МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«____»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

СИСТЕМА ВИЯВЛЕННЯ КОМП'ЮТЕРНИХ МЕРЕЖЕВИХ АТАК ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інтелектуальні інформаційні системи»

Здобувач _____ Кирило КОЛОДЯЖНИЙ

«___»_____ 2024 р.

Керівник д-р. техн. наук, доцент _____ Ірина КАЛІНІНА

«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Колодяжного Кирила Олексійовича
((прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Система виявлення комп’ютерних мережевих атак за допомогою методів машинного навчання».

Керівник роботи: Калініна Ірина Олександрівна, в. о. професора кафедри інтелектуальних інформаційних систем, д-р техн. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «03» червня 2024 р. № 140/1.

2. Строк представлення кваліфікаційної роботи «16» грудня 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: система виявлення комп’ютерних мережевих атак з використанням машинного навчання; набір даних потоку комп’ютерної мережі.

4. Перелік питань, що підлягають розробці: аналіз мережевого трафіку; огляд існуючих методів виявлення мережевих атак; аналіз результатів обраних методів машинного навчання для розв'язання поставленої задачі.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Ірина КАЛІНІНА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Кирило КОЛОДЯЖНИЙ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «07» червня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Система виявлення комп'ютерних мережевих атак за допомогою методів машинного навчання

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	03.06.2024	07.06.2024	Виконано
2	Аналіз предметної області та постановка задачі	10.06.2024	19.06.2024	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема аналіз публікацій та аналогічних систем, щодо виявлення комп'ютерних мережевих атак	20.06.2024	06.07.2024	Виконано
4	Огляд існуючих архітектур штучних нейронних мереж для вирішення поставленої задачі	01.09.2024	15.10.2024	Виконано
5	Реалізація обраних технологій з аналізом отриманих результатів	14.10.2024	16.11.2024	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	22.11.2024	22.11.2024	Виконано
7	Корегування роботи за результатами попереднього захисту	23.11.2024	05.12.2024	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	06.12.2024	06.12.2024	Виконано
9	Доробка та остаточне оформлення КР	07.12.2024	10.12.2024	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.12.2024	17.12.2024	Виконано

Керівник роботи _____ Ірина КАЛІНІНА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Кирило КОЛОДЯЖНИЙ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«19» червня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 601м ЧНУ ім. Петра Могили
Колодяжного Кирила Олексійовича
на тему: **“СИСТЕМА ВИЯВЛЕННЯ КОМП’ЮТЕРНИХ МЕРЕЖЕВИХ АТАК ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ”**

Актуальність роботи полягає у необхідності постійного підвищення рівня захищеності мережі, шляхом аналізу трафіку та виявлення атак.

Об’єктом дослідження є процеси передання інформації в комп’ютерних мережах (КМ).

Предметом дослідження є моделі машинного навчання в КМ.

Метою дослідження є підвищення виявлення КМ атак за допомогою мультикласової класифікації.

В результаті виконання роботи було досліджено чотири методи машинного навчання (метод логістичної регресії, k-найближчих сусідів, випадковий ліс та дерево рішень) та розроблено інформаційну систему для класифікації типів атак в мережі.

Дана робота складається з чотирьох розділів. Кожен розділ відповідно присвячений: аналіз предметної області; методам машинного навчання, що використані у роботі; моделюванню інформаційної системи для виявлення атак; аналізу отриманих результатів. Загальний обсяг роботи – 89 сторінок. Кваліфікаційна робота містить 2 додатки, 40 рисунків, 1 таблицю і 57 джерел посилання.

Ключові слова: виявлення мережевих атак, класифікація, нормалізація набору даних, метод логістичної регресії, метод k-найближчих сусідів, метод випадкового лісу, метод дерева рішень.

ABSTRACT

to the qualification work by the student of the group 601m of Petro Mohyla Black Sea National University

Kolodiazhnyi Kyrylo

“A SYSTEM FOR DETECTING COMPUTER NETWORK ATTACKS USING MACHINE LEARNING METHODS”

A relevance of the work is the need to constantly improve the level of network security by analyzing traffic and detecting attacks.

An object of study is the processes of information transmission in computer networks.

A subject of the study is machine learning models in computer networks (CN).

A purpose of the study is to provide recommendations for improving the protection of CNs through the future implementation of the proposed machine learning model.

As a result of the work, four machine learning methods (logistic regression, k-nearest neighbors, random forest, and decision tree) were investigated, and an information system was developed to classify types of attacks in the network.

This work consists of four sections. Each of them is devoted to analysis of the subject area; machine learning methods used in the work; modeling of the information system for detecting attacks; analysis of the results. The total volume of the work is 89 pages. The qualification work contains 2 appendixes, 40 figures, 1 table and 57 references.

Key words: network attack detection, classification, data set normalization, logistic regression method, k-nearest neighbors method, random forest method, decision tree method.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ – програмне забезпечення

ОС – операційна система

API – application programming interface

CVE – common vulnerabilities and exposures

DDoS – distributed denial of service

DoS – denial of service

IDS – intrusion detection systems

IPS – intrusion prevention systems

KNN – k-nearest neighbors

MAE – mean absolute error

MSE – mean squared error

ВСТУП

Сучасний кіберпростір характеризується стрімким зростанням кількості та складності мережевих атак, що ускладнюють захист інформаційних систем та зумовлюють необхідність систематичного, поглибленого вивчення їхньої природи, характерних ознак та способів протидії. Постійна еволюція атак, зокрема поява нових технологічних рішень з боку зловмисників, вимагає використання новітніх підходів до їхнього виявлення.

Критичним фактором успішного протистояння кібератакам є глибокий аналіз їхніх механізмів, класифікація відповідно до різних критеріїв та розуміння їхньої еволюції. На основі цього аналізу розробляються та вдосконалюються стратегії захисту, включно з впровадженням нових стандартів безпеки, такими як удосконалення алгоритмів виявлення.

Створення системи, здатної виявляти та аналізувати спектр мережевих атак, може суттєво підвищити загальний рівень безпеки інформаційної системи. Такий підхід забезпечує активне реагування на потенційні загрози, мінімізує негативні наслідки успішних нападів та сприяє формуванню комплексної системи захисту.

Актуальність – необхідність постійного підвищення рівня захищенності системи, шляхом аналізу та виявлення комп’ютерних мережевих атак.

Завдання які мають бути вирішеними:

- аналіз атак на механізм надання сесійних токенів авторизації;
- знаходження механізму, котрий буде відповідати всім вимогам безпеки;
- розробка, на основі проаналізованих даних, більш безпечного механізму.

Об’єкт дослідження – процес передання інформації в комп’ютерних мережах.

Предмет дослідження – моделі машинного навчання в комп’ютерних мережах.

Метою дослідження є підвищення ефективності виявлення комп’ютерних мережевих атак за допомогою машинного навчання.

Методи, що мають бути використані: мова програмування Python для реалізації моделей машинного навчання та візуалізації даних та мова програмування R для нормалізації набору даних.

Основні положення, системи виявлення комп’ютерних мережевих атак за допомогою методів машинного навчання, пройшли апробацію під час Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів (м. Миколаїв, 2-4 грудня 2024 р.). За результатами опубліковано тези доповіді [1].

1 АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1 Аналіз мережевих атак

Сучасний кіберпростір характеризується динамічним зростанням кількості та складності мережевих атак, що потребує поглибленого дослідження їхньої природи, структури та методів протидії. Динаміка цих загроз зумовлена як появою нових технологічних досягнень, так і винахідливістю зловмисників. Усе це робить надзвичайно важливим не тільки спостереження за існуючими атаками, але й їх глибокий аналіз та класифікацію для побудови та покращення стратегій захисту. Розуміння структури різних видів мережевих атак дозволяє розробляти більш надійні механізми безпеки та ефективно запобігати потенційним загрозам.

Мережеві атаки можна розділити на кілька категорій залежно від їхньої мети, характеру дії та використаних технологій [2]. До основних типів належать атаки на відмову в обслуговуванні (DoS/DDoS), fuzzing-атаки, зловмисне сканування систем для виявлення вразливостей, використання бекдорів для забезпечення постійного доступу до системи, атаки з використанням мережевих хробаків та використання шелл-кодів. Кожен із цих видів атак становить унікальну загрозу для інформаційної безпеки та потребує спеціальних заходів протидії. У цьому розділі буде детально проаналізовано кожен із цих типів атак.

1.1.1 DoS та DDoS-атаки

Атаки типу «відмова в обслуговуванні» (Denial of Service, DoS) спрямовані на виведення з ладу інформаційних систем шляхом навмисного перевантаження їхніх ресурсів. Зловмисник надсилає велику кількість запитів, або ж один запит з великим вмістом, через що цільова система змушена витрачати процесорний час, оперативну пам'ять та мережеві можливості винятково на обробку цих нелегітимних звернень. Відповідно, доступність сервісу для легітимних користувачів значно погіршується або й повністю припиняється [3].

Розподілені атаки на відмову в обслуговуванні (Distributed Denial of Service, DDoS) є більш досконалим варіантом DoS-атак. У таких сценаріях одночасну атаку здійснює велика кількість компрометованих хостів, об'єднаних у ботнет під контролем зловмисника. Масовість дій, а також географічна та мережева різниця задіяних машин, значно ускладнюють виявлення джерел проблеми та ефективну протидію атаці (рис. 1.1) [4].

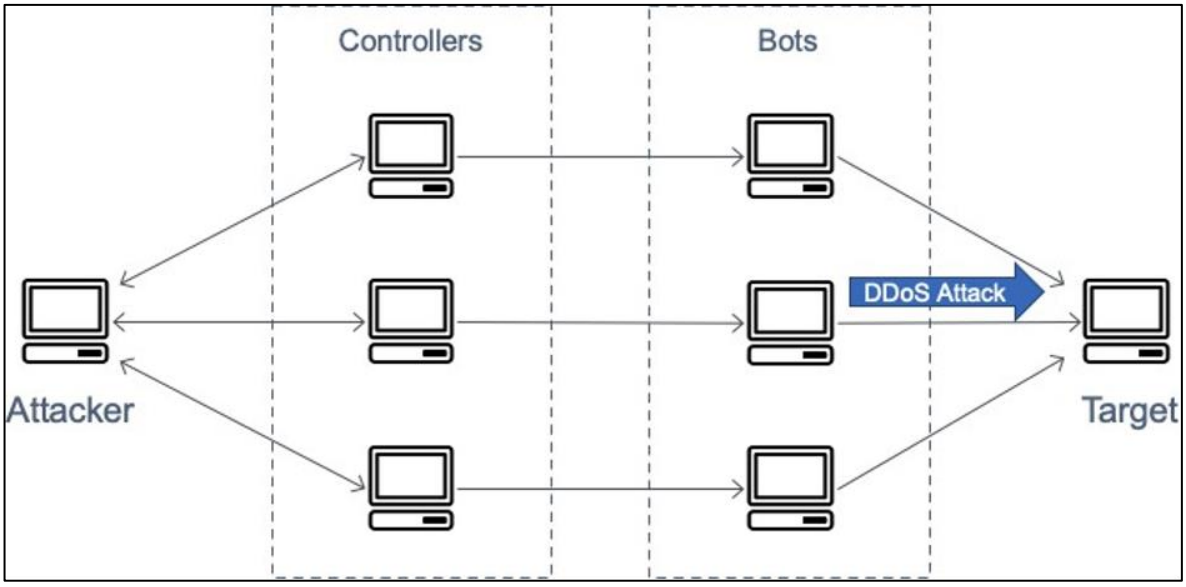

Рисунок 1.1 – Зображення DDoS-атаки зловмисником

Під час подібних атак, цільова система стикається з потоком нелегітимних запитів, що перевищує її номінальну пропускну здатність. Замість обслуговування коректних звернень, сервер використовує свої ресурси на відповіді зловмисним запитам. Це призводить до значного збільшення часу відгуку, а у найкритичніших випадках до повної зупинки роботи ресурсу. Таке порушення доступності інформаційного сервісу завдає шкоди репутації організації, може спричинити фінансові втрати та знизити довіру користувачів.

Відновлення нормального функціонування після DDoS-атаки зазвичай потребує суттєвого часу, залучення додаткового технічного персоналу та впровадження додаткових захисних механізмів.

У результаті, легітимні користувачі стикаються з повільною роботою сервісу або взагалі не можуть отримати від нього відповіді. Це робить сервіс недоступним

та може призвести до втрати довіри з боку користувачів, фінансових збитків та негативного впливу на репутацію організації [5].

Сигнатура DoS/DDoS-атак зазвичай включає наявність великої кількості однотипних або схожих запитів у короткі часові інтервали. Часто ці звернення мають однакову довжину, схему чи повторювані параметри. Такі закономірності можуть слугувати індикаторами для систем виявлення вторгнень, дозволяючи швидше ідентифікувати та блокувати шкідливий трафік.

Прикладом реалізації такої атаки можна навести атаку на українську енергетичну інфраструктуру в 2015 році. Під час цього інциденту, оператори телефонних ліній енергетичних компаній зазнали атаки відмови в обслуговуванні, через що не могли приймати дзвінки від клієнтів щодо відключень. Також операторам було заблоковано доступ до їхніх підпорядкованих пристроїв, оскільки мікропрограмне забезпечення конвертерів із послідовного інтерфейсу в Ethernet було перезаписано, що фактично вивело ці пристрої з ладу [6].

1.1.2 Fuzzing-атаки

Fuzzing-атаки (або атаки типу «перерахування»), як і багато інших сучасних методів компрометації, спрямовані на виявлення прихованих чи неправильно захищених ресурсів у мережевій інфраструктурі та інформаційних системах. Атака проводиться шляхом надсилання великої кількості випадкових або цілеспрямовано згенерованих вхідних даних для активного сканування інтерфейсів, файлових систем, веб-додатків, API а також відкритих портів і служб [7]. Завдяки цьому зловмисник має змогу зібрати детальну інформацію про об'єкти, системні компоненти та потенційно вразливих точок входу.

На відміну від DoS/DDoS-атак, метою яких є вичерпування ресурсів та виведення системи з ладу, fuzzing-атаки зосереджені переважно на пошуку інформації. Зловмисник не має на меті зробити сервіс недоступним, натомість він намагається знайти корисну інформацію, котра буде використовуватись для побудування векторів атак на інфраструктуру: приховані директорії,

адміністративні панелі, невідомі IP-адреси, ідентифікатори користувачів, токени доступу та інші потенційні точки входу [8]. Це дає змогу сформувати комплексний образ системи без належної автентифікації (рис. 1.2).

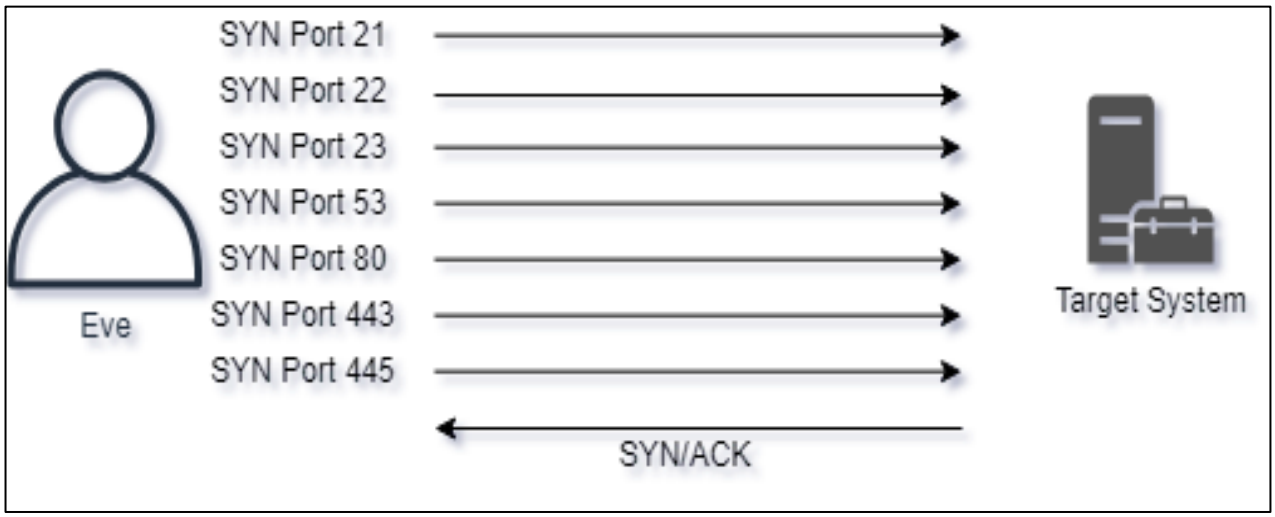

Рисунок 1.2 – Зображення перерахування портів шляхом надсилання SYN пакетів до об'єкта мережі

Одним із ключових аспектів fuzzing-атак є різноманітність технік та підходів до генерування вхідних даних. Наприклад, застосування спеціальних словників та баз даних типових шляхів, паролів чи портів, комбінування довільних та випадкових значень, а також використання автоматизованих інструментів може значно підвищити ефективність атаки. Зловмисник може варіювати формат запитів, кодування символів, параметри протоколів передачі даних, імітуючи дії легітимних користувачів або сервісів. Такий підхід ускладнює розпізнавання шкідливої активності за рахунок нестандартних даних та різноманітності вмісту. Сигнатурою цієї атаки є кількість надісланих запитів від одного або ж декількох користувачів та популярність об'єктів в запиті, по типу популярні порти, сторінки, паролі і т.д. [9].

Для fuzzing-атак характерна велика кількість різноманітних запитів, які часто не відповідають типовій поведінці легітимних користувачів чи сервісів. Ці запити можуть містити малозрозумілі параметри, випадкові рядки, різні кодування або формат даних. Часто метою є надсилання звернень до популярних сторінок чи

шляхів (наприклад, «/admin», «/docs», «/config»), відомих паролів або стандартних портів, що підвищує ймовірність виявлення прихованих об'єктів.

Прикладом використання fuzzing-атак можна привести кібератаки у відношенні українських провайдерів. Зловмисники використовували різні автономні інструменти для отримання інформації та різного доступу до інфраструктури провайдера, шляхом сканування мережевих портів, підбір автентифікаційних даних, пошук публічно доступних сервісів (біллінг, особистий кабінет користувача, хостингові сервери тощо) [10].

1.1.3 Сканери вразливостей

Сканування на існуючі вразливості є важливим елементом сучасних кібератак, оскільки воно спрямоване на цілеспрямоване виявлення недоліків у конфігураціях, програмному забезпеченні чи мережевих протоколах. На відміну від атак перерахування або fuzzing, де пріоритетом є виявлення прихованих об'єктів чи збоїв у обробці даних, сканування вразливостей зосереджується на вже раніше встановлених та відомих векторів атак. Зловмисники у цьому разі використовують тестові запити, націлюючи їх на певні системні компоненти, версії програмного забезпечення, відкриті порти або веб-застосунки, що потенційно уразливі до відомих експлойтів [11-12].

Принцип дії таких атак передбачає перевірку стану системи на наявність конкретних вразливостей, описаних у відповідних базах даних, наприклад, у базі Common Vulnerabilities and Exposures (CVE) [13]. Якщо цільова система містить застарілі модулі, плагіни, бібліотеки або неверифіковані точки доступу до Application Programming Interface (API), зловмисник може виявити їх завдяки одиничним спрямованим запитам, що містять шкідливий код або специфічні послідовності вхідних даних (рис. 1.3). У разі позитивного результату, атакуючий використовує виявлений компонент для подальшого компрометування системи та впровадження бекдорів, які у перспективі гарантують йому непомітний доступ до системи [14].

Особливістю сканування на вразливості є його спрямованість на конкретні цілі: замість надсилання величезних обсягів даних, як у DDoS, або використання загальних наборів даних, як у fuzzing-атаках, дії зловмисника є методичними та вибірковими.

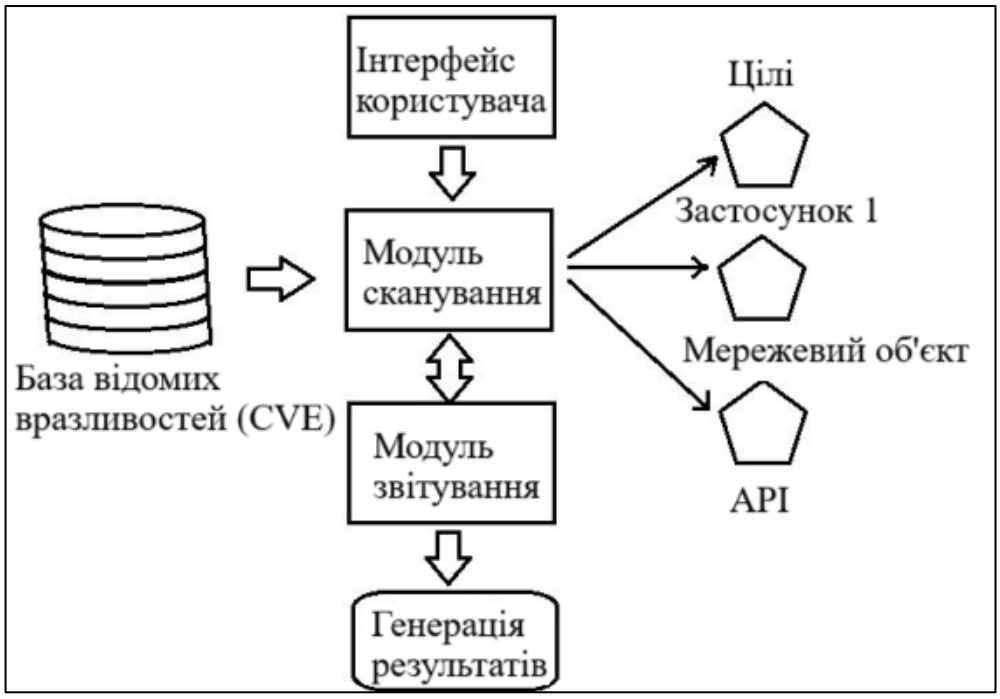

Рисунок 1.3 – Зображення класичної архітектури сканерів вразливостей

Такий підхід суттєво ускладнює виявлення з боку традиційних систем моніторингу та аналізу трафіку. Ці системи відстежують масовані аномалії в діяльності користувачів та сервісів, але окремі запити з конкретною метою можуть виглядати як цілком легітимні операції. Водночас, незважаючи на цю вибіркову методику, сканування часто базується на добре відомих шкідливих вхідних даних, сигнатурах атак або характерних ланцюжках символів у запитах, що дозволяє системам з глибоким аналізом трафіку або спеціалізованим інструментам захисту, такі як Web Application Firewall (WAF), визначати потенційно шкідливу активність.

Для сканеру вразливостей існує характерна наявність цільових запитів, що спрямовані на конкретні версії програмного забезпечення (ПЗ), протоколи чи слабкі системи. Часто ці запити містять характерні елементи (наприклад, посилання на файли конфігурації, ін'єкційні рядки коду, згадки про специфічні

технології або CVE-ідентифікатори), які допомагають класичним системам виявлення ідентифікувати запити як підозрілу активність. Зловмисні пейлоади зазвичай не масові, але мають чітке спрямування, що відрізняє їх від випадкових чи хаотичних дій [15-17].

Використання сканерів вразливостей є фундаментальним інструментом для кіберзлочинців. Приклад Pawn Storm в 2019 році демонструє ефективність цього методу для проведення масштабних фішингових кампаній та компрометації серверів. Зловмисники використовували сканери вразливостей для пошуку серверів Microsoft SQL Server та Active Directory Services, які можуть бути вразливими [18].

1.1.4 Backdoor-атаки

Цей тип атаки належить до одних із найнебезпечніших форм компрометації інформаційних систем та мереж. Особливість атак полягає у тайному впровадженні вразливих модулів, програмних застосунків або конфігурацій, що забезпечують несанкціонований доступ до системи в обхід стандартних механізмів автентифікації та контролю. Головна мета бекдору – гарантувати зловмиснику тривалий, сталий та непомітний канал керування скомпрометованою інфраструктурою [19-20].

Встановлення або імплементація backdoor-компонента зазвичай відбувається після успішної атаки іншого типу: наприклад, після виявлення вразливості в додатку чи протоколі, або ж унаслідок використання фішингу, соціальної інженерії чи зараження ланцюга постачань (Supply Chain Attack). Потрапивши до системи, бекдор може виступати у ролі сервісу, невидимого процесу, бібліотеки або навіть внесення несанкціонованих змін до налаштувань операційних систем (ОС) чи конфігурації мережевого обладнання. Часто ця присутність практично не впливає на нормальну роботу системи, що дозволяє зловмисникові залишатися «в тіні» протягом тривалого часу [21-23].

Потенціал бекдору значно розширюється завдяки гнучкості його функціоналу. Він здатен надавати віддалений доступ до файлової системи, дозволяючи зчитувати, змінювати або видаляти дані, запускати довільні процеси, змінювати права доступу чи перехоплювати мережевий трафік. Крім того, сучасні backdoor-атаки дедалі частіше використовують методи шифрування та стеганографії, що ускладнюють їх виявлення за допомогою традиційних механізмів. Зловмисник може також налаштувати тригери для активації бекдору: певні дії адміністратора, настання конкретної дати, спроби аналізу коду або рестарту системи [24].

Сигнатури backdoor-атак можуть включати аномальні мережеві з'єднання на нестандартних портах, підозрілі виконувані файли з криптографічно нестабільними хешами, невідомі службові процеси або раптову появу додаткових користувачів із надмірними правами. Часто сигнатурою може бути й незвичайна активність у часи, коли системою зазвичай не користуються, або маніпуляції з журналами подій, спрямовані на приховування слідів [25].

Використання бекдорів також поширений вектор атак після знаходження вразливостей для закріплення у системі. Під час атаки на енергетичну інфраструктуру України в 2022, зловмисники використовували бекдор виду веб-оболонки Neo-REGEORG після компрометації системи [26].

1.1.5 Worm-атаки

Атаки типу мережевих хробаків (Worm-атаки) належать до однієї з найбільш динамічних та небезпечних форм шкідливого програмного забезпечення. На відміну від традиційних вірусів, які зазвичай потребують дії користувача для розповсюдження (наприклад, відкриття зараженого файлу), хробаки поширюються автоматично, використовуючи вразливості мережевих протоколів, ОС, додатків або некоректно налаштованих служб.

Особливість атаки полягає у здатності самостійно відтворюватися та переміщуватися мережею без будь-якої взаємодії з боку користувача, що робить їх надзвичайно складними для стримування.

Сигнатури worm-атак можуть включати аномальну кількість однотипних мережевих підключень, постійні спроби звернень до певних портів чи сервісів, збільшення кількості некоректних запитів у журналах відстеження, а також типові патерни шкідливого коду, відомі за історією попередніх заражень. Крім того, зміни у конфігураційних файлах, поява нових виконуваних процесів чи раптове зростання мережевого трафіку без очевидної причини можуть слугувати індикаторами присутності черв'яка у системі.

Одним із найвідоміших випадків була поява мережевого хробака Conficker у 2008–2009 роках. Цей шкідливий код скористався вразливістю у Windows-системах для масового поширення по всьому світу. Conficker швидко інфікував мільйони пристроїв, об'єднуючи їх у мережу під контролем зловмисників (рис. 1.4).

Хробак не тільки ускладнив роботу системних адміністраторів, які намагалися відновити контроль над ураженими машинами, але й продемонстрував, наскільки небезпечними можуть бути worm-атаки при відсутності належних оновлень, моніторингу та системного захисту [28].

Рисунок 1.4 – Зображення мережевого хробака Conficker та причини зараження компонентів

1.1.6 Shellcode-атаки

Shellcode-атака – це вид кібератаки, під час якої зловмисник намагається впровадити та виконати спеціально сформований фрагмент машинного коду (так званий «shellcode») безпосередньо в пам’ять цільової системи або процесу. На відміну від традиційних шкідливих програм, що потребують запуску окремого файлу чи сценарію, shellcode виконується всередині вразливого застосунку, використовуючи недоліки у його обробці даних або верифікації вхідних параметрів.

У багатьох випадках, shellcode є інструментом, який використовується після знаходження уразливості типу buffer overflow, stack overflow, heap overflow або інших помилок управління пам’яттю [29-30]. Зловмисник замінює частину даних у буфері програми власним виконуваним кодом і спричиняє перенаправлення потоку виконання до цих впроваджених інструкцій. У результаті програма починає виконувати шкідливий код часто з підвищеними привілеями або всупереч встановленим політикам безпеки.

Сучасні атаки можуть використовувати шифрування, кодування, поліморфізм або використовувати різні методи «сліпого» виконання для обходу антивірусів та систем виявлення вторгнень. Це ускладнює розпізнавання шкідливого коду, особливо коли він створений під конкретну систему.

1.2 Аналіз методів виявлення мережевих атак

Виявлення мережевих атак включає в себе процес моніторингу подій, що відбуваються в комп'ютерній системі або мережі, та їх аналіз на предмет виявлення ознак можливих інцидентів, які є порушеннями або неминучими загрозами порушення політик комп'ютерної безпеки, політик прийнятного використання або стандартних практик безпеки. Конфіденційність, доступність і цілісність – три основні цілі систем мережевої безпеки [31]. Методи виявлення та запобігання мережевим вторгненням можна класифікувати на основі підходу, який

використовується для виявлення мережевих загроз, їх запобігання або комбінації обох підходів. Ці методи розробляються як програмні, апаратні або комбіновані [32]. Їх можна розділити на два класи: системи виявлення вторгнень (IDS) і системи запобігання вторгненням (IPS) [33], які описанні нижче.

Система виявлення вторгнень (intrusion detection systems, IDS) – або ж мережева IDS (network-based IDS, NIDS). Ця система відстежує зловмисну мережеву активність і сповіщає у разі виявлення атаки, не маючи можливостей для її запобігання (рис. 1.5). Виявлення на основі сигнатур та аномалій – це два найпоширеніші підходи, що використовуються IDS для виявлення загроз. Процедури на основі сигнатур застосовуються для виявлення тільки відомих загроз, покладаючись на базу даних, що містить список попередньо існуючих характеристик відомих атак (сигнатур атак) для виявлення підозрілих подій. База даних повинна постійно оновлюватися для включення нових сигнатур.

Якщо ж зловмисник атакує, сигнатури котрої ще немає в базі або ж система виявлення не встигла оновити базу сигнатур, то атака пройде не помітною. Це може статись через різні причини: сигнатура атаки була не повною (зловмиснику вдалося виконати атаку, змінивши базові принципи котрі характеризують сигнатуру), існування атаки було невідоме (атака «нульового дня» [34-36] чи атака «першого дня» [37] не описана). З іншого боку, процедури, засновані на аномаліях, намагаються відрізнити зловмисний трафік від реального трафіку на основі змін у мережевому трафіку; таким чином, вони можуть виявити невідомі загрози. Невідповідності, такі як трафік великого розміру, мережеві затримки, трафік з незвичайних портів і аномальна продуктивність системи – все це являє собою зміни в нормальній поведінці системи і може вказувати на наявність мережевих атак.

Однак, атака котра має на меті доступ до важливої інформації може відбуватись місяцями [38], що робить її складною для виявлення.

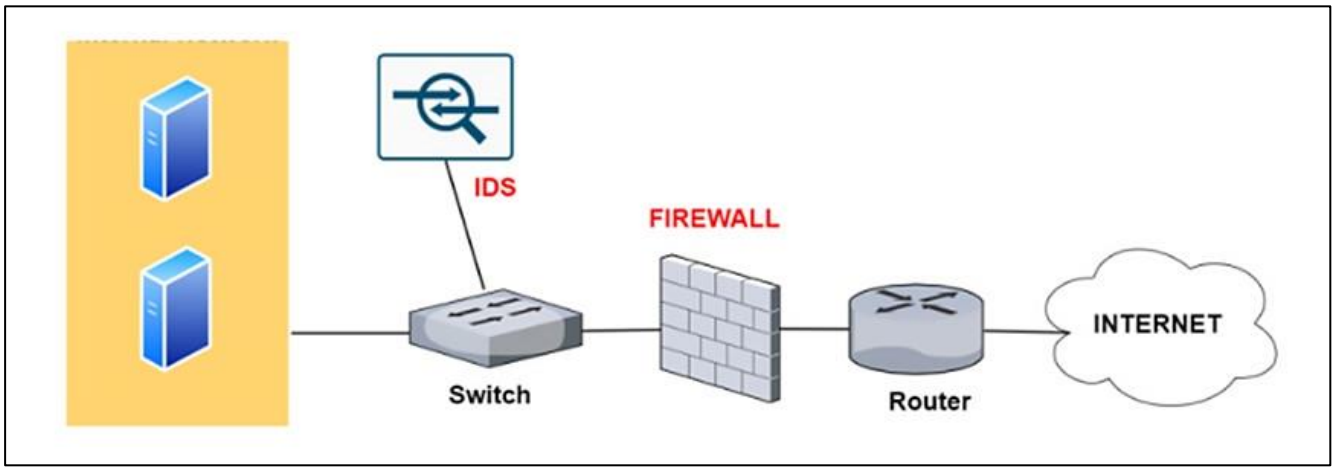

Рисунок 1.5 – Архітектура мережі при використанні IDS

Система запобігання вторгненням (intrusion prevention systems, IPS) – відома також як система виявлення та запобігання вторгнень (intrusion detection and prevention systems, IDPS). Вона безперервно сканує мережу на наявність нелигітимних або несанкціонованих контрольних точок, які виявляються на основі змін у поведінці (рис. 1.6). Система автоматично вживає контрзаходи для усунення загроз і захисту системи.

Основна мета IDPS – не допустити, щоб шкідливі або небажані пакети та атаки завдали шкоди. IDPS є більш ефективною, ніж IDS, оскільки вона не тільки виявляє загрози, але й здатна вживати заходів проти них. Існує два типи IDPS: мережеві системи виявлення та запобігання вторгнень (network-based intrusion prevention systems, NIDPS), які аналізують мережевий протокол для виявлення будь-яких підозрілих дій, і системи виявлення та запобігання вторгнень на рівні хостів (host-based intrusion prevention systems, HIDPS), які використовуються для моніторингу діяльності хостів на предмет будь-яких підозрілих подій всередині хоста.

Слід зазначити, якщо система не здатна виявити атаку на будь-яких етапах, інформаційна система та/або мережа є скомпрометованою, незалежно від запобігання певних етапів атак [39].

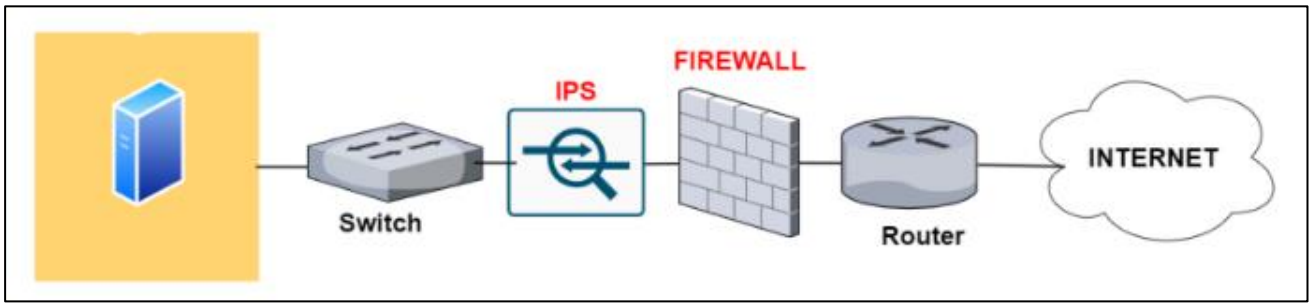

Рисунок 1.6 – Архітектура мережі при використанні IDS

Описані методи чудово захищають проти простих атак та системи виявляють, оповіщають та запобігають втручанню при явних сигнатурах та змін кількості пакетів в мережі.

Проте, при більш складних атаках, де зловмисник змінює атаку, підхід та вектори під певну інфраструктуру, класичні системи виявлення та запобігання можуть не допомогти при захисті.

Виявлення складних та багатовекторних атак, зменшення кількості хибних спрацьовувань та підвищення точності виявлення є успішним поєднанням систем виявлення та запобігання з моделями машинного та глибокого навчання. Додатковий шар, у вигляді машинного навчання, збільшує відсоток виявлених атак.

1.3 Огляд та аналіз наявних аналогів та публікацій

Серед аналогів поєднання класичних методів та штучного інтелекту можна виявити рішення, описанні нижче.

Darktrace Enterprise Immune System – продукт від Darktrace, компанії у сфері кібербезпеки, яка спеціалізується на застосуванні штучного інтелекту та машинного навчання для виявлення та реагування на кіберзагрози в режимі реального часу. EIS використовує технології машинного навчання для вивчення та моделювання нормальної поведінки користувачів і пристроїв у мережі, що дозволяє системі ідентифікувати аномалії та потенційні загрози без залежності від попередньо визначених сигнатур або правил. Функціональні можливості Darktrace EIS включають автоматичний аналіз мережевого трафіку, виявлення невідомих атак, включаючи атаки нульового дня, та автономне реагування на інциденти за

допомогою модуля Antigena, котрий може самостійно вживати заходів для ізоляції або нейтралізації загроз. Рішення охоплюють різні середовища, зокрема локальні мережі, хмарні сервіси та Інтернет речей (Internet of Things, IoT), забезпечуючи комплексний підхід до безпеки. Програмне забезпечення Darktrace IES є закритим проектом [40];

Cisco Secure Network Analytics – рішення в галузі кібербезпеки, розроблене компанією Cisco для моніторингу мережевого трафіку та виявлення загроз у режимі реального часу. Cisco SNA застосовує методи поведінкового аналізу та машинного навчання для аналізу та виявлення аномалій в мережі. Система дозволяє збирати та аналізувати телеметричні дані з різних джерел, таких як NetFlow, IPFIX та інших мережевих протоколів, що дозволяє охопити поведінку користувачів і пристроїв у мережі. Як і з Darktrace Enterprise Immune System, Cisco Secure Network Analytics дає змогу виявляти складні загрози, включаючи внутрішні атаки, компрометацію облікових записів та ексфільтрацію даних, адаптуючись до змін у мережевому середовищі та мінімізуючи кількість хибних сповіщень. Функціональні можливості системи включають детальний аналіз трафіку, виявлення аномалій, розслідування інцидентів та підтримку відповідності нормативним вимогам. Програмне забезпечення є закритим кодом, та його внутрішні алгоритми і реалізації недоступні для публічного огляду або модифікації, що може обмежувати можливості кастомізації та незалежного аудиту [41].

Для кожного рішення, закритий код та моделі машинного та глибокого навчання є основою для продажів. Таким чином, постачальник програмного застосунку сам визначає орієнтовано базу клієнтів та залишає невеликий простір налаштування фінального продукту під клієнтську мережу.

Публікації на тему використання машинного та глибокого навчання в сфері кібербезпеки також демонструють високий рівень виявлення та аналізу атак.

Одним зі знайдених досліджень є «A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks» [42]. В ньому автори досліджують застосування рекурентних нейронних мереж для побудови системи виявлення

мережевих вторгнень. Для цього вони пропонують модель RNN-IDS, яка використовує глибинне навчання для підвищення точності ідентифікації різних типів мережевих атак. Дослідження фокусується як на бінарній класифікації (нормальний трафік проти аномального), так і на багатокласовій класифікації, розрізняючи між нормальним, легітимним трафіком так і чотирма типами атак: DoS, U2R, R2L та Probe.

Результати показали, що RNN-IDS досягає вищої точності як у бінарній, так і в багатокласовій класифікації, перевершуючи традиційні алгоритми. Зокрема, модель продемонструвала високу здатність до моделювання складних патернів у даних, характерних для мережевого трафіку. Важливим висновком є те, що використання глибинного навчання та RNN дозволяє покращити виявлення вторгнень, що свідчить про потенціал глибинних нейронних мереж у сфері кібербезпеки, особливо для обробки великомасштабних і високорозмірних даних.

Автори також зазначають, що їхня модель може слугувати основою для подальших досліджень та розвитку більш досконалих систем IDS.

Однак, слід зазначити, що від цільової мережі, обирається різний набір даних, котрий буде найкраще підходити до створення моделей для виявлення атак.

На прикладі дослідження «Abnormal Behavior Detection Based on Traffic Pattern Categorization in Mobile Networks» [43], можна виявити залежність компонентів мережі від обраного набору даних. Дослідження зосереджується на аналізі аномальної поведінки в мобільних клітинних мережах і її можливих наслідках, наприклад: виникнення проблем у мережі та відключення вишок, що призводить до збільшення оперативних витрат і втрати прибутку для операторів. Дослідження визначає аномальну поведінку як ключовий фактор для уникнення подібних проблем та розглядає можливості виявлення аномалій як важливий компонент управління мережею та контролю самовідновлення.

Автори дослідження пропонують використовувати методи навчання без учителя для виявлення аномалій в мобільних мережах, використовуючи дані звітів про дзвінки (Call Detail Records, CDR). Запропонована система оцінюється на

реальному наборі даних CDR, наданому італійським оператором. Важливим висновком є демонстрація значущості врахування різних моделей трафіку в різних географічних областях для виявлення аномалій в мобільних мережах.

Дослідження щодо виявлення аномалій в мобільних мережах, є схожим до обраної теми, але через інший вид мережі, було взято інші набори даних. На основі висновку, можна визначити що обраний набір даних вплинув на вибір методу навчання та на сам результат.

1.4 Постановка задачі

Опираючись на розглянуті види мережевих атак, класичні методи виявлення, аналізу та протидії цим атакам та огляд наявних аналогів і публікацій було вирішено дослідити виявлення комп'ютерних мережевих атак за допомогою моделей машинного та глибокого навчання.

Для виконання поставленої задачі необхідно:

– дослідити мережеві атаки та їхню сигнатуру;

– розглянуті методи та системи виявлення, аналізу та запобігання атакам і їхні слабкі місця;

– розглянути теоретичні відомості про моделі машинного та глибокого навчання;

– підібрати актуальний та доступний набір даних і попередньо обробити його;

– на основі підготовлених даних навчити моделі для класифікації мережевих атак;

– проаналізувати отриманні результати.

Висновки до розділу 1

В першому розділі було розглянуто основні мережеві атаки, їхню шкоду мережі та окремим компонентам. Також було зазначено їхні сигнатури, на котрі

спираються класичні системи виявлення та протидії мережевим атакам. Однак, швидка еволюція атак, відсутність чи невелика база сигнатур впливають на безпеку мережі та її компонентам.

Поєднання систем аналізу та запобігання з моделями машинного та глибокого навчання допомагають виявляти складні та багатовекторні атаки при зменшеній кількості хибних спрацьовувань та підвищеній точності виявлення.

Наведенні аналоги Darktrace Enterprise Immune System та Cisco Secure Network Analytics використовують в своїх системах моделі машинного та глибокого навчання, що допомагає виявляти атаки на основі стандартних сигнатур та набору даних.

Додатково, було розглянуто наукові дослідження, котрі також демонструють успіх у використанні моделей машинного та глибокого навчання для виявлення та запобігання мережевим атакам.

Поставлена задача описує визначенні цілі та основну мету дослідження – покращення виявлення комп’ютерних мережевих атак за допомогою використання моделей машинного та глибокого навчання.

2 РОЗРОБКА СТРУКТУРИ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

Машинне навчання є одним з ключових розділів штучного інтелекту, який дозволяє розробити алгоритми, здатні навчити машину виконувати певні завдання. Для досягнення цього, необхідно мати набір даних, на основі якого можна досліджувати взаємозв'язки між даними, виявляти закономірності та застосовувати алгоритми. Це дозволяє, з одного боку, аналізувати вхідні дані, а з іншого – генерувати результати, обґрунтовані на визначених закономірностях. Кожен запис в наборі даних має свої унікальні характеристики, що сприяє створенню моделей, які можна узагальнити для виконання конкретного завдання. Ідея використання величезного обсягу даних для самостійно навчання машин може бути спрощеним визначенням машинного навчання.

Загалом, машинне навчання дозволяє в автономному режимі аналізувати й відокремлювати закономірності з великої кількості даних, створюючи на цій основі модель навчання. Цей підхід дозволяє покращувати результати в майбутньому, спираючись на попередній досвід. Після того, як алгоритм було навчено, він здатний виконувати складні та динамічні завдання, більш точно класифікувати, прогнозувати та реагувати на різні ситуації.

Зростання інтересу до машинного навчання в останні роки обумовлене низкою факторів, які мають аналогії із інтелектуальним аналізом даних. Це одна з найважливіших задач сучасної епохи через збільшення обсягів і різноманіття доступних даних. Крім того, затрати на ресурси для реалізації та підтримки моделей машинного навчання і зберігання великих об'ємів даних зменшуються. Крім того, результатом є високі прогнози, які можливо отримати за короткий період часу, що демонструє виконання самостійних дій в режимі реального часу і прийняття кращих рішень без будь-якого втручання людини.

Реалізація методів машинного навчання на основі набору даних належить до класу навчання з учителем. Цей процес базується на отриманні вхідних даних, де кожен зразок має додаткову інформацію про категорію чи значення. Головна мета

полягає в знаходженні закономірності між ознаками вхідних даних і відповідними мітками, що дозволяє моделі працювати з новими наборами даних. Слід зазначити, що метод навчання з учителем дозволяє використовувати великі набори даних, що допомагає навчити модель швидше.

Класифікація є одним із найважливіших завдань у машинному навчанні, що належить до класу задач навчання з вчителем. Мета класифікації полягає у визначенні категорій, наданих зразків даних, на основі попереднього аналізу промаркованих даних. Під час навчання модель тренується ідентифікувати закономірності, в наборі даних, котрі пов’язані із кожним зразком. Надалі використовує це для класифікації нових наборів даних.

Серед видів класифікації можна розрізнити бінарну класифікацію та мультикласову класифікацію. Під час використання бінарної класифікації, дані поділяються чітко на два класи. У мультикласової класифікації модель не лише розділяє дані, а й визначає, до якого саме класу належить кожен зразок.

2.1 Метод логістичної регресії

Логістична регресія є одним із найпоширеніших методів у машинному навчанні, який використовується для вирішення задач, де необхідно передбачити ймовірність належності об'єкта до певної категорії. Метод належить до узагальнених лінійних моделей і використовує логістичну функцію для опису зв’язку між залежною змінною та незалежними змінними.

Основна ідея полягає в зміні залежної змінної за допомогою логістичної функції, яка перетворює значення в діапазон ймовірностей від 0 до 1 (рис. 2.1).

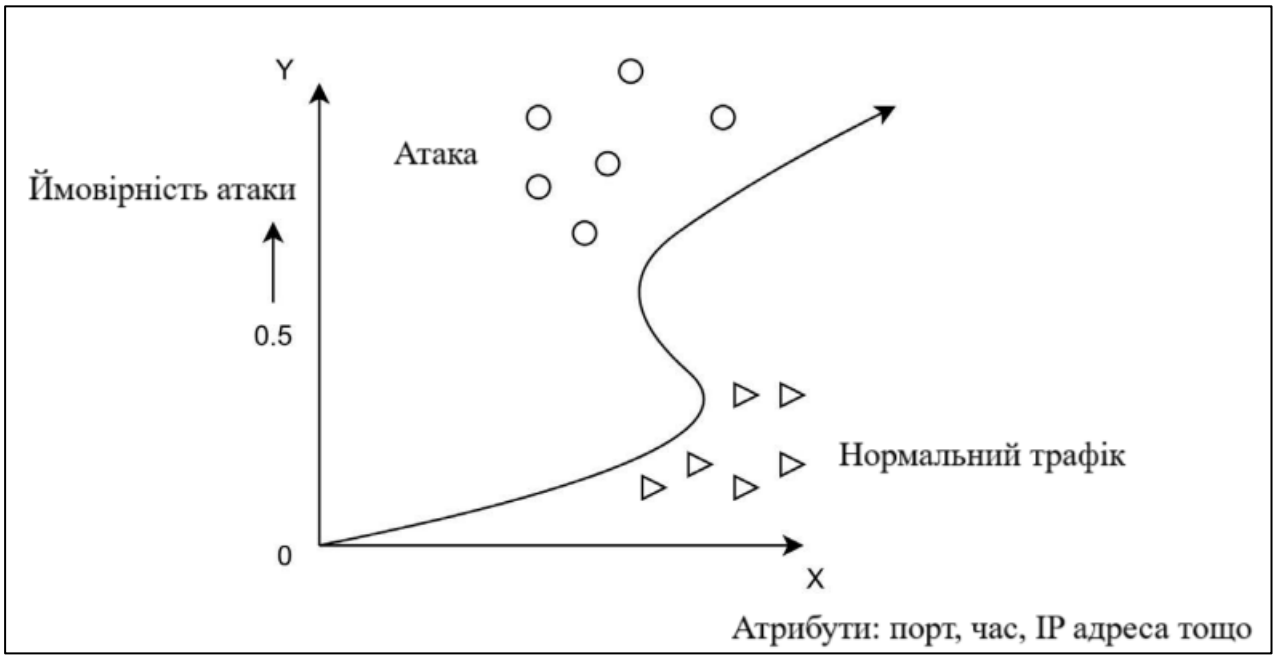

Рисунок 2.1 – Зображення логістичної функції на основі класифікації мережевого трафіку

Такий підхід дозволяє оцінювати ймовірність належності конкретного випадку до позитивного класу (наприклад, нормальний трафік) або негативного класу (наприклад, трафік атаки).

Для моделювання ймовірності належності до класу логістична регресія використовує наступне рівняння:

$$y = \frac{e^{(\beta_0+\beta_1 x)}}{1+e^{(\beta_0+\beta_1 x)}}, \quad (2.1)$$

де y – ймовірність належності до позитивного класу;

β_0 – константа, як враховує базовий рівень результату, коли всі незалежні змінні дорівнюють нулю;

β_1 – коефіцієнт, що відображає вплив змінної на результат;

x – значення незалежної змінної (ознаки).

Процес навчання моделі логістичної регресії передбачає оптимізацію параметрів β_0 та β_1 для мінімізації похибки передбачень. Для цього

використовується метод максимальної правдоподібності, який дозволяє знайти такі значення параметрів, що максимізують відповідність моделі тренувальним даним.

Після оптимізації модель може бути використана для класифікації нових даних. Для кожного нового зразка обчислюється значення y, яке інтерпретується як ймовірність належності до позитивного класу. Для прийняття рішень зазвичай встановлюється порогове значення, наприклад 0.5 та якщо ймовірність перевищує поріг, зразок відносять до позитивного класу, інакше – до негативного [44].

2.2 Метод k-найближчих сусідів

k-найближчих сусідів (k-Nearest Neighbors, KNN) – це метод машинного навчання, застосовуваний для вирішення задач класифікації та регресії. Він ґрунтується на принципі подібності, згідно з яким об'єкти з близькими значеннями ознак повинні мати подібні результати або належати до одного класу.

Для вимірювання відстані між зразками використовується мангеттенська відстань, яка визначається як:

$$d(x, x_i) = \sum_{j=1}^{n} \left| x_j - x_{ij} \right|, \tag{2.2}$$

де $d(x, x_i)$ – відстань між зразками x_j та x_{ij};

x_j – значення ознаки відповідно для наданих зразків;

n – кількість ознак.

Метод KNN обчислюються відстані до всіх зразків навчальної вибірки, для нового зразка, після цього вибираються k зразків з найменшими відстанями – що і є найближчими сусідами (рис. 2.2).

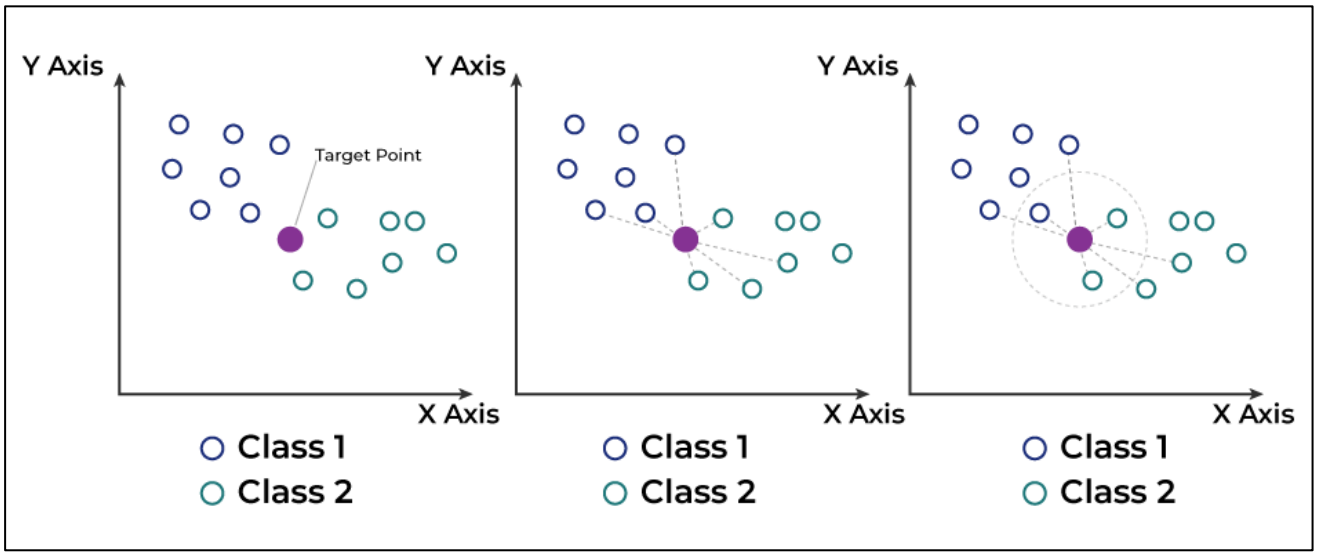

Рисунок 2.2 – Приклад класифікації методом k-найближчих сусідів

У випадку класифікації визначення класу нового зразка здійснюється на основі класів цих сусідів, шляхом простої більшості голосів або з використанням зваженого голосування, де ваги можуть бути обернено пропорційними до відстаней.

Вибір параметра k є критичним для роботи алгоритму. Низькі значення k можуть зробити модель чутливою до шуму в даних, тоді як високі значення можуть призвести до врахування занадто великої кількості далеких сусідів, що може знизити точність. Оптимальне значення k часто визначається експериментально.

Перед застосуванням KNN важливо виконати попередню обробку даних, оскільки масштабування ознак впливає на результати через використання метрик відстані. Стандартизація або нормалізація ознак допомагає забезпечити, щоб всі ознаки мали рівний вплив на розрахунок відстаней [45].

2.3 Метод випадкового лісу

Метод випадкового лісу (Random Forest) – це потужний ансамблевий метод машинного навчання, призначений для задач класифікації та регресії. Метод ґрунтується на ідеї об'єднання великої кількості дерев рішень. Ключовою концепцією є зниження варіативності та перенавчання шляхом усереднення результатів багатьох моделей.

Формула для прогнозування результату в задачі класифікації визначається як мода прогнозів окремих дерев:

$$\hat{y} = \arg max_k \sum_{i=1}^{N} \mathbb{I}(h_i(x) = k), \quad (2.3)$$

де $\hat{y}$ – остаточний прогноз моделі для вхідного зразка x;

N – кількість дерев у лісі;

$h_i(x)$ – прогноз і-го дерева для зразка;

k – можливі класи;

$\mathbb{I}$ – індикаторна функція, яка дорівнює 1, якщо умова виконується, і 0 в іншому випадку.

На основі формули, метод працює шляхом побудови великої кількості дерев рішень, кожне з яких навчається на випадковій підвибірці навчальних даних, обраній з поверненням. Для кожного вузла дерева при розбитті вибирається випадкова підмножина ознак із загального набору. Ця випадковість у виборі даних та ознак зменшує кореляцію між окремими деревами, що сприяє підвищенню загальної точності моделі.

Після побудови всіх дерев, їхні прогнози об'єднуються для отримання остаточного результату. У задачах класифікації використовується принцип голосування більшості: клас, який був найчастіше передбачений окремими деревами, стає остаточним прогнозом для зразка (рис. 2.3).

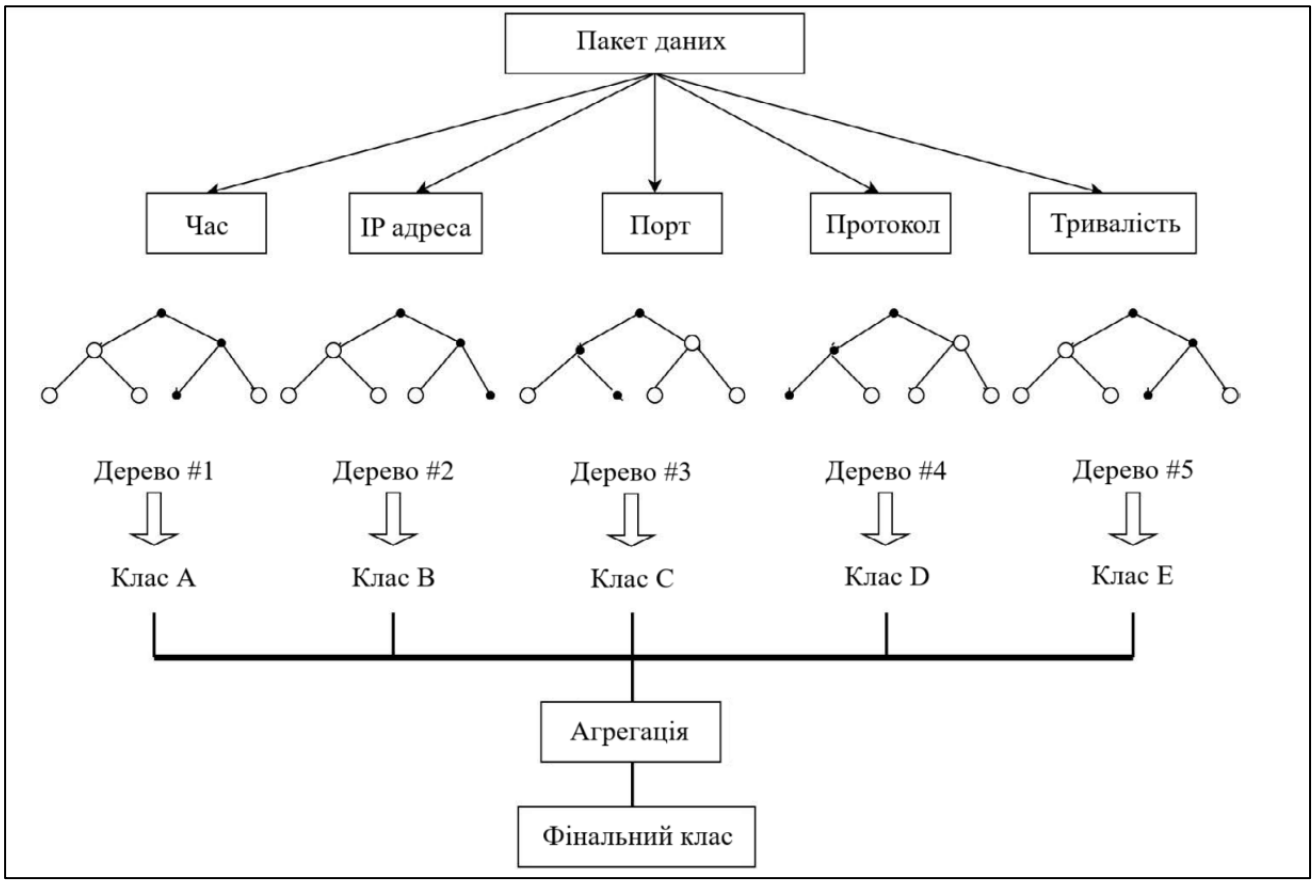

Рисунок 2.3 – Приклад структури методу випадкового лісу

Додатково, випадковий ліс надає можливість оцінювати важливість ознак у моделі. Це здійснюється шляхом аналізу того, наскільки часто певні ознаки використовуються для розбиття вузлів у деревах та який внесок вони роблять у зниження критерію помилки. Така інформація є цінною для інтерпретації моделі та виявлення ключових факторів, що впливають на результат.

Також метод відзначається стійкістю до шуму та викидів у даних. Завдяки використанню бутстреп-агрегування та випадкового вибору ознак, модель менш чутлива до аномальних спостережень, що покращує її здатність до узагальнення на нових даних [46].

2.4 Метод дерев рішення

Дерева рішень – це ефективний метод машинного навчання з наглядом, який використовується для вирішення задач класифікації та регресії. Вони будують

деревоподібну структуру рішень та можливих результатів, дозволяючи алгоритму навчатися та генерувати прогнози на основі вхідних ознак.

Формально, процес розбиття даних у дереві рішень можна описати за допомогою критерію, такого як індекс Джині або інформаційна ентропія, та можна визначити як:

$$E(S) = \sum_{i=1}^{k} - p_i \log_2 p_i, \quad (2.4)$$

де $E(S)$ – ентропія набору даних S;

k – кількість класів;

p_i – ймовірність того, що випадковий зразок належить до класу k.

Алгоритм починає з кореневого вузла, що містить набір даних, і поетапно ділить дані на менші підмножини на основі ознак, котрі максимально знижують ентропію.

Кожний внутрішній вузол у дереві містить ознаку або властивість, за якою відбувається розбиття, а кожний листовий вузол є кінцевим прогнозом. Розбиття триває доти, доки не будуть виконані певні умови зупинки, такі як мінімальна кількість зразків у листових вузлах, максимальна глибина дерева або мінімальне зменшення ентропії (рис. 2.4).

У результаті дерево рішень перетворює дані на набір правил, які можуть бути використані для прогнозування цільової змінної для нових наборів даних.

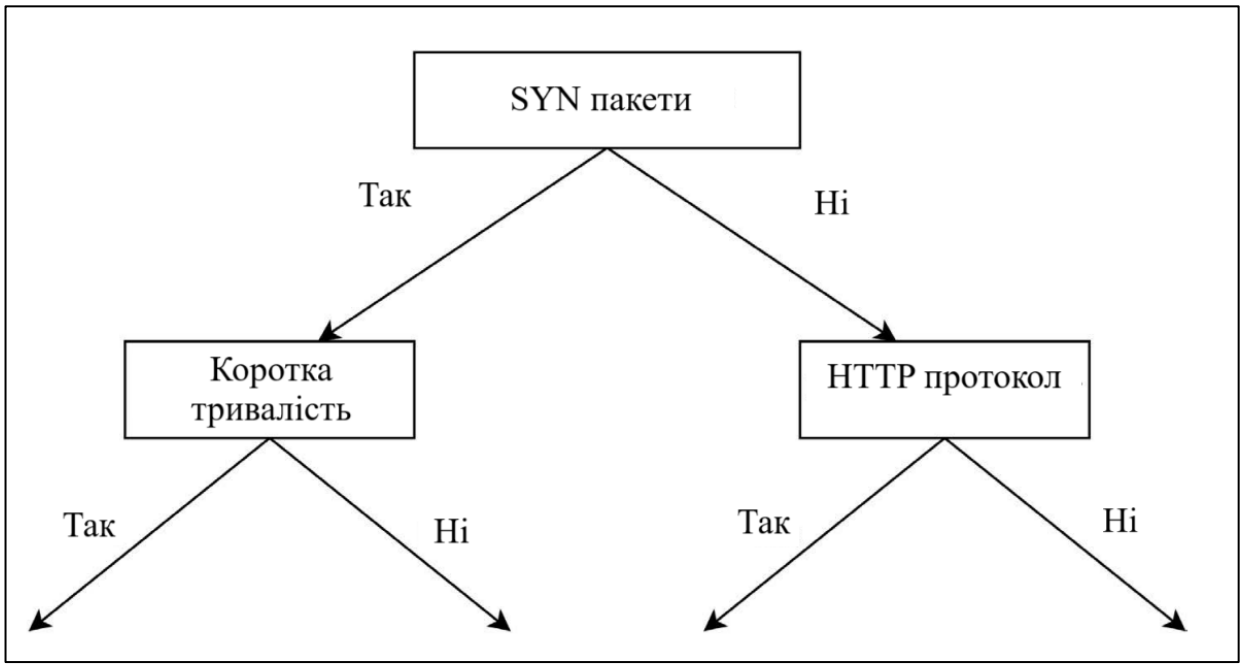

Рисунок 2.4 – Приклад моделі дерев рішень

Таким чином, метод дерева рішень працює шляхом ітеративного розбиття простору ознак на підпростори, що відповідають різним значенням цільової змінної. Алгоритм вибирає на кожному кроці ту ознаку та поріг розбиття, які максимально підвищують "чистоту" підмножин, тобто зменшують неоднорідність даних у них.

Додатково, дерева рішень можуть обробляти як числові, так і категоріальні ознаки, що робить їх універсальними для різних типів даних. Однією з ключових переваг дерев рішень є їхня інтерпретація, оскільки структура дерева чітко ілюструє зв'язки між ознаками та цільовою змінною. Це дозволяє легко пояснити рішення моделі та зрозуміти, які фактори найбільше впливають на результат.

Крім того, метод дерев рішень має можливість враховувати нелінійні залежності між ознаками та цільовою змінною, а також взаємодії між ознаками. Однак, однією з потенційних проблем є ризик перенавчання, особливо коли дерево стає занадто глибоким і складним. Для запобігання цьому застосовуються методи обрізання дерева або встановлення обмежень на його глибину.

Узагальнюючи, дерева рішень є потужним та інтуїтивно зрозумілим інструментом машинного навчання, що дозволяє ефективно моделювати складні залежності в даних і забезпечує високу інтерпретацію результатів [47].

2.5 Метрики оцінки якості моделей

Оцінка якості моделей класифікації є критично важливим етапом у процесі машинного навчання, оскільки дозволяє визначити, наскільки ефективно модель здатна класифікувати правильні класи на нових даних. Коректна оцінка моделі забезпечує розуміння її продуктивності, виявляє можливі недоліки та сприяє вдосконаленню алгоритмів для досягнення оптимальних результатів у практичних застосуваннях.

Одним із основних інструментів для аналізу результатів класифікації є матриця помилок або матриця невідповідностей. Вона надає детальне уявлення про те, як модель класифікує зразки між різними класами, дозволяючи ідентифікувати типи помилок, які вона здійснює. У випадку бінарної класифікації матриця невідповідностей має наступний вигляд (рис. 2.5).

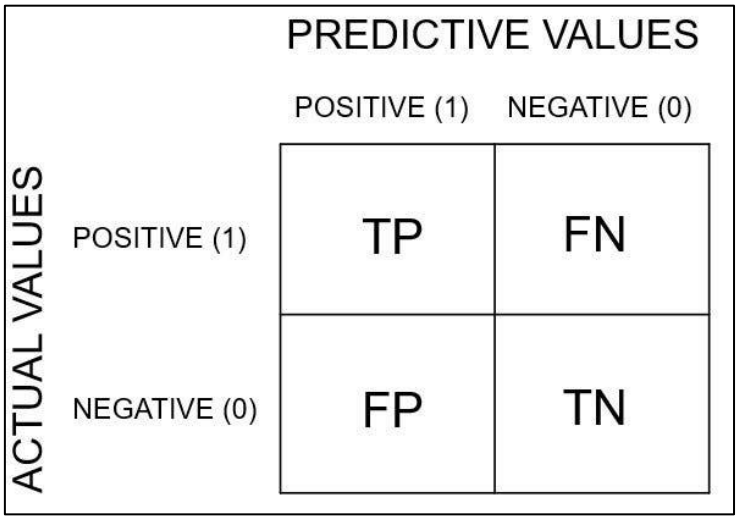

Рисунок 2.5 – Матриця невідповідностей для бінарної класифікації

Скорочення на рис. 2.5 розшифровуються як:

– TP (True Positive) – кількість правильно передбачених позитивних зразків;

– TN (True Negative) – кількість правильно передбачених негативних зразків;

– FP (False Positive) – кількість негативних зразків, помилково класифікованих як позитивні;

– FN (False Negative) – кількість позитивних зразків, помилково класифікованих як негативні.

У випадку багатокласової класифікації матриця помилок розширюється до розміру C_n, де n – кількість класів. Кожен елемент матриці представляє кількість зразків фактичного класу (рис. 2.6).

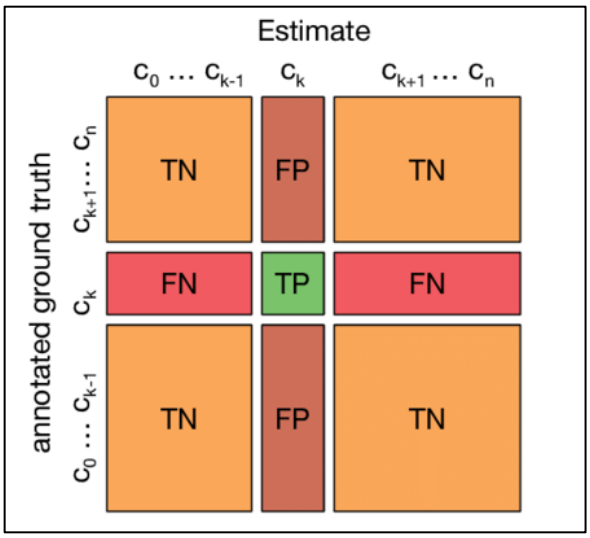

Рисунок 2.6 – Матриця невідповідностей для мультикласової класифікації

На основі матриці помилок розраховуються різні метрики для оцінки продуктивності моделі, які описані нижче.

Точність (Accuracy) визначається як відношення кількості правильно передбачених зразків до загальної кількості зразків:

$$Accuracy = \frac{\sum_{i=1}^{K} M_{ii}}{N}, \quad (2.5)$$

де N – загальна кількість зразків;

K – кількість класів;

M_{ij} – кількість правильно передбачених зразків класу i.

Точність (Precision) для кожного класу iii вимірює, яка частка зразків, передбачених як клас iii, дійсно належить до цього класу:

$$Precision_i = \frac{M_{ii}}{\sum_{j=1}^{K} M_{ji}}, \quad (2.6)$$

де $\sum_{i=1}^{K} M_{ij}$ – загальна кількість зразків, передбачених як клас i;

K – кількість класів.

Повнота (Recall) або чутливість для класу iii вимірює, яка частка зразків фактичного класу iii була правильно передбачена:

$$Recall_i = \frac{M_{ii}}{\sum_{j=1}^{K} M_{ij}}, \quad (2.7)$$

де $\sum_{i=1}^{K} M_{ij}$ – загальна кількість зразків, передбачених як клас i;

K – кількість класів.

F1-міра (F1-score) для класу i є гармонійним середнім між точністю та повнотою:

$$F1_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i}, \quad (2.8)$$

де N – загальна кількість зразків;

M_{ij} – кількість правильно передбачених зразків класу i.

Підтримка (Support) вказує на кількість зразків у кожному класі в наборі даних. Вона не є метрикою продуктивності, але надає контекст для інтерпретації інших метрик, показуючи, як розподілені дані між класами.

Додатково до метрик, заснованих на матриці помилок, для оцінки якості моделей класифікації також використовуються інші метрики, які дозволяють більш детально аналізувати точність та ефективність моделі.

Однією з таких метрик є середня абсолютна помилка (Mean Absolute Error, MAE), яка вимірює середню абсолютну різницю між передбаченими значеннями та фактичними мітками класів:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|, \quad (2.9)$$

де N – загальна кількість зразків;

y_i – мітка класу i;

$\hat{y}_i$ – передбачене значення для класу i.

Середня квадратична помилка (Mean Squared Error, MSE) розраховується як середнє значення квадратів різниць між передбаченими та фактичними значеннями:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \quad (2.10)$$

Ця метрика підкреслює більші помилки завдяки квадратичному характеру формули, що робить її особливо чутливою до великих відхилень у передбаченнях. MSE використовується для оцінки дисперсії помилок моделі.

Корінь середньої квадратичної помилки (Root Mean Squared Error, RMSE) є квадратним коренем з MSE і повертає метрику до початкових одиниць виміру, що спрощує інтерпретацію результатів:

$$RMSE = \sqrt{MSE} \quad (2.11)$$

RMSE дозволяє інтерпретувати середню помилку передбачення в контексті вихідних значень, що є корисним для розуміння практичної значущості помилок моделі.

Коефіцієнт детермінації (R^2), або R-квадрат, оцінює, наскільки добре модель пояснює варіативність у даних:

$$MAE = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}, \quad (2.12)$$

де $\bar{y}_i$ – середнє значення фактичних міток класу i.

Застосування додаткових метрик, таких як MAE, MSE, RMSE та R^2, у поєднанні з традиційними метриками класифікації, дозволяє отримати більш повне та глибоке розуміння ефективності моделі. Це особливо важливо у випадках, коли модель надає передбачення у вигляді ймовірностей або коли необхідно оцінити ступінь відхилення передбачень від фактичних значень. Комплексний підхід до оцінки якості моделей класифікації сприяє виявленню слабких місць та напрямків для покращення, що є критично важливим для досягнення високої точності та надійності моделі в практичних застосуваннях.

2.6 Структура системи виявлення

Система виявлення комп'ютерних мережевих атак на рис. 2.7, та включає в собі наступні основні блоки: збір даних (аналіз структури та атрибутів набору даних), підготовка даних (нормалізація та стандартизація набору даних для отримання кращого відсотку точності виявлення), навчання моделей (поділення набору даних та реалізація описаний методів машинного навчання), визначення точності моделей (використання описаних метрик оцінки якості моделей) та аналіз результатів (порівняння метрик та визначення кращої моделі).

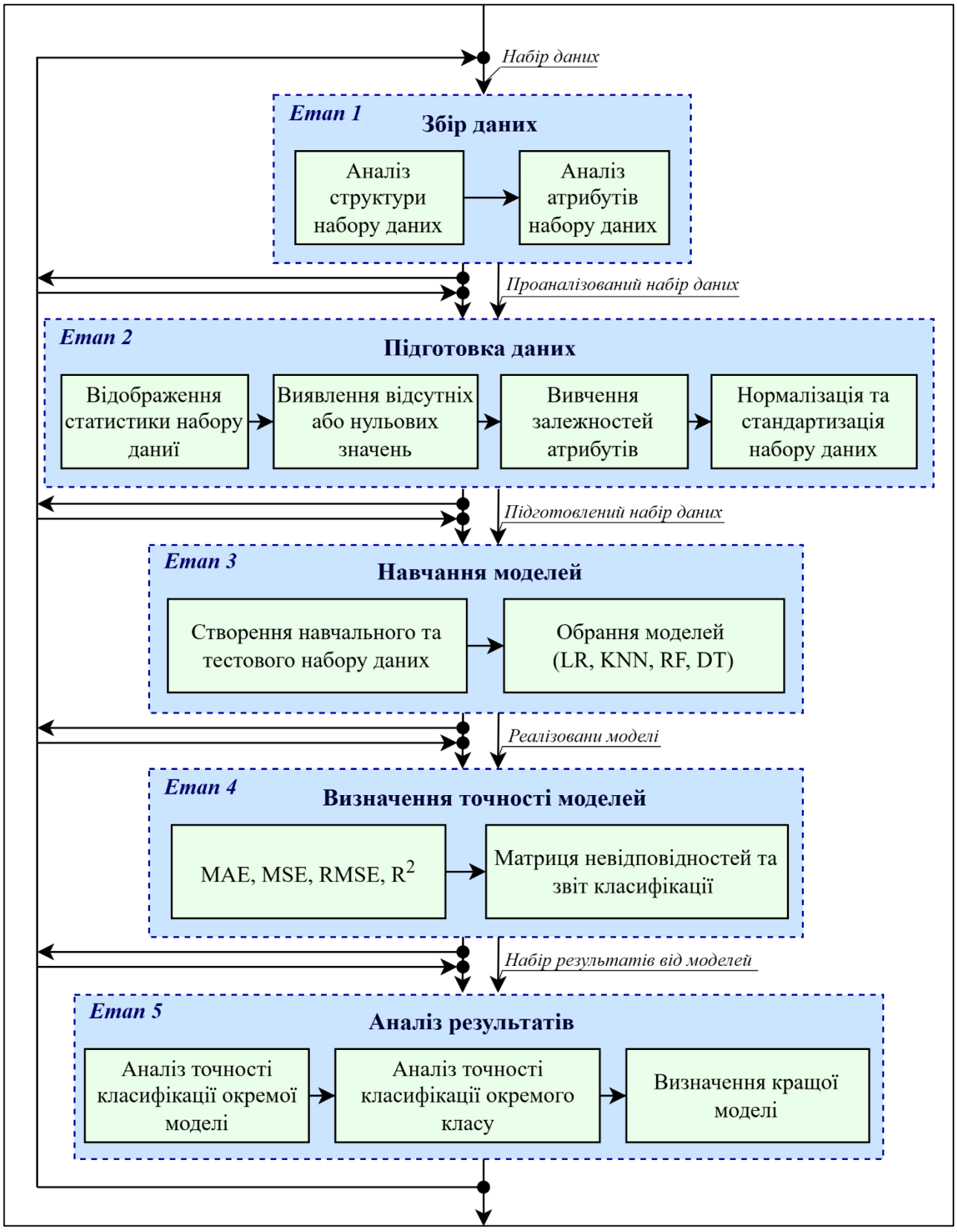

Рисунок 2.7 – Структура системи виявлення комп'ютерних мережевих атак

Висновки до розділу 2

В другому розділі було розглянуто методи машинного навчання та створено структуру системи виявлення комп'ютерних мережевих атак. Спочатку було висвітлено основи машинного навчання, зокрема важливість навчання з учителем та задачі класифікації, які є критичними для аналізу великих обсягів даних та виявлення закономірностей.

Було проаналізовано чотири основні методи машинного навчання: логістичну регресію, метод k-найближчих сусідів, метод випадкового лісу та дерева рішень. Логістична регресія використовується для передбачення ймовірності належності об'єкта до певної категорії, що дозволяє ефективно класифікувати мережевий трафік та визначати класи атак. Метод k-найближчих сусідів ґрунтується на принципі подібності, де класифікація нових зразків здійснюється на основі найближчих сусідів у просторі ознак. Вибір оптимального значення параметра k та попередня обробка даних є критичними для продуктивності цього методу. Метод випадкового лісу представляє собою ансамблевий підхід, який об'єднує велику кількість дерев рішень для зниження варіативності та підвищення точності моделі. Метод дерев рішень будує деревоподібну структуру на основі розбиття даних за критеріями, такими як інформаційна ентропія. Вони можуть обробляти як числові, так і категоріальні ознаки та враховувати нелінійні залежності між ними.

У розділі також були розглянуті метрики оцінки якості моделей класифікації, такі як точність, точність класифікації, повнота, F1-міра та підтримка. Ці метрики дозволяють детально аналізувати продуктивність моделей та ідентифікувати типи помилок. Додатково, використання метрик MAE, MSE, RMSE та R^2 надає можливість оцінити точність передбачених ймовірностей та ступінь відхилення передбачень від фактичних значень, що є важливим при прийнятті рішень на основі ступеня впевненості моделі.

На основі проаналізованої інформації, було реалізовано структуру системи виявлення мережевих атак, яка включає етапи збору та підготовки даних, навчання моделей, визначення їх точності та аналізу результатів. Такий комплексний підхід дозволяє ефективно застосовувати розглянуті методи машинного навчання для класифікації багатокласових атак у комп'ютерних мережах.

3 АНАЛІЗ ТА ПІДГОТОВКА ДАНИХ

3.1 Аналіз набору даних

Структура та підготовка набору даних є ключовим етапом в процесі реалізації моделей машинного та глибокого навчання для виявлення мережевих атак. Набір даних є основою, на якій базується навчання алгоритмів, та його якість безпосередньо впливає на кінцеві результати моделювання. Невідповідність даних, наявність шуму, неповних або хибних записів може призвести до неточних прогнозів або невірної інтерпретації результатів.

Для отримання якісних та ефективних моделей виявлення атак необхідний добре структурований, повний та релевантний набір даних, котрий охоплює широкий спектр мережевого трафіку, включаючи як легітимний так і нелегітимний трафік. У галузі дослідження мережевої безпеки існують різні набори даних, які використовуються для навчання та тестування алгоритмів виявлення атак.

Одним із найвідоміших є KDD Cup 99 набір даних, що був широко використовуваний для задач виявлення мережевих атак. Попри свою популярність, він має низку суттєвих недоліків, таких як застарілість даних та надмірна кількість синтетичних атак, що не відображають сучасні вектори атак та не допоможуть у вирішенні поставленої задачі [48].

Іншим аналогом є NSL-KDD, що є вдосконаленою версією KDD Cup 99 з виправленими багатьма недоліками, такими як усунення надмірного дублювання записів. Проте, NSL-KDD також має обмеження, пов'язані з тим, що він не повністю відображає сучасний стан загроз у мережах і не містить нових типів атак, які з'явилися в останні роки.

Ще одним популярним набором даних є DARPA 1998, який був створений з метою моделювання мережевих атак на реальних даних. Цей набір даних був одним із перших, який охоплював широкий спектр мережевого трафіку, включаючи різноманітні атаки. Однак він також має свої недоліки, оскільки його дані вже є застарілими, що не дозволяє повністю враховувати сучасні атаки та загрози [49].

Останні роки почав набувати популярності набір даних UNSW-NB15. Він був створений для вирішення проблем, пов'язаних із застарілістю попередніх наборів і відображає новітні типи мережевих атак. UNSW-NB15 був зібраний з використанням сучасної інфраструктури, яка відображає реальні мережеві умови (рис. 3.1).

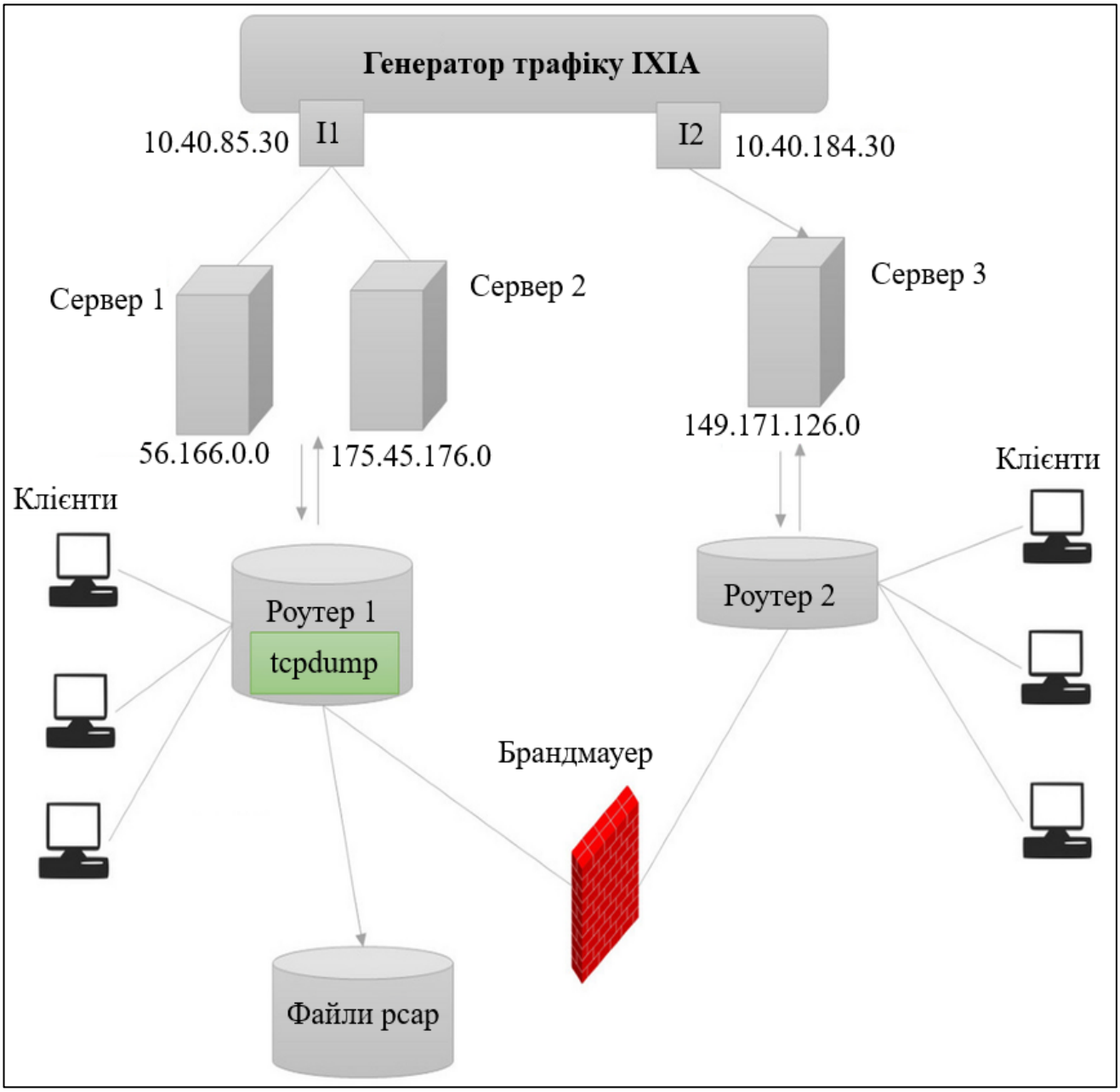

Рисунок 3.1 – Зображення мережевої інфраструктури котра використовувалась під час створення набору даних

Для створення набору даних використовувався генератор трафіку IXIA. Обраний генератор дозволяє відтворювати складні сценарії атак, які відповідають реальним загрозам в комп'ютерних мережах. Гнучкість конфігурації, інтеграція з сучасними стандартами та підтримка багатоетапних атак дозволяють генератору застосовувати наступні атаки: DDoS, спроби проникнення, експлуатацію вразливостей протоколів і послуг (на основі MITRE та CVE), а також шкідливий трафік, пов'язаний із ботнетами, вірусами та шпигунським програмним забезпеченням.

IXIA котрий був налаштований з трьома серверами. Перший та третій сервери були призначені для легітимного трафіку, котрий імітує звичайну діяльність користувачів в мережі, таких як: передача даних, запит на сервер та інші типові дії в інфраструктурі. Другий сервер був призначений для створення не легітимної активності, що включає в себе різні види атак та аномальна діяльність.

Для забезпечення комунікацій між серверами та мережевими вузлами було налаштовано два віртуальні інтерфейси, котрі були підключені до двох маршрутизаторів які забезпечували передачу даних до брандмауера, що пропускав весь трафік, незалежно від його характеристик (нормальний чи аномальний) [50].

При виконання моделювання, файли формату pcap були отримані за допомогою інструменту tcpdump. Характеристики набору даних UNSW-NB15 були отримані використовуючи інструменти Argus, Bro-IDS, а також шляхом розробки дванадцяти алгоритмів, як це відображено на рис. 3.2. Інструмент Argus відповідає за обробку мережевих пакетів та генерує атрибути потоків мережевих пакетів. В свою чергу Bro-IDS орієнтований на моніторинг мережевого трафіку, та використовувався для інспекції мережевого трафіку для виявлення потенційно шкідливої активності.

Результуючі файли, отримані з інструментів Argus та Bro-IDS, зберігаються в базі даних SQL Server та подалі оброблюються в csv формат для більш зручного аналізу [51].

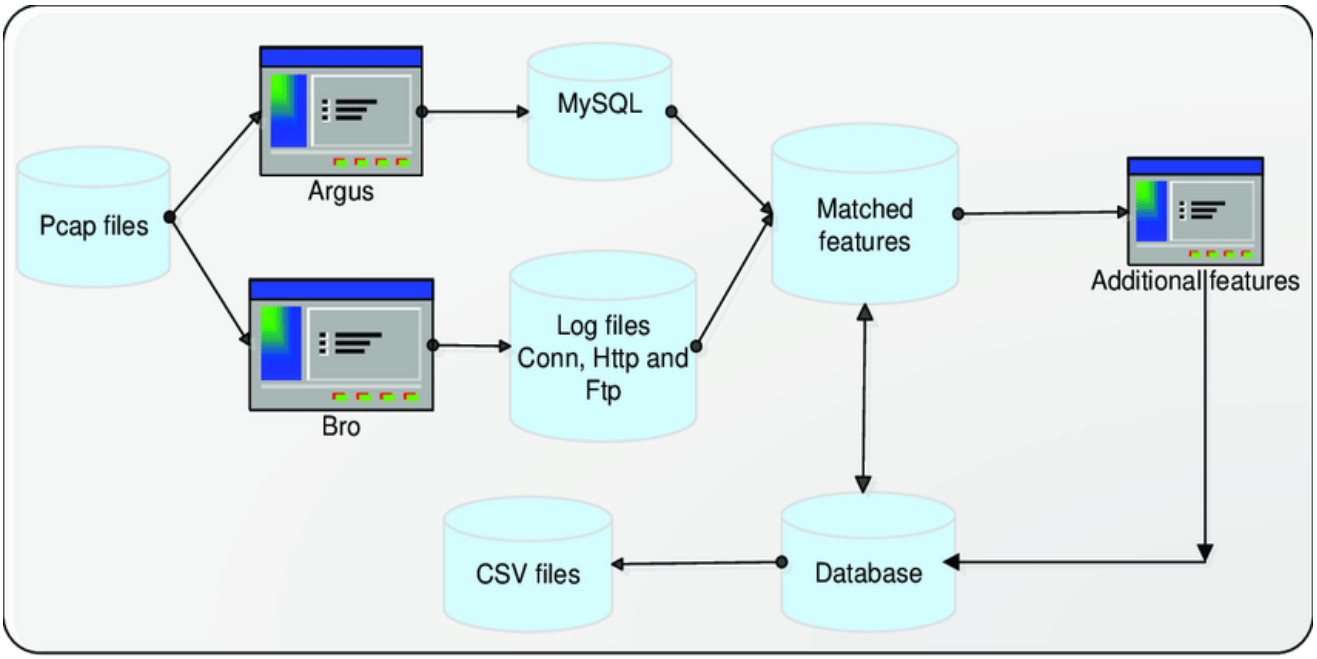

Рисунок 3.2 – Структура фреймворку використаного для генерації набору даних

3.2 Структура набору даних

Набір даних UNSW-NB15 має 49 атрибутів, які детально описують кожне мережеве з'єднання. Атрибути можна поділити на наступні категорії:

а) атрибути характеристики потоку – містять основні дані про запит:

- srcip – IP-адреса джерела;
- sport – порт джерела;
- dstip – IP-адреса призначення;
- dsport – порт призначення;
- proto – використаний протокол;

б) атрибути базових характеристик – надають фундаментальну інформацію про кожне мережеве з'єднання або сеанс:

1) state – стан з'єднання та залежний протокол. Прикладами станів можуть бути:

- ACC (Accepted) – з'єднання встановлено успішно;
- CLO (Closed) – з'єднання закрито;
- FIN (Finished) – процес передачі даних завершено;

– CON (Connected) – з'єднання встановлено;
– RST (Reset) – з'єднання скинуто;

2) dur – тривалість сеансу;
3) sbytes – байти від джерела до призначення;
4) dbytes – байти від призначення до джерела;
5) sttl – значення TTL з боку джерела. Показує початкове значення поля "Time To Live" у пакеті, відправленому від джерела;
6) dttl – значення TTL з боку призначення;
7) sloss – втрачені або повторно передані пакети з боку джерела;
8) dloss – втрачені або повторно передані пакети з боку призначення;
9) service – сервіс або застосунок, що використовується. Вказує на тип сервісу або протоколу прикладного рівня, який використовувався в сеансі, наприклад:

– http – протокол передачі гіпертексту;
– ftp – протокол передачі файлів;
– ssh – протокол безпечного віддаленого доступу;
– dns – система доменних імен;

10) sload – навантаження з боку джерела в бітах за секунду;
11) dload – навантаження з боку призначення в бітах за секунду;
12) spkts – кількість надісланих пакетів від джерела до призначення.
13) dpkts – кількість надісланих пакетів від призначення до джерела;

в) атрибути вмісту – надають детальну інформацію про параметри з'єднання та особливості переданих даних на транспортному рівні:

– swin – оголошення вікна TCP джерела;
– dwin – оголошення вікна TCP призначення;
– stcpb – номер послідовності TCP джерела;
– dtcpb – номер послідовності TCP призначення;
– smeansz – середній розмір пакета, переданого джерелом;

– dmeansz – середній розмір пакета, переданого призначенням;

– trans_depth – глибина або ж кількість запитів/відповідей HTTP у межах одного з'єднання;

– res_bdy_len – довжина тіла відповіді HTTP;

г) атрибути часу – характеризують часові аспекти з'єднань:

– sjit – джиттер з боку джерела (варіація часу між послідовними пакетами, відправленими джерелом);

– djit – джиттер з боку призначення;

– stime – час початку з'єднання;

– ltime – час завершення з'єднання;

– sintpkt – середній інтервал між пакетами з боку джерела;

– dintpkt – середній інтервал між пакетами з боку призначення;

– tcprtt – час затримки TCP (TCP Round Trip Time);

– synack – час між SYN та SYN_ACK пакетами;

– ackdat – час між SYN_ACK та ACK пакетами;

д) додаткові атрибути – були згенеровані на основі атрибутів базових характеристик, часу та вмісту, та їх можна поділити на дві категорії: універсальні атрибути та атрибути з'єднання. Перша категорія атрибутів слугують для загального аналізу мережевих характеристик і допомагають виявляти аномальні сигнатури. Наприклад, атрибут is_sm_ips_ports визначає, чи є джерело та призначення однаковими IP-адресами та портами, що може сигналізувати про можливі спроби атаки типу «IP-спуфінг». Атрибути з'єднання, з іншого боку, націлені на детальний аналіз активності в межах 100 послідовних з'єднань, що дозволяє відстежувати поведінкові закономірності. Атрибути ct_srv_src та ct_srv_dst допомагають оцінити, як часто певний сервіс використовується одним джерелом або призначенням, що може бути ознакою сканування портів або інших атак перерахування від зловмисника;

е) марковані атрибути – є поєднанням записів про запит зі звітом інструменту IXIA, що дозволяє визначити легітимність запиту та категорію атаки, якщо запит є частиною атаки:

- attack_cat – категорія атаки;
- label – чи є запит не легітимним.

Збагачення набору даних шляхом використання звіту покращує якість набору даних, адже відсутність маркованих атрибутів ускладнювала б його використання. На основі цього, можна визначити всі типи атак котрі використовуються в наборі даних:

- Fuzzers – тип атаки спрямований на призупинення процесів, сервісів та програмних застосунків шляхом надання випадково згенерованих даних;
- Analysis – тип атаки спрямований на збір детальної інформації про цільову систему або мережу для подальшого використання в більш складних атаках;
- Backdoors – тип атаки, котрий використовує вже знайдені варіанти обходу механізмів безпеки для отримання несанкціонованого доступу до комп'ютера або мережі;
- DoS – атаки типу відмова в обслуговуванні спрямовані на тимчасове або постійне переривання роботи сервера чи мережевого ресурсу;
- Exploits – тип атаки перерахуванням відомих вразливостей в будь-яких системах чи програмних забезпеченнях для подальшого несанкціонованого доступу;
- Generic – тип атаки спрямований проти блочних шифрів зі заданими розмірами блоку та ключа і мають на меті злам криптографічних алгоритмів;
- Shellcode – атаки типу шеллкод, є запити з невеликим фрагментом виконуваного коду, які використовуються для експлуатації вразливостей програмного забезпечення чи систем;

– Worms – тип атаки що самостійно поширюється на інші комп'ютери без втручання користувача, для цього, шкідливі програми, використовують мережі для розповсюдження.

Оглянувши структуру набору даних, перед початком навчання слід підготувати обраний набір даних.

3.3 Підготовка набору даних

Після завантаження набору даних та відкриття в середі програмування, набір має наступний вигляд (рис. 3.3).

```
> print(data)
            id      dur  proto service  state spkts dpkts sbytes dbytes        rate  sttl  dttl        sload      dload sloss dloss    sinpkt    dinpkt        sjit
         <int>    <num> <char>  <char> <char> <int> <int>  <int>  <int>       <num> <int> <int>        <num>      <num> <int> <int>     <num>     <num>       <num>
     1:      1 0.121478    tcp       -    FIN     6     4    258    172    74.08749   252   254    14158.942   8495.365     0     0  24.29560   8.37500    30.17755
     2:      2 0.649902    tcp       -    FIN    14    38    734  42014    78.47337    62   252     8395.112 503571.312     2    17  49.91500  15.43286    61.42693
     3:      3 1.623129    tcp       -    FIN     8    16    364  13186    14.17016    62   252     1572.272  60929.230     1     6 231.87557 102.73720 17179.58686
     4:      4 1.681642    tcp     ftp    FIN    12    12    628    770    13.67711    62   252     2740.179   3358.622     1     3 152.87655  90.23573   259.08017
     5:      5 0.449454    tcp       -    FIN    10     6    534    268    33.37383   254   252     8561.499   3987.060     2     1  47.75033  75.65960  2415.83763
    ---
175337: 175337 0.000009    udp     dns    INT     2     0    114      0 111111.10720   254     0 50666664.000      0.000     0     0   0.00900   0.00000     0.00000
175338: 175338 0.505762    tcp       -    FIN    10     8    620    354    33.61265   254   252     8826.286   4903.492     2     1  54.40011  66.98057  3721.06879
175339: 175339 0.000009    udp     dns    INT     2     0    114      0 111111.10720   254     0 50666664.000      0.000     0     0   0.00900   0.00000     0.00000
175340: 175340 0.000009    udp     dns    INT     2     0    114      0 111111.10720   254     0 50666664.000      0.000     0     0   0.00900   0.00000     0.00000
175341: 175341 0.000009    udp     dns    INT     2     0    114      0 111111.10720   254     0 50666664.000      0.000     0     0   0.00900   0.00000     0.00000
             djit  swin        stcpb        dtcpb  dwin    tcprtt    synack    ackdat smean dmean trans_depth response_body_len ct_srv_src ct_state_ttl ct_dst_ltm
            <num> <int>        <i64>        <i64> <int>     <num>     <num>     <num> <int> <int>       <int>            <int>      <int>        <int>      <int>
     1:    11.8306   255 3.071965e-315 1.088196e-314   255 0.000000 0.000000 0.000000    43    43           0                0          1            0          1
     2:  1387.7783   255 7.005278e-315 1.520432e-314   255 0.000000 0.000000 0.000000    52  1106           0                0         43            1          1
     3: 11420.9262   255 1.045517e-314 1.463973e-314   255 0.111897 0.061458 0.050439    46   824           0                0          7            1          2
     4:  4991.7847   255 5.469896e-315 5.175055e-315   255 0.000000 0.000000 0.000000    52    64           0                0          1            1          2
     5:   115.8070   255 1.203612e-314 9.768440e-315   255 0.128381 0.071147 0.057234    53    45           0                0         43            1          2
    ---
175337:     0.0000     0  0.000000e+00  0.000000e+00     0 0.000000 0.000000 0.000000    57     0           0                0         24            2         24
175338:   120.1777   255 1.738506e-314 1.706054e-314   255 0.099440 0.036895 0.062545    62    44           0                0          1            1          1
175339:     0.0000     0  0.000000e+00  0.000000e+00     0 0.000000 0.000000 0.000000    57     0           0                0         12            2          3
175340:     0.0000     0  0.000000e+00  0.000000e+00     0 0.000000 0.000000 0.000000    57     0           0                0         30            2         30
175341:     0.0000     0  0.000000e+00  0.000000e+00     0 0.000000 0.000000 0.000000    57     0           0                0         30            2         30
        ct_src_dport_ltm ct_dst_sport_ltm ct_dst_src_ltm is_ftp_login ct_ftp_cmd ct_flw_http_mthd ct_src_ltm ct_srv_dst is_sm_ips_ports attack_cat label
                   <int>           <int>          <int>        <int>      <int>           <int>      <int>      <int>           <int>     <char> <int>
     1:                1               1              1            0          0               0          1          1               0     Normal     0
     2:                1               1              2            0          0               0          1          6               0     Normal     0
     3:                1               1              3            0          0               0          2          6               0     Normal     0
     4:                1               1              3            1          1               0          2          1               0     Normal     0
     5:                2               1             40            0          0               0          2         39               0     Normal     0
    ---
175337:               24              13             24            0          0               0         24         24               0    Generic     1
175338:                1               1              2            0          0               0          1          1               0 Shellcode     1
175339:                3               3             13            0          0               0          3         12               0    Generic     1
175340:               30              14             30            0          0               0         30         30               0    Generic     1
175341:               30              16             30            0          0               0         30         30               0    Generic     1
```

Рисунок 3.3 – Вигляд завантаженого набору даних

Набір даних містить 175341 рядків та 45 колонок. Атрибут label є бінарним, та демонструю категорію атак на нормального трафіку. Використовуючи його, можна зобразити кількість зобразити кількість легітимних пакетів, та нелегітимних, на основі колонки label (рис. 3.4). Набір даних містить 68% пакетів котрі є частинами атак.

За допомогою атрибуту attack_cat, можливо зобразити кількість пакетів кожної категорії атак (рис. 3.5).

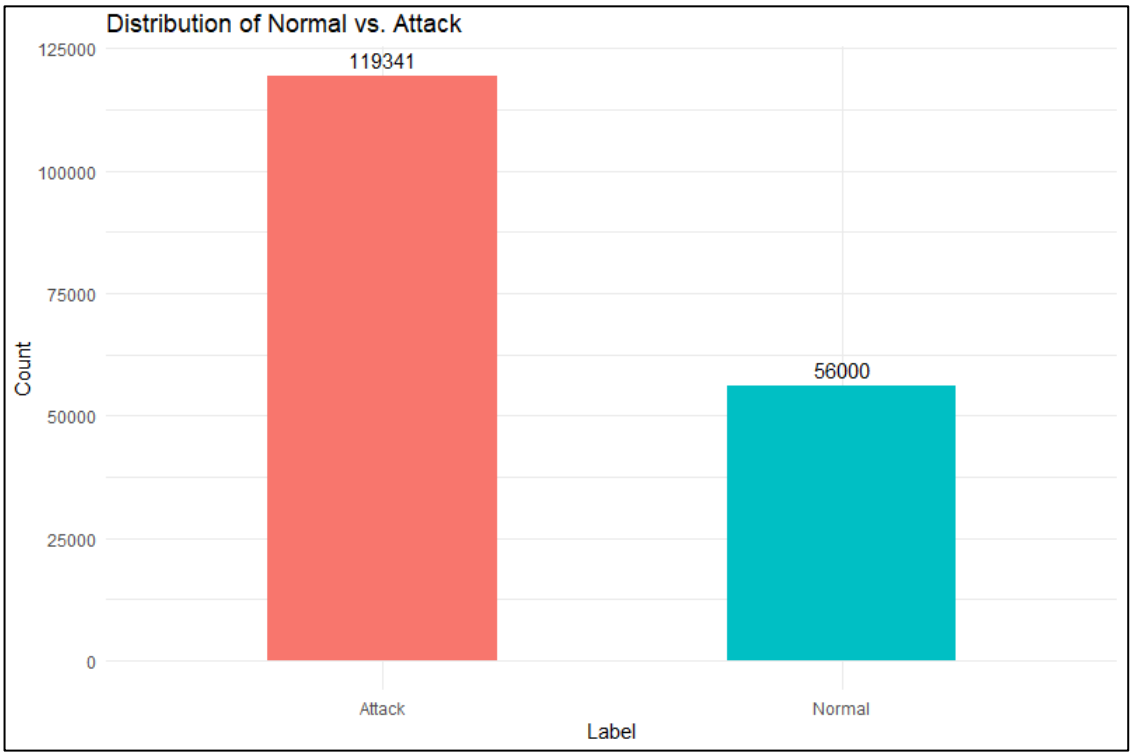

Рисунок 3.4 – Зображення кількості пакетів з категорією атак та нормального трафіку

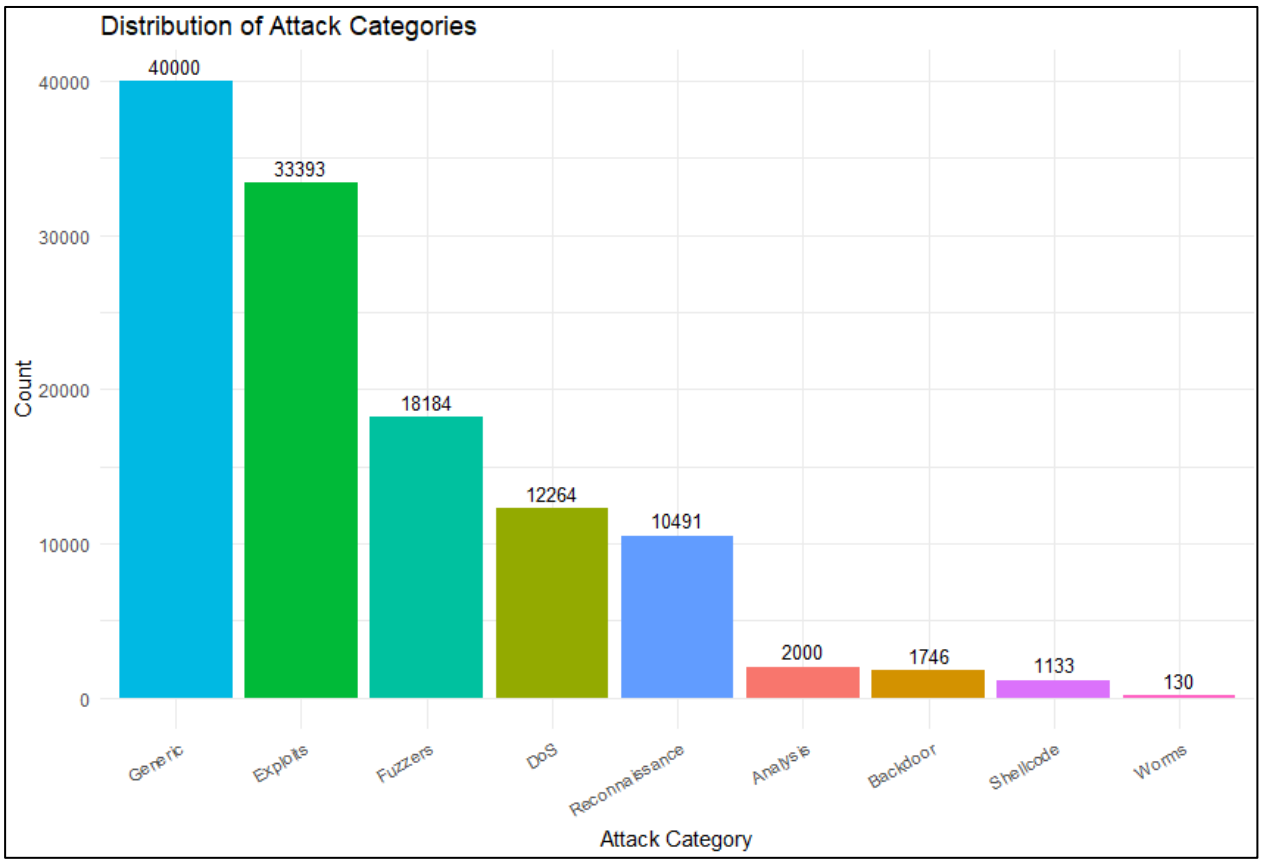

Рисунок 3.5 – Зображення кількості пакетів з категоріями атак

Аналізуючі отриману інформацію, можна помітити, що перші три категорії атак – «Generic», «Exploits» та «Fuzzers» – є 76.7% всіх атак в наборі даних. Це пов’язано з тим, що головна характеристика цих атак є генерація великої кількості запитів для перерахування або ж експлуатації. З іншого боку, три останні категорії – «Worms», «Shellcode» та «Backdoor» – є 2.5% від загальної кількості атак, хоча потенційна небезпека від них є більше ніж від перших трьох.

Під час подальшого аналізу набору даних, було виявлено що стовпець service містить нульові значення (рис. 3.6). Відсутність інформації в певних рядках можна пояснити тим, що деякі дії в мережі неможливо точно пов'язати з сервісом через шифрування, нестандартну поведінку або обмеження збору даних. Це підкреслює важливість обробки відсутніх даних для забезпечення точного моделювання та аналізу. Після видалення, данні підготовленні для нормалізації.

```
> print(missing_values)
                id                dur              proto            service              state
                 0                  0                  0              94168                  0
             spkts              dpkts             sbytes             dbytes               rate
                 0                  0                  0                  0                  0
             sttl               dttl              sload              dload              sloss
                 0                  0                  0                  0                  0
            dloss             sinpkt             dinpkt               sjit               djit
                 0                  0                  0                  0                  0
             swin               stcpb              dtcpb               dwin             tcprtt
                 0                  0                  0                  0                  0
           synack             ackdat              smean              dmean        trans_depth
                 0                  0                  0                  0                  0
response_body_len        ct_srv_src      ct_state_ttl        ct_dst_ltm  ct_src_dport_ltm
                 0                  0                  0                  0                  0
 ct_dst_sport_ltm    ct_dst_src_ltm       is_ftp_login        ct_ftp_cmd  ct_flw_http_mthd
                 0                  0                  0                  0                  0
       ct_src_ltm        ct_srv_dst    is_sm_ips_ports        attack_cat              label
                 0                  0                  0                  0                  0
```

Рисунок 3.6 – Відсутність даних в колонках

Нормалізація даних є критичним етапом підготовки набору даних, оскільки вона дозволяє узгодити різні числові ознаки за масштабом. Це особливо важливо для алгоритмів машинного навчання, які чутливо реагують на різницю в значеннях між категоріями. Без нормалізації, категорії з великими числовими значеннями можуть непропорційно впливати на результати моделі, що призводить до хибних висновків.

Для категоріальних ознак існують різні методи нормалізації, котрі обираються від значення цих ознак. Обраний набір даних містить наступні категоріальні ознаки: proto (протокол), service (сервіс), state (стан з'єднання) та attack_cat (категорія атаки), їхні унікальні значення зображенні на рис. 3.7.

```
> for (var in categorical_vars) {
+   cat("\n---", var, "---\n")
+   print(unique(data[[var]]))
+ }

--- proto ---
[1] "tcp" "udp"

--- service ---
 [1] "ftp"      "smtp"     "snmp"     "http"     "ftp-data" "dns"      "ssh"      "radius"   "pop3"     "dhcp"
[11] "ssl"      "irc"

--- state ---
[1] "FIN" "INT" "CON" "RST" "REQ"

--- attack_cat ---
[1] "Normal"         "Backdoor"       "Fuzzers"        "Reconnaissance" "Exploits"       "Analysis"
[7] "DoS"            "Worms"          "Generic"
```

Рисунок 3.7 – Зображення унікальних значень категоріальних ознак

Для цих ознак було використано метод one-hot кодування, що дозволяє перетворювати категоріальні змінні на бінарні індикатори. Кожна унікальна категорія після використання one-hot методу має окрему колонку, що містить значення 1, якщо зразок належить до цієї категорії, або 0 в іншому випадку (рис. 3.8). Використовуючи цей підхід, можна уникнути проблеми впорядкованості, що може виникати при прямому числовому кодуванні категорій, і забезпечує, що модель не інтерпретує певні категорії як більш важливі.

	proto.tcp <num>	proto.udp <num>	service.dhcp <num>	service.dns <num>	service.ftp <num>	service.ftp-data <num>	service.http <num>	service.irc <num>	service.pop3 <num>
1:	1	0	0	0	1	0	0	0	0
2:	1	0	0	0	0	0	0	0	0
3:	0	1	0	0	0	0	0	0	0
4:	1	0	0	0	0	0	1	0	0
5:	1	0	0	0	0	0	1	0	0

81169:	0	1	0	1	0	0	0	0	0
81170:	0	1	0	1	0	0	0	0	0
81171:	0	1	0	1	0	0	0	0	0
81172:	0	1	0	1	0	0	0	0	0
81173:	0	1	0	1	0	0	0	0	0

	service.radius <num>	service.smtp <num>	service.snmp <num>	service.ssh <num>	service.ssl <num>	state.CON <num>	state.FIN <num>	state.INT <num>	state.REQ <num>	state.RST <num>
1:	0	0	0	0	0	0	1	0	0	0
2:	0	1	0	0	0	0	1	0	0	0
3:	0	0	1	0	0	0	0	1	0	0
4:	0	0	0	0	0	0	1	0	0	0
5:	0	0	0	0	0	0	1	0	0	0

81169:	0	0	0	0	0	0	0	1	0	0
81170:	0	0	0	0	0	0	0	1	0	0
81171:	0	0	0	0	0	0	0	1	0	0
81172:	0	0	0	0	0	0	0	1	0	0
81173:	0	0	0	0	0	0	0	1	0	0

Рисунок 3.8 – Результат перетворення категоріальних ознак

Для інших числових ознак буде використовуватись один із найпоширеніших методів нормалізації мін-макс. Цей метод нормалізації полягає в лінійному перетворенні значень змінної таким чином, щоб вони були масштабовані до певного діапазону, зазвичай від 0 до 1. Формула для нормалізації кожного значення змінної виглядає наступним чином:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3.1)$$

де X – оригінальне значення змінної;

X_{min} – мінімальне значення змінної в наборі даних;

X_{max} – максимальне значення змінної в наборі даних;

X_{norm} – нормалізоване значення змінної.

Результат використання методу мін-макс зображено на рис. 3.9.

	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload
	<num>	<num>	<num>	<num>	<num>	<num>	<num>	<num>	<num>
1:	2.802780e-02	0.0011440458	0.0010934937	4.072461e-05	5.253982e-05	1.367711e-05	0.1383929	0.992126	1.189314e-06
2:	3.488529e-02	0.0063442538	0.0025514853	4.336940e-03	1.509326e-04	4.252097e-05	0.1383929	0.992126	9.193799e-05
3:	1.666693e-08	0.0001040042	0.0000000000	2.930938e-06	0.000000e+00	5.000000e-01	0.9955357	0.000000	1.197917e-01
4:	6.559354e-03	0.0009360374	0.0007289958	5.861876e-05	7.478396e-05	4.319589e-05	0.1383929	0.992126	6.828761e-06
5:	5.633690e-03	0.0009360374	0.0005467469	6.926269e-05	1.828659e-05	4.437647e-05	0.9955357	0.992126	9.234825e-06

81169:	8.333466e-08	0.0001040042	0.0000000000	1.079819e-06	0.000000e+00	1.666667e-01	0.9955357	0.000000	3.298611e-02
81170:	1.333355e-07	0.0001040042	0.0000000000	1.079819e-06	0.000000e+00	1.111111e-01	0.9955357	0.000000	2.199074e-02
81171:	1.333355e-07	0.0001040042	0.0000000000	1.079819e-06	0.000000e+00	1.111111e-01	0.9955357	0.000000	2.199074e-02
81172:	1.333355e-07	0.0001040042	0.0000000000	1.079819e-06	0.000000e+00	1.111111e-01	0.9955357	0.000000	2.199074e-02
81173:	1.333355e-07	0.0001040042	0.0000000000	1.079819e-06	0.000000e+00	1.111111e-01	0.9955357	0.000000	2.199074e-02

	dload	sloss	dloss	sinpkt	dinpkt	sjit	djit	swin	stcpb
	<num>	<num>	<num>	<num>	<num>	<num>	<num>	<num>	<i64>
1:	0.0001497865	0.0002082032	0.0005470460	1.092583e-02	0.006448963	0.0002752496	0.0172494368	1	5.469896e-315
2:	0.0003635846	0.0058296898	0.0014587892	2.452283e-03	0.005366704	0.0034563189	0.0003666819	1	9.015328e-315
3:	0.0000000000	0.0000000000	0.0000000000	1.429367e-07	0.000000000	0.0000000000	0.0000000000	0	0.000000e+00
4:	0.0008693878	0.0004164064	0.0003646973	3.125199e-03	0.003406816	0.0022574509	0.0002481298	1	1.918443e-314
5:	0.0002364346	0.0004164064	0.0001823487	2.516980e-03	0.004043643	0.0020489133	0.0002834333	1	1.317166e-314

81169:	0.0000000000	0.0000000000	0.0000000000	4.288100e-07	0.000000000	0.0000000000	0.0000000000	0	0.000000e+00
81170:	0.0000000000	0.0000000000	0.0000000000	6.432150e-07	0.000000000	0.0000000000	0.0000000000	0	0.000000e+00
81171:	0.0000000000	0.0000000000	0.0000000000	6.432150e-07	0.000000000	0.0000000000	0.0000000000	0	0.000000e+00
81172:	0.0000000000	0.0000000000	0.0000000000	6.432150e-07	0.000000000	0.0000000000	0.0000000000	0	0.000000e+00
81173:	0.0000000000	0.0000000000	0.0000000000	6.432150e-07	0.000000000	0.0000000000	0.0000000000	0	0.000000e+00

	dtcpb	dwin	tcprtt	synack	ackdat	smean	dmean	trans_depth	response_body_len
	<i64>	<num>	<num>	<num>	<num>	<num>	<num>	<num>	<num>
1:	5.175055e-315	1	0.00000000	0.0000000	0.00000000	0.006777108	0.04389575	0.000000000	0.000000e+00
2:	4.252506e-315	1	0.08797198	0.1030002	0.05497517	0.652108434	0.05418381	0.000000000	0.000000e+00
3:	0.000000e+00	0	0.00000000	0.0000000	0.00000000	0.019578313	0.00000000	0.000000000	0.000000e+00
4:	1.523734e-314	1	0.08070895	0.1153734	0.04291389	0.032379518	0.09396433	0.005813953	1.570587e-05
5:	1.739784e-314	1	0.10359750	0.1073988	0.06974701	0.042921687	0.03086420	0.005813953	0.000000e+00

81169:	0.000000e+00	0	0.00000000	0.0000000	0.00000000	0.010542169	0.00000000	0.000000000	0.000000e+00
81170:	0.000000e+00	0	0.00000000	0.0000000	0.00000000	0.010542169	0.00000000	0.000000000	0.000000e+00
81171:	0.000000e+00	0	0.00000000	0.0000000	0.00000000	0.010542169	0.00000000	0.000000000	0.000000e+00
81172:	0.000000e+00	0	0.00000000	0.0000000	0.00000000	0.010542169	0.00000000	0.000000000	0.000000e+00
81173:	0.000000e+00	0	0.00000000	0.0000000	0.00000000	0.010542169	0.00000000	0.000000000	0.000000e+00

Рисунок 3.9 – Результат перетворення числових ознак

Після підготовки даних, було почато відбір ознак котрі будуть використовуватись для класифікації. З цією метою було проаналізовано кореляційні взаємозв'язки між ознаками та цільовою змінною.

Створена матриця кореляції містить коефіцієнти кореляції Пірсона між усіма парами числових ознак у наборі даних. Коефіцієнти кореляції варіюються від -1 до 1, де значення, близькі до 1 або -1, вказуються на сильний лінійний зв'язок між ознаками, а значення близькі до 0 – на слабкий або відсутній (рис. 3.10).

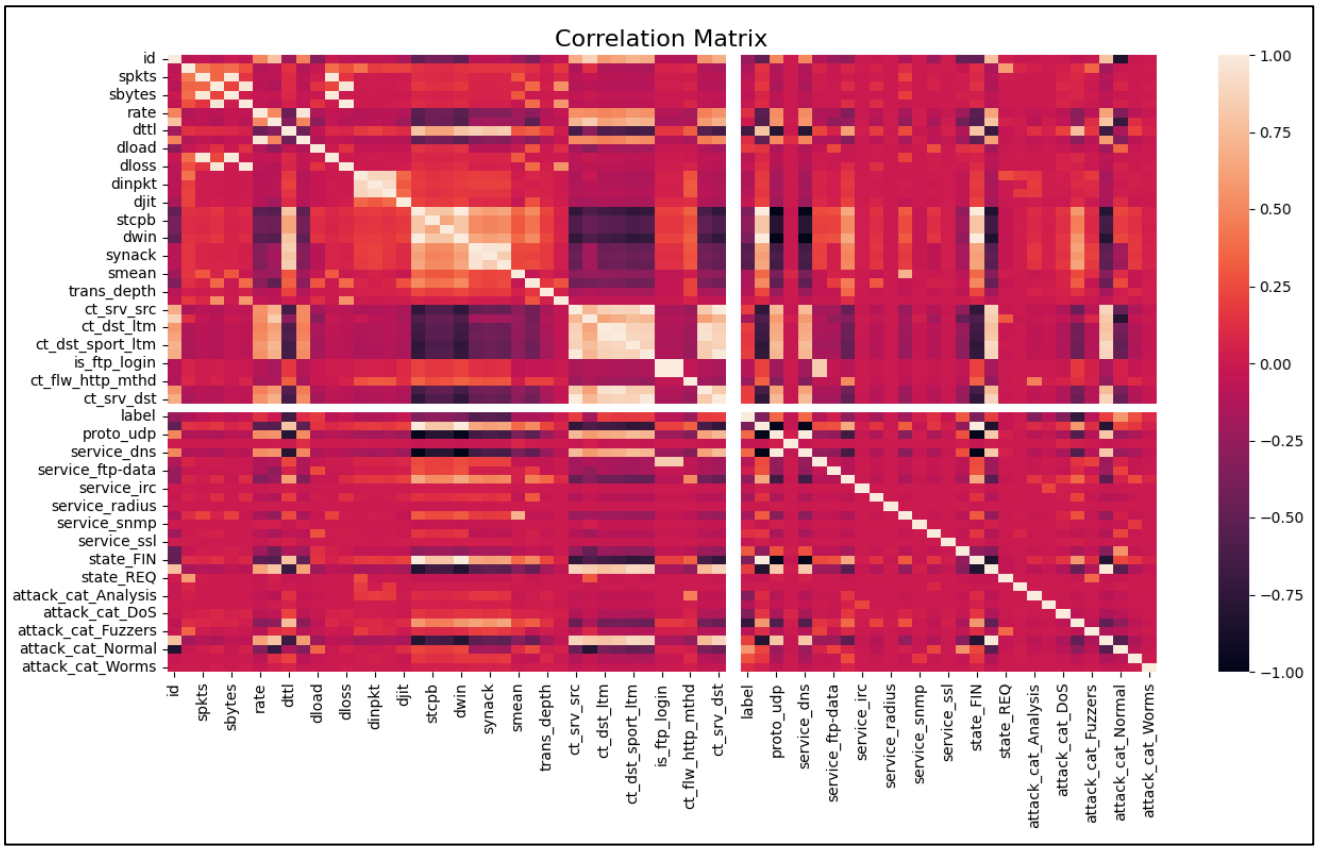

Рисунок 3.10 – Матриця кореляції

Обчислення абсолютних значень коефіцієнтів кореляції між усіма ознаками та цільовою змінною label дозволяє враховувати як позитивні так і негативні кореляції фокусуючись на силі зв'язку без врахування його напрямку, що підвищує інформативність для моделі. Встановивши поріг абсолютної кореляції 0.25 дозволяє досягати оптимального балансу між кількістю інформативних ознак та складністю моделі. Крім того, такий підхід забезпечує кращу здатність

узагальнення моделі на нових даних, оскільки вона фокусується на найбільш релевантних ознаках. Відібрані ознаки відображені на рис. 3.11.

	Feature	Value
0	dttl	0.646589
1	swin	0.364393
2	dwin	0.364393
3	tcprtt	0.570205
4	synack	0.524027
5	ackdat	0.570098
6	label	1.000000
7	proto_tcp	0.364393
8	proto_udp	0.364393
9	service_dns	0.365346
10	state_CON	0.302853
11	state_FIN	0.361750
12	attack_cat_Analysis	0.326209
13	attack_cat_DoS	0.339669
14	attack_cat_Exploits	0.719733
15	attack_cat_Normal	0.570858

Рисунок 3.11 – Відібрані ознаки на основі порогу абсолютної кореляції

Відібрані ознаки поєднанні в набір даних та готові для використання у навчанні моделей.

Висновки до розділу 3

У третьому розділі було детально розглянуто процес обрання, структурування та підготовки набору даних для реалізації моделей машинного навчання з метою виявлення мережевих атак. Підкреслено важливість якісного та актуального набору даних, адже його відповідність сучасним загрозам безпосередньо впливає на ефективність та точність моделювання.

Аналіз існуючих наборів даних, таких як KDD Cup 99, NSL-KDD та DARPA 1998, виявив їхні суттєві обмеження, пов'язані із застарілістю та невідповідністю

сучасним мережевим атакам. У зв'язку з цим, було обрано набір даних UNSW-NB15, який відображає новітні типи атак та реальні мережеві умови.

Розглянуто інфраструктуру котра була реалізована для генерації мережевої активності структуру набору даних UNSW-NB15 та архітектуру фреймворку для отримання результатів.

Набір даних містить 49 атрибутів, що детально описують кожне мережеве з'єднання. Атрибути були поділені на категорії: характеристики потоку, базові характеристики, атрибути вмісту, часові атрибути, додаткові атрибути та марковані атрибути. Особливу увагу приділено маркованим атрибутам «attack_cat» та «label», які дозволяють визначити легітимність запиту та категорію атаки.

У процесі підготовки даних було проведено аналіз розподілу атак, що виявив дисбаланс між різними категоріями атак. Зокрема, три категорії: «Generic», «Exploits» та «Fuzzers» – складають понад 76% усіх атак, тоді як «Worms», «Shellcode» та «Backdoor» – лише 2.5%.

Виявлено наявність відсутніх значень у стовпці service, що могло негативно вплинути на результати моделі. Для забезпечення точності моделювання були видалені рядки з відсутніми значеннями та проведено обробку даних.

Для категоріальних ознак (proto, service, state) було застосовано кодування унітарним кодом (one-hot encoding), що дозволило перетворити їх у числовий формат, придатний для моделей машинного навчання. Стовпець attack_cat свідомо не був включений у процес нормалізації та моделювання, щоб уникнути витоку даних та штучного підвищення точності моделі.

Проведені етапи підготовки даних забезпечили створення якісного та збалансованого набору даних, готового для подальшого використання в моделях машинного навчання. Такий підхід сприяє підвищенню точності виявлення мережевих атак та забезпечує надійну основу для подальших досліджень у галузі кібербезпеки.

4 РЕАЛІЗАЦІЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

Після завершення попередньої обробки даних, набір можна використовувати для навчання моделей. Для цього етапу буде використовуватись мова програмування Python та ряд спеціалізованих бібліотек, які забезпечують широкий спектр інструментів для аналізу даних, побудови моделей та візуалізації результатів. Нижче детально розглянуто бібліотеки та технології, які будуть застосовані у процесі моделювання.

4.1 Бібліотеки та технології для моделювання моделей

Мова програмування Python є одним із найпопулярніших інструментів у сфері машинного навчання та аналізу даних. Вона відзначається простим та зрозумілим синтаксисом та потужними можливостями для швидкого прототипування та експериментування.

Python підтримує об'єктно-орієнтоване, процедурне та функціональне програмування, що робить її гнучкою для різних типів задач. Проте, базова версія мови не містить вбудованих бібліотек для машинного навчання або складного аналізу даних. Ці можливості реалізуються через численні сторонні бібліотеки, розроблені спільнотою, які надають необхідний функціонал та оптимізовані алгоритми.

У контексті нашого дослідження, Python виступає як основний інструмент для реалізації моделей машинного навчання завдяки своїй сумісності з багатьма бібліотеками та фреймворками. Ця екосистема дозволяє ефективно виконувати повний цикл роботи з даними: від їхнього завантаження та попередньої обробки до побудови моделей та оцінки результатів.

NumPy – фундаментальна бібліотека для наукових обчислень у Python, яка забезпечує підтримку багатовимірних масивів та матриць, а також містить велику кількість високорівневих математичних функцій для роботи з ними. Бібліотека є основою для багатьох інших бібліотек машинного навчання та аналізу даних [52].

Pandas – потужна бібліотека для маніпулювання та аналізу даних, яка надає гнучкі структури даних, такі як Series та DataFrame. Вона дозволяє легко імпортувати дані з різних форматів, обробляти відсутні значення, трансформувати та очищати дані, а також виконувати складні операції групування та агрегування. Pandas є незамінною для попередньої обробки даних перед моделюванням, що включає перетворення типів даних, кодування категоріальних змінних та підготовку даних до подальшого аналізу [53].

Matplotlib – бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python. Вона надає широкий спектр інструментів для побудови різноманітних графіків, таких як лінійні графіки, гістограми, діаграми розсіювання, теплові карти та інші. Ця бібліотека допоможе візуалізувати кореляційної матриці та інші графічні представлень, що допомагають інтерпретувати дані та результати моделювання [54].

Seaborn – бібліотека для статистичної візуалізації, яка надає високорівневі інтерфейси для створення яскравих та інформативних графіків. Seaborn інтегрується з Pandas та Matplotlib, що дозволяє легко візуалізувати складні статистичні взаємозв'язки, такі як кореляційні матриці, розподіли даних, категоріальні діаграми тощо [55].

Scikit-learn – є провідною бібліотекою для машинного навчання в Python, яка містить широкий набір ефективних інструментів для статистичного моделювання та аналізу даних. Вона охоплює алгоритми класифікації, регресії, кластеризації, зниження розмірності та попередньої обробки даних. Scikit-learn відзначається єдиним стилем інтерфейсу для різних алгоритмів, що спрощує їх використання та порівняння [56]. Також буде використовуватись бібліотека Yellowbrick, яка розширює можливості scikit-learn, і надає інструменти для візуалізації процесу навчання моделей, їх продуктивності та помилок класифікації [57]. Після обрання моделей, можна чітко визначити які саме компоненти scikit-learn будуть використовуватись. Компоненти котрі будуть використовуватись, описанні нижче.

LogisticRegression – клас доступний у модулі linear_model і реалізує метод логістичної регресії для задач класифікації. Шляхом створення екземпляру, де можемо налаштувати параметри, такі як тип регуляризації, обернена сила регуляризації та алгоритм оптимізації, здійснюється навчання моделі. LogisticRegression підтримує багатокласову класифікацію через параметр multi_class, дозволяючи застосовувати стратегії, такі як «один проти решти» або мультинормальна логістична регресія.

KNeighborsClassifier – цей класифікатор знаходиться в модулі neighbors і реалізує алгоритм k-ближчих сусідів для класифікації. Шляхом ініціалізації KNeighborsClassifier, задаючи параметри, такі як кількість сусідів, що визначає, скільки сусідів враховувати при прогнозуванні, схема вагових коефіцієнтів, яка може бути рівномірною або залежати від відстані, та метрика для обчислення відстані між точками, відбувається навчання моделі. Ці параметри впливають на те, як модель вимірює схожість між зразками та приймає рішення про класифікацію.

RandomForestClassifier – модель імпортується з модуля ensemble і надає інструменти для побудови випадкового лісу, який є ансамблем дерев рішень. У коді ми ініціалізуємо екземпляр RandomForestClassifier, де можемо вказати різні гіперпараметри, такі як кількість дерев у лісі, максимальна глибина кожного дерева та випадковий стан для відтворюваності результатів. Після ініціалізації модель навчається на навчальній вибірці за допомогою методу fit, куди передаються матриця ознак та вектор міток.

DecisionTreeClassifier – модель імпортується з модуля tree та надає інструменти для побудови дерева рішень. Модель дозволяє налаштовувати різні параметри котрі впливають на поведінку та продуктивність моделі: функція для оцінки якості розбиття, максимальна глибина дерева, мінімальна кількість зразків тощо. Після ініціалізації модель навчається на навчальній вибірці за допомогою методу fit, куди передаються матриця ознак та вектор міток.

4.2 Навчання моделей

Після розділення набору на навчальний та тестовий, дані були надання для навчання методу Лінійної Регресії.

В процесі реалізації логістичної регресії для багатокласової класифікації, максимальна кількість ітерацій було встановлено у розмірі 2000, що шляхом аналізу було визначено як достатню кількість кроків для досягнення збіжності. Метод Ньютона використовується в параметрі solver та є ефективним для задач з великою кількістю ознак. Додаткове налаштування параметру multi_class використовує мультинормального підходу для багатоклаосові класифікації, дозволяючи моделі одночасно моделювати ймовірності належності зразків до кожного з кількох класів. Результатом є правильність класифікованих зразків 98.9% (рис. 4.1).

Metrics	Values
Mean Absolute Error	0.0172
Mean Squared Error	0.0394
Root Mean Squared Error	0.1984
Explained Variance Score (R^2)	97.3481
Accuracy	98.8995

Рисунок 4.1 – Точність класифікації методу Логістичної Регресії

Звіт класифікації демонструє що класи типу «Reconnaissance», «Normal», «Generic», «Exploits», «DoS» та «Analysis» були класифіковані майже ідеально. Клас «Fuzzers» має точність 0.71 і відгук 0.81 що призводить до оцінки f1 – 0.75 та правильно класифікує 81% випадків атак «Fuzzers» (рис. 4.2).

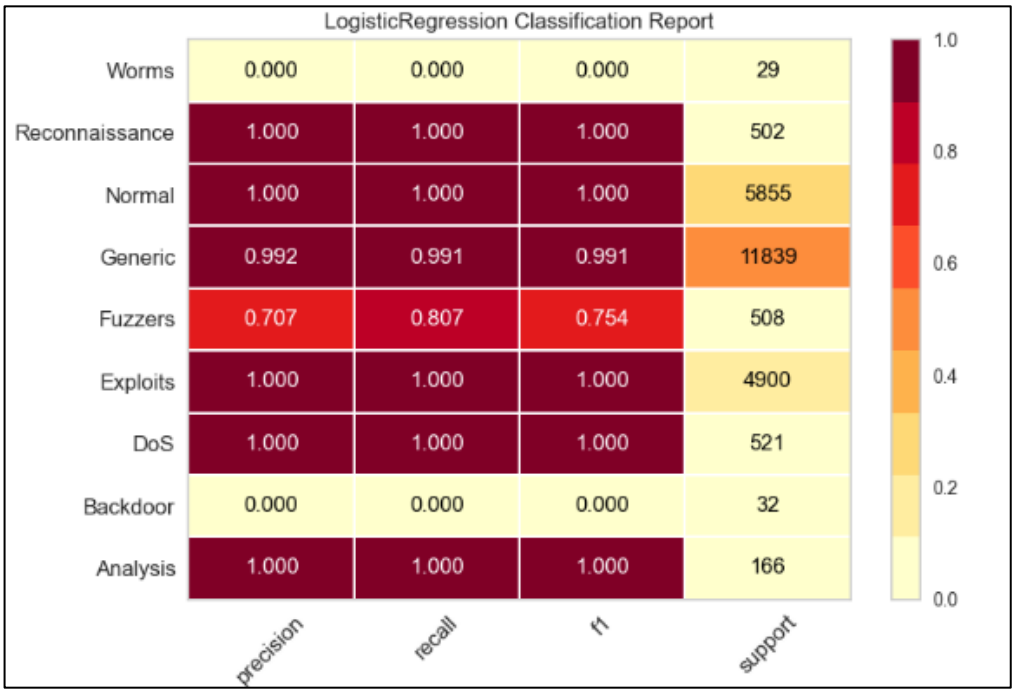

Рисунок 4.2 – Звіт класифікації методом Логістичної регресії

Матриця невідповідностей демонструє що клас «Fuzzers» має відгук в 81% через класифікацію частково як «Generic», та навпаки, певні данні класу «Generic» було визначено як клас «Fuzzers». Згідно звіту класифікації та матриці невідповідностей, класи «Backdoor» та «Worms» мають 0% точності. Як і з класом «Generic», вони були класифіковані як клас «Fuzzers» (рис. 4.3).

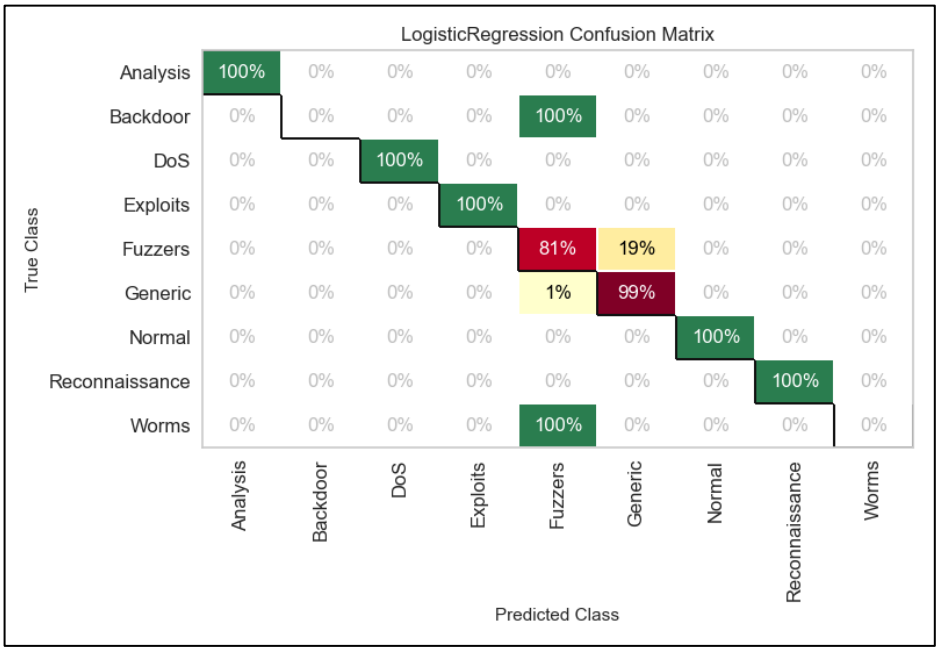

Рисунок 4.3 – Матриця невідповідностей методу Логістичної Регресії

Наступним методом машинного навчання було реалізовано метод k-найближчих сусідів.

В процесі реалізації було проаналізовано залежність точності від кількості найближчих сусідів (рис. 4.4).

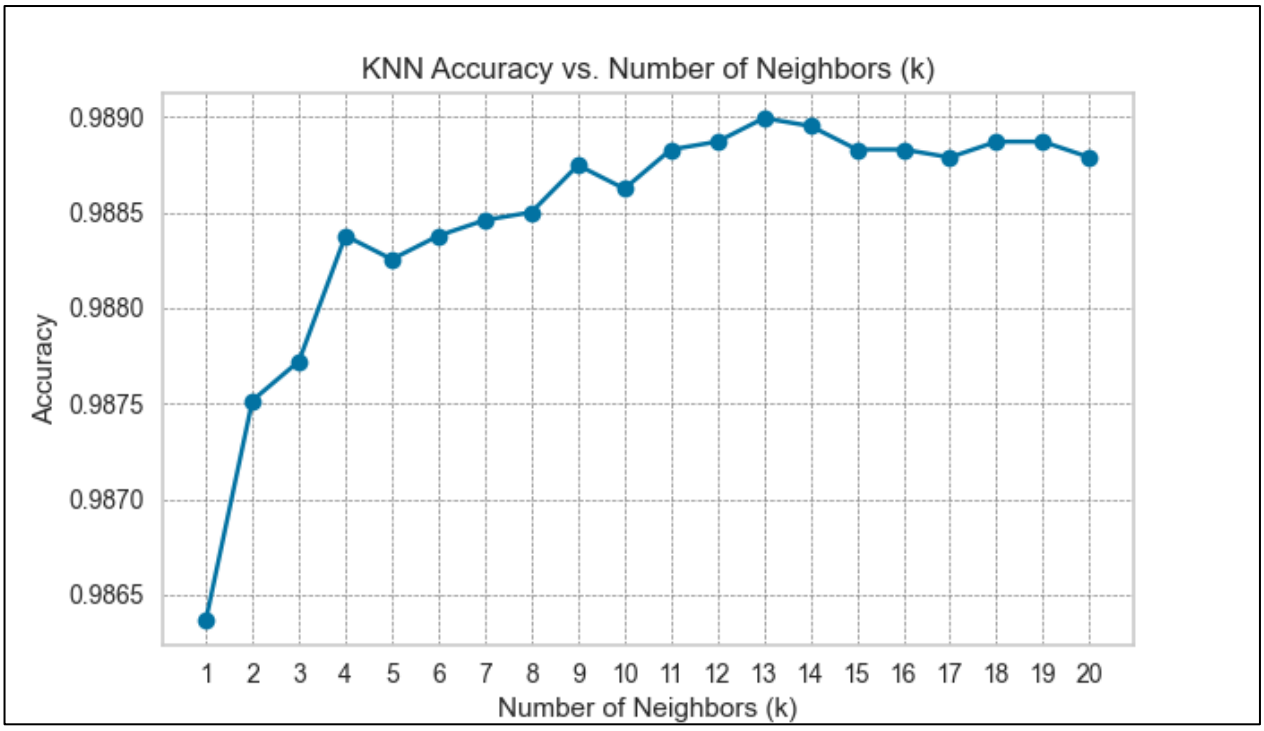

Рисунок 4.4 – Графік залежності точності від кількості k-найближчих сусідів

Згідно отриманих результатів точності, різниця між найгіршою на найкращою точністю із 20 використаних k є 0.25%.

Однак, проаналізувавши отримані матриці невідповідностей, можна визначити, що при використанні k=1, клас «Backdoor» має 6%, «Fuzzers» – 37%, «Reconnaissance» – 53% та «Worms» – 7% точності визначення (рис. 4.5)

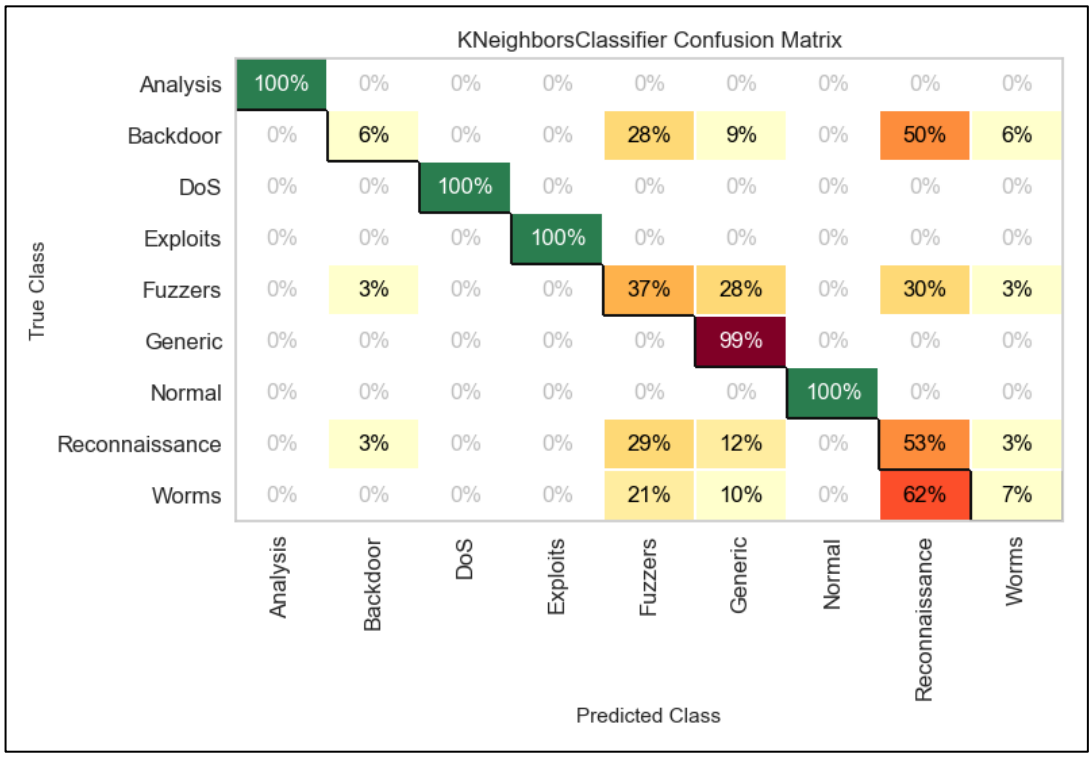

Рисунок 4.5 – Матриця невідповідностей методу KNN при k=1

При використанні k=4, точність класу «Backdoor» збільшується з 6% до 9%, «Fuzzers» – зростає з 37% до 73%, «Reconnaissance» – має 100% до 52% та «Worms» – залишається з 7% точності визначення (рис. 4.6).

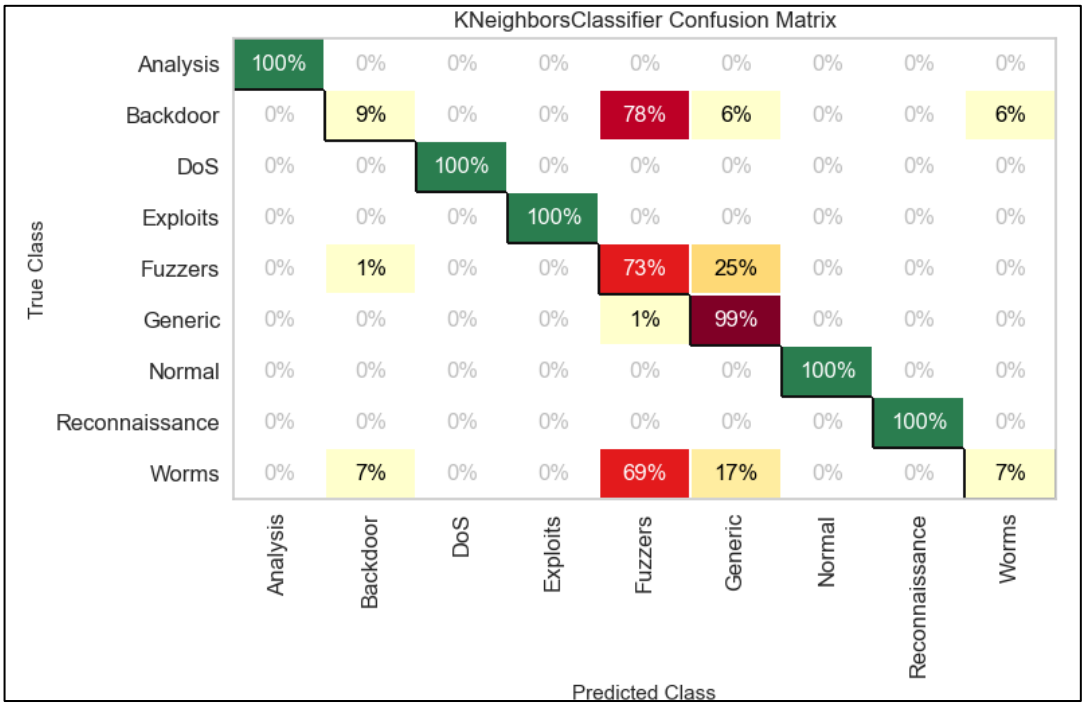

Рисунок 4.6 – Матриця невідповідностей методу KNN при k=4

При використання k=13 (найкраща точність з 20 проаналізованих), окрім зростання точності класу «Fuzzers» на 6% та 100% точності інших класів, класи «Backdoor» та «Worms» втрачають її та мають 0% (рис. 4.7).

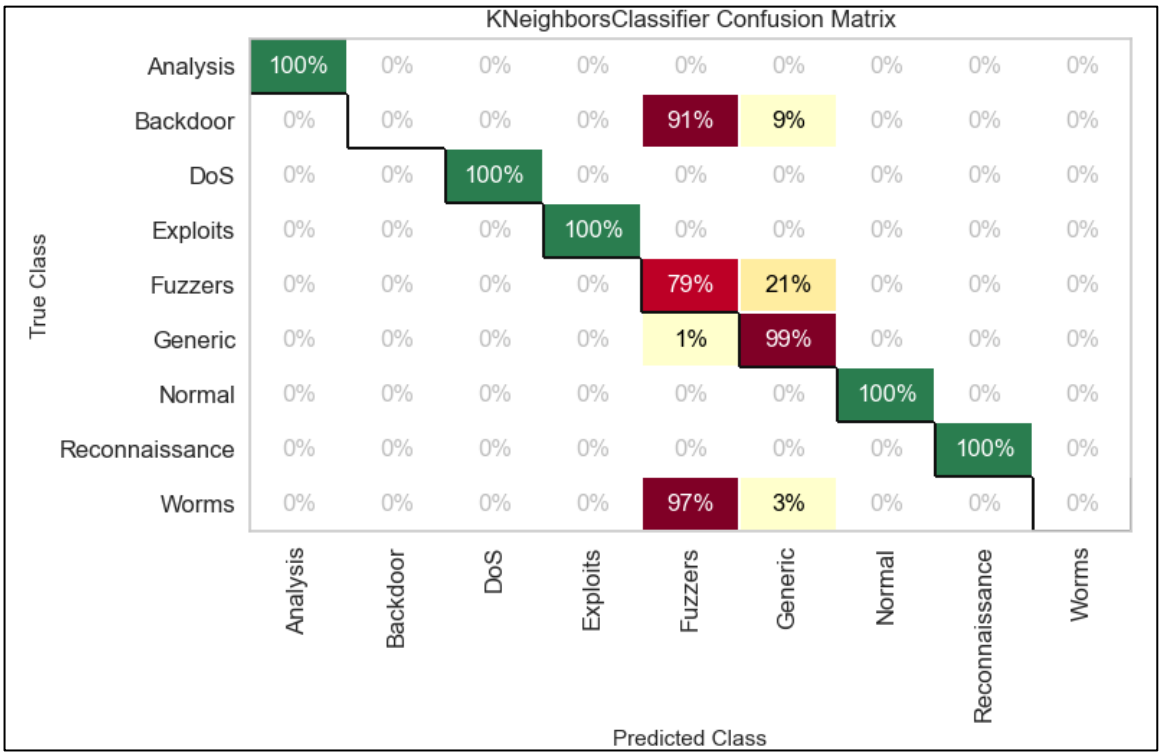

Рисунок 4.7 – Матриця невідповідностей методу KNN при k=13

На основі цих порівнянь, можна визначити, що незважаючи на покращення точності, збереження малого відсотку класифікацій класів «Worms» та «Backdoor» є пріоритетними за покращення класифікації класу «Fuzzers».

Результат використання методу k-найближчих сусідів становить правильність класифікованих зразків 98.9% (рис. 4.8).

Metrics	Values
Mean Absolute Error	0.0255
Mean Squared Error	0.0746
Root Mean Squared Error	0.2731
Explained Variance Score (R^2)	94.9799
Accuracy	98.6367

Рисунок 4.8 – Точність класифікації методу k найближчих сусідів

Звіт класифікації демонструє що класи типу «Reconnaissance», «Normal», «Generic», «Exploits», «DoS» та «Analysis» були класифіковані майже ідеально. Клас «Fuzzers» має точність 0.75 і відгук 0.73 що призводить до оцінки f1 – 0.74 та правильно класифікує 73% випадків атак «Fuzzers» (рис. 4.9). Класи «Worms» та «Backdoor» – 7% та 9% відповідно.

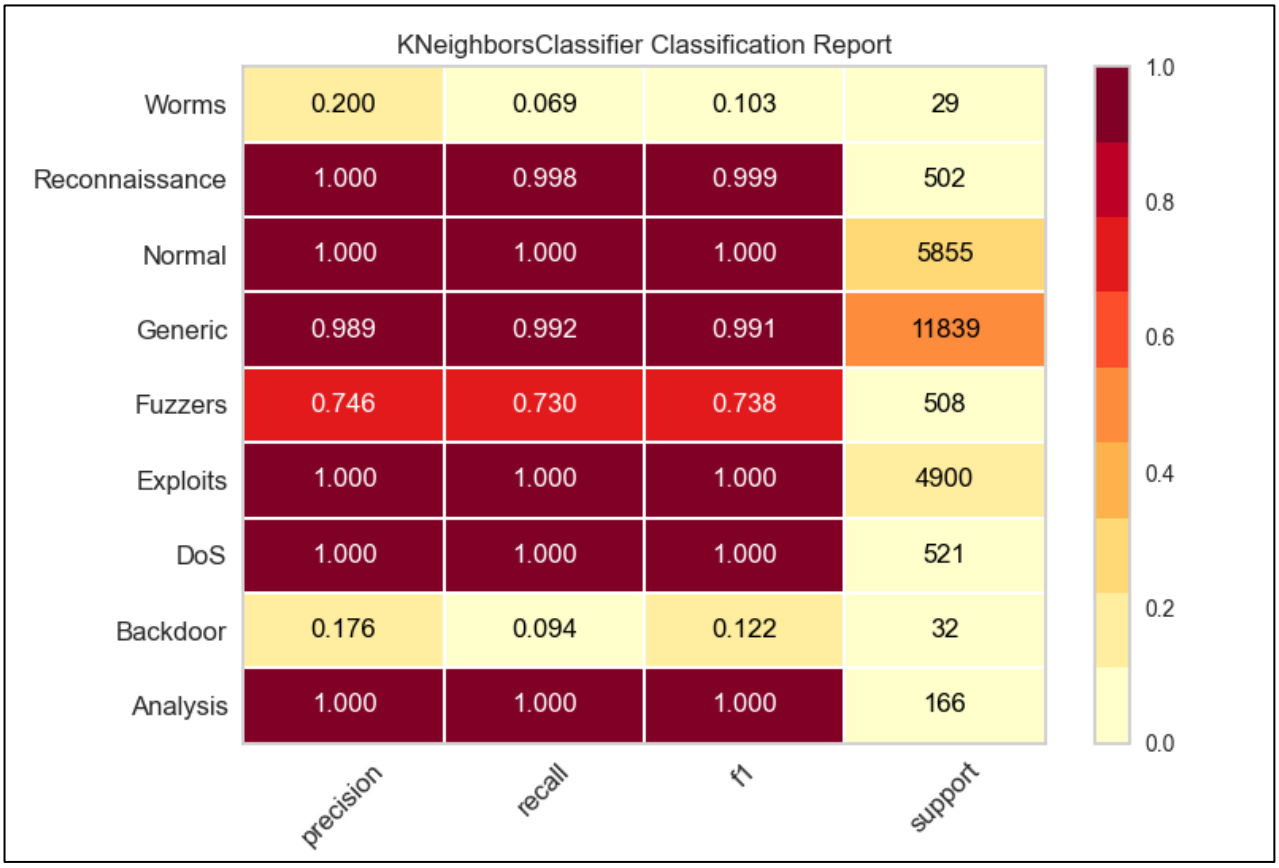

Рисунок 4.9 – Звіт класифікації методом k найближчих сусідів

Наступним методом машинного навчання було реалізовано метод Random Forest. Результатом є правильність класифікованих зразків 98.9% (рис. 4.10).

Metrics	Values
Mean Absolute Error	0.0172
Mean Squared Error	0.0394
Root Mean Squared Error	0.1984
Explained Variance Score (R^2)	97.3481
Accuracy	98.8995

Рисунок 4.10 – Точність класифікації методу Логістичної Регресії

Звіт класифікації демонструє що класи типу «Reconnaissance», «Normal», «Generic», «Exploits», «DoS» та «Analysis» були класифіковані майже ідеально. Клас «Fuzzers» має точність 0.76 і відгук 0.77 що призводить до оцінки f1 – 0.77 та правильно класифікує 77% випадків атак «Fuzzers». Правильність класифікації класів «Worms» та «Backdoor» є 17% та 3% відповідно (рис. 4.11).

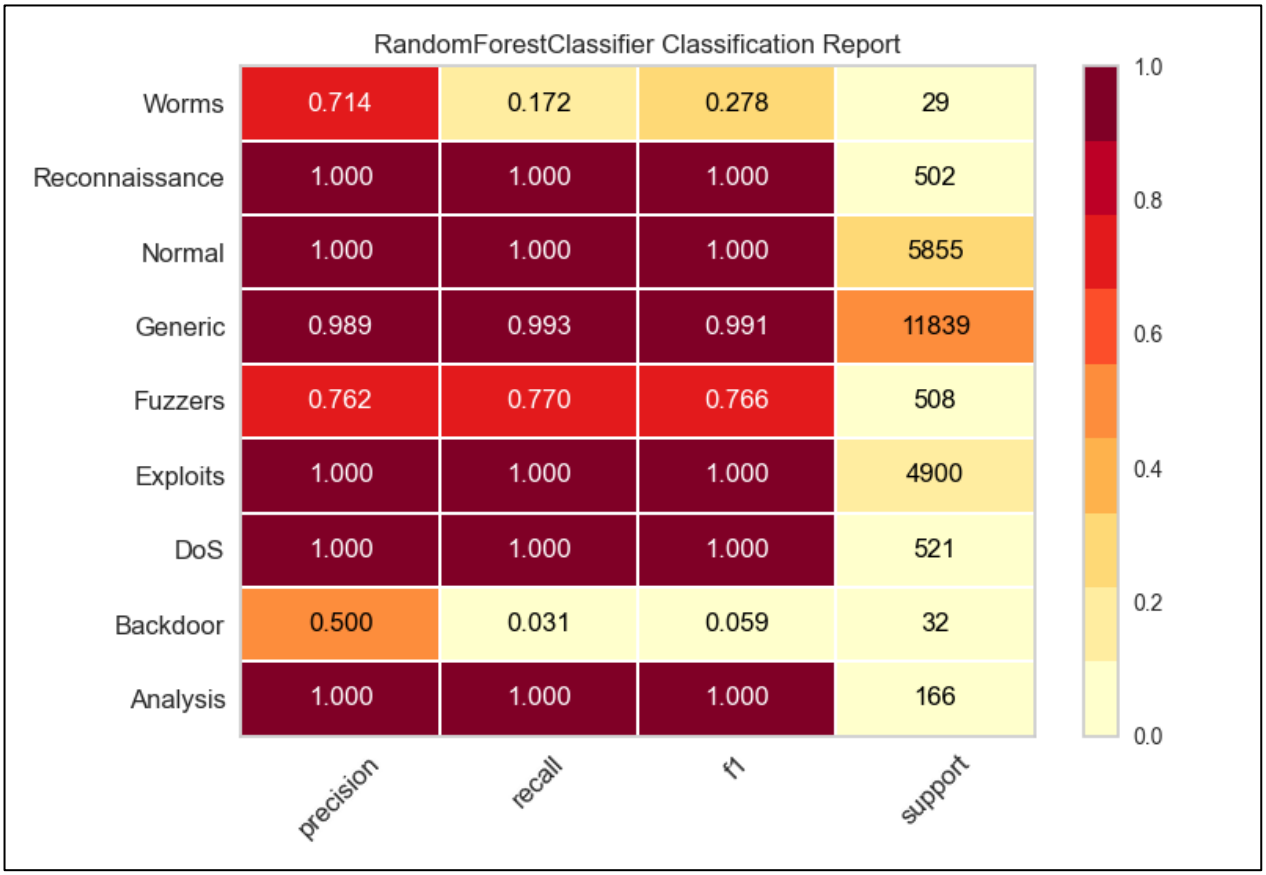

Рисунок 4.11 – Звіт класифікації методом Random Forest

Матриця невідповідностей демонструє що класи «Backdoor» та «Worms» мають не велику точність через помилкову класифікацію як «Fuzzers» та «Generic» (рис. 4.12).

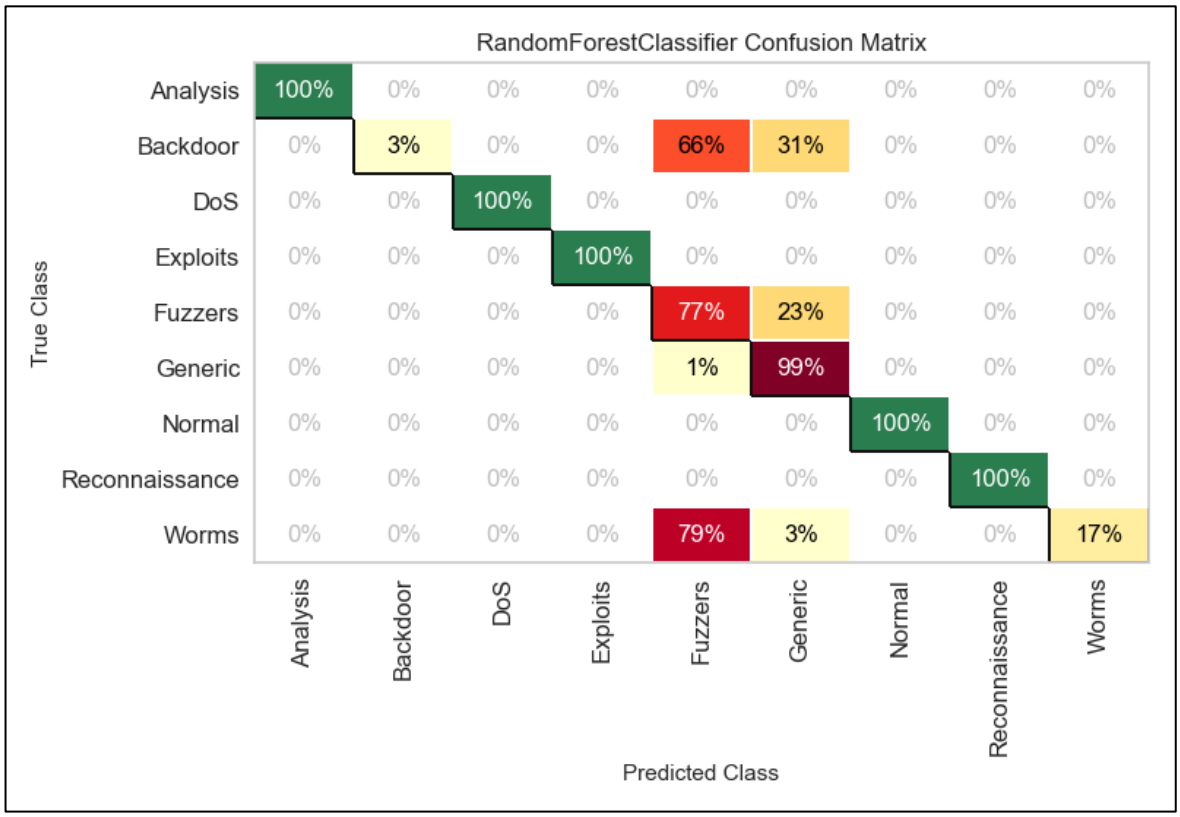

Рисунок 4.12 – Матриця невідповідностей методу Random Forest

Останнім методом машинного навчання було реалізовано метод Дерева Рішень. Під час реалізації моделі, було обрано критерій Джіні як функцію для оцінки якості розбиття на кожному з вузлів дерева, що сприятиме підвищенню точності моделі. Встановлення максимального рівня глибини дерева до 10 буде обмежувати складність моделі, що допоможе у запобіганні перенавчання. Мінімальна кількість зразків в листових вузлах була визначена як 1, а для мінімальної кількості зразків для розбиття внутрішніх вузлів було обрано 10.

Результатом реалізації моделі є правильність класифікованих зразків 98.9% (рис. 4.13).

	Metrics	Values
0	Mean Absolute Error	0.0174
1	Mean Squared Error	0.0406
2	Root Mean Squared Error	0.2014
3	Explained Variance Score (R^2)	97.2679
4	Accuracy	98.9077

Рисунок 4.13 – Точність класифікації методу Дерева Рішень

Звіт класифікації демонструє схожі результати отриманні раніше, де класи Worms та Backdoor мають слабкий відсоток класифікації (рис. 4.14).

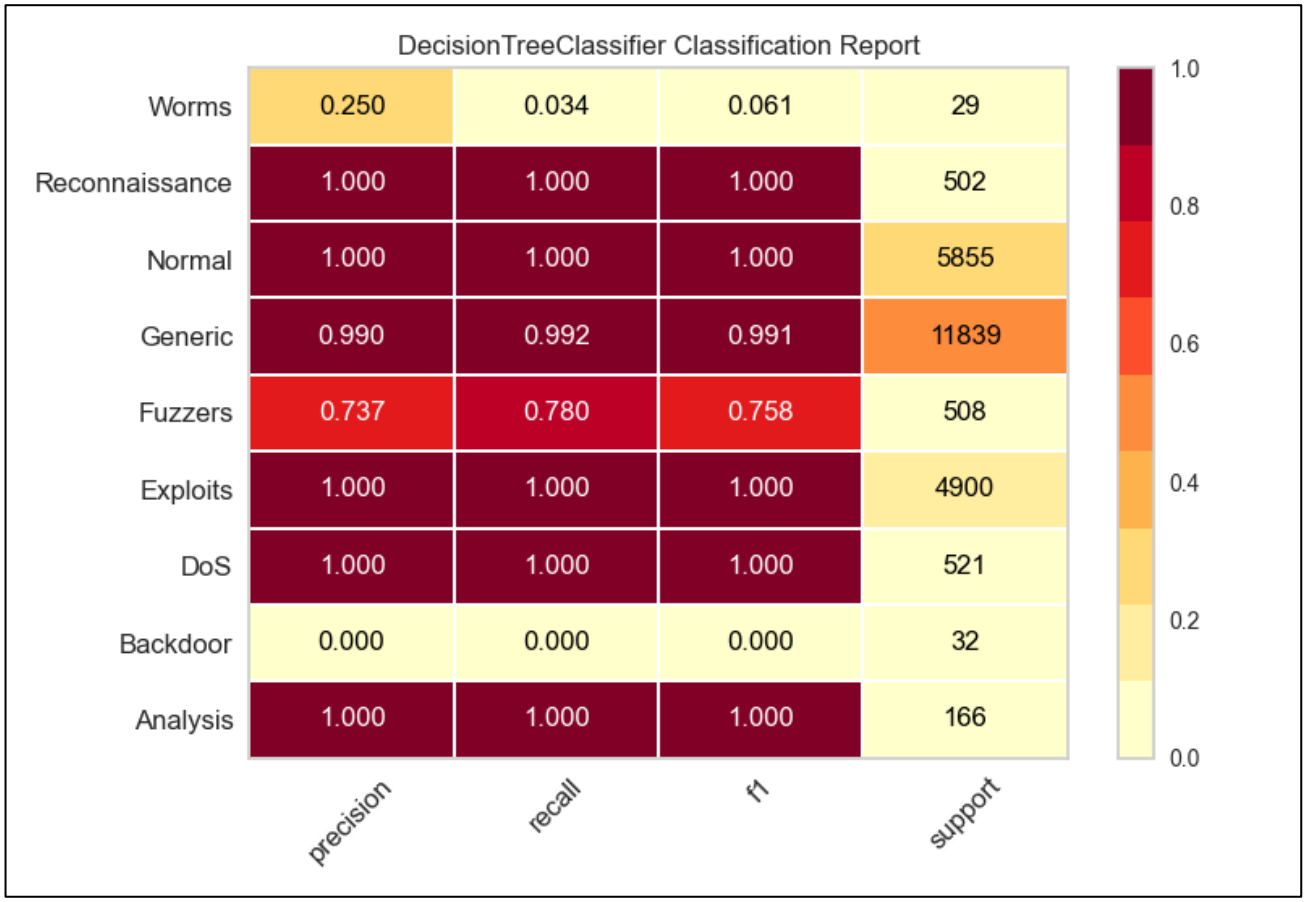

Рисунок 4.14 – Звіт класифікації методом Дерева Рішень

На основі матриці невідповідностей, можна визначити, що класи «Worms» та «Backdoor» помилково визначаються як «Fuzzers» та «Generic». Та самі класи «Fuzzers» і «Generic» помилково класифікуються між собою (рис. 4.15).

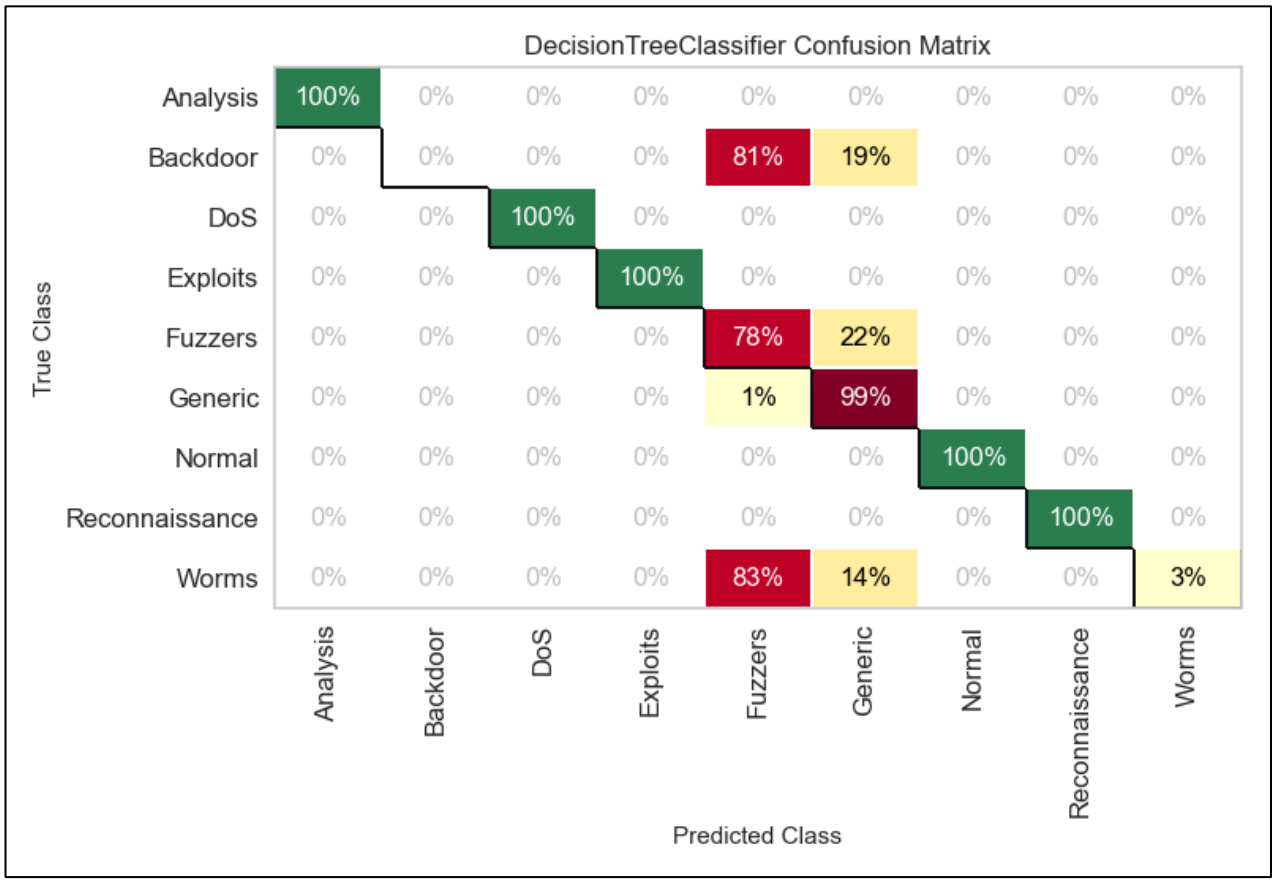

Рисунок 4.15 – Матриця невідповідностей методу Дерева Рішень

Отриманні результати від чотирьох моделей машинного навчання дозволяють перейти до аналізу та визначення кращої моделі.

4.3 Аналіз результатів

Отримані результати всіх моделей машинного навчання демонструюсь гарний результат класифікації – більше 95% точності кожен. Слід зазначити, що незважаючи на високу точність, моделі машинного навчання мали певні нюанси котрі можливо було спостерігати в кожній матриці невідповідностей.

Логістична регресія показала високу загальну точність класифікації, досягнувши 98.8995%. Середня абсолютна помилка становила 0.0172, корінь середньоквадратичної помилки – 0.1984, а коефіцієнт детермінації (R^2) – 97.3481%. Аналізуючи класифікаційний звіт, можна побачити, що моделі вдалося досягти ідеальної точності, повноти та F1-міри для класів «Analysis», «DoS», «Exploits», «Normal» та «Reconnaissance», що свідчить про здатність моделі ефективно

розпізнавати ці типи атак. Однак, для класів «Backdoor» та «Worms» показники точності, повноти та F1-міри дорівнювали 0.00, що вказує на повну неспроможність моделі ідентифікувати ці атаки. Клас «Fuzzers» мав помірні показники: точність 0.71, повнота 0.81 та F1-міра 0.75.

Метод KNN також показав високу загальну точність – 98.6367%, з середньою абсолютною помилкою 0.0255 та коренем середньоквадратичної помилки 0.2731. Коефіцієнт детермінації (R^2) склав 94.9799%. У класифікаційному звіті спостерігається збереження високих показників для більшості класів, однак клас «Backdoor» отримав незначне покращення з точністю та повнотою 0.06. Клас «Worms» показав точність 0.14, повноту 0.17 та F1-міру 0.15. Клас «Fuzzers» мав точність 0.76, повноту 0.60 та F1-міру 0.67, що свідчить про певні труднощі моделі в розпізнаванні цього типу атак.

Випадковий ліс продемонстрував найвищу загальну точність серед розглянутих моделей – 98.9652%. Середня абсолютна помилка становила 0.0165, корінь середньоквадратичної помилки – 0.1987, а коефіцієнт детермінації (R^2) – 97.3428%. Класифікаційний звіт показує значне покращення для класу «Backdoor» з точністю 0.50, хоча повнота залишилася низькою – 0.03. Клас «Worms» мав точність 0.71, повноту 0.17 та F1-міру 0.28. Клас «Fuzzers» показав покращення з точністю та повнотою 0.76 та 0.77 відповідно.

Дерево рішень показало високу загальну точність – 98.9077%, з середньою абсолютною помилкою 0.0174 та коренем середньоквадратичної помилки 0.2014. Коефіцієнт детермінації (R^2) склав 97.2679%. У класифікаційному звіті спостерігається збереження високих показників для більшості класів, однак клас «Backdoor» має нульові значення. Клас «Worms» показав точність 0.25, повноту 0.03 та F1-міру 0.06. Клас «Fuzzers» мав точність 0.74, повноту 0.78 та F1-міру 0.76, що свідчить про певні труднощі моделі в розпізнаванні цього типу атак.

Порівнюючи результати моделей, можна зробити висновок, що всі чотири моделі ефективно класифікують більшість класів з високою точністю та повнотою. Особливо це стосується класів з великою кількістю зразків у наборі даних, таких

як «Generic», «Normal» та «Exploits». Це свідчить про те, що моделі добре навчаються на класах з достатнім обсягом даних та чіткими відмінностями у характеристиках.

Однак, слабка класифікація класів «Backdoor» та «Worms» є спільною проблемою для всіх моделей. Основною причиною цього є значний дисбаланс у наборі даних. Класи «Backdoor» та «Worms» мають лише 32 та 29 зразків відповідно, що є мізерною часткою від загальної кількості зразків (24352). Такий дисбаланс призводить до того, що моделі не отримують достатньо інформації для навчання характеристикам цих атак. Крім того, через малу кількість зразків моделі можуть розглядати ці класи як шум або аномалії, що ще більше ускладнює їх правильну класифікацію.

Логістична регресія, будучи лінійною моделлю, може не фіксувати складні нелінійні взаємозв'язки у даних, що необхідно для розрізнення схожих класів. KNN, залежний від метрики відстані, може бути чутливим до «прокляття розмірності» у високовимірних просторах, де відстані між точками стають менш інформативними. Дерева рішень та їх ансамблі, такі як випадковий ліс, здатні моделювати нелінійні залежності та взаємодії між ознаками, проте вони також можуть стикатися з проблемами при класифікації рідкісних класів, таких як «Backdoor» або «Worms». Представлення недостатньої кількості даних призводить до того, що навіть якщо загальні результати моделі високі, точність класифікації для цих класів залишається низькою.

Однак, атаки «Backdoor» та «Worms» сприяють їхній неправильній класифікації. Атака «Backdoor» є особливо складною для класифікації через їхню мету бути не поміченою серед мережевого трафіку. Такі з'єднання можуть мати малу кількість пакетів в співвідношенні до атак «Exploits» чи «Reconnaissance» та приховуватись під нормальним мережевим трафіком. Приховування може проявлятись надсилання запитів через протокол HTTP або ж HTTPS, що робить ознаки розмитими та складними для моделювання. Клас «Worms» має кращі показники ніж «Backdoor», адже мусить розповсюджувати свої копії для

досягнення мети атаки, через це часто класифікується моделями як атаки «Fuzzers», які спрямовані на виявлення вразливостей шляхом надсилання випадкових або неправильних даних, що підвищує аномальність з'єднань та незвичайний патерн трафіку.

Способи експлуатації атак «Backdoor» та «Worms» є варіативними та змінюються залежно від: складності та мети атаки і архітектури комп'ютерної мережі. Використання широкого спектру технік та експлуатація різних вразливостей, ускладнює створення універсальних ознак для їх виявлення. А адаптація та зміна власної поведінки, наприклад, шляхом шифрування свого трафіку та імітація під нормальний, мають на меті уникнення виявлення.

Саме тому, детально оглянувши результати моделей машинного навчання, для вибору кращої моделі для класифікації мережевих атак, слід порівняти не тільки загальну точність класифікації, а й точність окремих класів «Worms», «Backdoor», «Fuzzers» (рис. 4.16).

Рисунок 4.16 – Аналіз результатів класифікації моделей

Наведена детальна інформація в табл. 4.1 та гістограма дозволяє проаналізувати кожен аспект результатів методів.

Метод логістичної регресії має помірне значення f1 для класу «Fuzzers» та нульові значення для класів «Worms» та «Backdoor». Загальна точність класифікації методу також є помірною серед інших методів.

Метод дерева рішень має кращі показники за метод логістичної регресії окрім класу «Backdoor», котрий також не було класифіковано правильно.

Метод k найближчих сусідів зміг класифікувати клас «Backdoor» з точністю в 6% та клас «Worms» – 15%. Однак клас «Fuzzers» був класифікований гірше за інші методи. Загальна точність методу є найгіршою з усіх отриманих.

Останній метод, метод випадкового лісу, класифікував клас «Worms» з точністю в 28%, та «Backdoor» – 6%. Клас «Fuzzers» також має кращу точність класифікації за інші методи. Також метод має найкращу загальну точність серед усіх реалізованих методів.

Таблиця 4.1 – Узагальнення проаналізованих результатів моделей

Метрика / Клас		**Логістична регресія**	**KNN**	**Випадковий ліс**	**Дерево рішень**
Загальна точність (%)		98.8995	98.6367	**98.9652**	98.9077
Середня абсолютна помилка		0.0172	0.0255	**0.0165**	0.0174
Середньоквадратична помилка		0.0394	0.0746	**0.0395**	0.0406
RMSE		0.1984	0.2731	**0.1987**	0.2014
R^2 (%)		97.3481	94.9799	**97.3428**	97.2679
Клас «Fuzzers»					
	Точність	0.7069	0.7580	**0.7622**	0.7374
	Повнота	0.8071	0.6043	**0.7697**	0.7795
	F1-міра	0.7537	0.6725	**0.7659**	0.7579
Клас «Worms»					
	Точність	0.0000	0.1389	**0.7143**	0.2500
	Повнота	0.0000	0.1724	**0.1724**	0.0345
	F1-міра	0.0000	0.1538	**0.2778**	0.0606
Клас «Backdoor»					
	Точність	0.0000	0.0625	**0.5000**	0.0000
	Повнота	0.0000	0.0625	**0.0313**	0.0000
	F1-міра	0.0000	0.0625	**0.0588**	0.0000

Покращення точності класифікації певного класу навіть на 5% є важливою перевагою для використання методу для безпеки комп'ютерних мереж.

Висновки до розділу 4

В четвертому розділі було розглянуто техніки та фреймворки котрі використовувались для нормалізації даних та реалізацію методів машинного навчання

Отримання результати були графічно продемонстровані матрицею невідповідностей та звітом класифікації, та свідчать про досягнення високої точності класифікації для більшості класів, особливо тих, що мають значну кількість зразків у наборі даних, таких як «Generic», «Normal» та «Exploits». Це свідчить про здатність моделей ефективно навчатися на добре представлених даних та розпізнавати чіткі патерни атак.

Однак було виявлено слабку класифікацію менш представлених та складних класів, зокрема «Backdoor» та «Worms». Незважаючи на загальну високу точність моделей, ці класи залишаються проблемними для правильного розпізнавання. Основними причинами є значний дисбаланс у наборі даних та особливості природи цих атак. Атаки типу «Backdoor» можуть маскуватися під нормальний трафік, використовуючи різні протоколи та порти, що робить їх ознаки неоднозначними та складними для класифікації. Аналогічно, атаки «Worms» можуть імітувати поведінку інших атак або нормальної активності, що ускладнює їх розпізнавання моделями.

Серед всіх результатів, метод Random Forest продемонстрував найвищу загальну точність серед розглянутих моделей – 98.9652%. Середня абсолютна помилка становила 0.0165, корінь середньоквадратичної помилки – 0.1987, а коефіцієнт детермінації (R^2) – 97.3428%. Класифікаційний звіт показує значне покращення для класу «Backdoor» з точністю 0.50, хоча повнота залишилася низькою – 0.03. Клас «Worms» мав точність 0.71, повноту 0.17 та F1-міру 0.28. Клас «Fuzzers» показав покращення з точністю та повнотою 0.76 та 0.77 відповідно.

Слід зазначити, що інші реалізовані моделі машинного навчання класифікували класи «Worms» та «Backdoor» помилково як «Fuzzers» що свідчить про явне визначення як тип атаки, а не нормальний трафік мережі.

Для забезпечення надійного захисту інформаційних систем необхідно приділити особливу увагу складним та рідкісним атакам, вдосконалюючи методи класифікації та підходи до підготовки даних. Подальші дослідження в цьому напрямку сприятимуть підвищенню рівня кібербезпеки та розробці більш ефективних систем виявлення загроз, націлених на певні типи атаки для комп’ютерних мереж.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було створено систему для виявлення комп’ютерних мережевих атак за допомогою методів машинного навчання.

Був проведений аналіз існуючих мережевих атак, стандартних методів виявлення та огляд наявних аналогів та публікації. На основі цього було поставлено задачі, котрі необхідно реалізувати.

Обрані методи машинного навчання, а саме: зокрема логістичної регресії, методу k-найближчих сусідів, дерева рішень та випадкового лісу, було проаналізовано та детально описано.

Серед схожих наборів даних, набір UNSW-NB15 було обрано, для кращого відображення реальних умов функціонування мережевої інфраструктури, порівняно з більш застарілими наборами даних на кшталт KDD Cup 99 чи DARPA 1998.

Проведений аналіз отриманих результатів підтвердив можливості моделей машинного навчання ефективно класифікувати більшість типів атак з високою точністю. Особливо це стосується поширених та добре представлених у наборі даних категорій, як-от «Generic», «Exploits» та «Normal», для яких показники точності, повноти та F1-міри були майже ідеальними.

Водночас, результати чітко засвідчили проблему класифікації менш розповсюджених та складних для виявлення атак, таких як «Backdoor» та «Worms». Незважаючи на загально високу точність систем (понад 98% для більшості моделей), ці рідкісні категорії залишались важкими для коректного розпізнавання, що пояснюється більше як прихований характер поведінки таких атак, які можуть імітувати властивості нормального трафіку або інші шкідливі дії. Ці результати наголошують на необхідності додаткових кроків, для поліпшення виявлення складних багатовекторних загроз, такі як використання методів штучного

балансування даних, використання інших наборів даних або збагачення на основі обраного.

Серед розглянутих у роботі алгоритмів найкращі показники продемонстрував випадковий ліс, який забезпечив найвищу загальну точність та деяке, хоча й обмежене, поліпшення у класифікації складних атак.

Таким чином, дослідження підтвердило доцільність використання машинного навчання для автоматизованого виявлення мережевих атак та окреслило напрямки подальшої оптимізації підходів до аналізу складних, рідкісних і складних типів загроз. Поглиблене вивчення цих питань сприятиме підвищенню надійності й адаптивності систем захисту, забезпечуючи більш ефективне виявлення та нейтралізацію сучасних кібернетичних небезпек.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Колодяжний К. О., Калініна І. О. Система виявлення комп'ютерних мережевих атак за допомогою методів машинного навчання / Тези всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів, 2–4 грудня 2024 р., м. Миколаїв.

2. Machine learning Beginners Guide Algorithms: Supervised & Unsupervised learning, Decision Tree & Random Forest Introduction. CreateSpace Independent Publishing Platform, 2017. 266 p.

3. Gawande R. DOS Attacks. International Journal of Science and Research (IJSR). 2018. Vol. 7, no. 2. P. 450.

4. Denial of Service. URL: https://attack.mitre.org/techniques/T0814/ (дата звернення: 03.10.2024).

5. Network Denial of Service. URL: https://attack.mitre.org/techniques/T1498/ (дата звернення: 03.10.2024).

6. Analysis of the Cyber Attack on the Ukrainian Power Grid. URL: https://nsarchive.gwu.edu/sites/default/files/documents/3891751/SANS-and-Electricity-Information-Sharing-and.pdf (дата звернення: 03.10.2024).

7. Active Scanning: Wordlist Scanning. URL: https://attack.mitre.org/techniques/T1595/003/ (дата звернення: 05.10.2024).

8. Fuzzing. URL: https://owasp.org/www-community/Fuzzing (дата звернення: 05.10.2024).

9. Z. Hu and Z. Pan, "A Systematic Review of Network Protocol Fuzzing Techniques," 2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2021, pp. 1000-1005.

10. Особливості деструктивних кібератак Sandworm у відношенні українських провайдерів (CERT-UA#7627). URL: https://cert.gov.ua/article/6123309 (дата звернення: 07.10.2024).

11. Pranav P., Patel A., Jain S. Machine Learning in Healthcare and Security. Boca Raton : CRC Press, 2023. URL: https://doi.org/10.1201/9781003388845 (date of access: 07.10.2024).

12. Technical Guide to Information Security Testing and Assessment. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-115.pdf (дата звернення: 15.10.2024).

13. About the CVE Program. URL: https://www.cve.org/About/Overview (дата звернення: 15.10.2024).

14. Stallings W. Network Security Essentials Applications and Standards. Pearson Education, Limited, 2013.

15. Vulnerability Scanning Tools. URL: https://owasp.org/www-community/Vulnerability_Scanning_Tools (дата звернення: 16.10.2024).

16. Vulnerability Scanning. URL: https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-014_Vulnerability_Scanning (дата звернення: 23.10.2024).

17. Active Scanning: Vulnerability Scanning. URL: https://attack.mitre.org/techniques/T1595/002/ (дата звернення: 23.10.2024).

18. Pawn Storm in 2019. A Year of Scanning and Credential Phishing on High-Profile Targets. URL: https://documents.trendmicro.com/assets/white_papers/wp-pawn-storm-in-2019.pdf (дата звернення: 26.10.2024).

19. Applied Network Security Monitoring. Elsevier, 2014. URL: https://doi.org/10.1016/c2013-0-00546-4 (date of access: 26.10.2024).

20. Zero Trust Networks. / R. Rais та ін. 2-ге вид. O'Reilly Media, 2024. 334 с.

21. Stage Capabilities: Upload Malware. URL: https://attack.mitre.org/techniques/T1608/001/ (дата звернення: 05.11.2024).

22. User Execution: Malicious Image. URL: https://attack.mitre.org/techniques/T1204/003/ (дата звернення: 05.11.2024).

23. Process Injection. URL: https://attack.mitre.org/techniques/T1055/ (дата звернення: 05.11.2024).

24. Server Software Component: Web Shell. URL: https://attack.mitre.org/techniques/T1505/003/ (дата звернення: 05.11.2024).

25. Validating Security Configurations and Detecting Backdoors in New Network Devices. URL: https://www.sans.org/white-papers/35472/ (дата звернення: 05.11.2024).

26. Sandworm Disrupts Power in Ukraine Using a Novel Attack Against Operational Technology. URL: https://cloud.google.com/blog/topics/threat-intelligence/sandworm-disrupts-power-ukraine-operational-technology/ (дата звернення: 05.11.2024).

27. Malicious Code. URL: https://csrc.nist.rip/publications/nistir/threats/section3_3.html (дата звернення: 06.11.2024).

28. Conficker. URL: https://attack.mitre.org/software/S0608/ (дата звернення: 06.11.2024).

29. CWE-121: Stack-based Buffer Overflow. URL: https://cwe.mitre.org/data/definitions/121.html (дата звернення: 06.11.2024).

30. CWE-122: Heap-based Buffer Overflow. URL: https://cwe.mitre.org/data/definitions/122.html (дата звернення: 06.11.2024).

31. NIST SPECIAL PUBLICATION 1800-26A. URL: https://www.nccoe.nist.gov/publication/1800-26/VolA/index.html (дата звернення: 06.11.2024).

32. Rahman R.U., Tomar D.S. Emerging Wireless Communication and Network Technologies: Principle, Paradigm and Performance. Springer; Singapore: 2018. Security attacks on wireless networks and their detection techniques; pp. 241–270.

33. Azeez N.A., Bada T.M., Misra S., Adewumi A., der Vyver C., Ahuja R. Intrusion Detection and Prevention Systems: An Updated Review. In: Sharma N., Chakrabarti A., Balas V.E., editors. Data Management, Analytics and Innovation. Springer; Singapore: 2020. pp. 685–696.

34. NISTIR 8011 Volume 3. Automation Support for Security Control Assessments Software Asset Management. URL: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8011-3.pdfa (дата звернення: 14.11.2024).

35. Mukherjee A. Network Security Strategies: Packt Publishing, 2020. 339 с.

36. A review of zero-day in-the-wild exploits in 2023. URL: https://blog.google/technology/safety-security/a-review-of-zero-day-in-the-wild-exploits-in-2023/ (дата звернення: 05.11.2024).

37. J. Oh, "Fight against 1-day exploits: Diffing binaries vs anti-diffing binaries", Black Hat, 2009.

38. Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations. URL: https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-352a (дата звернення: 14.11.2024).

39. A.H. Almutairi, N.T. Abdelmajeed. Innovative signature based intrusion detection system: Parallel processing and minimized database, 2017 International Conference on the Frontiers and Advances in Data Science (FADS), IEEE (2017).

40. Darktrace Launches Enterprise Immune System Version 4. URL: https://darktrace.com/news/darktrace-launches-enterprise-immune-system-version-4 (дата звернення: 16.11.2024).

41. Cisco Secure Network Analytics. URL: https://www.cisco.com/site/us/en/products/security/security-analytics/secure-network-analytics/index.html (дата звернення: 16.11.2024).

42. C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," in IEEE Access, vol. 5, pp. 21954-21961, 2017.

43. Jonathan, D., Camila, P., Luiz, D., Cristiano, B., João, G., Celia, R., Marcelo, M.: Abnormal Behavior Detection Based on Traffic Pattern Categorization in Mobile Networks. IEEE Transactions on Network and Service Management 18(4), 2021.

44. Logistic Regression. URL: https://web.stanford.edu/~jurafsky/slp3/5.pdf (дата звернення: 25.11.2024).

45. Eknath, Prof. Upasani Dhananjay, 1st, Shete, Prof. Virendra Virbhadra, 1st. Machine Learning with Python: Machine Learning with Python. INSC International Publisher (IIP), 2021.

46. Machine learning Beginners Guide Algorithms: Supervised & Unsupervised learning, Decision Tree & Random Forest Introduction. CreateSpace Independent Publishing Platform, 2017. 266 p.

47. Decision Tree Algorithm, Explained. URL: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html (дата звернення: 26.11.2024).

48. M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 2009, pp. 1-6.

49. Choudhary S., Kesswani N. Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT. Procedia Computer Science. 2020. Vol. 167. P. 1561–1573. URL: https://doi.org/10.1016/j.procs.2020.03.367 (date of access: 26.11.2024).

50. Zoghi Z., Serpen G. UNSW-NB15 computer security dataset: Analysis through visualization. SECURITY AND PRIVACY. 2023. URL: https://doi.org/10.1002/spy2.331 (date of access: 26.11.2024).

51. Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015.

52. NumPy documentation. URL: https://numpy.org/doc/stable/ (дата звернення: 27.11.2024).

53. pandas documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 27.11.2024).

54. Matplotlib 3.9.3 documentation. URL: https://matplotlib.org/stable/ (дата звернення: 27.11.2024).

55. seaborn: statistical data visualization. URL: https://seaborn.pydata.org/ (дата звернення: 27.11.2024).

56. scikit-learn. URL: https://scikit-learn.org/stable/ (дата звернення: 28.11.2024).

57. Yellowbrick: Machine Learning Visualization. URL: https://www.scikit-yb.org/en/latest/ (дата звернення: 28.11.2024).

ДОДАТОК А
Код програмної реалізації

main.py

```
import warnings
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
import seaborn as sns
from yellowbrick.classifier import ConfusionMatrix, ClassificationReport
from sklearn.metrics import confusion_matrix

data = pd.read_csv('UNSW_NB15.csv', encoding="utf-8")
features = pd.read_csv('UNSW_NB15_features.csv', encoding="utf-8")

data['service'] = data['service'].replace('-', np.nan)
data.dropna(inplace=True)

for col in data.columns.intersection(features['Name'][features['Type '] == 'integer']):
    data[col] = pd.to_numeric(data[col], errors='coerce')

for col in data.columns.intersection(features['Name'][features['Type '] == 'binary']):
    data[col] = pd.to_numeric(data[col], errors='coerce')

for col in data.columns.intersection(features['Name'][features['Type '] == 'float']):
    data[col] = pd.to_numeric(data[col], errors='coerce')

num_col = data.select_dtypes(include='number').columns
cat_col = data.columns.difference(num_col)[1:]
data_cat = data[cat_col].copy()
data_cat = pd.get_dummies(data_cat, columns=cat_col)
data = pd.concat([data, data_cat], axis=1)

data.drop(columns=cat_col, inplace=True)

num_col = list(data.select_dtypes(exclude='object').columns)
num_col.remove('id')
num_col.remove('label')

minmax_scale = MinMaxScaler(feature_range=(0, 1))

def normalization(df, col):
    for i in col:
        arr = np.array(df[i])
        df[i] = minmax_scale.fit_transform(arr.reshape(-1, 1))
    return df
```

```
data = normalization(data.copy(), num_col)

multi_data = data.copy()
multi_label = pd.DataFrame(multi_data.attack_cat)
multi_data = pd.get_dummies(multi_data,columns=['attack_cat'])

le = preprocessing.LabelEncoder()
enc_label = multi_label.apply(le.fit_transform)
multi_data['label'] = enc_label

num_col.append('label')
num_col = list(multi_data.select_dtypes(include=('number','bool')).columns)

plt.figure(figsize=(15,10))
corr_multi = multi_data[num_col].corr()
sns.heatmap(corr_multi,vmax=1.0,annot=False)
plt.title('Correlation Matrix',fontsize=16)
plt.subplots_adjust(bottom=0.3)
plt.show()

corr_ymulti = abs(corr_multi['label'])
highest_corr_multi = corr_ymulti[corr_ymulti >0.25]
multi_cols = highest_corr_multi.index
multi_data = multi_data[multi_cols].copy()


X = multi_data.drop(columns=['label'],axis=1)
Y = multi_data['label']
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.30, random_state=100)

lr = LogisticRegression(random_state=123, max_iter=1500,solver='newton-cg',multi_class='multinomial')
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

metrics_data_LR = {
   'Metrics': ['Mean Absolute Error', 'Mean Squared Error', 'Root Mean Squared Error', 'Explained Variance Score (R²)',
'Accuracy'],
   'Values': [
      metrics.mean_absolute_error(y_test, y_pred),
      metrics.mean_squared_error(y_test, y_pred),
      np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
      metrics.explained_variance_score(y_test, y_pred) * 100,
      accuracy_score(y_test, y_pred) * 100
   ]
}

df_metrics = pd.DataFrame(metrics_data_LR)
df_metrics['Values'] = df_metrics['Values'].apply(lambda x: f"{x:.4f}" if isinstance(x, float) else x)
print("Logistic Regression Results:\n",df_metrics)

visualizer = ClassificationReport(lr, classes=le.classes_, support=True)
visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
visualizer.show()
classification_report_LR = classification_report(y_test, y_pred, target_names=le.classes_, output_dict=True)
print("\n\nClassification Report of Logistic Regression:\n", classification_report_LR)

cm_viz = ConfusionMatrix(lr, classes=le.classes_, percent=True)
cm_viz.fit(X_train, y_train)
cm_viz.score(X_test, y_test)
cm_viz.show()
```

```
knc = KNeighborsClassifier(n_neighbors=4)
knc.fit(X_train,y_train)
y_pred = knc.predict(X_test)

metrics_data_KNC = {
   'Metrics': ['Mean Absolute Error', 'Mean Squared Error', 'Root Mean Squared Error', 'Explained Variance Score (R²)',
'Accuracy'],
   'Values': [
      metrics.mean_absolute_error(y_test, y_pred),
      metrics.mean_squared_error(y_test, y_pred),
      np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
      metrics.explained_variance_score(y_test, y_pred) * 100,
      accuracy_score(y_test, y_pred) * 100
   ]
}
df_metrics = pd.DataFrame(metrics_data_KNC)
df_metrics['Values'] = df_metrics['Values'].apply(lambda x: f"{x:.4f}" if isinstance(x, float) else x)
print("\n\n\nKNN Results:\n",df_metrics)

cm_viz = ConfusionMatrix(knc, classes=le.classes_, percent=True)
cm_viz.score(X_test, y_test)
cm_viz.show()

visualizer = ClassificationReport(knc, classes=le.classes_, support=True)
visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
visualizer.show()
classification_report_KNC = classification_report(y_test, y_pred, target_names=le.classes_, output_dict=True)
print("\n\nClassification Report of Randomw Forest:\n", classification_report_KNC)

rfc = RandomForestClassifier(random_state=100)
rfc.fit(X_train,y_train)
y_pred = rfc.predict(X_test)

metrics_data_RFC = {
   'Metrics': ['Mean Absolute Error', 'Mean Squared Error', 'Root Mean Squared Error', 'Explained Variance Score (R²)',
'Accuracy'],
   'Values': [
      metrics.mean_absolute_error(y_test, y_pred),
      metrics.mean_squared_error(y_test, y_pred),
      np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
      metrics.explained_variance_score(y_test, y_pred) * 100,
      accuracy_score(y_test, y_pred) * 100
   ]
}
df_metrics = pd.DataFrame(metrics_data_RFC)
df_metrics['Values'] = df_metrics['Values'].apply(lambda x: f"{x:.4f}" if isinstance(x, float) else x)
print("\n\n\nRandom Forest Results:\n",df_metrics)

cm_viz = ConfusionMatrix(rfc, classes=le.classes_, percent=True)
cm_viz.score(X_test, y_test)
cm_viz.show()

visualizer = ClassificationReport(rfc, classes=le.classes_, support=True)
visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
visualizer.show()
classification_report_RFC = classification_report(y_test, y_pred, target_names=le.classes_, output_dict=True)
print("\n\nClassification Report of Randomw Forest:\n", classification_report_RFC)
```

```
dtc = DecisionTreeClassifier(criterion="gini", max_depth=10, min_samples_split=10, min_samples_leaf=1, random_state=42)
dtc.fit(X_train,y_train)
y_pred = dtc.predict(X_test)

metrics_data_DTC = {
   'Metrics': ['Mean Absolute Error', 'Mean Squared Error', 'Root Mean Squared Error', 'Explained Variance Score (R²)', 'Accuracy'],
   'Values': [
      metrics.mean_absolute_error(y_test, y_pred),
      metrics.mean_squared_error(y_test, y_pred),
      np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
      metrics.explained_variance_score(y_test, y_pred) * 100,
      accuracy_score(y_test, y_pred) * 100
   ]
}
df_metrics = pd.DataFrame(metrics_data_DTC)
df_metrics['Values'] = df_metrics['Values'].apply(lambda x: f"{x:.4f}" if isinstance(x, float) else x)
print("\n\n\nDecision Tree Results:\n",df_metrics)
cm_viz = ConfusionMatrix(dtc, classes=le.classes_, percent=True)
cm_viz.score(X_test, y_test)
cm_viz.show()

visualizer = ClassificationReport(dtc, classes=le.classes_, support=True)
visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
visualizer.show()
classification_report_DTC = classification_report(y_test, y_pred, target_names=le.classes_, output_dict=True)
print("\n\nClassification Report of Decision Tree:\n", classification_report_DTC)

accuracy, r2_scores, worms_f1, backdoor_f1, fuzzers_f1 = [],[],[],[],[]
models = {
   'Logistic Regression': [classification_report_LR, metrics_data_LR['Values']],
   'KNN': [classification_report_KNC, metrics_data_KNC['Values']],
   'Random Forest': [classification_report_RFC, metrics_data_RFC['Values']],
   'Decision Tree': [classification_report_DTC, metrics_data_DTC['Values']]
}

for model_name, reports in models.items():
   classification_report_dict = reports[0]
   accuracy.append(reports[1][4])
   r2_scores.append(reports[1][3])

   worms_f1.append(classification_report_dict.get('Worms', {}).get('f1-score', 0.0))
   backdoor_f1.append(classification_report_dict.get('Backdoor', {}).get('f1-score', 0.0))
   fuzzers_f1.append(classification_report_dict.get('Fuzzers', {}).get('f1-score', 0.0))

plt.figure(figsize=(12, 10))
plt.subplot(3, 1, 1)
plt.bar(models.keys(), accuracy, color='blue', alpha=0.7)
plt.title('Overall Accuracy Comparison')
plt.ylabel('Accuracy (%)')
plt.ylim(90, 100)
for i, val in enumerate(accuracy):
   plt.text(i, val + 0.2, f"{val:.2f}%", ha='center', va='bottom', fontsize=10)

plt.subplot(3, 1, 2)
plt.bar(models.keys(), r2_scores, color='green', alpha=0.7)
plt.title('R² (Explained Variance) Comparison')
plt.ylabel('R² (%)')
plt.ylim(90, 100)
```

```
for i, val in enumerate(r2_scores):
    plt.text(i, val + 0.2, f"{val:.2f}%", ha='center', va='bottom', fontsize=10)

x = np.arange(len(models))
plt.subplot(3, 1, 3)
plt.bar(x - 0.25, worms_f1, 0.25, label='Worms', color='orange', alpha=0.7)
plt.bar(x, backdoor_f1, 0.25, label='Backdoor', color='purple', alpha=0.7)
plt.bar(x + 0.25, fuzzers_f1, 0.25, label='Fuzzers', color='teal', alpha=0.7)
plt.title('F1-Score Comparison for Worms, Backdoor, and Fuzzers')
plt.ylabel('F1-Score')
plt.ylim(0, 0.9)
plt.xticks(x, models.keys())
plt.legend()

for i, val in enumerate(worms_f1):
    plt.text(i - 0.25, val + 0.02, f"{val:.2f}", ha='center', va='bottom', fontsize=10)

for i, val in enumerate(backdoor_f1):
    plt.text(i, val + 0.02, f"{val:.2f}", ha='center', va='bottom', fontsize=10)

for i, val in enumerate(fuzzers_f1):
    plt.text(i + 0.25, val + 0.02, f"{val:.2f}", ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.show()
```

ДОДАТОК Б
Матеріали апробації роботи

МАШИННЕ НАВЧАННЯ ТА ШТУЧНИЙ ІНТЕЛЕКТ

04.12.2024 р.
15-30
Посилання: https://meet.google.com/qaz-vtch-njh

Голова секції: д.т.н., проф. Гожий О.П.

Секретар секції: Сомряков Б.

1. **Пещерьова В. С., Калініна І. О.** Інтелектуальне прогнозування на основі багатошарових ансамблевих структур.
2. **Димо В.В.** Семантична сегментація пошкоджених будівель з використанням згорткових нейронних мереж
3. **Yu Fei, E. Malakhov** Implementation of self-attention mechanism into information technology of high-resolution remote sensing image semantic segmentation.
4. **Колодяжний К. О., Калініна І. О.** Система виявлення комп'ютерних мережевих атак за допомогою методів машинного навчання.
5. **Рева В.В.** Інтегрована система для навчання агентів Unity ML AGENTS з використанням моделей глибоких нейронних мереж.

УДК 004.85

Колодяжний К. О., Калініна І. О.
Чорноморський національний університет
ім. Петра Могили,
Миколаїв, Україна

СИСТЕМА ВИЯВЛЕННЯ КОМП'ЮТЕРНИХ МЕРЕЖЕВИХ АТАК ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Проведення мережевих атак безперервно розвивається, збільшуючи арсенал методів і тактик, що створює серйозні виклики для сучасної сфери кібербезпеки. Виявлення шляхом використання методів машинного навчання допоможе зберегти безпеку комп'ютерної мережі. Традиційні системи виявлення вторгнень часто виявляються неефективними для невідомих атак або складних векторів загроз, що підкреслює важливість використання машинного навчання.

Дослідження та аналоги по виявленню атак в мережах [1-2] демонструють високий рівень виявлення та аналізу атак. Наприклад, системи, оптимізовані для аналізу трафіку в корпоративних мережах, можуть показувати нижчу точність при роботі у середовищах IoT або мобільних мережах. Такі розбіжності обумовлені відмінностями у структурі мережевого трафіку, обсягах даних, та типах атак, які переважають у кожному з цих середовищ.

Тому в дослідженні поставлено задачу обрати модель машинного навчання, яка найкраще виявляє мережеві атаки шляхом розв'язання задачі класифікації.

Набір даних UNSW-NB15 обрано для дослідження через його актуальність та широкий спектр сучасних атак, включаючи DoS, DDoS, Fuzzing, експлойти та бекдори. Цей набір даних створено за допомогою генератора IXIA, що дозволяє імітувати реальні мережеві сценарії. Основною перевагою UNSW-NB15 є його відповідність сучасним викликам у сфері кібербезпеки, на відміну від застарілих наборів, таких як KDD Cup 99 або NSL-KDD.

Перед реалізацією моделей, набір даних підготовлено, шляхом прибирання відсутніх значень, нормалізації числових характеристик та кодування категоріальних ознак використовуючи one-hot метод.

Набір даних [3] містить 175341 записів та 45 ознак. Характеристики включають: атрибути характеристики потоку (дані що містять основну інформацію про запит), атрибути базових характеристик (інформація про мережеве з'єднання), атрибути вмісту (інформація про параметри з'єднання та особливості переданих даних