах кіберзагроз, а також у поєднанні методів аналізу з класичними алгоритмами класифікації. Такий підхід дозволяє точніше виявляти потенційно небезпечні дії у критичних системах реального часу.

Практичною цінністю отриманих результатів: можливість впровадження розробленої системи класифікації кібератак для підвищення надійності та безпеки систем аварійного електропостачання АЕС або результати роботи можуть бути використані під час побудови системи раннього виявлення кіберзагроз на об'єктах критичної інфораструктури. Запропоноване рішення дозволяє своєчасно виявляти аномальні дії у критичних інформаційно-керуючих системах, мінімізуючи ризики порушення стабільної роботи обладнання в

умоваї кібератак, оскільки складність та потужність шкідливого програмного забезпечення постійно зростає.

Апробація: Зіпман Р.О. Штучний інтелект в оцінці уразливості систем аварійного електропостачання АЕС до кібератак / Зіпман Р.О., Браіловський М.М. // Інформаційні технології : ІІ Міжнародна науково-практична конференція, 11 квітня 2025 року : тези доп. – К.: Київський нац. ун-т імені Тараса Шевченка, 2025. – С. 32-34.

Оцінка сучасного стану проблеми на основі вітчизняної та зарубіжної літератури. Проблематика кіберфізичної безпеки в галузі ядерних технологій являє собою вкрай популярне та науково-доцільне направлення, що приваблює велику кількість дослідників, зокрема в сфері захисту інформації та кібербезпеки. Дослідження способів захисту інформації та систем об’єктів критичної інфраструктури популярна тематика як серед українських дослідників, так й закордоном.

Зарубіжні дослідники, зокрема Wilson, Clay [1], Haber, Eldar, and Tal Zarsky [2], Brechbühl, Hans [3] досліджують основні принципи та підходи до забезпечення кібербезпеки на різних об’єктах критичної інфраструктури. У роботах згаданих дослідників наголошується на таких методах захисту інформації як сегментація мережі, моніторинг активності та регулярне оновлення програмного забезпечення.

Також, враховуючи збільшення кількості атак на ядерні об’єкти, проводяться активні дослідження, спрямовані на аналіз побідних інцидентів, з метою розробки рекомендацій з унеможливлення їх негативного впливу, в майбутньому, зокрема у роботі S. Karnouskos [4] розглядається відомий інцидент зі Stuxnet, який продемонстрував вразливість промислових систем управління до кібератак. Автор наголошує на необхідності розробки ефективних методів виявлення та протидії таким загрозам.

Серед найбільш вагомих дослідників процесів захисту об’єктів критичної інфраструктури доцільно виділити В.В. Мохора [5], О. М. Дибач [6], що розглядають способи оцінювання ризиків виникнення кібератак на ядерні

об’єкти, зокрема АЕС, а також займаються розробкою специфічних методів аналізу та моделювання поширення кіберфізичних загроз, що дозволяє розроблювати практичні рекомендації по їх стриманню. Особливу увагу шляхам попередження виникнення кібератак на об’єкти критичної інфраструктури розглядають такі дослідники як Kovaliv, M., Skrynkovskyy, R., Nazar, Y. [7]. Однак, незважаючи на значну кількість сучасних досліджень та запропонованих методів, використання інтелектуальних агентів та методів штучного інтелекту, для захисту об’єктів критичної інфраструктури, досі, залишається непопулярним підходом і тільки починає поширювати в розвинених країнах заходу, а тому й потребує проведення постійних досліджень та кількісних експериментів, що, потенційно, пришвидшить поширення такого підходу.

Галузь застосування. Отримані результати та програмний комплекс може бути використаних на реальних об’єктах критичної інфраструктури, зокрема на ядерних об’єктах, інші об’єкти критичної інфраструктури, також у процесах промислової автоматизації, що працює з критичними даними та становлять інтерес для національної безпеки.

РОЗДІЛ 1
АНАЛІЗ ДЖЕРЕЛ І ПРОБЛЕМИ

1.1 Кібербезпека в системах критичної інфраструктури

Об’єкти критичної інфраструктури складають основний масив засобів та ресурсів, що дозволяє ефективно функціонувати суспільству та розвиватись в різних сферах життя. Функція об’єктів критичної інфраструктури прямо корелює з причиною виникнення необхідності забезпечення безпеки таких об’єктів на всіх рівнях, що спричиняє динамічне удосконалення методів запобігання та мінімізації загроз, що пов’язані з кібератаками на об’єкти критичної інфраструктури [8]. Оскільки сучасний стан розвитку технологій покликаний частково замінити людський ресурс, в більшості процесів, які пов’язані з обчисленнями, то на об’єктах критичної інфраструктури постійно впроваджують системні модулі, підсистеми та цілі системи, що дозволяють автоматизувати процес збору та керування даними, в будь-якому вигляді, це, у свою чергу, створює певні шляхи для проведення комплексних кіберфізичних атак з метою нанесення шкоди суспільству [9]. Більшість сучасних дослідників сходяться в думці, що потенційні кібератаки на об’єкти критичної інфраструктури, однозначно, призведуть до втрати важливих й стратегічних даних, фінансових збитків [10].

Галузь ядерних досліджень та функціонування АЕС є найбільш небезпечною сферою, за більшістю класифікацій ризиків, це пов’язується зі специфікою сировини що обробляється на АЕС. Також це є причиною використання складних обчислювальних систем та великої кількості персоналу на таких станціях, оскільки забезпечення ефективної роботи – ключовий аспект функціонування будь-якої АЕС [11].

Розглядаючи реальні прецеденти атак на критичну інфраструктуру, зокрема на ядерні об’єкти, варто згадати Stuxnet – шкідливе програмне забезпечення, яке було спеціально розроблене для атаки на ядерні об'єкти Ірану

[12]. Саме ця атака довела, що професійно підготовлене та спроектоване програмне забезпечення, шкідливого спрямування, може впливати не лише на витік даних, а й на процес функціонування ядерного об’єкта, тобто, наразі, найбільшу небезпеку становлять саме атаки, спрямовані на зміну налаштувань програмного забезпечення (ПЗ) та модулів систем об’єктів критичної інфраструктури [13].

Процес захисту об’єктів критичної інфраструктури доволі складний, оскільки є сукупністю залежних між собою підпроцесів, до яких можна віднести взаємодію людських ресурсів, технічне оснащення, організаційні та безпекові аспекти [14]. Найбільш складним та важливим, очевидно, є впровадження належного рівня технічного оснащення, зокрема впровадження міжмережевих каналів, систем виявлення та попередження вторгнень в системи та підсистеми, криптографічне шифрування даних а також регулярні створення резервних копій серверів та кластерів. Не менш важливим є розробка коректних, юридично обґрунтованих політик забезпечення безпеки на об’єктах критичної інфраструктури, тобто розробка чітких інструкцій та алгоритмів поведінки є важливим аспектом, що корелює з ефективністю роботи технічного оснащення, оскільки компетенція персоналу визначає те, наскільки ефективним буде та чи інша кібератака [15]. Враховуючи окреслену складність організації безпеки на об’єктах критичної інфраструктури, першочергово необхідна співпраця між урядовими представниками та науковим товариством, оскільки імплементація наукових результатів, алгоритмів та підходів, можлива лише на найвищому рівні влади, оскільки зміна політики керування тим чи іншим об’єктом вимагає перенавчання персоналу, створення нових інструкцій, стандартів та координації між різними відомствами, та регуляторами [16].Удосконалення способів захисту СКІ постійно супроводжується удосконаленням шкідливого програмного забезпечення та стратегій атак, тому розробка нових методів захисту від кібератак та виявлення загроз повинно відбуватись динамічно, що спроможні досягти саме інтелектуальні агенти та моделі, оскільки можуть діяти і навчатись відповідно до наданих даних.

1.2 Специфікація атак на системи критичної інфраструктури

Для атак на об’єкти критичної інфраструктури, зловмисники використовують різноманітні методи та підходи, що варіюються, в залежності від мети атаки, а також об’єкт атаки, оскільки атака може відбуватись на різні підсистеми об’єктів критичної інфраструктури, бо ідентифікувати атаку на довільну підсистему складніше, ніж на цілий об’єкт [17]. Одним з найпоширеніших типів атак на системи критичної інфраструктури є атаки на промислові системи управління (Industrial Control Systems, ICS) та системи диспетчерського управління та збору даних (Supervisory Control and Data Acquisition, SCADA) [17]. Саме ці об’єкти є критично важливими ланками у процесі якісного функціонування об’єктів критичної інфраструктури. Розглянемо найбільш популярні специфікації атак на об’єкти критичної інфраструктури:

1. Зловмисники можуть отримати несанкціонований доступ до систем ICS/SCADA через вразливості у віддалених підключеннях, таких як віртуальна приватна мережа (Virtual Private Network, VPN), схему такої атаки доцільно представити у форматі діаграма послідовності атаки через віддалене підключення (рис. 1.1) [18]:

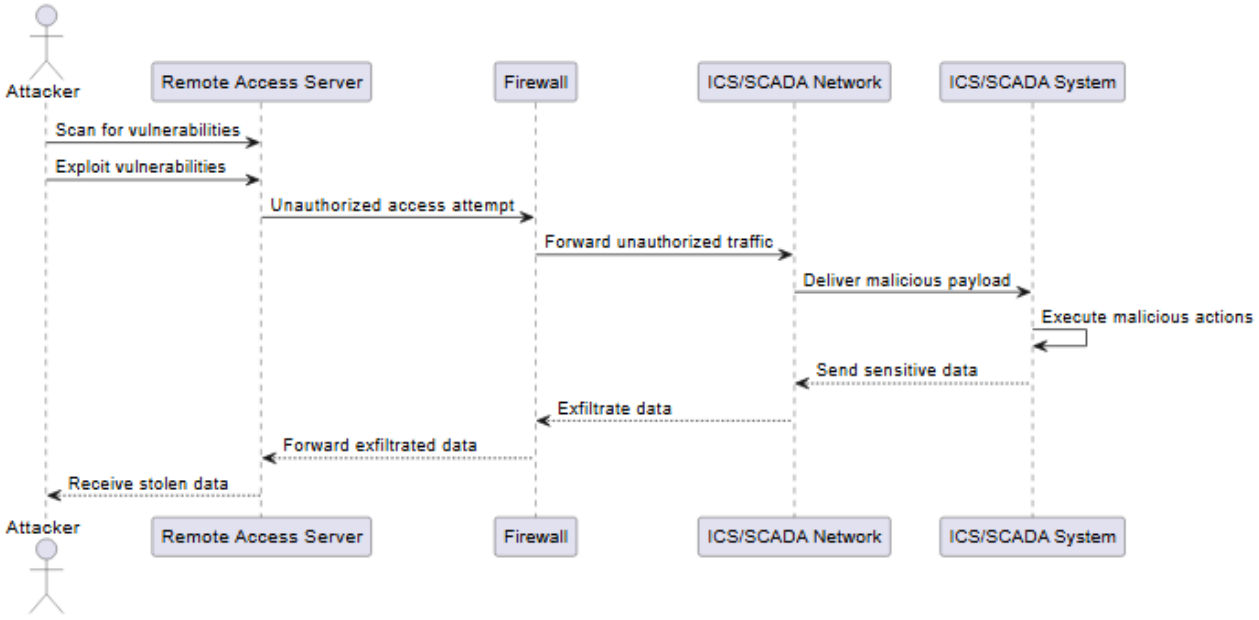

Рисунок 1.1. Діаграма послідовності атаки через віддалене підключення

2. Найбільш популярними типами атак є атаки, що передбачають розробку шкідливого програмного забезпечення не лише для етапу переналаштувань системи, а й для етапу проникнення в систему, до такого ПЗ можна віднести Stuxnet. Такий підхід є найбільш небезпечним, бо активні засоби захисту, можуть бути знешкоджені, а шкода системі може мати катастрофічні наслідки, особливо, якщо шкідливе програмне забезпечення поширюється безперешкодно, даний тип атак доцільно представити у вигляді діаграми послідовності атаки через шкідливе програмне забезпечення (рис. 1.2) [19]:

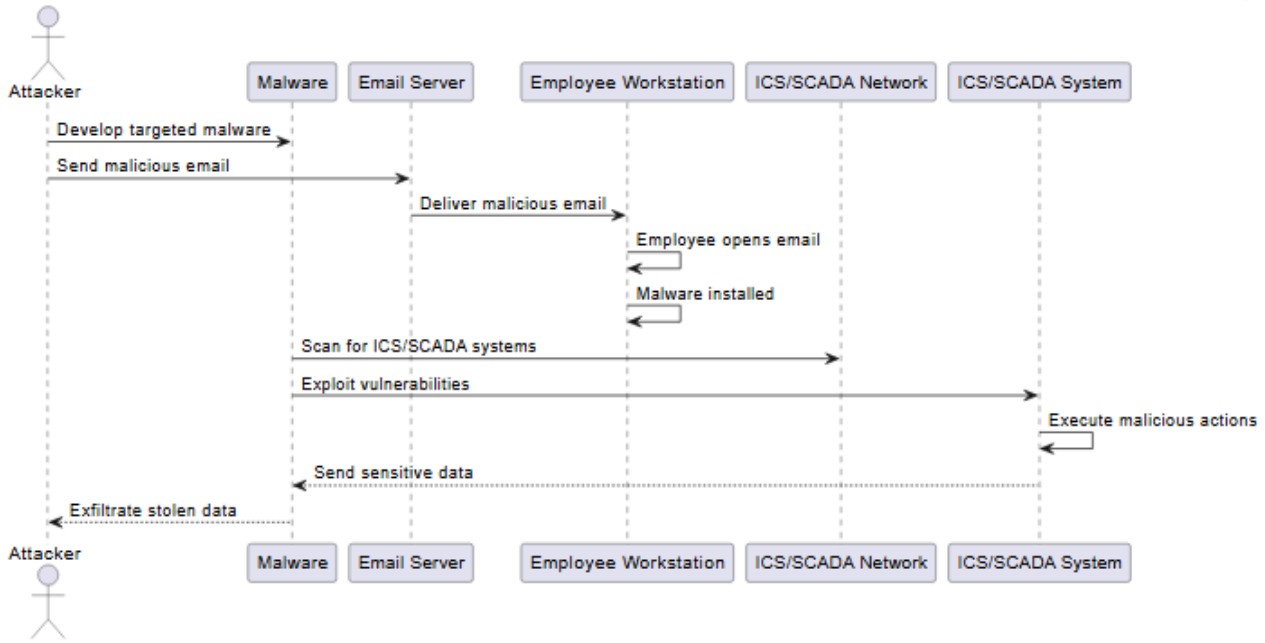

Рисунок 1.2. Діаграма послідовності атаки через шкідливе програмне забезпечення

3. Зловмисники можуть отримати фізичний доступ до компонентів ICS/SCADA, таких як програмовані логічні контролери (Programmable Logic Controller, PLC) або людино-машинні інтерфейси (Human Machine Interface, HMI), і здійснювати атаки безпосередньо зсередини інфраструктури, що є найбільш небезпечно, оскільки виявити подібну активність вкрай складно, даний тип атак представимо у вигляді діаграми послідовності атаки через отримання фізичного доступу (рис. 1.3) [20]:

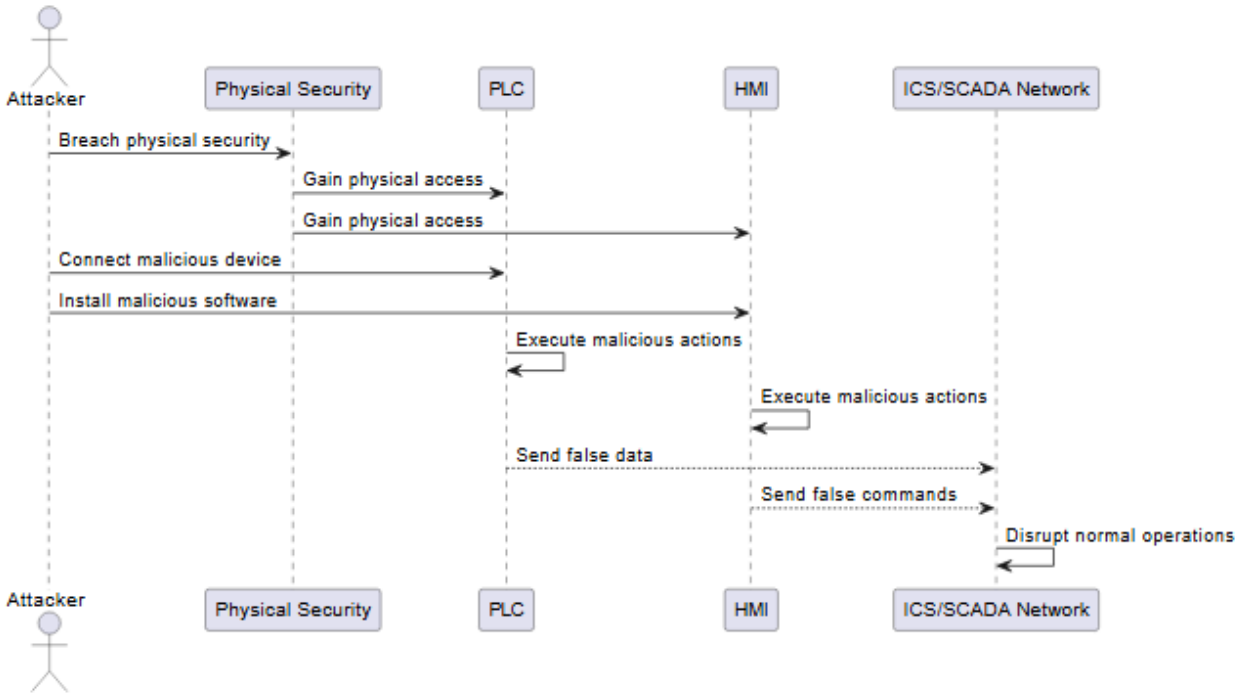

Рисунок 1.3. Діаграма послідовності атаки через отримання фізичного доступу

4. Не менш небезпечним є застосування методів соціальної інженерії для отримання доступу до даних працівників об’єктів кричної інфраструктури, що дозволить отримати дані до модулів відповідних систем та підсистем, що є небезпечним, оскільки психологічний вплив засобами фішингу та претекстингу є вкрай нестабільним, бо важко зробити прогноз поведінки того чи іншого працівника. Такі атаки можна представити у вигляді діаграми послідовності атаки через методи соціальної інженерії (рис. 1.4) [21]:

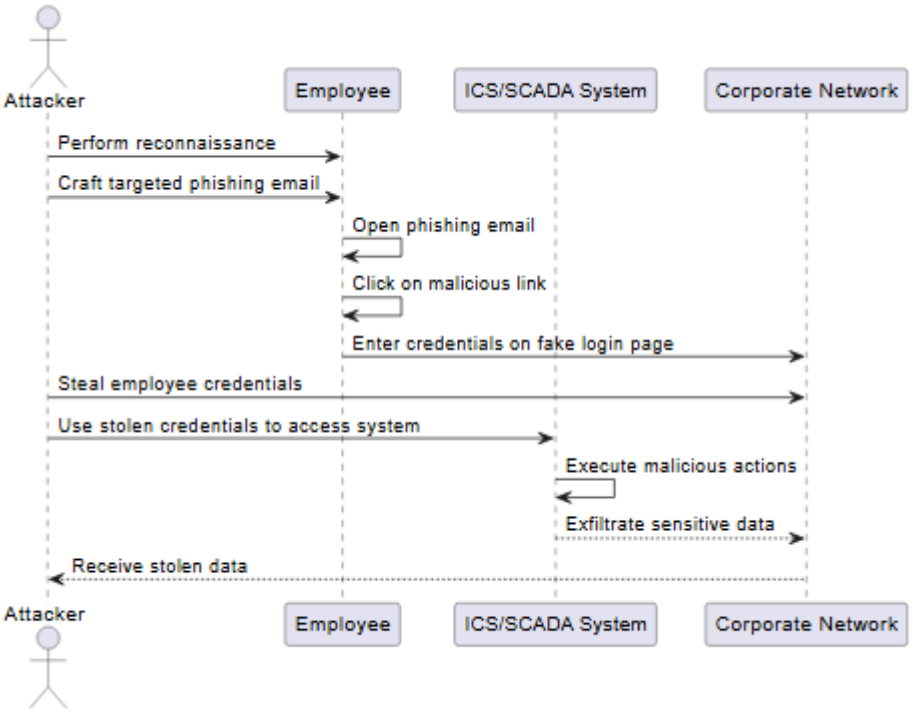

Рисунок. 1.4. Діаграма послідовності атаки через методи соціальної інженерії

5. Менш популярний, однак, вкрай небезпечний тип атак – атаки в процесі розробки програмного забезпечення, тобто, в такому випадку, зловмисник імплементує шкідливе ПЗ, у «сплячому режимі», на етапі розробки систем або підсистем і запускає його у довільний час. За умови отримання зловмисником доступу до архітектурного шаблону програмного забезпечення (Model View Controller, MVC), виявити шкідливі модулі, практично, неможливо, тому регулювання повинно відбуватись на етапі відбору розробників, шляхом ретельної перевірки особистості. Такі атаки можна представити у вигляді діаграми послідовності атаки через ланцюг поставки ПЗ (рис. 1.5) [22]:

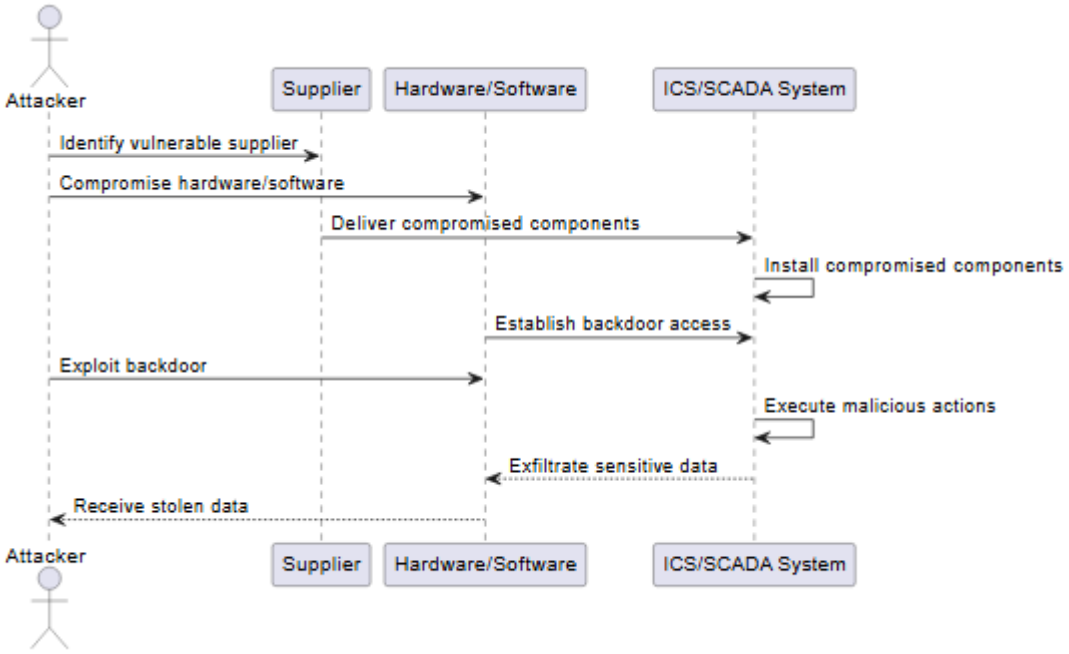

Рисунок 1.5. Діаграма послідовності атаки через ланцюг поставки ПЗ

6. Найбільш популярними та шкідливими й небезпечними є кібератаки на ядерні об'єкти або об'єкти, що пов'язані з ядерними програмами. Саме цей тип атак є найбільш небезпечним, оскільки несе загрозу не лише обмеження ресурсів для користування, а й техногенного характеру. Більш того, специфіка роботи ядерних об’єктів така, що вимагає постійного контролю за системами охолодження, водопостачання, електропостачання і навіть незначний перебій призведе до порушення роботи алгоритмів мережі. Атаки на ядерні об’єкти, за своєю природою, складні і узагальнити їх доцільно у вигляді діаграми послідовності атаки на ядерні об’єкти (рис. 1.6) [23]:

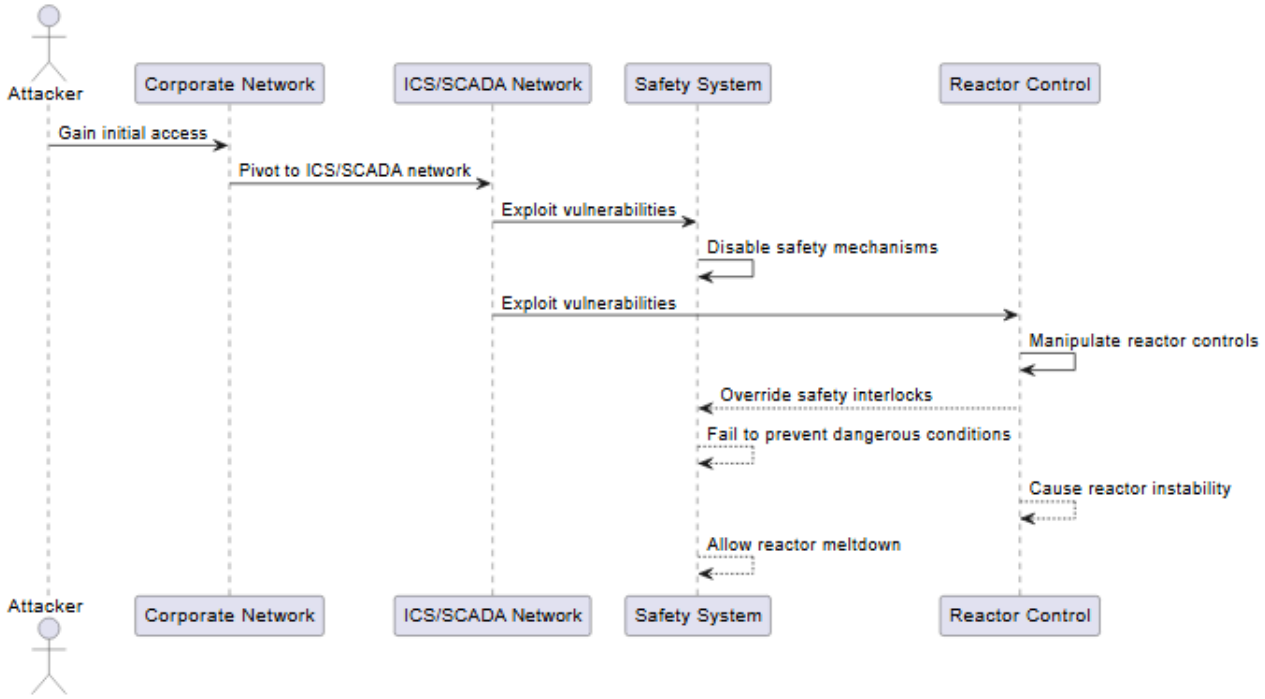

Рисунок 1.6. Діаграма послідовності атаки на ядерні об’єкти

З огляду на окреслені категорії атак на об'єкти критичної інфраструктури, доцільно розглянути відомі та дієві шляхи реагування та попередження кібефізичних атак на СКІ.

1.3 Підходи до захисту об’єктів критичної інфраструктури

Захист об'єктів критичної інфраструктури від кібератак є складним завданням, яке вимагає комплексного підходу. Він повинен включати в себе технічні, організаційні та людські аспекти безпеки [24]. Одним із найбільш вагомих аспектів захисту об’єктів критичної інфраструктури від кібератак є розробка багаторівневої, комплексної інфраструктури таких систем, підсистем та навіть окремих модулів, що називається Defense-in-Depth, тобто необхідність посилення безпеки через вертикальне розширення систем. Для цього необхідно використовувати декілька рівнів захисту даних та інформації, що обов’язково повинно включати сегментацію мережі на підмережі, активне використання брандмаузерів NAT-технологій, підсистем, що здатні виявляти та автоматично

запобігати виділеній множині вторгнень (Intrusion Detection System, IDS та Intrusion Prevention System, IPS). Структуру багаторівневого захисту доцільно представити у вигляді такої схеми побудови Defense-in-Depth архітектури (рис. 1.7) [24]:

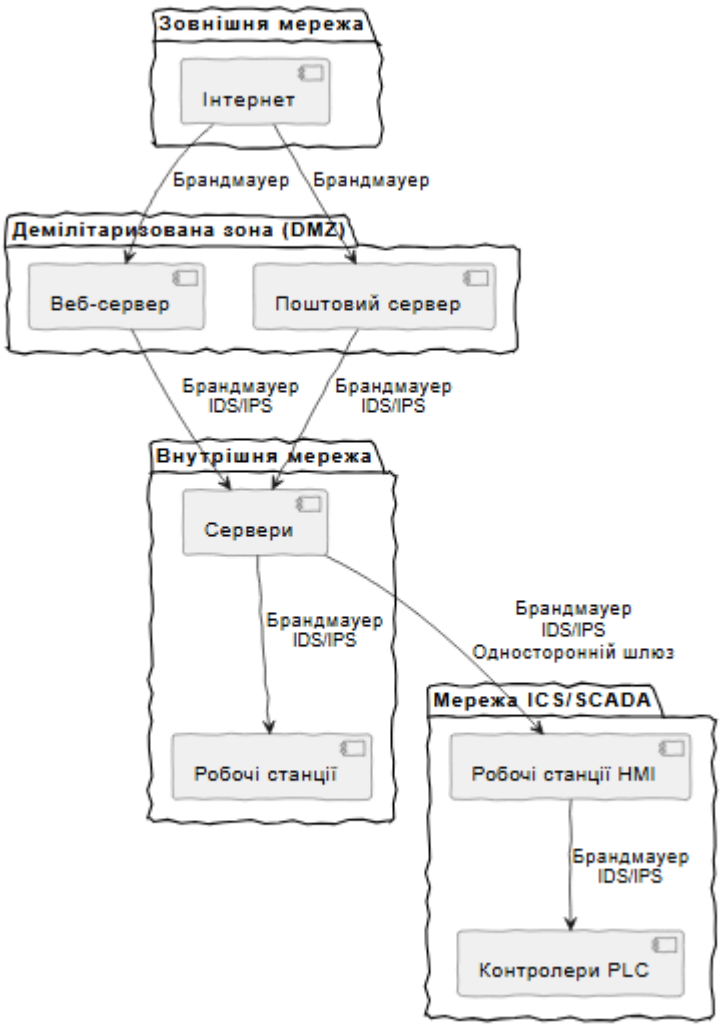

Рисунок 1.7. Схема побудови Defense-in-Depth архітектури

Не менш важливим рішенням є розмежування прав доступу та динамічне регулювання й ідентифікація режиму доступу з усіх пристроїв мережі, таким чином стає можливим запобігання несанкціонованого доступу, а також децентралізація збереження даних, тобто зловмисник, за умови вторгнення, не зможе отримати повноцінні масиви релевантної інформації. Для цього необхідно імплементувати в мережу багатофакторної автентифікації (Multi Factor Authentication, MFA), принципу найменших привілеїв (Principle of Least

Privilige, PoLP) та регулярний моніторинг активності користувачів, схему такого захисту доцільно узагальнити таким чином як діаграма послідовності моніторингу активності користувачів мережі (рис. 1.8) [25]:

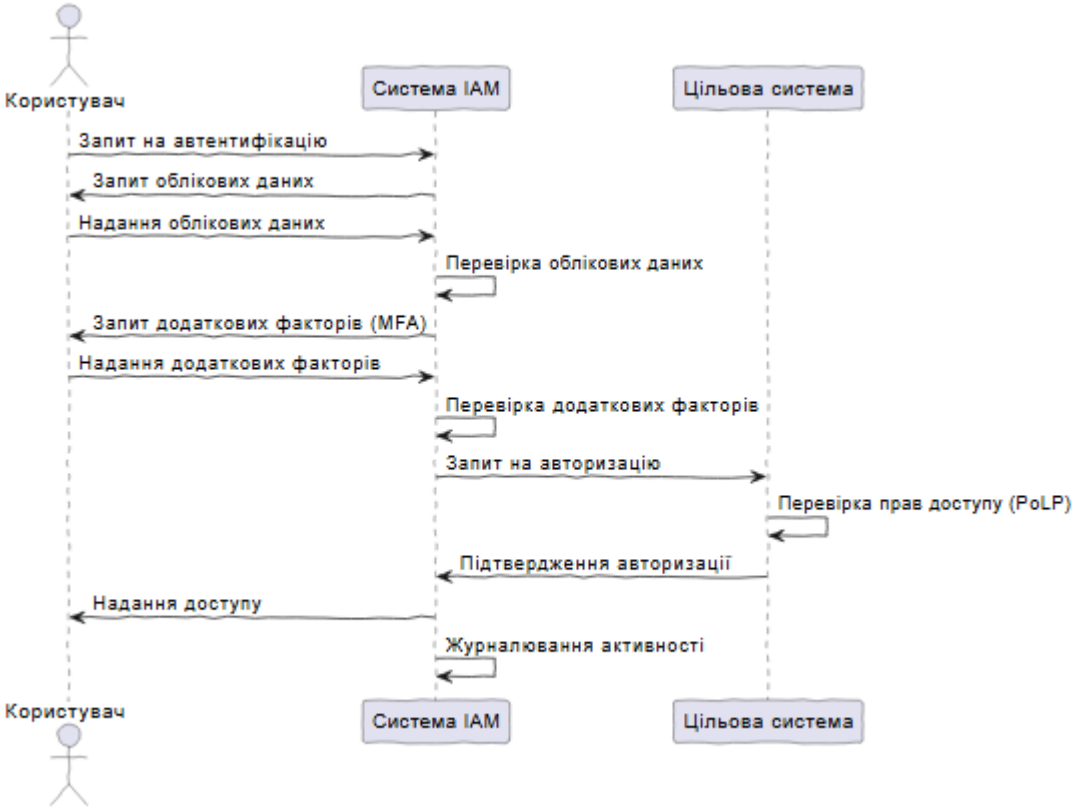

Рисунок 1.8. Діаграма послідовності моніторингу активності користувачів мережі

Технічно важливим процесом, що має викликатись як підпроцес на всіх системах є детекція аномалій в мережевій активності та в структурі даних, що поступають на системи критичної інфраструктури. Оскільки аномалії, у своєму статистичному виявленні чітко вказують або на технічні проблеми та несправності, або на потенційні загрози і їх виявлення дозволяє уникнути витоку інформації із системи. Процес виявлення аномалій включає в себе використання систем управління інформаційною безпекою та подіями (Security Information And Event Management, SIEM), які збирають та автоматично аналізують журнали подій та логів підсистем. Процес детекції аномалій відобразимо у вигляді такої схеми роботи підсистеми виявлення аномалій (рис. 1.9) [26]:

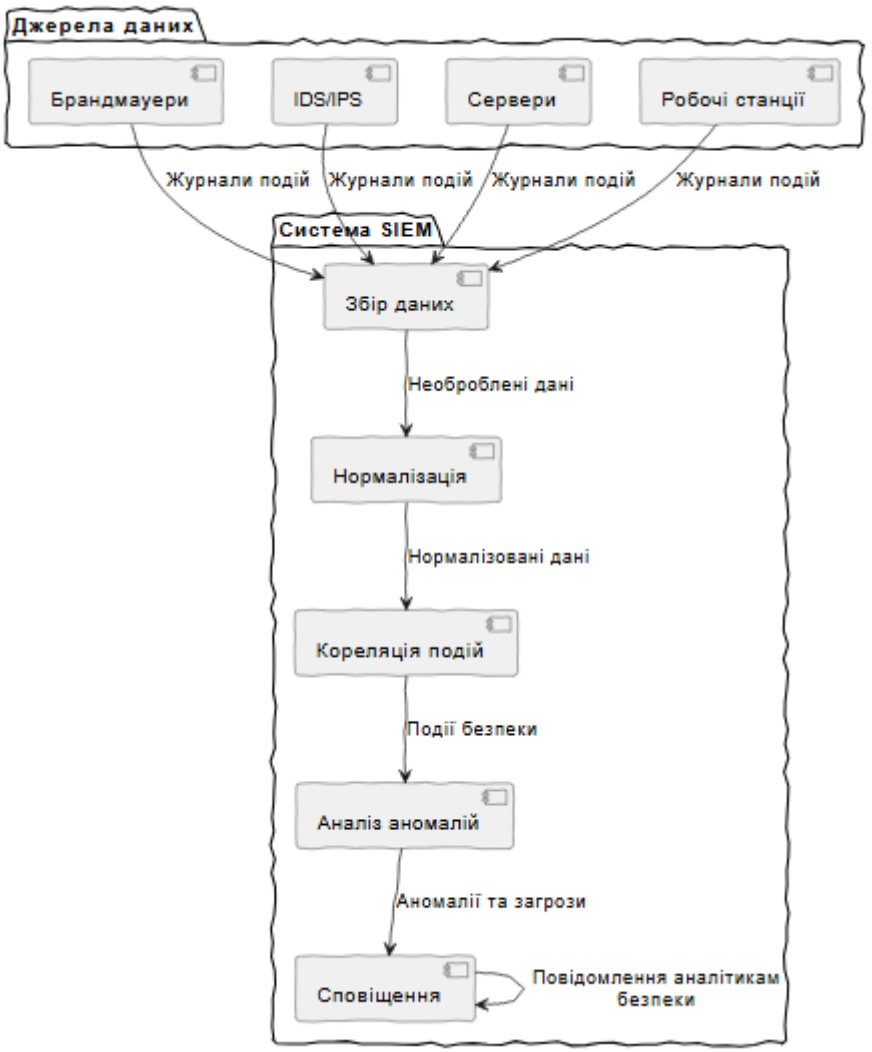

Рисунок 1.9. Схема роботи підсистеми виявлення аномалій

Моделювання процесів проникнення в системи об'єктів критичної інфраструктури ще один із методів уникнення загрози витоку даних та, загалом, кібератак, оскільки моделюючи ситуацію поширення шкідливого ПЗ або несанкціонованого доступу, стає можливим превентивно розробляти комплекс дій, спрямований на унеможливлення проникнення в систему ззовні. Більш того, такий підхід дозволяє виявляти вразливості в системах та точки входу, що, потенційно, можуть бути використані при спробі кіберфізичної атаки. План моделювання проникнення в систему представимо таким чином як діаграма послідовності моделювання проникнення в систему (рис. 1.10) [27]:

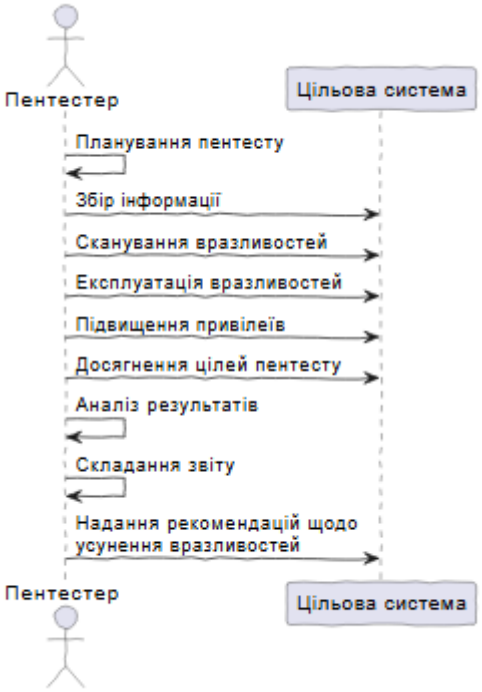

Рисунок 1.10. Діаграма послідовності моделювання проникнення в систему

1.4 Обґрунтування необхідності проведення дослідження

Сучасний стан розвитку систем інформаційної та кіберфізичної безпеки на об'єктах критичної інфраструктури вказує, згідно з більшістю досліджень, що постійне удосконалення методів та підходів до захисту останніх є критично важливим на шляху забезпечення процесу безперебійного функціонування СКІ. Часто виникає потреба не в заміні існуючих методів та архітектур, а в надбудові нових алгоритмів додаткового захисту як на етапах виявлення, так і на етапах знешкодження шкідливого програмного забезпечення.

З огляду на це, ті методи, що існують як для виявлення аномалій в системах та логах систем, так і для прогнозування загроз повинні бути удосконалені сучасними алгоритмами, що підвищать відмовостійкість систем та дозволять операторам відповідних модулів проводити більш якісний аналіз вхідних даних. Більш того, імплементація сучасних алгоритмів в існуючих системах критичної інфраструктури, викликана ще й постійним процесом удосконалення шкідливого програмного забезпечення, його функціональних

можливостей та здатності обходити захисні бар’єри кіберфізичних систем. Це є причиною вибору, в якості додаткового механізму захисту, алгоритмів штучного інтелекту, бо вони здатні динамічно підлаштовуватись до зміни даних, за умови попереднього навчання, а також мають меншу кількість обмежень, що притаманні евристичних і аналітичним алгоритмам прогнозування. А у ситуації виявлення нових типів атак, лише засоби інтелектуальних штучних агентів дозволять визначити хоча б мінімальну ймовірність виявлення атаки або аномалії в логах даних систем та підсистем.

Резюмуючи, можна виділити такі аспекти, що викликають необхідність проведення даного дослідження:

1. Постійно зростаюча загроза кібератак на СКІ, зокрема ядерні об’єкти та їх підсистеми, такі як системи аварійного електропостачання;
2. Недостатній рівень розширення архітектури більшості об’єктів критичної інфраструктури та необхідність надбудови сучасних алгоритмів виявлення ненормальної поведінки в системі;
3. Сильна обмеженість детермінованих методів оцінки ризику виникнення загроз та запобігання останнім, особливо, в умовах постійного розвитку якості шкідливого ПЗ;
4. Високий та зростаючий потенціал класифікаційних алгоритмів штучного інтелекту, що дасть змогу більш якісно оцінювати ризики потрапляння шкідливого ПЗ в системи об’єктів критичної інфраструктури.

Результати даного дослідження можуть стати основою для подальшого формування нових більш стійких та просунутих алгоритмів виявлення кіберзагроз та попередження їх появи.

Висновок до розділу 1

Проаналізовано актуальні загрози кібербезпеки для систем критичної інфраструктури, особливо вразливих до кібератак через високий рівень автоматизації та використання ICS/SCADA-систем. Визначено, що об’єкти

критичної інфраструктури є ключовими для стабільного функціонування держави, а їх компрометація – загрозою національній безпеці. На прикладі атак, таких як Stuxnet, підкреслено реальність і потенційні наслідки кіберфізичних впливів на ядерні об’єкти. Розглянуто основні типи атак: через віддалене підключення, шкідливе ПЗ, фізичний доступ, соціальну інженерію, закладення в процесі розробки ПЗ та атаки на ядерні об’єкти. Захист СКІ потребує комплексного підходу, що поєднує технічні рішення, нормативну підтримку, підготовку персоналу та активну взаємодію між урядовими і науковими структурами.

РОЗДІЛ 2
МЕТОДИКА ДОСЛІДЖЕННЯ І ВИБІР МАТЕМАТИЧНОГО АПАРАТУ

2.1 Обґрунтування вибору алгоритмів машинного навчання

Для вирішення поставленої задачі класифікації кіберзагроз в системі аварійного електропостачання АЕС, було прийняте рішення провести дослідження декількох алгоритмів штучного інтелекту, з метою вибору найбільш оптимального. Такий підхід дозволить виявити потенційні слабкості та преваги кожного з методів, а також дозволить кількісно оцінити складність класифікації загроз при їх масштабуванні.

Як відомо, в задачах класифікації даних вибір методів корелює з декількома умовами: швидкістю отримання результату та обсягом вхідних даних (навчальної вибірки). Більшість алгоритмів, які працюють доволі швидко, не оброблюють великі обсяги даних, що, очевидно, виникають на об’єктах критичної інфраструктури, де логування відбуваються щосекунди, або протягом декількох секунд. З огляду на це, у процесі даного дослідження, було прийняте рішення використовувати складні архітектурні рішення для вирішення задачі класифікації: глибокі нейронні мережі, рекурентні нейронні мережі та ансамблеву архітектуру, що поєднує велику кількість моделей.

Спершу дослідимо модель глибокого навчання, що базується на багатошарових нейронних мережах прямого поширення похибки, такі архітектури здатні виявляти складні закономірності в даних та класифікувати нелінійні залежності й, більш того, вони здатні працювати з розрідженими даними великих об’ємів, що робить їх потужним інструментом при розв’язанні класифікаційних задач [28].

Математично такі моделі являють собою послідовність прихованих шарів, що містять встановлену кількість нейронів, що поєднані між собою зв’язками. Таким чином дані, що подаються на вхід мережі передають їх на усі приховані шари, а на вихід з моделі отримуються мітки класів для кожного спостереження

з вхідної вибірки. Це досягається завдяки тому, що кожен прихований шар моделі виконує певне нелінійне перетворення вхідних даних витягуючи ознаки останніх, це перетворення, часто, називається активаційною функцією. Математично дану модель можна представити таким чином [29]:

Нехай $x -$ вхідний вектор ознак, W_i та $b_i -$ матриця ваг та вектор зміщення для i −го шару відповідно, $f_i -$ функція активації i −го шару, $d_i -$ рівень дропауту (dropout rate) для i −го шару. Тоді для довільного прихованого шару i виконується така послідовність дій [29]:

Лінійне перетворення:

$$z_i = W_i a_{\{i-1\}} + b_i \quad (2.1)$$

де $a_0 = x -$ вхідний вектор ознак.

Нелінійне перетворення за допомогою функції активації:

$$a_i = f_i(z_i) \quad (2.2)$$

де $f_i -$ функція активації.

Дропаут:

$$\tilde{a}_i = dropout(a_i, d_i) \quad (2.3)$$

де $dropout(x, d) -$ функція, що випадково встановлює елементи x в 0 з ймовірністю d.

Для останнього шару мережі, що є вихідним, застосовуються такі перетворення:

Лінійне перетворення такого вигляду:

$$z_o = W_o \tilde{a}_{o-1} + b_o \quad (2.4)$$

Softmax активація:

$$\hat{y} = softmax(z_o) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (2.5)$$

Паралельно з налаштуванням вагових коефіцієнтів, за описаними правилами, відбувається мінімізація функціоналу якості моделі, що оцінюється функцією втрат, що представлена категоріальною кросентропією у такому вигляді:

$$L = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} log(\hat{y}_{ij}) \quad (2.6)$$

де N – кількість прикладів, K – кількість класів, y_{ij} – справжня мітка (0 або 1) для i –го прикладу та j –го класу, $\hat{y}_{ij}$ – передбачена ймовірність j –го класу для i –го прикладу.

Узагальнити дану модель можна таким чином як архітектура моделі класифікатора глибокого навчання (рис. 2.1):

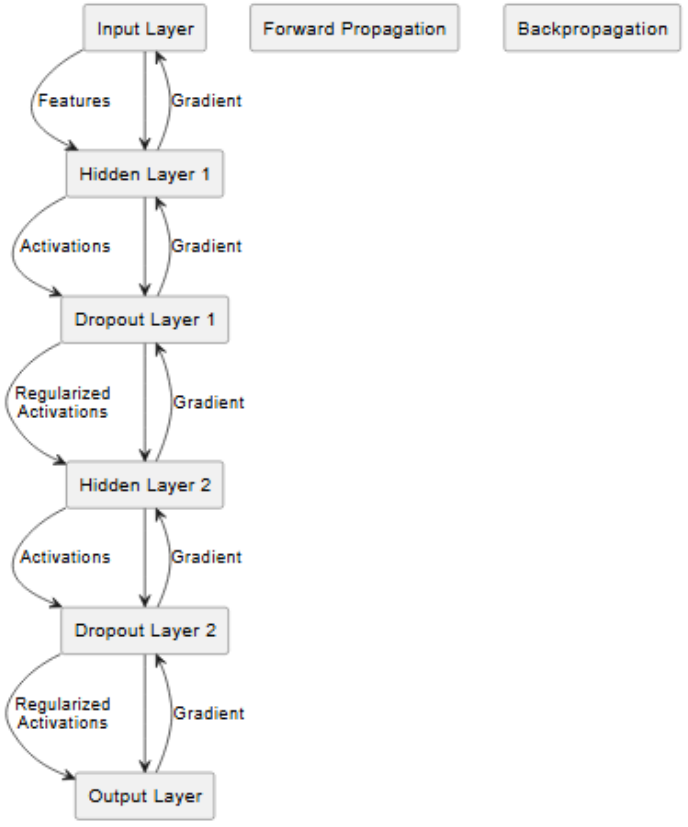

Рисунок 2.1. Архітектура моделі класифікатора глибокого навчання

Наступною розглянемо модель рекурентних нейронних мереж, що є вкрай потужним методом для розв'язання задач класифікації, оскільки ці моделі

враховують контекст в послідовностях даних, а отже ефективно вибудовують залежності між значеннями спостережень, що може бути ефективним для задач виявлення аномалій та попередження кібеінцедентів. В якості основної моделі було обрано модель довгої короткочасної пам’яті (Long Short Term Memory, LSTM), що представляє собою певну ділянку пам’яті, що складається з таких елементів: вхідного гейту (input gate), вихідного гейту (output gate), гейту забування (forget gate) та стану комірки (cell state). Саме набір цих параметрів дозволяє випадковим чином зберігати інформацію про залежності між ознаками вхідних даних. Математично дану модель можна представити таким чином [30]:

Нехай x_t – вхідний вектор ознак на кроці t, h_t – прихований стан LSTM-комірки на кроці t, c_t – стан комірки на кроці t. Тоді рівняння для LSTM-комірки можна записати наступним чином [30]:

Вхідний гейт:

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \quad (2.7)$$

Гейт забування:

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \quad (2.8)$$

Кандидат на оновлення стану комірки:

$$\widetilde{c_t} = \tanh tanh\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) \quad (2.9)$$

Оновлення стану комірки:

$$c_t = f_t c_{t-1} + i_t \widetilde{c_t} \quad (2.10)$$

Вихідний гейт:

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \quad (2.11)$$

Прихований стан:

$$h_t = o_t \tanh tanh\left(c_t\right) \quad (2.12)$$

де $\sigma(\cdot)$ – сигмоїдна функція;

$tanh(\cdot)$ – гіперболічний тангенс.

Для більшої ясності доцільно представити архітектуру моделі рекурентних нейронних мереж схематично, щоб зрозуміти яким чином та в якій послідовності відбувається обмін даними (рис. 2.2):

Рисунок 2.2. Архітектура моделі рекурентних нейронних мереж

Третьою було обрано ансамблеву модель, що поєднує в собі велику кількість простих моделей, такий підхід дозволить якісніше обробляти складні залежності в даних, а також оброблювати розріджені дані, в особливо великих масштабах. Основна проблема більшості класифікаторів, що повинні оброблювати великі масиви даних – перенавчання. Математично модель випадкового лісу, що складається з ансамблю дерев можна представити таким чином [31]:

1. Генерація M бутстреп-вибірок S_1, S_2, ..., S_M з початкового набору даних D шляхом випадкового вибору N прикладів з поверненням (N – розмір початкового набору даних);
2. Для кожної бутстреп-вибірки S_i:
   2.1. Побудова дерева рішень T_i на основі S_i, випадково вибираючи k ознак з K доступних ознак на кожному вузлі ($k \approx \sqrt{K}$ для класифікації);
   2.2. Вирощування дерева T_i до максимальної глибини або до досягнення мінімального розміру листового вузла, використовуючи критерій розщеплення Джині;
3. Для класифікації нового прикладу x:

3.1. Отримання прогнозу $\hat{y}$ від кожного дерева T_i шляхом проходження x через T_i до листового вузла;

3.2. Агрегація прогнозів $\hat{y}_1, \hat{y}_2, ..., \hat{y}_M$ шляхом більшості голосів (для класифікації) або усереднення (для регресії) для отримання фінального прогнозу $\hat{y}$.

Окремо розглянемо процедуру розщеплення за індексом Джині, що є ключовим аспектом в даній моделі:

$$Gini(p) = \sum_{i=1}^{K} p_i\left(1 - p_i\right) \quad (2.13)$$

де p_i – частка прикладів класу i в даному вузлі.

Випадковий ліс не потребує окремої фази навчання, оскільки кожне дерево будується незалежно на своїй бутстреп-вибірці. Саме цей аспект дозволяє ансамблю якісно вчитись на нерівномірно розподілених даних, тобто таких, де міток певних класів суттєво більше за кількість інших міток. Для звичайних моделей це може створити проблеми, перенавчання та хибну класифікацію.

Дану архітектуру ансамблю випадкового лісу доцільно представити у вигляді такої схеми (рис. 2.3):

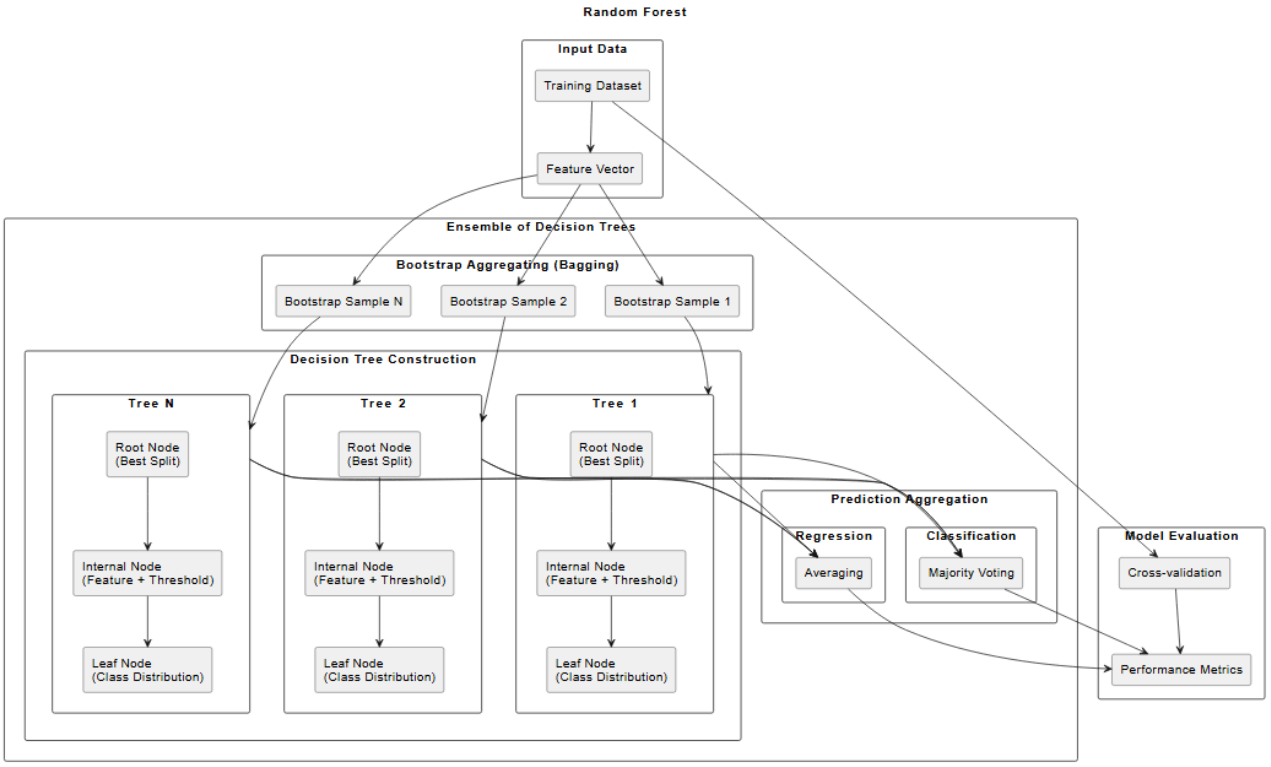

Рисунок 2.3. Архітектура ансамблю випадкового лісу

2.2 Розробка підходу до збирання і генерації даних

При розробці підходу до збирання та генерації даних для дослідження кібератак на систему аварійного електропостачання АЕС було враховано специфіку протоколу Modbus, який широко використовується в промислових системах управління та SCADA-системах [32]. Дані були отримані шляхом захоплення мережевого трафіку через послідовну лінію зв'язку, що дозволило отримати як інформацію про мережеві транзакції, так і про корисне навантаження на систему. Для генерації даних з подібних контролерів, використовують спеціалізований фреймворк, логіка якого може бути описана таки чином як схема логіки роботи фреймворку для збирання даних (рис. 2.4):

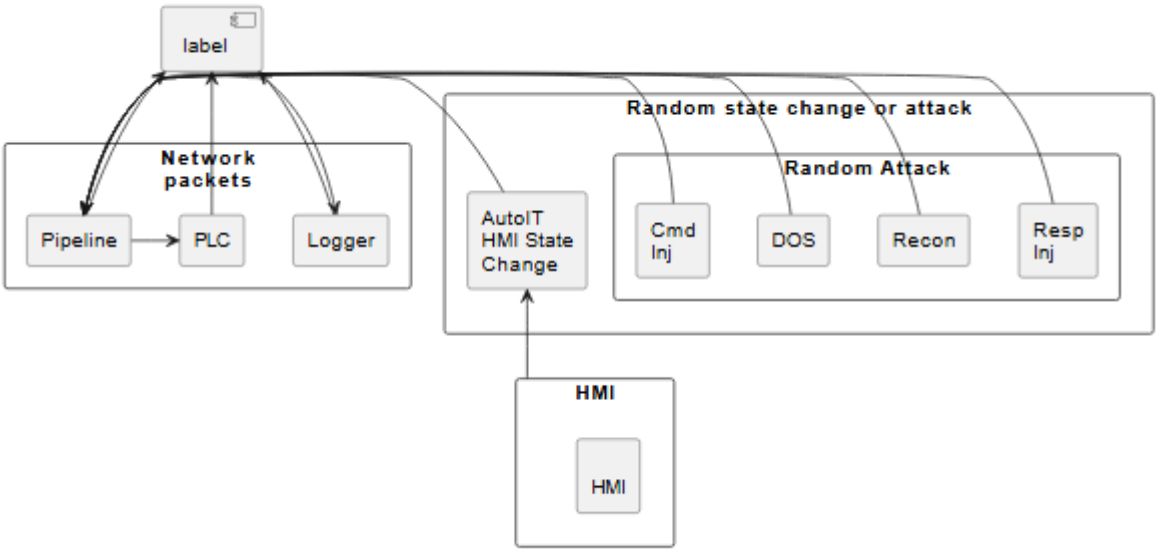

Рисунок 2.4. Логіка роботи фреймворку для збирання даних

Датасет представлений у форматі CSV, де кожен рядок відповідає одній мережевій транзакції. Це забезпечує гранулярність даних та дозволяє аналізувати кожну транзакцію окремо, що, в свою чергу, дозволить проводити більш точний аналіз та класифікацію, оскільки кількість параметрів в моделі значно збільшиться, у порівнянні з попередніми наборами даних, а отже дасть змогу ефективніше знаходити залежності між логуванням та типом атаки. Кожен запис у датасеті містить як інформацію про мережеву взаємодію (адреса станції, часова мітка, CRC тощо), так і інформацію про корисне навантаження (системні команди та дані про стан системи). Процес збирання даних доцільно представити у вигляді діаграми послідовності збору даних з системи (рис. 2.5):

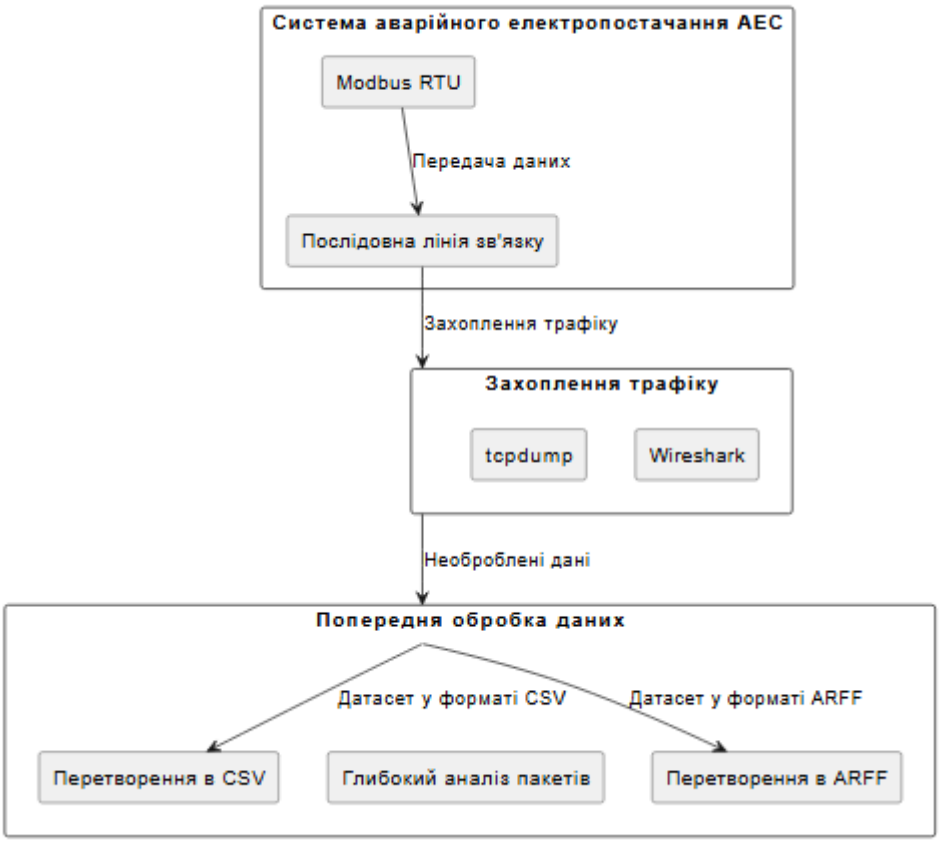

Рисунок 2.5. Діаграма послідовності збору даних з системи

Однак окрім збирання даних, за допомогою вказаного вище фреймворку, було згенеровано аномальну активність, та розроблено різні шаблони атак, що відображались у вигляді логувань, тобто вхідний масив даних був перетворений на такий, що містить випадкове число різних атак, цей процес представимо у вигляді окремої діаграми послідовності генерації різних типів атак, що дозволить точніше описати яким чином формуються атаки та які саме атаки розглядаються (рис. 2.6):

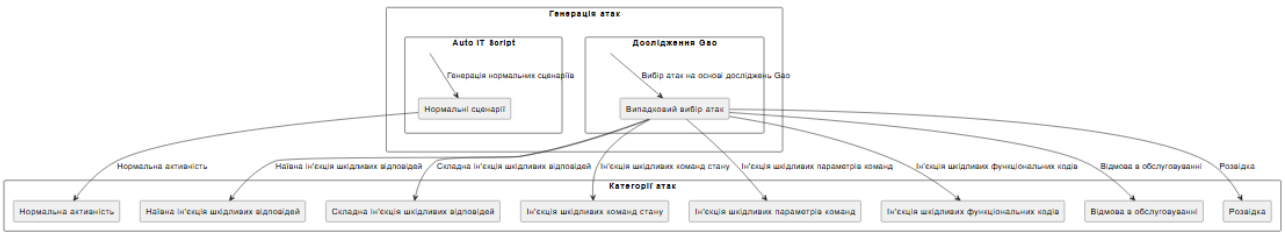

Рисунок 2.6. Діаграма послідовності генерації різних типів атак

2.3 Опис та специфікація даних для навчання

Спершу розглянемо структуру масиву даних, що буде передаватись в моделі для подальшого навчання та класифікації кібератак. Датасет містить такі поля:

1. «Modbus Frame»: містить повний кадр Modbus, який представляє собою послідовність байтів, що передаються через мережу. Цей кадр включає в себе мережеву інформацію та корисне навантаження;
2. «categorization»: числове значення, що відповідає категорії атаки. Це поле використовується для класифікації атак на основі їх загальних характеристик;
3. «specific attack»: числове значення, що відповідає специфічному типу атаки. Це поле надає більш детальну інформацію про конкретний тип атаки в рамках певної категорії;
4. «source»: рядкове значення, що відображає джерело атаки або нормальної активності. Воно може містити інформацію про IP-адресу, ідентифікатор пристрою або інші відомості про джерело;
5. «destination»: рядкове значення, що відображає ціль атаки або нормальної активності. Воно може містити інформацію про IP-адресу, ідентифікатор пристрою або інші відомості про ціль;
6. «time stamp»: часова мітка, що вказує на момент часу, коли була зафіксована певна активність або атака.

Для розуміння структури масиву даних, доцільно представити всі зв’язки між сутностями у вигляді OLTP-схеми сховища даних для класифікації та навчання моделей, що описує процес запису та формування масиву (рис. 2.7):

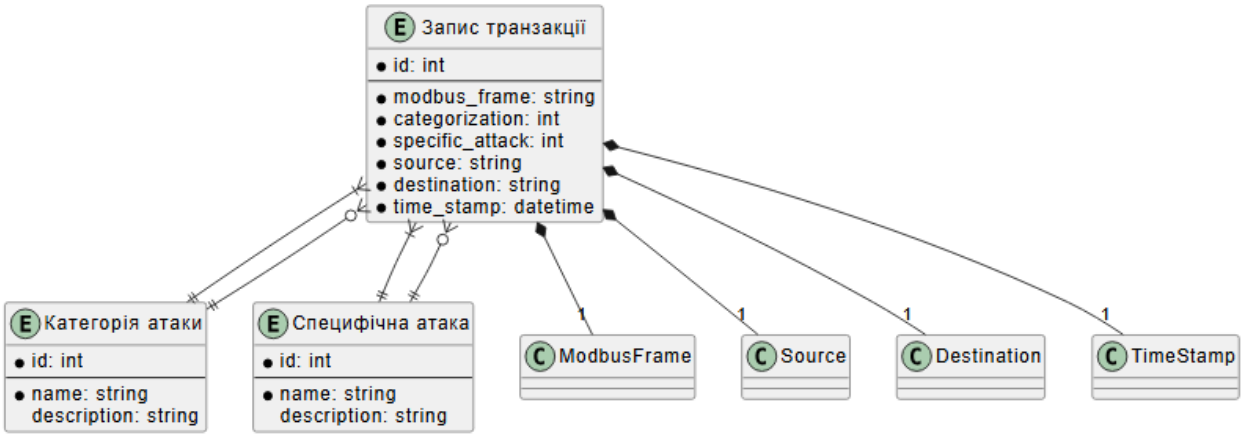

Рисунок 2.7. OLTP-схема сховища даних для класифікації та навчання моделей

Також для зручності обробки даних, всередині моделей, було прийняте рішення створити два словники category_mapping та specific_attack_mapping, що будують відображення між числовими мітками класів та їх рядковими значеннями, це корисно для інтерпретації результатів, продукованих моделями.

Розглянемо класи атак, що будуть аналізуватися моделями:

1. «Normal»: нормальна активність, відсутність атаки;
2. «Naïve Malicious Response Injection»: наївна ін'єкція шкідливих відповідей;
3. «Complex Malicious Response Injection»: складна ін'єкція шкідливих відповідей;
4. «Malicious State Command Injection»: ін'єкція шкідливих команд стану;
5. «Malicious Parameter Command Injection»: ін'єкція шкідливих параметрів команд;
6. «Malicious Function Code Injection»: ін'єкція шкідливих функціональних кодів;
7. «Denial of Service»: відмова в обслуговуванні;
8. «Reconnaissance»: розвідка.

Очевидно, що розподіл кількості міток класів нерівномірний, що створює певні складнощі для обробки даних моделями, оскільки детерміновані засоби оцінюють частоти входження кожної мітки у кінцевий масив даних, а отже це породжує труднощі з інтерпретацією результатів, оскільки моделі можуть

автоматично надавати перевагу тому класу, частота якого переважає над іншими. У процесі проведення розвідувального аналізу, було побудовано основні графіки розподілу даних в масиві, щоб, на основі цього, була змога коригувати архітектуру моделей, з метою мінімізації похибки від нерівномірності даних. Спершу відобразимо розподіл міток класів за категоріями атак у форматі гістограми розподілу кількості міток кожного класу з категорії атак (рис. 2.8):

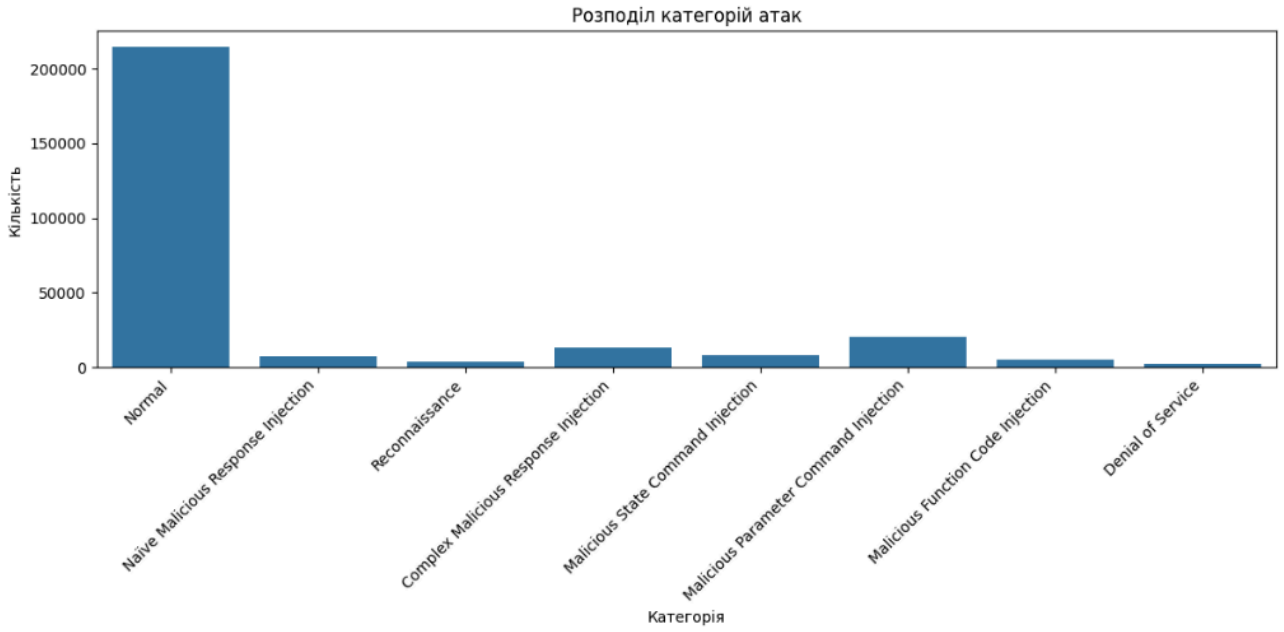

Рисунок 2.8. Гістограма розподілу кількості міток кожного класу з категорії атак

Як можна помітити, дійсно, спостережень, що сигналізують про атаки, вкрай мало, а отже класичні моделі, з високим ступенем ймовірності, дадуть некоректні результати класифікації, а отже, більш ймовірно, що використання стохастичних методів дозволить нівелювати таку розрідженість даних. Також було відображено розподіл кінцевих цілей атак у форматі гістограми розподілу кількості міток цілей атак (рис. 2.9):

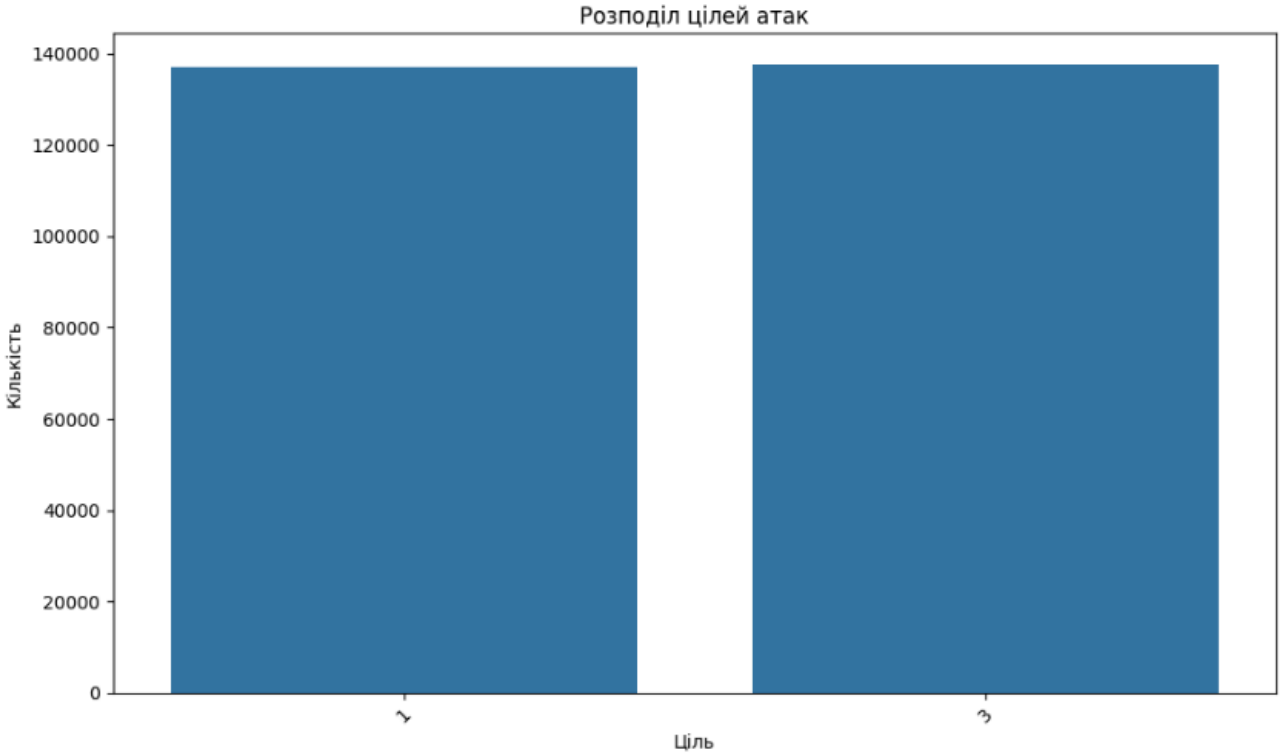

Рисунок 2.9. Гістограма розподілу кількості міток цілей атак

У даному випадку, рівномірний розподіл покращує ситуацію, оскільки таким чином, не потрібно проводити додаткову обробку та налаштування параметрів даного поля, а як можна помітити, кількість спостережень в масиві близька до 280000, що дозволить нівелювати вплив розрідженості, за рахунок побудови складних архітектур в моделях.

2.4 Обґрунтування підходу до тестування моделей

Тестування моделей було розбито на кілька основних етапів, які можна представити таким чином:

1. Кожна з трьох моделей, що аналізуються, буде програмно реалізована та навчена на підготовлених, оброблених даних, а також оцінена на тестовій вибірці даних;
2. Для кожної моделі буде оцінена точність її роботи та класифікації на основі відомих методів;

3. Для кожної моделі буде побудовано F1-крива, що дозволяє оцінити баланс між точністю (precision) та повнотою (recall) моделі при різних порогах прийняття рішення [34];
4. Для кожної моделі буде побудовано ROC-крива (Receiver Operating Characteristic), що відображає співвідношення між часткою правильно класифікованих позитивних прикладів (істинно позитивними) та часткою неправильно класифікованих негативних прикладів (хибно позитивними) при різних порогах прийняття рішення [35];
5. Для кожної моделі буде побудовано криву точності-повноти показує залежність між точністю та повнотою моделі при різних порогах прийняття рішення [36];
6. На заключному етапі тестування, для кожної моделі, буде сформовано classification report, що надає зведену інформацію про основні метрики класифікації, такі як точність, повнота, F1-міра та підтримка (support) для кожного класу [37].

За результатами аналізу отриманих результатів класифікації, буде обрана найкраща та найбільш оптимальна модель для вирішення поставленої задачі дослідження і буде реалізовано веб-додаток, який буде використовувати збережену модель для можливості клієнтської класифікації атак на основі вхідних даних, в реальному часі.

Висновок до розділу 2

Для класифікації кіберзагроз у системі аварійного електропостачання АЕС було обрано підхід дослідження та порівняння трьох архітектур штучного інтелекту: глибоких нейронних мереж, рекурентних нейронних мереж (LSTM) та ансамблевої моделі (випадковий ліс). Кожна з них має свої переваги у роботі з великими обсягами даних, складними залежностями та аномаліями. Глибокі мережі добре працюють з розрідженими даними, LSTM враховує часові

залежності, а ансамблеві моделі дозволяють уникнути перенавчання. Для забезпечення якісного навчання моделей, дані були зібрані з урахуванням специфіки протоколу Modbus, що дозволяє точно моделювати реальні кіберзагрози в індустріальних мережах.

РОЗДІЛ 3
АНАЛІЗ ЕКСПЕРИМЕНТАЛЬНИХ РЕЗУЛЬТАТІВ

3.1 Проведення експериментів з виявлення атак

На першому кроці було реалізовано та протестовано модель глибокого навчання для класифікації кібератак. Модель глибокого навчання побудована таким чиногм:

1. Створення послідовної моделі з використанням бібліотеки Keras;
2. Додавання вхідного шару з 128 нейронами та функцією активації ReLU;
3. Додавання шару Dropout з коефіцієнтом 0.5 для регуляризації;
4. Додавання прихованого шару з 64 нейронами та функцією активації ReLU;
5. Додавання ще одного шару Dropout з коефіцієнтом 0.5;
6. Додавання вихідного шару з кількістю нейронів, що дорівнює кількості класів, та функцією активації Softmax.

Програмна реалізація моделі глибокого навчання має такий вигляд (рис. 3.1):

```
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))

learning_rates = [ 0.01, 0.05, 0.1]
epochs_list = [10, 20, 30, 40]
best_model = None
best_loss = float('inf')

print("Навчання моделі з різними значеннями швидкості навчання та епох...")
loss_values = []
for lr in learning_rates:
    for epochs in epochs_list:
        print(f"Навчання моделі зі швидкістю навчання {lr} та {epochs} епохами...")
        model.compile(optimizer=Adam(learning_rate=lr), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
        history = model.fit(X_train, y_train, validation_split=0.2, epochs=epochs, batch_size=32, verbose=0)

        val_loss = history.history['val_loss'][-1]
        loss_values.append(val_loss)

        if val_loss < best_loss:
            best_loss = val_loss
            best_model = model
```

Рисунок 3.1. Програмна реалізація моделі глибокого навчання

Однак перед передачею даних в модель, було проведено попередню обробку даних, що включала такі етапи:

1. Видалення пропущених рядків з датасету за допомогою методу dropna();
2. Заповнення NaN значень нулями за допомогою методу fillna();
3. Кодування категоріальних змінних з використанням LabelEncoder з бібліотеки Scikit-learn.

У загальному, блок-схему роботи моделі глибокого навчання доцільно представити так (рис. 3.2):

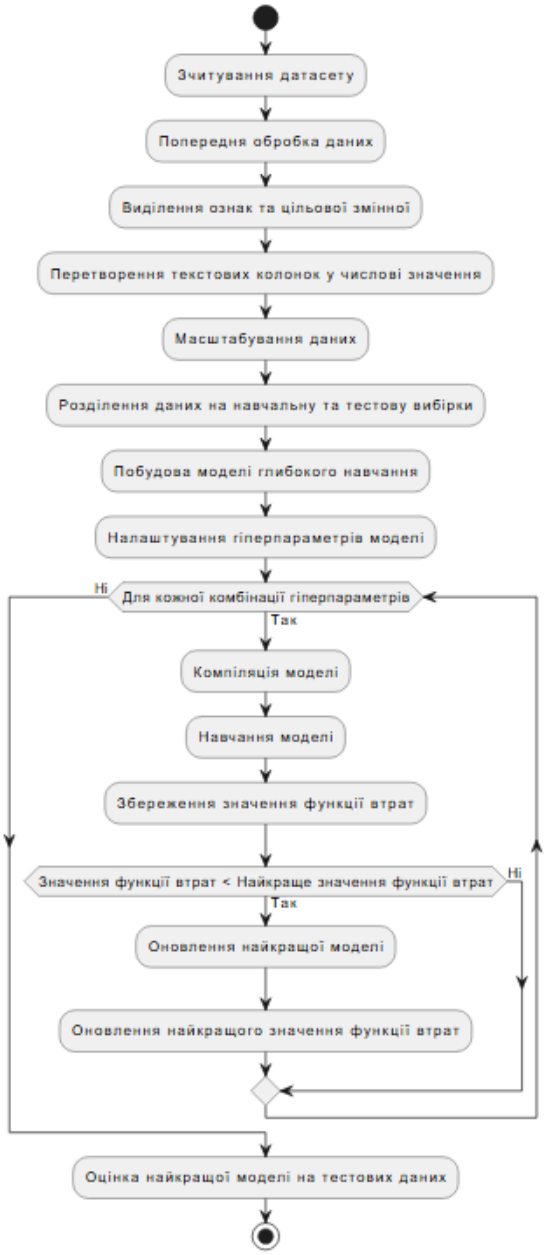

Рисунок 3.2. Блок-схема роботи моделі глибокого навчання

За результатами проведених експериментів, було отримано такі динаміки зміни точності моделі та критерію якості моделі, функції витрат (рис. 3.3):

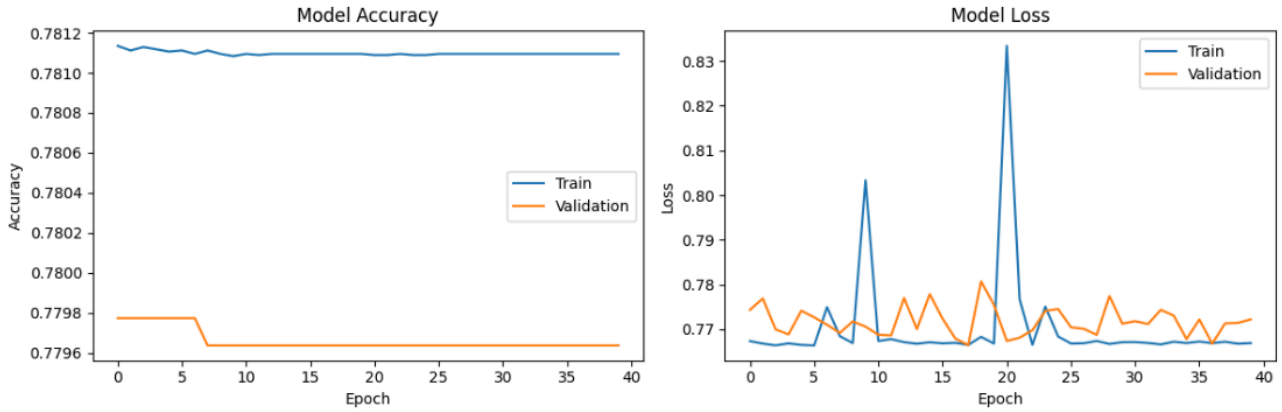

Рисунок 3.3. Динаміка зміни точності та критерію якості моделі

Також було побудовано таку матрицю плутанини класифікації моделі глибокого навчання, щоб оцінити кількість правильних та хибних класифікацій (рис. 3.4):

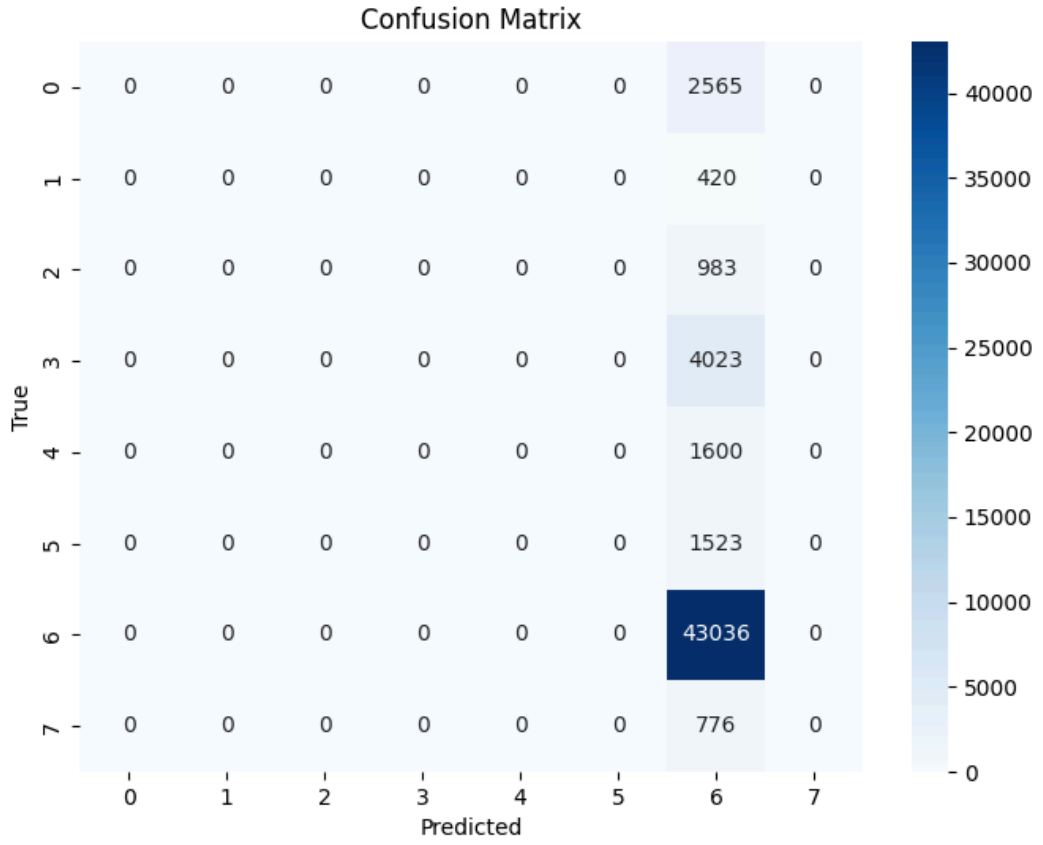

Рисунок 3.4. Матриця плутанини класифікації моделі глибокого навчання

На наступному кроці, було реалізовано модель LSTM, що представляє собою рекурентну нейронну мережу. Варто зазначити, що процес попередньої оброки вхідних даних аналогічний до моделі глибокого навчання. Програмна реалізація моделі LSTM має такий вигляд (рис. 3.5):

```
model = Sequential()
model.add(Embedding(input_dim=X_train.shape[1], output_dim=128, input_length=X_train.shape[1]))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(64, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(32))
model.add(Dropout(0.5))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))
model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

print("Навчання моделі...")
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_checkpoint = ModelCheckpoint('best_model_lstm.keras', monitor='val_accuracy', save_best_only=True)
history = model.fit(X_train, y_train, validation_split=0.2, epochs=10, batch_size=128,
                    callbacks=[early_stopping, model_checkpoint])
```

Рисунок 3.5. Програмна реалізація моделі LSTM

Модель містить такі складові:

1. Створення послідовної моделі з використанням бібліотеки Keras;
2. Додавання шару Embedding для перетворення числових ознак у щільні вектори;
3. Додавання трьох шарів LSTM з різною кількістю одиниць (128, 64, 32) та параметром return_sequences=True для перших двох шарів;
4. Додавання шарів Dropout з коефіцієнтом 0.5 після кожного шару LSTM для регуляризації;
5. Додавання вихідного шару з кількістю нейронів, що дорівнює кількості класів, та функцією активації Softmax.

У загальному, блок-схему роботи роботи моделі LSTM можна представити таким чином (рис. 3.6):

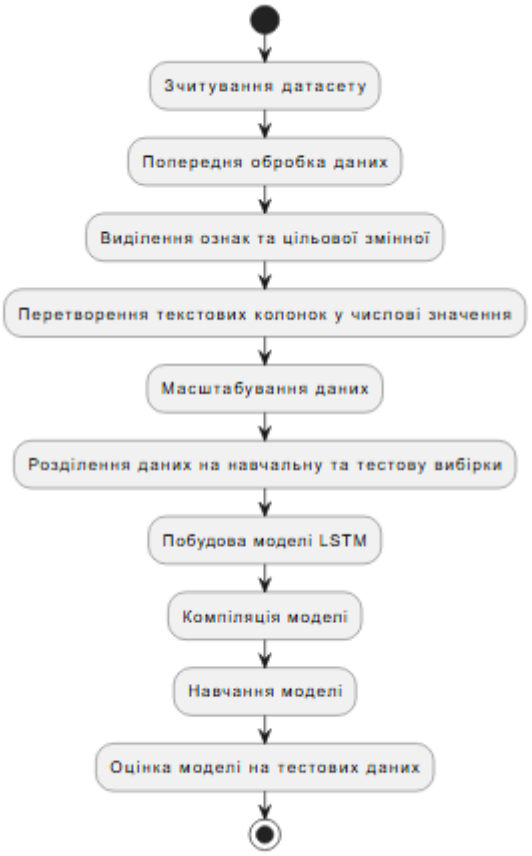

Рисунок 3.6. Блок-схема роботи моделі LSTM

У результаті були отримані такі показники динаміки точності зміни точності та критерію якості моделі (рис. 3.7):

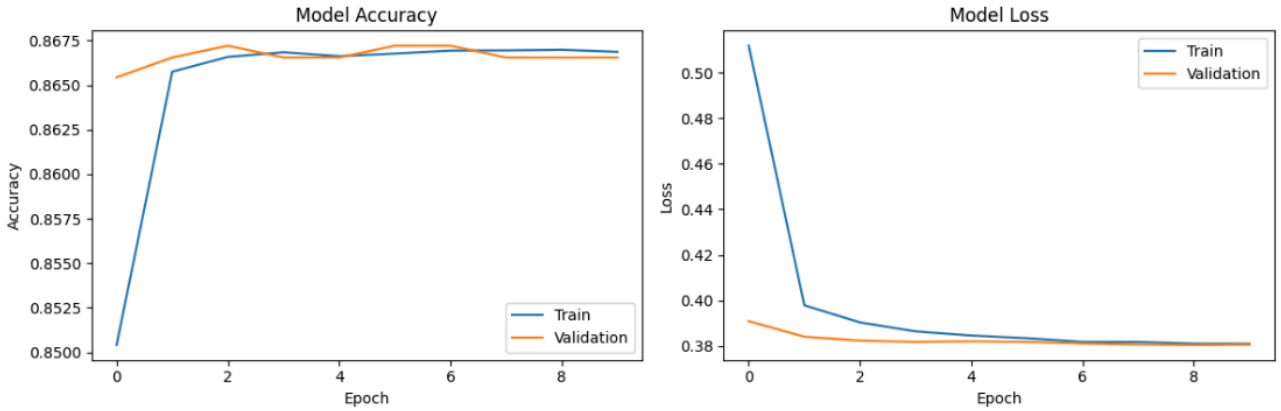

Рисунок 3.7. Динаміка зміни точності та критерію якості моделі

А також було отримано таку матрицю плутанини класифікації моделі LSTM, за результатами класифікації атак (рис. 3.8):

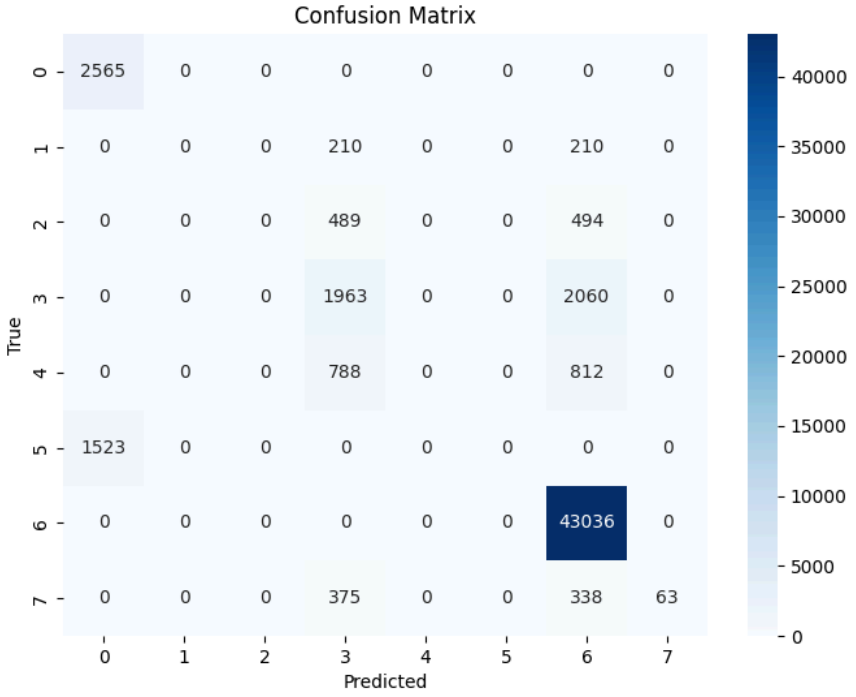

Рисунок 3.8. Матриця плутанини класифікації моделі LSTM

На заключному етапі була реалізована модель ансамблю випадкового лісу, що має таку програмну реалізацію (рис. 3.9):

```
param_grid = {
    'n_estimators': [200],
    'max_depth': [10],
    'min_samples_split': [2, 10],
    'min_samples_leaf': [2, 4]
}
rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
best_rf = grid_search.best_estimator_

print("Навчання моделі...")
best_rf.fit(X_train, y_train)
with open('random_forest_model.keras', 'wb') as file:
    pickle.dump(best_rf, file)

with open('encoders.pkl', 'wb') as file:
    pickle.dump(encoders, file)

with open('scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)
```

Рисунок 3.9. Програмна реалізація моделі випадкового лісу

Як вже зазначалось дана модель, по своїй суті, не проходить активну фазу навчання, але логіка її роботи дещо складніша за дві попередні моделі і її доцільно описати у вигляді блок-схеми роботи моделі випадкового лісу (рис. 3.10):

Рисунок 3.10. Блок-схема роботи моделі випадкового лісу

У результаті роботи моделі було отримано таку матрицю плутанини класифікації моделі випадкового лісу (рис. 3.11):

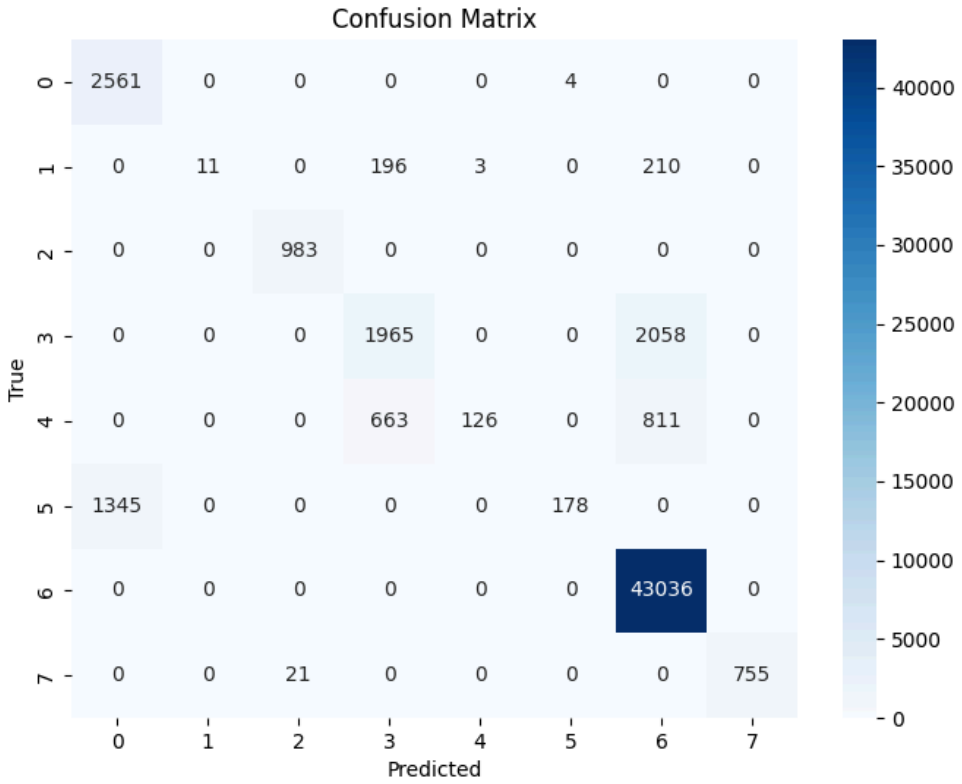

Рисунок 3.11. Матриця плутанини класифікації моделі випадкового лісу

Як можна помітити, на основі результатів навчання моделей, попередньо, найкращою виявляється ансамблева модель випадкового лісу, оскільки вона здатна класифікувати майже всі класи атак, що представлені у вхідному масиві даних, найменш спроможною виявилась модель глибокого навчання, що, скоріше за все, спричинене нерівномірним розподілом ознак в масиві.

3.2 Аналіз та порівняння результатів роботи алгоритмів

За результатами проведених експериментів, спершу, відобразимо характеристичні криві міри повноти та відгуку, ROC-AUC для моделі глибокого навчання (рис. 3.12 – 3.14):

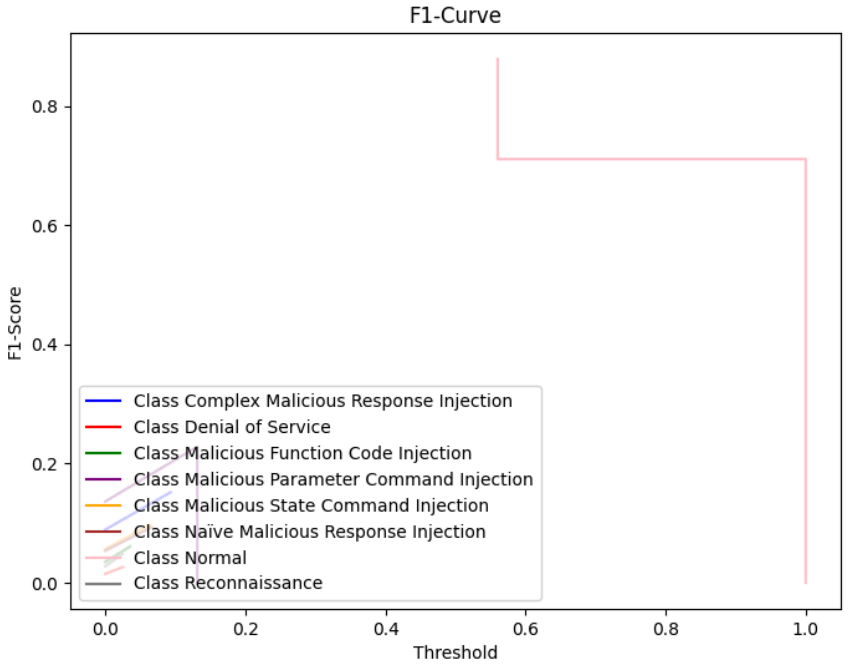

Рисунок 3.12. Крива міри повноти для моделі глибокого навчання

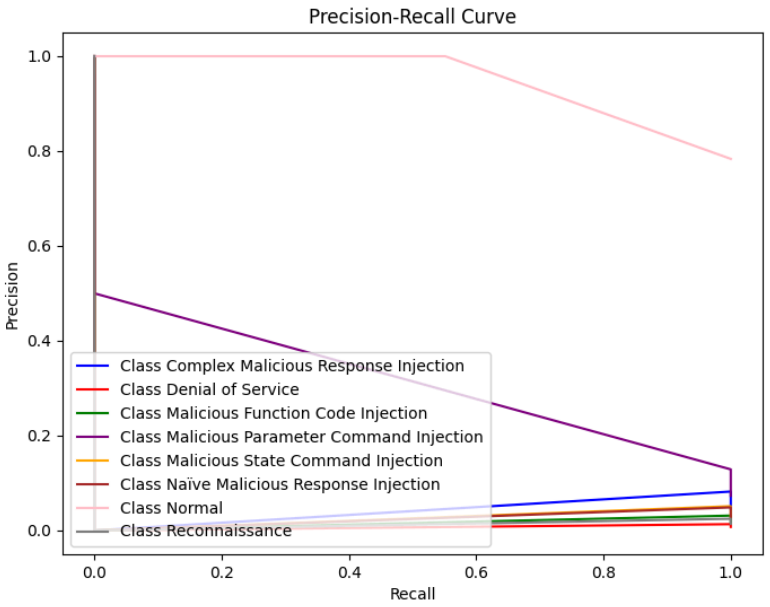

Рисунок 3.13. Крива відгуку та повноти для моделі глибокого навчання

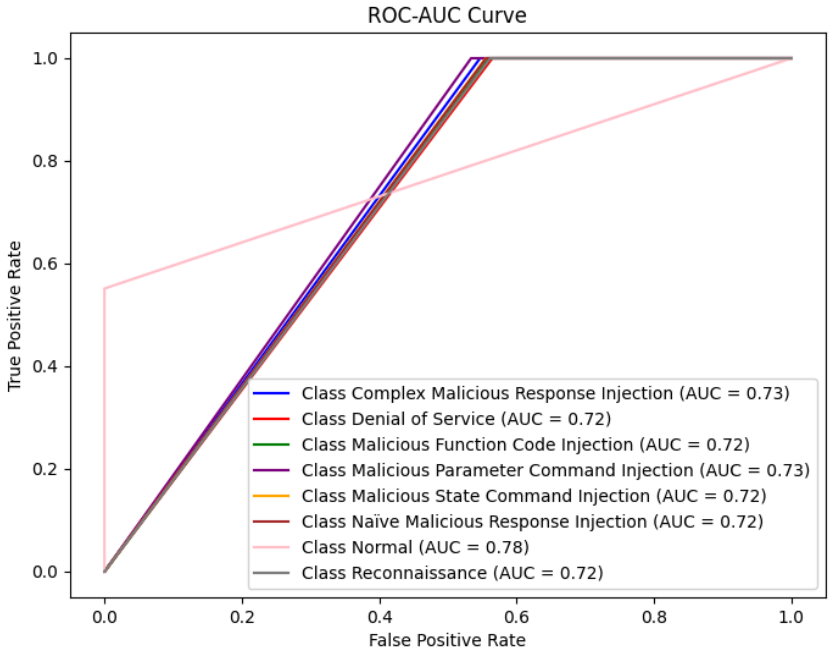

Рисунок 3.14. Крива ROC-AUC для моделі глибокого навчання

Відразу видно, що дана модель практично не справляється з класифікацією даних, відбувається перенавчання та класифікація більшості спостережень під один найбільший клас, що вказує на неспроможність обраної архітектури розв’язувати поставлену задачу.

Далі, аналогічним чином, відобразимо криві для моделі LSTM (рис. 3.15 – 3.17):

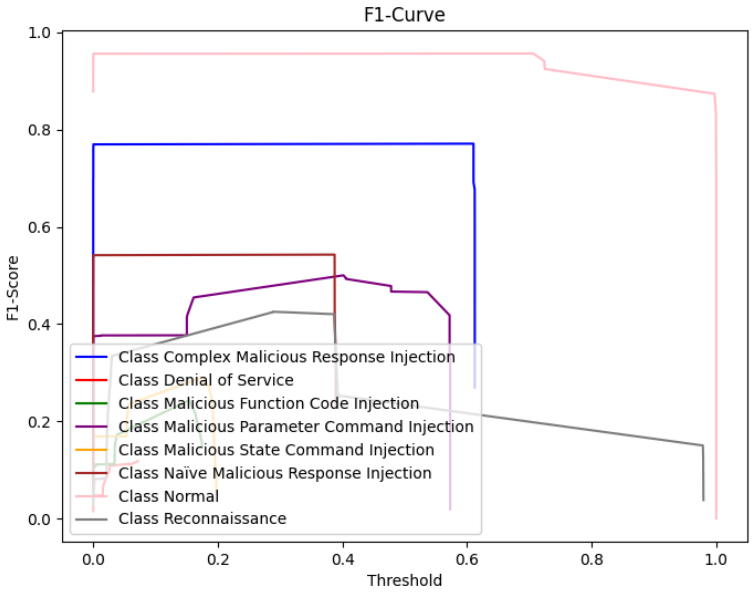

Рисунок 3.15. Крива міри повноти для моделі LSTM

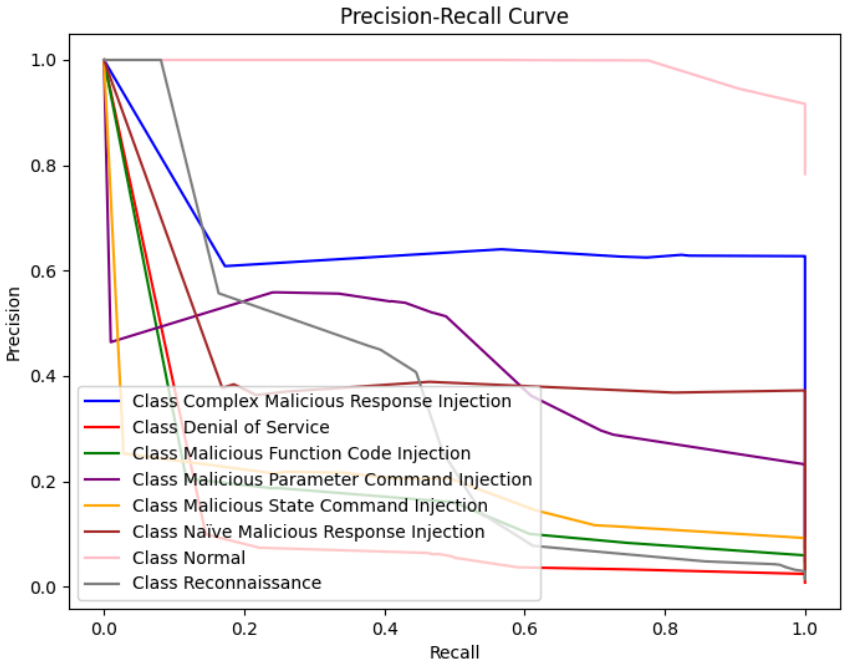

Рисунок 3.16. Крива відгуку та повноти для моделі LSTM

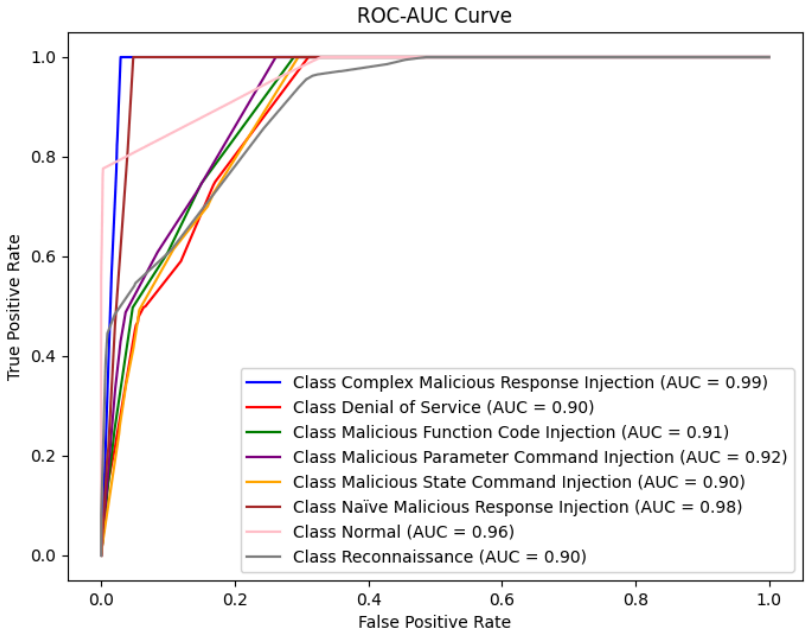

Рисунок 3.17. Крива ROC-AUC для моделі LSTM

Дана модель набагато краще справляється із задачею класифікації, як можна помітити баланс між повнотою та точністю вже набагато гладкіший, однак вона все ще робить велику кількість хибних класифікаційних висновків,

через відсутність балансу у кількості міток кожного класу, і деякі класи не здатна класифікувати.

На заключному етапі, аналогічно відобразимо характеристики моделі випадкового лісу (рис. 3.18 – 3.20):

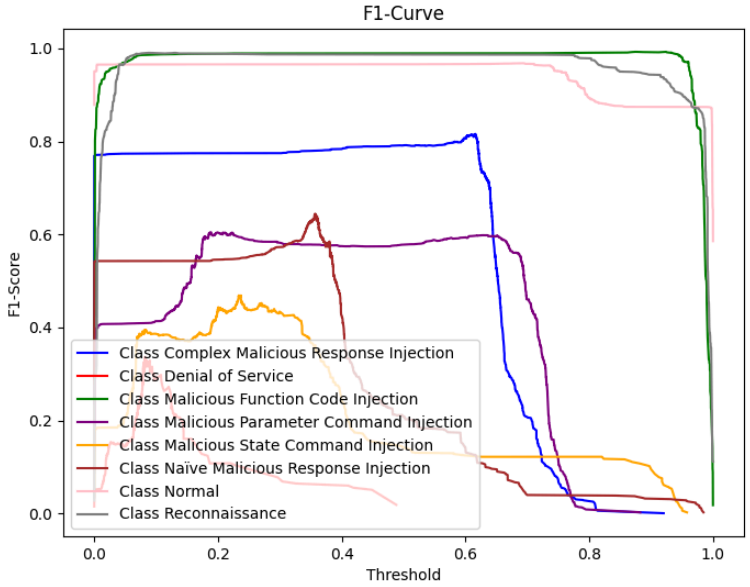

Рисунок 3.18. Крива міри повноти для моделі випадкового лісу

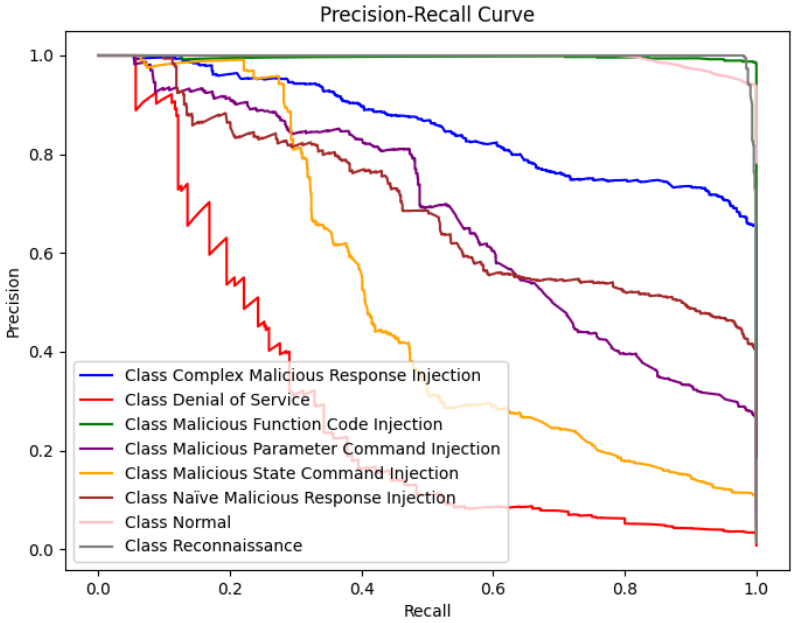

Рисунок 3.19. Крива відгуку та повноти для моделі випадкового лісу

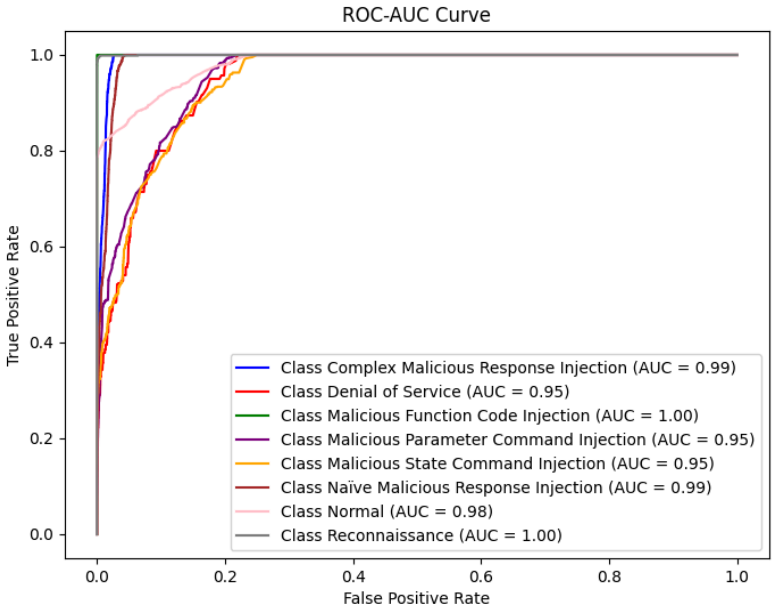

Рисунок 3.20. Крива ROC-AUC для моделі випадкового лісу

Модель випадкового лісу найкраще справляється із поставленою задачею, забезпечуючи кращий баланс між повнотою та відгуком моделі, також здатна класифікувати всі категорії атак із точністю не менше 95%, що вказує на доцільність використання стохастичних методів для розв'язання даної задачі.

Тепер для більш деталізовано висновку проаналізуємо отримані класифікаційні звіти кожної моделі. Були отримані такі звіти:

Deep model

Accuracy: 0.7232965845121235

Звіт класифікаційний представлений у таблиці 3.1.

Таблиця 3.1

Результати класифікаційї для моделі Deep Model

	Precision	Recall	F1-Score	Support
Complex Malicious Response Injection	0.53	0.76	0.20	2565
Denial of Service	0.00	0.00	0.00	420

Продовження таблиці 3.1

	1	2	3	4
Malicious Function Code Injection	0.00	0.00	0.00	983
Malicious Parameter Command Injection	0.21	0.39	0.71	4023
Malicious State Command Injection	0.00	0.00	0.00	1600
Naïve Malicious Response Injection	0.00	0.00	0.00	1523
Normal	0.92	1.00	0.96	43036
Reconnaissance	1.00	0.08	0.15	776
Accuracy			0.87	54926
Macro Avg	0.12	0.23	0.35	54926
Weighted Avg	0.54	0.54	0.71	54926

LSTM

Accuracy: 0.8643212569854875

Звіт класифікаційний представлений у таблиці 3.2

Таблиця 3.2

Результати класифікаційї для моделі LSTM

	Precision	Recall	F1-Score	Support
Complex Malicious Response Injection	0.63	1.00	0.77	2565
Denial of Service	0.00	0.00	0.00	420

Продовження таблиці 3.2

	1	2	3	4

Malicious Function Code Injection	0.00	0.00	0.00	983
Malicious Parameter Command Injection	0.51	0.49	0.50	4023
Malicious State Command Injection	0.00	0.00	0.00	1600
Naïve Malicious Response Injection	0.00	0.00	0.00	1523
Normal	0.92	1.00	0.96	43036
Reconnaissance	1.00	0.08	0.15	776
Accuracy			0.87	54926
Macro Avg	0.38	0.32	0.30	54926
Weighted Avg	0.80	0.87	0.82	54926

Random Forest

Accuracy: 0.9033062666132615

Звіт класифікаційний представлений у таблиці 3.3

Таблиця 3.3

Результати класифікаційї для моделі Random Forest

	Precision	Recall	F1-Score	Support
Complex Malicious Response Injection	0.66	1.00	0.79	2565
Denial of Service	1.00	0.03	0.05	420
Malicious Function Code Injection	0.98	1.00	0.99	983

Продовження таблиці 3.3

	1	2	3	4
Malicious Parameter	0.70	0.49	0.57	4023

Command Injection				
Malicious State Command Injection	0.98	0.08	0.15	1600
Naïve Malicious Response Injection	0.98	0.12	0.21	1523
Normal	0.93	1.00	0.97	43036
Reconnaissance	1.00	0.97	0.99	776
Accuracy			0.90	54926
Macro Avg	0.90	0.59	0.59	54926
Weighted Avg	0.91	0.90	0.88	54926

Випадковий ліс показує найвищу загальну точність 0.9033, порівняно з глибокою моделлю (0.7233) та LSTM (0.8643). Це свідчить про те, що випадковий ліс краще узагальнює та правильно класифікує більшість прикладів. Випадковий ліс демонструє високі значення точності (precision), повноти (recall) та F1-міри для більшості класів атак. Зокрема, він досягає високих результатів для класів «Complex Malicious Response Injection», «Malicious Function Code Injection», «Normal" та "Reconnaissance». Для деяких класів атак, таких як «Denial of Service», «Malicious State Command Injection» та «Naïve Malicious Response Injection», випадковий ліс показує нижчі значення повноти. Це може бути пов'язано з недостатньою кількістю прикладів цих класів у навчальній вибірці.

Випадковий ліс найкраще підходить для задачі класифікації кібератак на систему аварійного електропостачання АЕС, бо завдяки ансамблю 200 моделей він зменшує перенавчання і вибудовує складні шаблони залежності між вхідними характеристиками.

Таким чином, було прийняте рішення реалізувати веб-додаток для класифікації кібератак саме з моделлю випадкового лісу. Архітектура була

обрана клієнт-серверного типу, представлено у форматі схеми архітектурного рішення для веб-додатку (рис. 3.21):

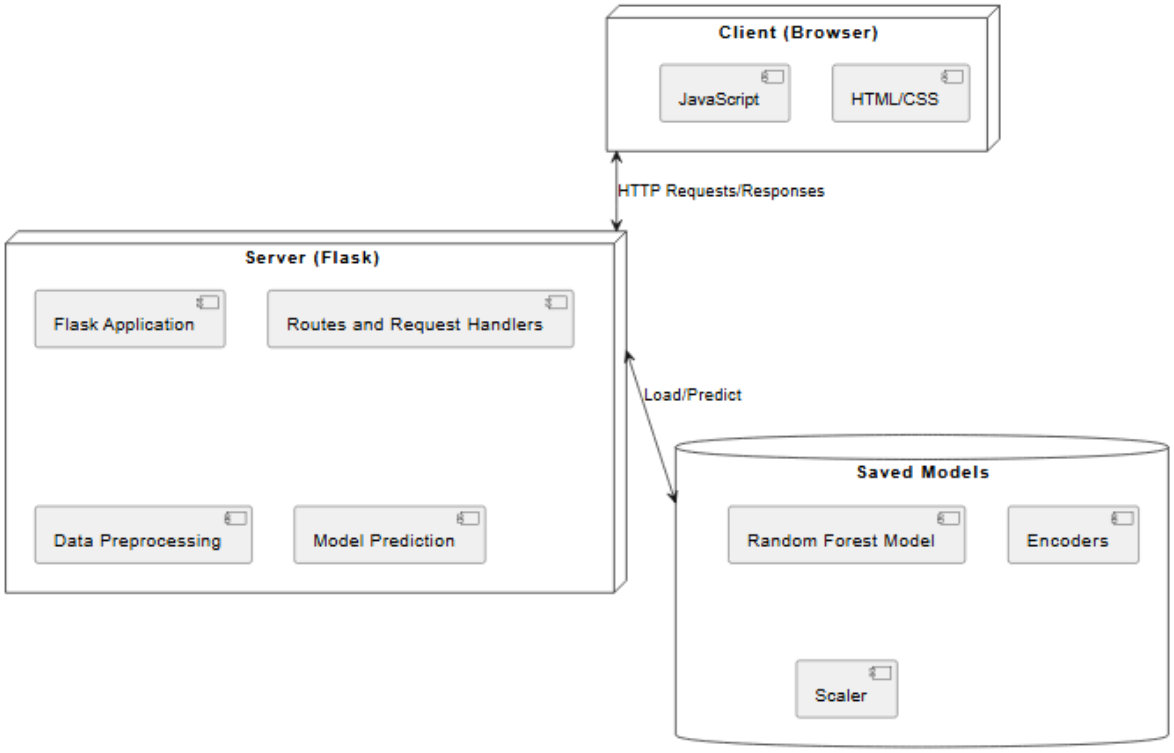

Рисунок 3.21. Схема архітектурного рішення для веб-додатку

При запуску серверу відбувається завантаження збереженої моделі випадкового лісу, енкодерів для кодування категоріальних змінних та скейлера для масштабування числових даних. Функція preprocess_input виконує попередню обробку вхідних даних, отриманих з форми. Вона створює DataFrame з вхідних даних, застосовує кодування категоріальних змінних з використанням збережених енкодерів та масштабує числові дані за допомогою збереженого скейлера. Попередньо оброблені дані передаються в модель випадкового лісу (model.predict) для класифікації атаки. Модель повертає передбачену категорію атаки та ймовірності приналежності до категорії.

Діаграму послідовності роботи з додатком клієнтської сторони додатка можна відобразити таким чином (рис. 3.22):

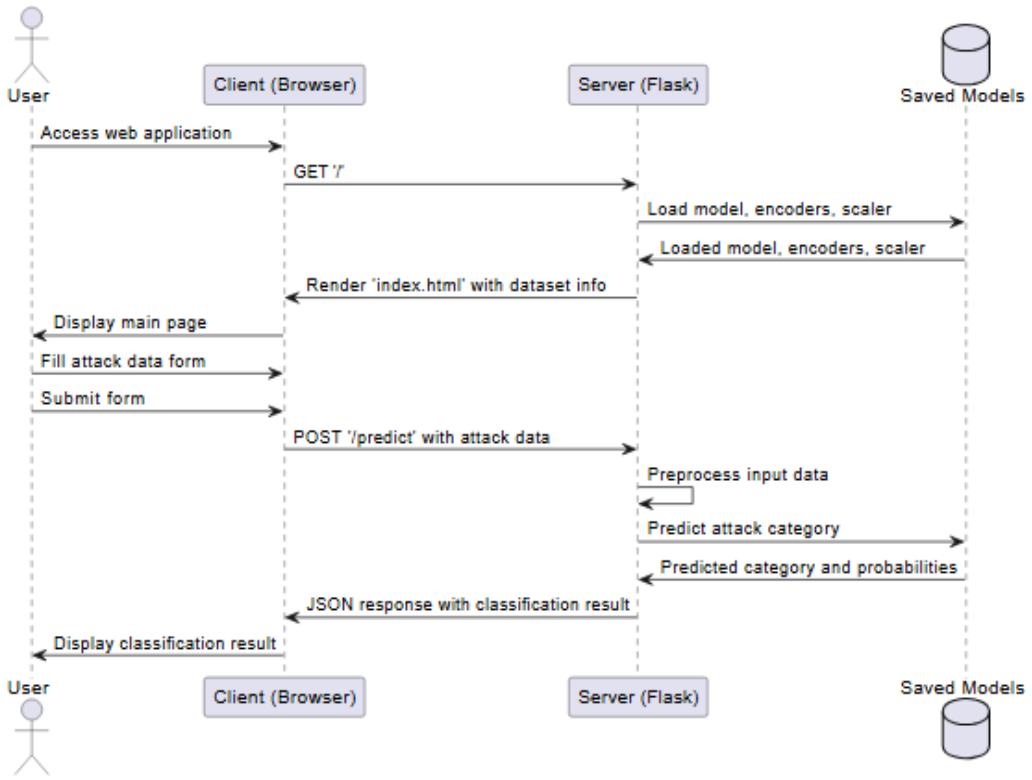

Рисунок 3.22. Діаграма послідовності роботи з додатком

Остаточно сформуємо блок-схему, що відображатиме логіку обробки даних роботи програми зі збереженою моделлю випадкового лісу (рис. 3.23):

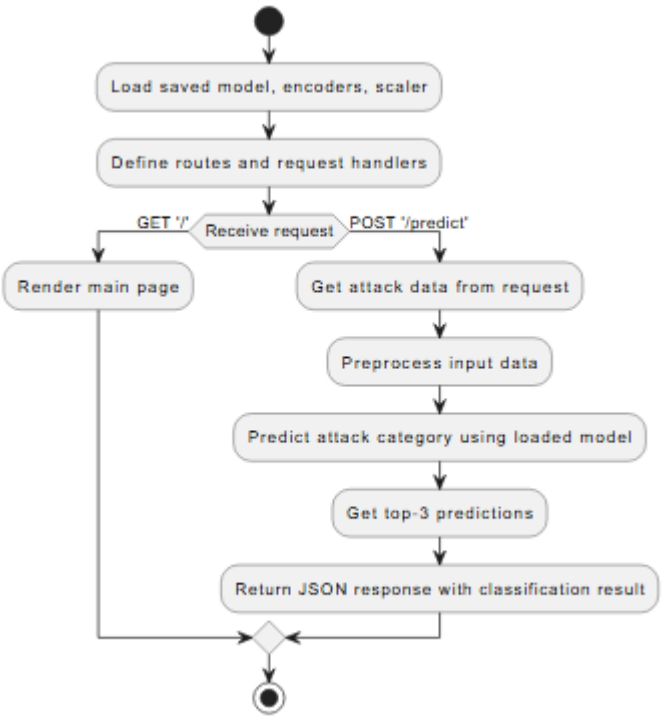

Рисунок 3.23. Блок-схема логіки обробки даних

Проведемо тестування розробленого додатку (рис. 3.24):

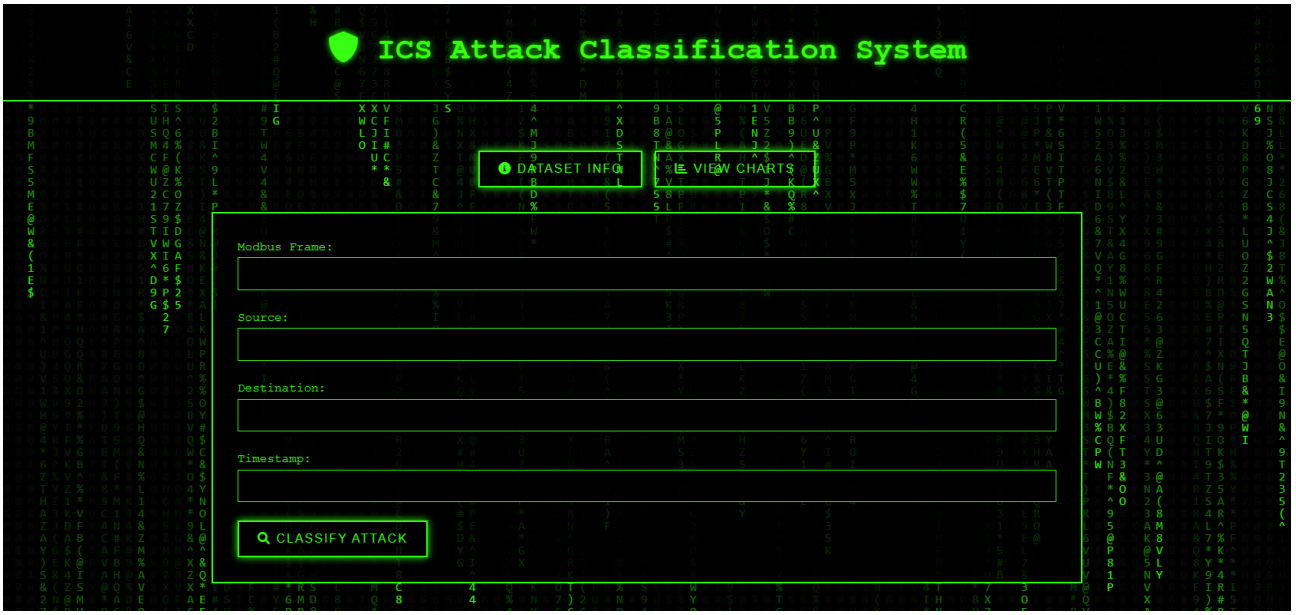

Рисунок 3.24. Результат запуску сервера

Тепер протестуємо роботу модальних вікон (рис. 3.25):

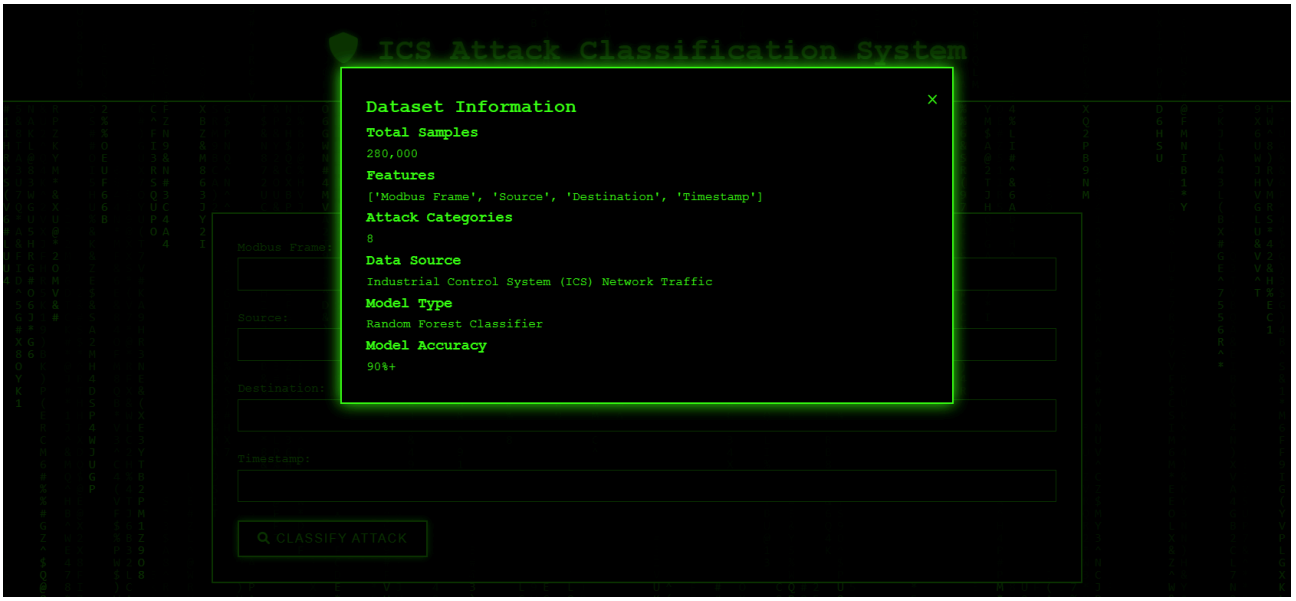

Рисунок 3.25. Виведення інформації про дані та модель

Далі протестуємо виведення розподілу міток класів для користувача в окреме модальне вікно (рис. 3.26):

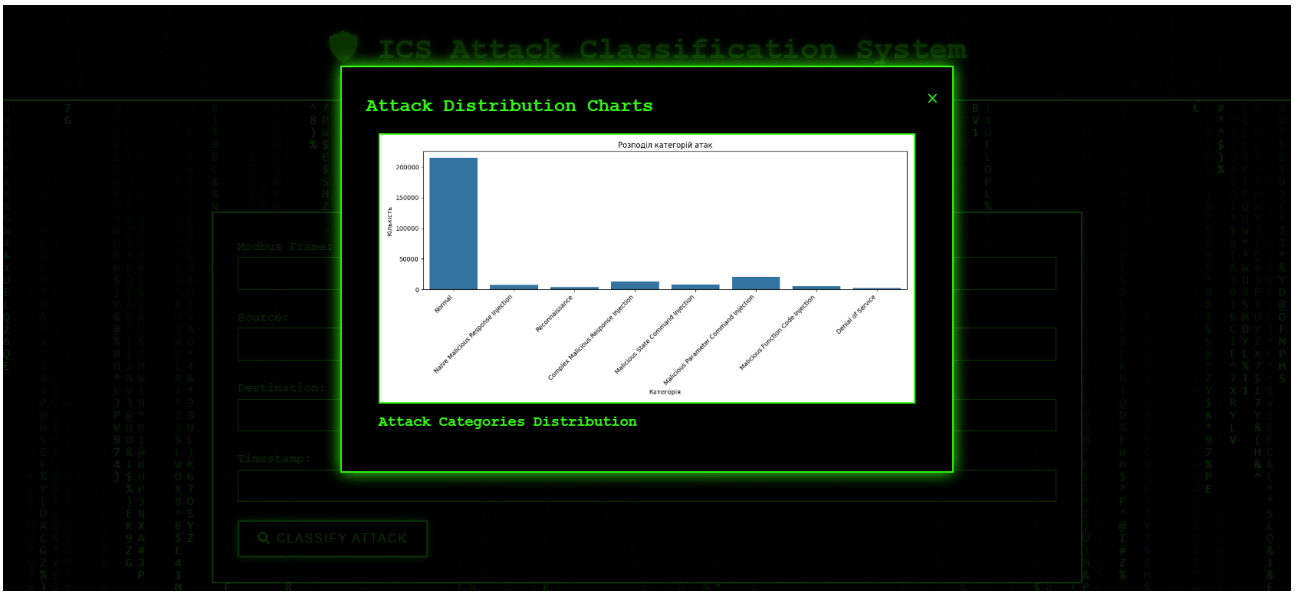

Рисунок 3.26. Виведення графіку розподілу міток класів

Тепер будемо подавати вхідні дані та тестувати роботу класифікатора (рис. 3.27):

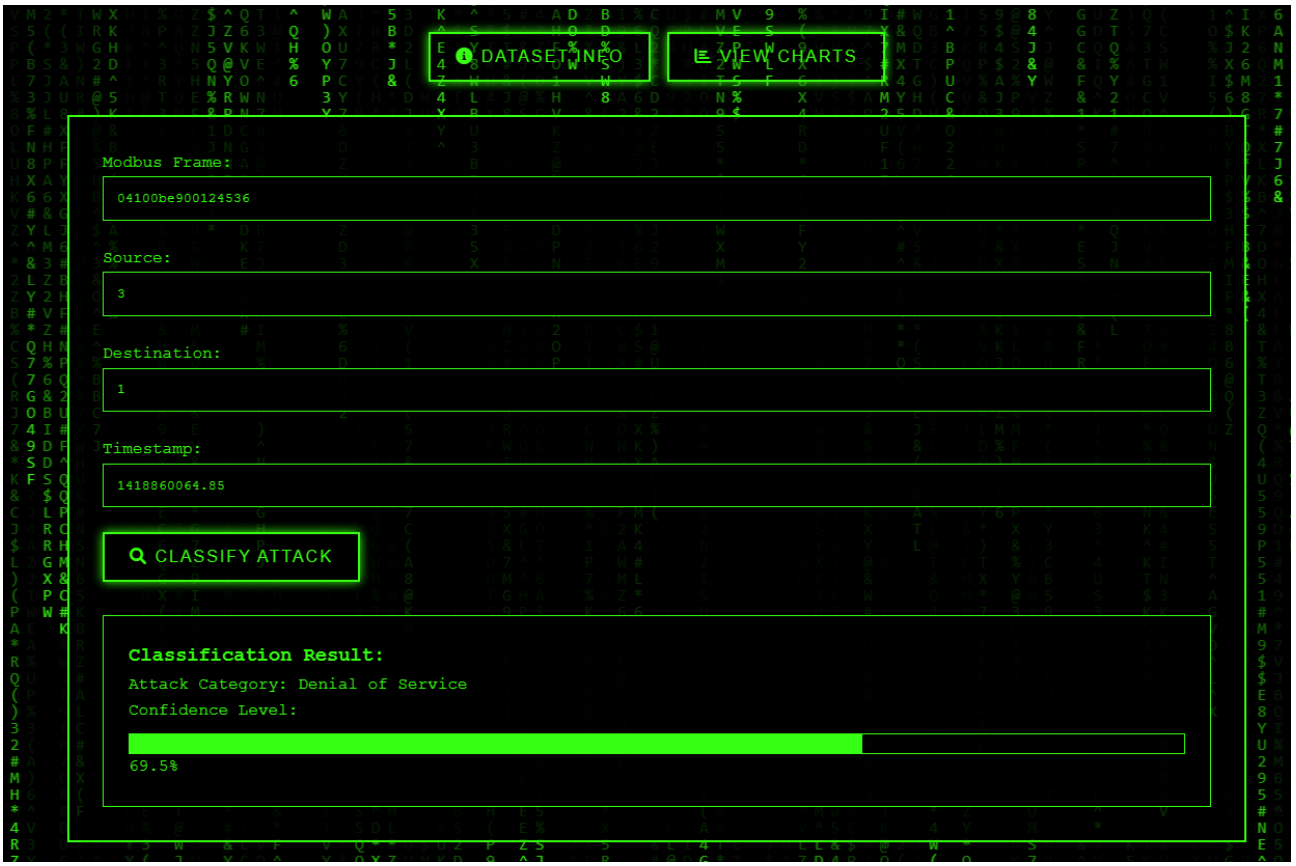

Рисунок 3.27. Результат класифікації атаки

Атака класифікована коректно і додатково, користувачу автоматично зберігається файл з трьома найбільш імовірними класами для цієї атаки у такому вигляді:

{"category": "Denial of Service", "confidence": 0.694743979882094, "top_predictions": [{"category": "Denial of Service", "probability": 0.694743979882094}, {"category": "Malicious State Command Injection", "probability": 0.22081654293850952}, {"category": "Malicious Parameter Command Injection", "probability": 0.061036115653789184}]}

Проведемо ще додатковий тест класифікатора (рис. 3.28):

Рисунок 3.28. Результат класифікації атаки

Також було отримано такі три найімовірніші типи атак:

{"category": "Malicious State Command Injection", "confidence": 69.28326891930871, "top_predictions": [{"category": "Malicious State Command Injection", "probability": 69.28326891930871}, {"category": "Malicious Parameter

Command Injection", "probability": 23.455372747088138}, {"category": "Na\u00efve Malicious Response Injection", "probability": 6.365566565168254}]}

Проведемо ще додаткове тестування (рис. 3.29):

Рисунок 3.29. Результат класифікації атаки

Також було отримано ймовірності, що це все ж аномальна поведінка і визначено такі потенційні атаки:

{"category": "Normal", "confidence": 38.13876428389957, "top_predictions": [{"category": "Normal", "probability": 38.13876428389957}, {"category": "Malicious Function Code Injection", "probability": 22.36123571610043}, {"category": "Malicious State Command Injection", "probability": 19.335526911328095}]}

Таким чином, було протестовано розроблений класифікатор кібератак на систему аварійного енергопостачання АЕС, що продемонстрував високий рівень точності, швидкості роботи та класифікації, а також спроможність

класифікувати логи і визначати ймовірність приналежності логів до різних типів атак.

3.3 Шляхи подальших досліджень

Одним з ключових факторів, що впливають на ефективність моделей машинного навчання, є якість та обсяг навчальних даних. Для подальшого покращення точності класифікації кібератак рекомендується розширити набір даних, включивши більше прикладів різних типів атак, особливо для класів з меншою представленістю.

Також рекомендується провести дослідження інших ансамблевих моделей, типу стекінг та попередньо натреновані моделі типу трансформерів. Такий підхід дозволить порівняти між собою ще більшу кількість алгоритмів та виявити найбільш оптимальні, за умови наявності достатньої кількості даних. Більш того, у подальших дослідженнях доцільно проводити оцінку важливості кожної ознаки в моделях, для різних конфігурацій останніх. Цей процес вимагає масштабних обчислювальних потужностей, однак дозволить оптимізувати роботу обраних моделей навіть для даних, що мають нерівномірний розподіл. До таких методів доцільно віднести SHAP та permutation importance, ці методи дозволять виділяти важливі ознаки динамічно протягом всього циклу навчання та оптимізації гіперпараметрів моделі.

Висновок до розділу 3

Було реалізовано та проаналізовано три моделі для класифікації кібератак: модель глибокого навчання, LSTM і модель випадкового лісу. Найгірші результати показала модель глибокого навчання через перенавчання та невдалу класифікацію більшості класів. Модель LSTM покращила показники, але також мала труднощі з рідкісними класами. Найвищу точність (90,3%) продемонструвала модель випадкового лісу, яка забезпечила найкращий баланс між точністю та повнотою і змогла коректно класифікувати більшість типів

атак. Це свідчить про доцільність використання ансамблевих стохастичних методів для задач виявлення кібератак.

ВИСНОВКИ

За результатами проведеного дослідження було досягнуто поставленої мети по розробці системи виявлення та класифікації кібератак на систему аварійного електропостачання АЕС з використанням алгоритмів машинного навчання.

У процесі проведення дослідження було проаналізовано сучасний стан кібербезпеки в системах критичної інфраструктури, зокрема в ядерній галузі, та визначено специфіку атак на такі системи. Було обґрунтовано необхідність розробки спеціалізованих методів захисту, які враховують особливості функціонування систем аварійного електропостачання АЕС. Для вирішення поставленої задачі було обрано та обґрунтовано використання алгоритмів машинного навчання, таких як випадковий ліс, глибокі нейронні мережі та LSTM. Розроблено підхід до збирання та генерації даних для навчання моделей, який включає в себе використання як реальних даних системи аварійного електропостачання АЕС, так і симуляцію різних типів кібератак. Проведено експерименти з навчання та тестування обраних моделей на зібраному наборі даних. Результати експериментів показали, що модель випадкового лісу демонструє найвищу ефективність у виявленні та класифікації кібератак, досягаючи точності 90.33% та збалансованості метрик для різних класів атак.

На основі найкращої моделі розроблено веб-додаток, який дозволяє в реальному часі аналізувати дані системи аварійного електропостачання АЕС та виявляти потенційні кібератаки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wilson, Clay. "Cyber threats to critical information infrastructure." Cyberterrorism: Understanding, Assessment, and Response. New York, NY: Springer New York, 2014. 123-136.
2. Haber, Eldar, and Tal Zarsky. "Cybersecurity for infrastructure: a critical analysis." Fla. St. UL Rev. 44 (2016): 515.
3. Brechbühl, Hans, et al. "Protecting critical information infrastructure: Developing cybersecurity policy." (2010): 83-91.
4. Karnouskos, Stamatis. "Stuxnet worm impact on industrial cyber-physical system security." IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2011.
5. Мохор, В. В., С. Ф. Гончар. "Оцінювання ризиків кібербезпеки інформаційних систем об’єктів критичної інфраструктури." Електронне моделювання 41.6 (2019): 65-76.
6. Мохор, В. В., С. Ф. Гончар, О. М. Дибач. "Методи оцінки сумарного ризику кібербезпеки об’єктів критичної інфраструктури." Ядерна та радіаційна безпека 2 (2019): 4-8.
7. Kovaliv, Myroslav, et al. "Legal support of cybersecurity of critical information infrastructure of Ukraine." Path of Science 7.4 (2021): 2011-2018.
8. Clark, Robert M., and Simon Hakim, eds. Cyber-physical security: protecting critical infrastructure at the state and local level. Vol. 3. Springer, 2016.
9. Rinaldi, Steven M., James P. Peerenboom, and Terrence K. Kelly. "Identifying, understanding, and analyzing critical infrastructure interdependencies." IEEE control systems magazine 21.6 (2001): 11-25.
10. Parfomak, Paul W. "Physical security of the US power grid: high-voltage transformer substations." 17 Jun. 2014,
11. Arinze, U. C., A. H. Eneh, and B. O. Longe. "Overview of Nuclear Cyber Security Requirements for Nuclear Power Plants (NPPs)." National Nuclear Security Regulations: 1-12.

12. Falliere, Nicolas, Liam O. Murchu, and Eric Chien. "W32. stuxnet dossier." White paper, symantec corp., security response 5.6 (2011): 29.
13. Langner, Ralph. "Stuxnet: Dissecting a cyberwarfare weapon." IEEE security & privacy 9.3 (2011): 49-51.
14. Barrett, Matthew P. "Framework for improving critical infrastructure cybersecurity version 1.1." (2018).
15. Siponen, Mikko, and Robert Willison. "Information security management standards: Problems and solutions." Information & management 46.5 (2009): 267-270.
16. Ericsson, Göran N. "Cyber security and power system communication—essential parts of a smart grid infrastructure." IEEE Transactions on Power delivery 25.3 (2010): 1501-1507.
17. Stouffer, Keith, Joe Falco, and Karen Scarfone. "Guide to industrial control systems (ICS) security." NIST special publication 800.82 (2011): 13-16.
18. Zhu, Bonnie, Anthony Joseph, and Shankar Sastry. "A taxonomy of cyber attacks on SCADA systems." 2011 International conference on internet of things and 4th international conference on cyber, physical and social computing. IEEE, 2011.
19. Kushner, David. "The real story of stuxnet." ieee Spectrum 50.3 (2013): 48-53.
20. Knapp, Eric D. Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems. Elsevier, 2024.
21. Mitnick, Kevin D., and William L. Simon. The art of deception: Controlling the human element of security. John Wiley & Sons, 2003.
22. Ladisa, Piergiorgio, et al. "Taxonomy of attacks on open-source software supply chains." arXiv preprint arXiv:2204.04008 (2022).
23. Shea, Thomas, Sandro Gaycken, and Maurizio Martellini. "Cyber security for nuclear power plants." Cyber Security: Deterrence and IT Protection for Critical Infrastructures. Cham: Springer International Publishing, 2013. 25-35.
24. Hurst, William, Madjid Merabti, and Paul Fergus. "A survey of critical infrastructure security." Critical Infrastructure Protection VIII: 8th IFIP WG 11.10

International Conference, ICCIP 2014, Arlington, VA, USA, March 17-19, 2014, Revised Selected Papers 8. Springer Berlin Heidelberg, 2014.

25. Nwoye, Chukwujekwu Charles. "Next-generation protection protocols and procedures for securing critical infrastructure." International Journal of Research Publication and Reviews 5.11 (2024): 4830-4845.
26. Ahmad, Atif, Justin Hadgkiss, and Anthonie B. Ruighaver. "Incident response teams–Challenges in supporting the organisational security function." Computers & Security 31.5 (2012): 643-652.
27. Siponen, Mikko, M. Adam Mahmood, and Seppo Pahnila. "Employees' adherence to information security policies: An exploratory field study." Information & management 51.2 (2014): 217-224.
28. Janiesch, Christian, Patrick Zschech, and Kai Heinrich. "Machine learning and deep learning." Electronic Markets 31.3 (2021): 685-695.
29. Mosavi, Amir, Sina Ardabili, and Annamaria R. Varkonyi-Koczy. "List of deep learning models." International conference on global research and education. Cham: Springer International Publishing, 2019.
30. Camargo, Manuel, Marlon Dumas, and Oscar González-Rojas. "Learning accurate LSTM models of business processes." Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17. Springer International Publishing, 2019.
31. Biau, Gérard, and Erwan Scornet. "A random forest guided tour." Test 25 (2016): 197-227.
32. Morris, Thomas, et al. "A control system testbed to validate critical infrastructure protection concepts." International Journal of Critical Infrastructure Protection 4.2 (2011): 88-103.
33. Morris, T., Thornton, Z., Turnipseed, I., Industrial Control System Simulation and Data Logging for Intrusion Detection System Research. 7th Annual Southeastern Cyber Security Summit. Huntsvile, AL. June 3 - 4, 2015.
34. DeVries, Zachary, et al. "Using a national surgical database to predict complications following posterior lumbar surgery and comparing the area under

the curve and F1-score for the assessment of prognostic capability." The spine journal 21.7 (2021): 1135-1142.

35. Bowers, Alex J., and Xiaoliang Zhou. "Receiver operating characteristic (ROC) area under the curve (AUC): A diagnostic measure for evaluating the accuracy of predictors of education outcomes." Journal of Education for Students Placed at Risk (JESPAR) 24.1 (2019): 20-46.
36. Boyd, Kendrick, Kevin H. Eng, and C. David Page. "Area under the precision-recall curve: point estimates and confidence intervals." Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13. Springer Berlin Heidelberg, 2013.
37. Bugalia, Nikhil, et al. "Machine learning-based automated classification of worker-reported safety reports in construction." Journal of Information Technology in Construction 27 (2022).

ДОДАТКИ

Додаток А

Лістинг програмного коду моделі глибокого навчання

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, precision_recall_curve, f1_score, roc_auc_score, roc_curve, auc, classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

print("Зчитування датасету...")
data = pd.read_csv("ModbusAttackDataset.csv")

print("Видалення пропущених рядків...")
data.dropna(inplace=True)

print("Заповнення NaN значень...")
data.fillna(0, inplace=True)

print("Кодування категоріальних змінних...")
label_encoder = LabelEncoder()
data["category_encoded"] = label_encoder.fit_transform(data["category_label"])
```

Продовження додатку А

```
print("Виділення ознак та цільової змінної...")
X = data[["Modbus Frame", "source", "destination", "time stamp"]]
y = data["category_encoded"]

print("Перетворення текстових колонок у числові значення...")
text_columns = ["Modbus Frame", "source", "destination"]
for column in text_columns:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column].astype(str))

print("Масштабування даних...")
scaler = StandardScaler()
X = scaler.fit_transform(X)

print("Розділення даних на навчальну та тестову вибірки...")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

print("Побудова моделі глибокого навчання...")
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))

learning_rates = [ 0.01, 0.05, 0.1]
epochs_list = [10, 20, 30, 40]
best_model = None
```

Продовження додатку А

```
best_loss = float('inf')

print("Навчання моделі з різними значеннями швидкості навчання та епох...")
loss_values = []
for lr in learning_rates:
    for epochs in epochs_list:
        print(f"Навчання моделі зі швидкістю навчання {lr} та {epochs} епохами...")
                                        model.compile(optimizer=Adam(learning_rate=lr),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
            history = model.fit(X_train, y_train, validation_split=0.2, epochs=epochs,
batch_size=32, verbose=0)

        val_loss = history.history['val_loss'][-1]
        loss_values.append(val_loss)

        if val_loss < best_loss:
            best_loss = val_loss
            best_model = model

print("Оцінка найкращої моделі на тестових даних...")
y_pred_proba = best_model.predict(X_test)
y_pred = np.argmax(y_pred_proba, axis=1)

print("Побудова графіків зміни точності та функції витрат для найкращої
моделі...")
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(history.history['accuracy'], label='Train')
ax1.plot(history.history['val_accuracy'], label='Validation')
ax1.set_title('Model Accuracy')
```

Продовження додатку А

```
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Accuracy')
ax1.legend()
ax2.plot(history.history['loss'], label='Train')
ax2.plot(history.history['val_loss'], label='Validation')
ax2.set_title('Model Loss')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Loss')
ax2.legend()
plt.tight_layout()
plt.savefig('accuracy_loss_curves_dl.png')

print("Побудова графіка Precision-Recall Curve для кожного класу...")
plt.figure(figsize=(8, 6))
colors = ['blue', 'red', 'green', 'purple', 'orange', 'brown', 'pink', 'gray']
for i, label in enumerate(label_encoder.classes_):
    precision, recall, _ = precision_recall_curve(y_test == i, y_pred_proba[:, i])
    plt.plot(recall, precision, color=colors[i], label=f'Class {label}')
plt.title('Precision-Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc='lower left')
plt.savefig('precision_recall_curve_dl.png')

print("Побудова графіка F1-Curve для кожного класу...")
plt.figure(figsize=(8, 6))
for i, label in enumerate(label_encoder.classes_):
    precision, recall, thresholds = precision_recall_curve(y_test == i, y_pred_proba[:,
i])
```

Продовження додатку А

```
    f1_scores = 2 * (precision * recall) / (precision + recall)
    plt.plot(thresholds, f1_scores[:-1], color=colors[i], label=f'Class {label}')
plt.title('F1-Curve')
plt.xlabel('Threshold')
plt.ylabel('F1-Score')
plt.legend(loc='lower left')
plt.savefig('f1_curve_dl.png')

print("Побудова графіка ROC-AUC Curve для кожного класу...")
plt.figure(figsize=(8, 6))
for i, label in enumerate(label_encoder.classes_):
    fpr, tpr, _ = roc_curve(y_test == i, y_pred_proba[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=colors[i], label=f'Class {label} (AUC = {roc_auc:.2f})')
plt.title('ROC-AUC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.savefig('roc_auc_curve_dl.png')

print("Побудова матриці помилок (Confusion Matrix)...")
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.savefig('confusion_matrix_dl.png')
```

Продовження додатку А

```
print("Побудова 3D графіка функції витрат...")
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
X, Y = np.meshgrid(learning_rates, epochs_list)
Z = np.array(loss_values).reshape(len(learning_rates), len(epochs_list))
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_xlabel('Learning Rate')
ax.set_ylabel('Epochs')
ax.set_zlabel('Loss')
ax.set_title('Loss Surface')
plt.savefig('loss_surface_dl.png')

print("Детальна статистика по метрикам:")
print("Accuracy:", best_model.evaluate(X_test, y_test)[1])
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred,
target_names=label_encoder.classes_))

print("Завершено.")
```

Додаток Б

Лістинг програмного коду моделі LSTM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, precision_recall_curve, f1_score, roc_auc_score, roc_curve, auc, classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, GRU, Dropout, LSTM
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Attention
from tensorflow.keras.layers import Conv1D, MaxPooling1D

print("Зчитування датасету...")
data = pd.read_csv("ModbusAttackDataset.csv")

print("Видалення пропущених рядків...")
data.dropna(inplace=True)
print("Заповнення NaN значень...")
data.fillna(0, inplace=True)

print("Кодування категоріальних змінних...")
label_encoder = LabelEncoder()
data["category_encoded"] = label_encoder.fit_transform(data["category_label"])
```

Продовження додатку Б

```
print("Виділення ознак та цільової змінної...")
X = data[["Modbus Frame", "source", "destination", "time stamp"]]
y = data["category_encoded"]

print("Перетворення текстових колонок у числові значення...")
text_columns = ["Modbus Frame", "source", "destination"]
for column in text_columns:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column].astype(str))

print("Масштабування даних...")
scaler = StandardScaler()
X = scaler.fit_transform(X)

print("Розділення даних на навчальну та тестову вибірки...")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = Sequential()
model.add(Embedding(input_dim=X_train.shape[1], output_dim=128,
input_length=X_train.shape[1]))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(64, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(32))
model.add(Dropout(0.5))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))
```

Продовження додатку Б

```
model.compile(optimizer=Adam(learning_rate=0.001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])

print("Навчання моделі...")
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
model_checkpoint = ModelCheckpoint('best_model_lstm.keras',
monitor='val_accuracy', save_best_only=True)
history = model.fit(X_train, y_train, validation_split=0.2, epochs=10,
batch_size=128,
                    callbacks=[early_stopping, model_checkpoint])

print("Оцінка моделі на тестових даних...")
y_pred_proba = model.predict(X_test)
y_pred = np.argmax(y_pred_proba, axis=1)

print("Побудова графіків зміни точності та функції витрат...")
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

ax1.plot(history.history['accuracy'], label='Train')
ax1.plot(history.history['val_accuracy'], label='Validation')
ax1.set_title('Model Accuracy')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Accuracy')
ax1.legend()

ax2.plot(history.history['loss'], label='Train')
ax2.plot(history.history['val_loss'], label='Validation')
ax2.set_title('Model Loss')
```

```
ax2.set_xlabel('Epoch')
```

Продовження додатку Б

```
ax2.set_ylabel('Loss')
ax2.legend()

plt.tight_layout()
plt.savefig('accuracy_loss_curves_gru.png')

print("Побудова графіка Precision-Recall Curve для кожного класу...")
plt.figure(figsize=(8, 6))
colors = ['blue', 'red', 'green', 'purple', 'orange', 'brown', 'pink', 'gray']
for i, label in enumerate(label_encoder.classes_):
    precision, recall, _ = precision_recall_curve(y_test == i, y_pred_proba[:, i])
    plt.plot(recall, precision, color=colors[i], label=f'Class {label}')
plt.title('Precision-Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc='lower left')
plt.savefig('precision_recall_curve_gru.png')

print("Побудова графіка F1-Curve для кожного класу...")
plt.figure(figsize=(8, 6))
for i, label in enumerate(label_encoder.classes_):
     precision, recall, thresholds = precision_recall_curve(y_test == i, y_pred_proba[:,
i])
    f1_scores = 2 * (precision * recall) / (precision + recall)
    plt.plot(thresholds, f1_scores[:-1], color=colors[i], label=f'Class {label}')
plt.title('F1-Curve')
plt.xlabel('Threshold')
plt.ylabel('F1-Score')
```

```
plt.legend(loc='lower left')
```

Продовження додатку Б

```
plt.savefig('f1_curve_gru.png')

print("Побудова графіка ROC-AUC Curve для кожного класу...")
plt.figure(figsize=(8, 6))
for i, label in enumerate(label_encoder.classes_):
    fpr, tpr, _ = roc_curve(y_test == i, y_pred_proba[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=colors[i], label=f'Class {label} (AUC = {roc_auc:.2f})')
plt.title('ROC-AUC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.savefig('roc_auc_curve_gru.png')

print("Побудова матриці помилок (Confusion Matrix)...")
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.savefig('confusion_matrix_gru.png')

print("Детальна статистика по метрикам:")
print("Accuracy:", model.evaluate(X_test, y_test)[1])
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred,
target_names=label_encoder.classes_))
```

Продовження додатку Б

```
print("Завершено.")
```

Додаток В

Лістинг програмного коду моделі випадкового лісу

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, precision_recall_curve, f1_score, roc_auc_score, roc_curve, auc, classification_report
from sklearn.ensemble import RandomForestClassifier
import pickle

print("Зчитування датасету...")
data = pd.read_csv("ModbusAttackDataset.csv")

print("Видалення пропущених рядків...")
data.dropna(inplace=True)

print("Заповнення NaN значень...")
data.fillna(0, inplace=True)

print("Кодування категоріальних змінних...")
label_encoder = LabelEncoder()
data["category_encoded"] = label_encoder.fit_transform(data["category_label"])

print("Виділення ознак та цільової змінної...")
X = data[["Modbus Frame", "source", "destination", "time stamp"]]
y = data["category_encoded"]
```

Продовження додатку В

```
print("Перетворення текстових колонок у числові значення...")
encoders = {}
for column in ["Modbus Frame", "source", "destination"]:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column].astype(str))
    encoders[column] = le

print("Масштабування даних...")
scaler = StandardScaler()
X = scaler.fit_transform(X)

print("Розділення даних на навчальну та тестову вибірки...")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Налаштування гіперпараметрів моделі випадкового лісу...")
param_grid = {
    'n_estimators': [200],
    'max_depth': [10],
    'min_samples_split': [2, 10],
    'min_samples_leaf': [2, 4]
}
rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
best_rf = grid_search.best_estimator_

print("Навчання моделі...")
```

```
best_rf.fit(X_train, y_train)
```

Продовження додатку В

```
with open('random_forest_model.keras', 'wb') as file:
    pickle.dump(best_rf, file)

with open('encoders.pkl', 'wb') as file:
    pickle.dump(encoders, file)

with open('scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)

print("Оцінка моделі на тестових даних...")
y_pred = best_rf.predict(X_test)
y_pred_proba = best_rf.predict_proba(X_test)

print("Побудова графіка Precision-Recall Curve для кожного класу...")
plt.figure(figsize=(8, 6))
colors = ['blue', 'red', 'green', 'purple', 'orange', 'brown', 'pink', 'gray']
for i, label in enumerate(label_encoder.classes_):
    precision, recall, _ = precision_recall_curve(y_test == i, y_pred_proba[:, i])
    plt.plot(recall, precision, color=colors[i], label=f'Class {label}')
plt.title('Precision-Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc='lower left')
plt.savefig('precision_recall_curve_rf.png')

print("Побудова графіка F1-Curve для кожного класу...")
plt.figure(figsize=(8, 6))
for i, label in enumerate(label_encoder.classes_):
```

Продовження додатку В

```
    precision, recall, thresholds = precision_recall_curve(y_test == i, y_pred_proba[:, i])
    f1_scores = 2 * (precision * recall) / (precision + recall)
    plt.plot(thresholds, f1_scores[:-1], color=colors[i], label=f'Class {label}')
plt.title('F1-Curve')
plt.xlabel('Threshold')
plt.ylabel('F1-Score')
plt.legend(loc='lower left')
plt.savefig('f1_curve_rf.png')

print("Побудова графіка ROC-AUC Curve для кожного класу...")
plt.figure(figsize=(8, 6))
for i, label in enumerate(label_encoder.classes_):
    fpr, tpr, _ = roc_curve(y_test == i, y_pred_proba[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=colors[i], label=f'Class {label} (AUC = {roc_auc:.2f})')
plt.title('ROC-AUC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.savefig('roc_auc_curve_rf.png')

print("Побудова матриці помилок (Confusion Matrix)...")
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
```

```
plt.ylabel('True')
```

Продовження додатку В

```
plt.savefig('confusion_matrix_rf.png')

print("Детальна статистика по метрикам:")
print("Accuracy:", best_rf.score(X_test, y_test))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred,
target_names=label_encoder.classes_))

print("Завершено.")
```

Додаток Д

Лістинг програмного коду серверної частини додатку

```
from flask import Flask, render_template, request, jsonify
import pickle
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
import pandas as pd
from datetime import datetime
import json

app = Flask(__name__)

with open('random_forest_model.keras', 'rb') as file:
    model = pickle.load(file)

with open('encoders.pkl', 'rb') as file:
    encoders = pickle.load(file)

with open('scaler.pkl', 'rb') as file:
    scaler = pickle.load(file)

category_mapping = {
    0: "Normal",
    1: "Naïve Malicious Response Injection",
    2: "Complex Malicious Response Injection",
    3: "Malicious State Command Injection",
    4: "Malicious Parameter Command Injection",
    5: "Malicious Function Code Injection",
    6: "Denial of Service",
    7: "Reconnaissance"
```

```
}

dataset_info = {
    "total_samples": "280,000",
    "features": ["Modbus Frame", "Source", "Destination", "Timestamp"],
    "attack_categories": len(category_mapping),
    "data_source": "Industrial Control System (ICS) Network Traffic",
    "model_type": "Random Forest Classifier",
    "model_accuracy": "90%+"
}

def preprocess_input(input_data):
    """
    Preprocess input data using the same transformations as during training
    """
    try:
        df = pd.DataFrame([input_data])

        df['time stamp'] = df['time stamp'].astype(float)

        encoded_data = df.copy()
        for column, encoder in encoders.items():
            if column in df.columns:
                try:
                    encoded_data[column] = encoder.transform(df[column].astype(str))
                except ValueError:
                    encoder.fit(list(encoder.classes_) + list(df[column].astype(str)))
                    encoded_data[column] = encoder.transform(df[column].astype(str))
        numerical_data = encoded_data.values.astype(float)
```

Продовження додатку Д

```
        scaled_data = scaler.transform(numerical_data)

        return scaled_data
    except Exception as e:
        raise ValueError(f"Error in preprocessing: {str(e)}")

@app.route('/')
def home():
        return render_template('index.html', category_mapping=category_mapping,
dataset_info=dataset_info)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        input_data = {
            'Modbus Frame': request.form['modbus_frame'],
            'source': request.form['source'],
            'destination': request.form['destination'],
            'time stamp': float(request.form['timestamp'])
        }

        processed_data = preprocess_input(input_data)

        prediction = model.predict(processed_data)[0]
        probabilities = model.predict_proba(processed_data)[0]

      top_3_indices = np.argsort(probabilities)[-3:][::-1]
        top_3_predictions = [
            {
```

Продовження додатку Д

```
            "category": category_mapping[idx],
              "probability": float(probabilities[idx]*100)
            }
            for idx in top_3_indices
        ]
        classification_result = {
            'category': category_mapping[prediction],
            'confidence': float(max(probabilities)*100),
            'top_predictions': top_3_predictions
        }
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        with open(f"classification_result_{timestamp}.json", 'w') as file:
            json.dump(classification_result, file)

        return jsonify({
            'success': True,
            'category': category_mapping[prediction],
            'confidence': float(max(probabilities)*100),
            'top_predictions': top_3_predictions
        })
    except Exception as e:
        return jsonify({
            'success': False,
            'error': str(e)
        })
if __name__ == '__main__':
    app.run(debug=True)
```