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, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ВИЯВЛЕННЯ КІБЕРЗАГРОЗ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Спеціальність 122 «Комп'ютерні науки»

122 – КРМ – 601. 21810301

Виконав студент 6-го курсу, групи 601
_____ І. В. Барбулат
«19» лютого 2024 р.

Керівник: д-р техн. наук, проф.
_____О. П. Гожий
«19» лютого 2024 р.

Миколаїв – 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

МН – машинне навчання.

ACC – accuracy.

ACL – access control list.

FN – false negative.

FP – false positive.

IDS – intrusion detection system.

IoT – internet of things.

IT – information technology.

ML – machine learning.

OCR – overall classification rate.

PPV – positive predicted value.

SOC – security operations center.

TN – true negative.

TP – true positive.

TPR – true positive rate.

ВСТУП

Актуальність. Наше життя стало залежним від інтернету. Кожен день ми спілкуємося з друзями, шукаємо різноманітну інформацію, виконуємо банківські операції та багато чого іншого. Головне з цього, що кожна людина залишає свій інформаційний слід у Інтернеті. І не завжди ці ресурси є захищеними від атак злочинців, які можуть заволодіти конференційною інформацією і використовувати її у своїх цілях. Тому і постало питання захисту інформації. Спочатку для виявлення атак використовували системи виявлення вторгнень на основі сигнатур, статичні правила брандмауера для мережевого трафіку та інші. Але із розвитком технологій, а особливо із об'ємом даних, старі методи вже не такі ефективні. Тим паче злочинці не стоять на місці і з кожним роком вдосконалюють свої навички і знаходять нові вразливості.

Тому люди замислились над використанням машинного навчання для запобіганням атак. МН дозволяє вдосконалити вже наявні методи захисту і виводить їх на новий рівень, що призводить до укріпленню кібербезпеки. Особливо ефективним воно стало у виявленні аномалій. Моделі машинного навчання здатні аналізувати величезні обсяги даних, включно з мережевим трафіком, системними журналами і моделями поведінки користувачів, на основі чого можна виявляти аномалії і виявляти потенційні загрози. Моделі машинного навчання допомагають зрозуміти, що таке "нормальна" поведінка в мережі або системі. Завдяки цим знанням вони можуть сигналізувати про аномальну поведінку або потенційні загрози безпеці. Для цього використовують алгоритми класифікації, які допомагають серед великих масивів даних класифікувати загрози і нормальний трафік.

Об'єкт дослідження – процес виявлення кіберзагроз.

Предмет дослідження – методи і алгоритми машинного навчання для вирішення задач класифікації.

Метою кваліфікаційної роботи магістра є підвищення ефективності виявлення та класифікації кіберзагроз за допомогою методів машинного навчання.

Відштовхуючись від мети були поставлені наступні **завдання**:

- проаналізувати класичні засоби виявлення кіберзагроз;
- проаналізувати основні теоретичні засади машинного навчання;
- дослідити алгоритми класифікації;
- підібрати набір даних, вивчити його та підготувати для навчання моделей;
- навчити та протестувати моделі для бінарної та багатокласової класифікації.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Аналіз засобів виявлення кіберзагроз

Щойно вчені з'єднали достатню кількість комп'ютерів через інтернет для створення мережі обміну інформацією, що приносить реальну користь, інші люди виявили, що цей засіб вільного передавання і широкого розповсюдження даних являє собою чудовий спосіб рекламування зробленої наспіх продукції, викрадення секретних облікових і реєстраційних даних і поширення комп'ютерних вірусів [6].

Таблиця 1.1 – Класичні методи виявлення кіберзагроз.

Фаєрволи	використовуються для фільтрації мережевого трафіку, блокування небажаних підключень та застосування правил безпеки. Ці засоби можуть працювати на основі заздалегідь визначених правил
Системи виявлення вторгнень (IDS) на основі сигнатур	використовують сигнатури відомих атак для виявлення аномалій.
Системи аудиту безпеки	Ці системи зафіксовують події та активності в системі для подальшого аналізу. Вони можуть використовувати правила та заздалегідь визначені шаблони для виявлення невідповідностей
Системи аналізу трафіку	Деякі засоби аналізу трафіку можуть виявляти аномалії та незвичайні патерни без використання методів машинного навчання, використовуючи заздалегідь визначені правила та параметри.
Системи виявлення підроблених ідентифікаторів	Ці системи можуть виявляти спроби використання неправомірних або підроблених ідентифікаторів на основі стандартних правил та порівнянь.

Із розвитком технологій розвивались методи захисту та методи атак, одним із напрямів стало використання машинного навчання для захисту і виявлення кіберзагроз.

Машинне навчання (ML) - це термін, який сьогодні широко використовується майже в усіх галузях IT. І хоча ML часто використовується для аналізу великих даних - для покращення бізнес-процесів та покращення ефективності бізнесу, а також для прогнозування - воно також виявилося безцінним в інших сферах, зокрема, в кібербезпеці.

Потреба в машинному навчанні пов'язана зі складністю. Сьогодні багато організацій мають зростаючу кількість пристроїв Інтернету речей (IoT), про які IT-спеціалісти не знають і не керують ними. Всі дані та додатки не працюють локально, оскільки гібридні та мультихмарні технології є новою нормою. Користувачі більше не перебувають переважно в офісі, оскільки віддалена робота є загальноприйнятою [6].

Не так давно підприємства покладалися на виявлення шкідливого програмного забезпечення на основі сигнатур, статичні правила брандмауера для мережевого трафіку та списки контролю доступу (ACL) для визначення політик безпеки. У світі, де пристроїв стало більше, ніж будь-коли, старі способи виявлення потенційних загроз безпеці не встигають за масштабом, обсягом і складністю.

Для того, щоб вирішити всі нові проблеми безпеки, з якими стикаються організації, існує очевидна потреба в машинному навчанні. Тільки машинне навчання може вирішити зростаючу кількість проблем у сфері кібербезпеки: масштабування рішень безпеки, виявлення невідомих атак і виявлення складних атак, включаючи поліморфне шкідливе програмне забезпечення. Сучасне шкідливе програмне забезпечення може змінювати форму, щоб уникнути виявлення, а використання традиційного підходу, заснованого на сигнатурах, дуже ускладнює виявлення таких складних атак. ML виявляється найкращим рішенням для боротьби з ними.

Але постає питання який вид алгоритмів використовувати. Найкращим вибором є алгоритми класифікації з наступних причин:

– виявлення аномалій: Алгоритми класифікації можуть виявляти аномалії в мережевому трафіку, системних логів та інших даних, що може вказувати на потенційні кіберзагрози або вторгнення;

– класифікація шкідливого програмного забезпечення: Застосування алгоритмів класифікації може допомогти в розпізнаванні та класифікації шкідливих програм, що спрощує їх виявлення та усунення.

1.2 Машинне навчання

Машинне навчання – це одна із областей штучного інтелекту, яка має набір методів, алгоритмів для навчання на власному досвіді. Для навчання використовуються дані, зазвичай великі набори, у яких знаходяться закономірності. Чим більше даних обробляється, тим ефективніша навчена модель, але не треба забувати і про перенавчання моделі [36].

Але також постає питання щодо перенавчання моделі. Цей процес відбувається коли модель не узагальнена і сильно відповідає навчальному набору даних. За частіше це відбувається через наступні причини:

– розмір навчальних даних занадто малий і не містить достатньої кількості вибірок даних для точного представлення всіх можливих значень вхідних даних;

– навчальні дані містять велику кількість нерелевантної інформації або зашумлені дані;

– модель надто довго навчається на одному наборі даних;

– складність моделі висока, тому вона вивчає шум у навчальних даних.

Тому постає питання як не допустити перенавчання моделі. Для цього використовують різні підходи які описані нижче.

Регуляризація. Набір методів для навчання та оптимізації направлені на скорочення перенасичення. Вони направленні на усунення факторів які не

впливають на результат прогнозу шляхом оцінки об’єктів за важливістю. Наприклад, для математичних обчислень застосовують штрафи до об’єктів які мають мінімальний вплив. Маючи статистичну модель, яка намагається спрогнозувати ціни на житло у місті, регуляція дає менше штрафу для характеристик які описують зростання населення та середньорічний дохід, але більше штрафу для середньорічної температури у місті.

Рання зупинка. Зупиняє етап навчання до того, як модель дізнається про шуми у наборі. Однак важливо правильно визначити час, в іншому випадку результати навчання моделі дасть не точні результати.

Ансамблювання. Поєднує прогнози кількох алгоритмів. Моделі з неточними результатами відносяться до слабких і потім ці моделі об’єднуються задля більш точних результатів(обрання найбільш точних результатів). Два основні методи ансамблювання – це пакетування та бустинг. Під час бустингу навчаються різні моделі машинного навчання одна за одною, щоб отримати кінцевий результат, тоді як пакетування навчає їх паралельно.

Обрізка. При налаштуванні моделі можна визначити кілька параметрів чи об’єктів які мають вплив на остаточний результат прогнозу. Вибірка об’єктів визначає найважливіші функції в навчальному наборі та усуває непотрібні. Наприклад, щоб передбачити, чи є зображення твариною або людиною, можна подивитися на різні вхідні параметри, такі як форма обличчя, положення вух, структура тіла. Можна віддавати перевагу формі обличчя і ігнорувати форму очей.

Доповнення даних. При навчанні моделі дані трохи зазнають змін, кожен раз як модель їх обробляє. При помірному збільшенні даних навчальні набори виглядають унікальними для моделі і дозволяють моделі вивчати їх характеристики. Наприклад, застосування перетворень, таких як переміщення, відображення та поворот, до вхідних зображень.

Говорячи про перенавчання треба звернути увагу і на недонавчання. Недодовчання – це вид помилки, коли модель не може визначити зв’язок між

вхідними та вихідними даними. Як результат отримуються неточні моделі, якщо вони не навчалися протягом відповідного періоду часу на великій кількості точок даних.

Невідповідні моделі характеризуються високим ступенем усунення – вони дають неточні результати як для тренувальних даних, так і для тестових. З іншого боку перенавчені моделі мають високу дисперсію – вони дають точні результати для тренувального набору, але не для тестового набору. Більше ретельне навчання моделі призводить до меншої похибки, але дисперсія може збільшитися. Тож треба знайти золоту середину між недонавчанням та перенавчанням при підгонці моделі. Добре підігнана модель може швидко встановити домінуючу тенденцію для видимих та невидимих наборів даних.

1.2.1 Складові машинного навчання

Машинне навчання має три основних складових на яких все тримається.

Перша складова це дані, інакше кажучи набори даних на яких навчається модель. Чим більша вибірка, тим краще і точніша буде модель, але кількість не означає якість.

Самі дані залежать від поставленої задачі. Найважливіший етап – збір цих даних. Він може відбуватися вручну або автоматично. Перший спосіб повільний але у результаті більш точніший. Автоматичний спосіб швидкий але дані не завжди виходяться точними, присутні шуми , викиди, аномальні значення. Тому треба буде приділити особливу увагу підготовці даних, їх очищенню та аналізу.

Через це хороший набір даних є цінним ресурсом, адже від якості буде залежати результат. Важливо комбінувати способи збору даних, надавати розрізнену інформацію.

Другою складовою є ознаки: характеристики, метрики та властивості які є головними для вирішення поставленої задачі. Оскільки правильність властивостей безпосередньо впливає на результат, який отримується, їх відбір займає часто

більше часу, ніж сам процес навчання. Тут головне – не обмежувати набір характеристик, виходячи з особистої думки, щоб не спотворити машинне сприйняття. А разом з ним і кінцевий результат.

Третьою складовою є алгоритм, вибір якого залежить від поставленої задачі. Від вибору алгоритму буде залежати точність моделі та її швидкість навчання.

1.2.2 Види машинного навчання

У загальному розумінні машинне навчання поділяється на два види: навчання на прецедентах (індуктивне навчання) та дедуктивне.

Перший вид має три типи: контрольоване навчання, або навчання з учителем, неконтрольоване навчання, або навчання без учителя, і навчання з підкріпленням.

Другий вид відносять до експертних систем. Бази знань експертних систем важко поєднувати із реляційною моделлю даних, тому СУБД майже неможливо ефективно використовувати для наповнення баз знань.

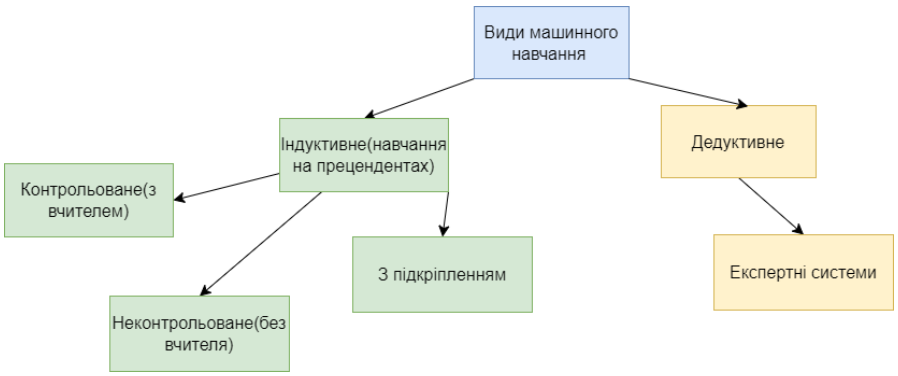

Рисунок 1.1 – види машинного навчання

Навчання з вчителем.

Для навчання використовується набір даних які містить не тільки вектор незалежних змінних(атрибутів, ознак), але і значення, яке повинна видавати модель після навчання(цільове значення). Відмінність між цільовими та фактичними(результат роботи навченої моделі) значеннями називається помилкою

навчання, яка під час навчання мінімізується і є “вчителем”. Значення помилки в подальшому використовується для обчислення корекцій параметрів моделі на кожній ітерації. Цей тип навчання використовують частіше, бо модель вчиться швидко та якісно.

Даний тип використовується у задачах класифікації та кластеризації. У першому випадку у якості цільової змінної використовується мітка класу, у другому числова змінна.

Також використовуються різні алгоритми для різних задач. Наприклад для задач кластеризації можна використати дерево рішень, метод k найближчих сусідів. А для задач регресії використовують лінійну та логістичну регресії. Але це ділення не є обов’язковим, бо кожен алгоритм можна оптимізувати під кожну задачу.

Навчання без вчителя.

У цьому навчанні набір даних не має цільової функції. Тобто дані не розмічені і модель повинна сама знайти зв’язки між даними, виявити наявні закономірності, виконати класифікацію даних. Такий підхід використовується рідше і зазвичай для аналізу та підготовки даних. Добре розмічені дані це рідкість і для їх розмітки використовують сторонні сервіси або алгоритми(машинне навчання не виключення).

Навчання без вчителя використовує принцип «ця річ така ж, як інші». Алгоритми вивчають подібності і можуть виявити відмінність і виконати виявлення аномалій, розпізнаючи, що є незвичайним або несхожим.

У випадках, коли розмітка даних неможлива, вдаються до методів навчання без учителя. До таких алгоритмів належать задачі кластеризації, зменшення розмірності і пошуку правил.

Навчання з підкріпленням.

Є окремим видом навчання з вчителем, але у ролі вчителя виступають не дані, а середовище у якому все відбувається. Модель яка навчається, в даному випадку є агентом, не має ніякої інформації про середовище але може виконувати будь-які

дії. Середовище у свою чергу реагує на дії агента і надає йому дані, що дає змогу навчатися, утворюючи систему із зворотнім зв’язком.

Це навчання дуже схоже на навчання людей, а скоріше дітей, які пізнають нове середовище та намагаються взаємодіяти з усім що бачать. У результаті деякі дії заохочуються, деякі караються. Найчастіше використовується в системах навігації для роботів, які навчаються уникати зіткнень з перешкодами шляхом набуття досвіду, отримуючи зворотний зв’язок при кожному зіткненні. Також в логістиці, при складанні графіків і плануванні завдань.

1.3 Системи виявлення аномального трафіку

1.3.1 Cyber AI Analyst

Cyber AI Analyst, продукт дослідницької ініціативи науково-дослідного центру Darktrace у Кембриджі, був створений для посилення команд безпеки та оптимізації розслідування загроз. Він постійно досліджує кожну подію, що виникає в імунній системі підприємства Darktrace, імітуючи експертні розумові процеси людини для автономного сортування та складання звітів.

Ця технологія поєднує в собі інтуїцію експерта-аналітика з послідовністю, швидкістю і масштабованістю штучного інтелекту. Він у будь-який момент часу виявляє загрози з найвищим пріоритетом і швидко синтезує весь контекст атаки в зручний для читання звіт.

Застосовуючи поєднання контрольованого і неконтрольованого машинного навчання, а також методів глибокого навчання і просунутої математики, Cyber AI Analyst може виконувати більшу частину важкої роботи, яку в іншому випадку довелося б виконувати людині. Він використовує інформацію, отриману від експертів світового класу Darktrace за роки розслідування загроз, для прийняття високоточних рішень.

Ключова особливість підходу, заснованого на імунній системі, і підтримувана штучним інтелектом, Cyber AI Analyst може аналізувати великі

обсяги даних зі швидкістю і в масштабі, розширюючи людські команди і економлячи час, щоб зосередитися на стратегічній роботі. Його розслідування проводяться в масштабах всього підприємства, що дає змогу технології збирати докупи розрізнені аномалії, перш ніж дійти висновку високого рівня про природу, основну причину і масштаб ширшого інциденту безпеки.

Cyber AI Analyst застосовує різні форми штучного інтелекту, включно з глибоким навчанням, а також контрольованим машинним навчанням на постійно зростаючому наборі даних, що відображає, як фахівці-аналітики Darktrace досліджують загрози. Володіючи цими знаннями, аналітик Cyber AI може зрозуміти, які загрози найважливіші для розслідування, які події є пов'язаним інцидентом і як слід керувати атакою. Це потужна можливість, враховуючи, що Імунна система підприємства часто виявляє складні та нові загрози, які застарілі інструменти не можуть ідентифікувати. Cyber AI Analyst надає експертний аналіз усіх типів кіберзагроз, навіть тих, що характеризуються інноваційними методами атак, які неможливо виявити та відреагувати на них за допомогою заздалегідь визначених сценаріїв.

Сама система базується на чотирьох компонентах, сама схема продемонстрована на рисунку 1.2.

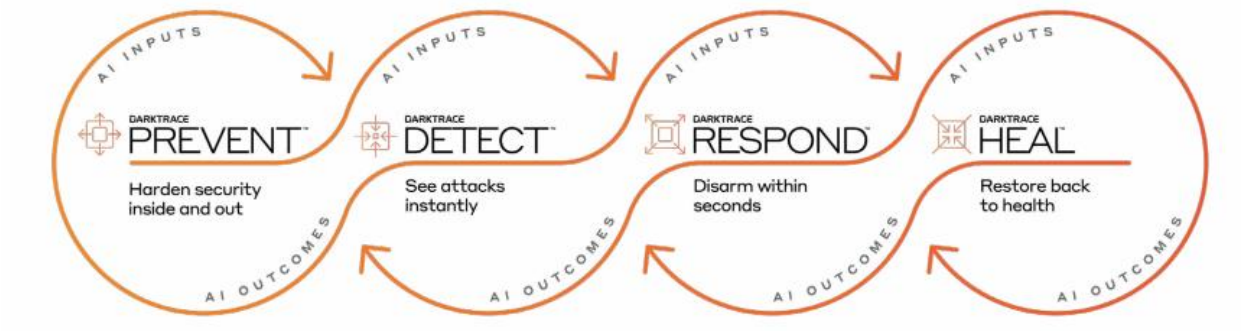

Рисунок 1.2 – Структура Cyber AI Analyst

Компонент PREVENT. Оцінює ризики з якими може стикнутися система, надаючи можливість завчасно підготуватися до атаки. Визначає та встановлює пріоритетність важливих цілей та шляхів захисту важливих компонентів системи.

Компонент DETECT. Вивчає систему в яку встановлений. Аналізує користувачів, їх дії у системі, пристрої та взаємозв'язки між ними. Повністю проаналізувавши систему може виявити найтонші відхилення від нормальної діяльності, які вказують на нові загрози.

Компонент RESPOND. Відповідає блокування атак. Наприклад заблокувати певне з'єднання через порт замість того, щоб помістити пристрій на карантин. Це дозволяє бізнес-операціям працювати безперебійно залишаючись при цьому безпечними. Також є можливість налаштувати роботу: поставити певний період у який буде працювати чи налаштувати на які події реагувати.

Компонент HEAL. Використовує штучний інтелект для розуміння даних системи, щоб забезпечити готовність до відновлення після активної кібератаки та швидкого відновлення системи.

1.3.2 Palo Alto Cortex XDR

Це вдосконалений додаток для виявлення та реагування, який може інтегрувати дані з кінцевих точок, мережі та хмарних сховищ для виявлення, запобігання та зупинки відомих і невідомих атак шкідливих програм на організацію. Ця система розроблена з використанням аналітики поведінки та штучного інтелекту.

Має єдине вікно контролю, що дозволяє отримати 360-градусний огляд IT-інфраструктури з однієї панелі. Завдяки цьому перевіряти системи, файли і процеси в хмарі, мережі і на кінцевих точках дуже просто. Оскільки наявна єдина інформаційна панель, можливо зупинити процес, усунути код або ізолювати кінцеву точку одним натисканням миші. Контроль над всією організацією з однієї

панелі дозволяє SOC-команді реагувати на загрози швидше, ніж це відбувається зазвичай.

Ця система безпеки надає чітку картину інциденту. Вона інформує де у бізнес-системі ховається вразливість. І не тільки це, є можливість перехресно перевірити дані телеметрії, щоб визначити першопричину проблеми. Крім того, виконується автоматичний аналіз.

Щодня команді SOC доводиться розслідувати безліч тривог від різних систем безпеки. Більшу частину часу вони витрачають на аналіз кожної тривоги, і більшість з них є хибно позитивними. XDR від Palo Alto добре справляється з цією проблемою, зменшуючи кількість сповіщень до 90 відсотків. Цей інструмент розроблений на основі уніфікованого механізму інцидентів. Він співвідносить сповіщення з усіх порталів, а потім надсилає розумні групові сповіщення.

Основні функції XDR наведені нижче.

Збір та кореляція даних. Збирає всі дані з кінцевих точок, серверів, хмарних станцій, мережі та інших місць. Всі ці дані зберігаються і збираються в режимі реального часу в єдиній консолі.

Поглиблений аналіз. Система та SOC-команда аналізують ці дані. Оскільки ці дані генеруються з усіх джерел у організації, це покращує видимість загроз і скорочує час на розслідування. Припустимо, що на організацію сталася атака. Тоді треба перевірити Cortex XDR Dashboard і дізнатися, чи сталася ця атака лише на одну кінцеву точку, чи поширилася на мережеві або хмарні файли. Через це не потрібно проводити дослідження вручну, оскільки система надсилає сповіщення про виявлення шкідливого файлу або атаки.

Реагування та сповіщення. Кожен продукт безпеки надсилає певні сповіщення, згруповані на основі пов'язаних подій. Таким чином, команда має справу з меншою кількістю сповіщень і більшою кількістю інформації. Palo Alto XDR розроблено з використанням інструментів штучного інтелекту та аналізу поведінки для профілювання активності та поведінки користувачів, щоб виявити

підозрілу активність. Машинне навчання дозволяє легко виявляти та зупиняти невидимі загрози.

Швидке розслідування. Аналіз першопричин - найкраща функція Cortex XDR. Вона дозволяє SOC-команді зрозуміти чітку картину атаки, щоб прискорити розслідування і блискавично реагувати на відомі та невідомі загрози.

1.4 Постановка задачі

З огляду на вищесказане було прийнято рішення досліди процес виявлення кіберзагроз із використанням класифікаційних алгоритмів машинного навчання. А саме дослідити бінарну і багатокласову класифікацію. Для виконання поставленого завдання потрібно:

- проаналізувати класичні засоби виявлення кіберзагроз;
- проаналізувати основні теоретичні засади машинного навчання;
- дослідити алгоритми класифікації;
- підібрати набір даних, вивчити його та підготувати для навчання моделей;
- навчити та протестувати моделі для бінарної та багатокласової класифікації;
- систематизувати отримані результати.

Висновки до розділу 1

У розділі було розглянуто і проаналізовано сучасний стан засобів виявлення кіберзагроз та взаємозв’язок цих засобів із машинним навчанням. Виявлено, що в сучасному інформаційному середовищі дедалі більше уваги приділяється застосуванню методів машинного навчання для виявлення та протидії кіберзагрозам.

Надано базову теорію про машинне навчання, де було розглянуто складові та види МН.

Також були проаналізовані системи виявлення аномального трафіку, такі як Cyber AI Analyst та Palo Alto Cortex XDR, які використовують різноманітні технології машинного навчання для ефективного виявлення та аналізу загроз.

В рамках постановки задачі були визначені основні цілі та завдання дослідження, спрямовані на вдосконалення методів виявлення кіберзагроз за допомогою методів машинного навчання.

2 АЛГОРИТМИ ТА МЕТРИКИ ОЦІНКИ МОДЕЛЕЙ

2.1 Логістична регресія

Цей тип статистичної моделі часто використовується для класифікації та прогнозної аналітики. Логістична регресія оцінює ймовірність виникнення події, наприклад, голосувати чи не голосувати, на основі заданого набору даних незалежних змінних. Оскільки результат є ймовірністю, залежна змінна обмежена діапазоном від 0 до 1. У логістичній регресії до шансів застосовується логіт-перетворення, тобто ймовірність успіху, поділена на ймовірність невдачі. Це також широко відоме як логарифм шансів або натуральний логарифм шансів, і ця логістична функція представлена такими формулами:

$$Logit(pi) = \frac{1}{1+\exp(-pi)} \tag{2.1}$$

$$\ln\left(\frac{pi}{1-pi}\right) = B_o + B_1 * X_1 + \ldots + B_k * K_k$$

де $Logit(pi)$ – є залежною змінною або змінною відгуку;

X – незалежна змінна;

B – коефіцієнт, у цій моделі зазвичай оцінюється за допомогою оцінки максимальної правдоподібності.

Всі ці ітерації створюють функцію правдоподібності, а логістична регресія намагається максимізувати цю функцію, щоб знайти найкращу оцінку параметра. Після того, як оптимальний коефіцієнт (або коефіцієнти, якщо є більше однієї незалежної змінної) знайдено, умовні ймовірності для кожного спостереження можна обчислити, записати в журнал і підсумувати разом, щоб отримати прогнозовану ймовірність. Для бінарної класифікації ймовірність, менша за 0,5, означає 0, а більша за 0 - 1.

Існує три типи моделей логістичної регресії, які визначаються на основі категоріальної відповіді.

Бінарна логістична регресія. У цьому підході відповідь або залежна змінна є дихотомічною за своєю природою - тобто вона має лише два можливих результати (наприклад, 0 або 1). Деякі популярні приклади його використання включають прогнозування того, чи є електронний лист спамом, чи ні, або чи є пухлина злоякісною чи не злоякісною. У логістичній регресії це найпоширеніший підхід, а в більш загальному плані - один з найпоширеніших класифікаторів для бінарної класифікації.

Мультиноміальна логістична регресія. У цьому типі моделі логістичної регресії залежна змінна має три або більше можливих результатів, однак ці значення не мають визначеного порядку. Наприклад, якщо ми розглядаємо мультиноміальну логістичну регресію для виявлення різних типів кіберзагроз (наприклад, атаки типу DoS, атаки типу SQL Injection, атаки типу Phishing тощо), модель може враховувати різні фактори (ознаки), такі як характеристики мережі, підозрілі патерни трафіку, інформацію про аутентифікацію тощо. Однак у цьому випадку класи або види кіберзагроз не мають визначеного порядку, і модель дозволяє оцінити ймовірність належності кожного об'єкта (в даному випадку, інциденту кіберзагрози) до різних класів.

Порядкова логістична регресія. Цей тип моделі логістичної регресії використовується, коли змінна відгуку має три або більше можливих результатів, але в цьому випадку ці значення мають певний порядок. Прикладами порядкових відповідей є шкали оцінювання від A до F або рейтингові шкали від 1 до 5.

2.2 Метод опорних векторів

Відноситься до навчання з вчителем, який використовується у задачах класифікації. Являє собою лінійний класифікатор, який дозволяє створювати гіперплощину у векторному просторі, щоб розділити два класи у заданому наборі

даних. Метод опорних векторів використовує функцію залежних втрат, яка штрафує точки розташовані на не правильній стороні відносно гіперплощини або дуже близькі до неї і знаходяться на правильній стороні. SVM-класифікатор намагається знайти максимальну гіперплощину, що розділяє два класи, де грань(margin) є відстанню від роздільної площини до ближніх точок на кожній стороні. Коли дані розділені не прямою лінією, точки в середині границі штрафуються пропорціонально їх віддаленості від границі, що проілюстровано на рисунку 2.1.

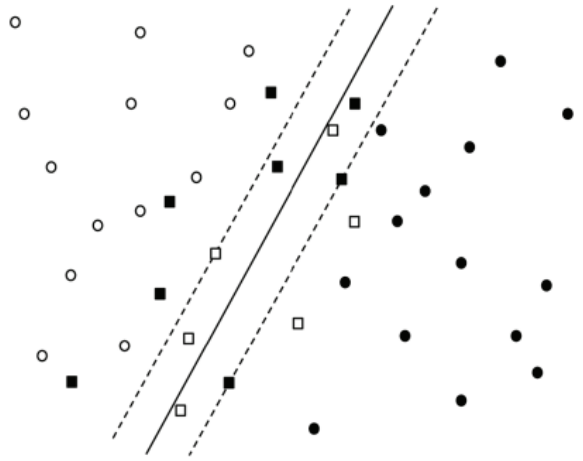

Рисунок 2.1 – Графік SVM

На рисунку 2.1 зображено два класи у вигляді точок(білих та чорних). Суцільна лінія – роздільна площина, а штрихові – границі. Квадратні точки це опорні вектори, тобто вектори які вносять нульовий вклад у функцію втрат. Функція втрат має наступний вигляд:

$$\beta + C \sum_{i=1}^{N} \xi_i \qquad (2.2)$$

де β-границя;

C-гіперпараметр моделі який визначає відносний вклад двох членів формули;

ξ_i-відстань від вектору до границі.

Для класифікації нової точки треба визначити яку сторону роздільної площини вона розміщується. При необхідності отримати оцінку реального значення обраховують відстань від точки до роздільної площини і потім застосовують сигмоідну функцію для відображення інтервалу [0,1].

Метод опорних векторів розкривається на повну за допомогою “трюку з ядром”. Це математичне перетворення, яке на вході приймає лінійну границю рішення і на виході видає нелінійну границю. На найвищому рівні ядро виконує перетворення одного векторного виміру V_1 у інше V_2. З математичної точки зору ядро є функцією $V_1 \times V_2$ і визначається як:

$$K(x, y) \tag{2.3}$$

Кожне значення $x \in V_1$ відображається яка функція $K(x)$, а V_2- простір, охвачений всіма цими функціями. Можна відновити лінійне SVM-простір, визначивши функцію як добуток змінних:

$$K(x, y) = x \times y \tag{2.4}$$

У випадку нелінійного ядра, лінійний класифікатор у просторі V_1 буде перетворений в нелінійний класифікатор у просторі V_2. Найчастішим вибором є радіально-базисна функція:

$$K(x, y) = e^{-y|x-y|} \tag{2.5}$$

SVM із ядерною функцією RBF можна уявити як результат згладженою лінійною комбінацією сфер навколо кожної точки x. При цьому простір всередині кожної такої сфери класифікується так само як x, а простір за межами сфери відноситься до протилежного класу. Параметр у визначає радіус сфери або максимальну відстань на якій повинна знаходитись точка, щоб класифікуватися як точка центру сфери. Параметр C визначає ступінь згладжування. Велике значення параметра C призведе до того, що класифікатор буде складатися з об'єднаних сфер, тоді як мале значення C дає більш звивисту межу, на яку тією чи іншою мірою впливає кожна сфера. Таким чином, можна зрозуміти, що при занадто великому значенні параметра C створюється модель із перепідгонкою тренувальних даних, а при занадто малому значенні C класифікатор буде слабким з погляду точності. Параметр у для кожного RBF ядра є єдиним у своєму роді, тоді як параметр C визначає одні й ті самі властивості для будь-якого ядра, зокрема і для лінійного методу опорних векторів. Оптимальні значення цих параметрів зазвичай обчислюються з використанням пошуку по решітці.

2.3 Наївний байєсівський класифікатор

Є одним із найстаріших статичних класифікаторів. Оснований на суворих статистичних вихідних припущеннях, а саме: ознаки обирають незалежно з деякого (невідомого заздалегідь) розподілу, але таке вихідне припущення ніколи не зустрічається у реальному житті. Розглянемо класифікатор спаму, для якого ознаками є слова в повідомленні. Вихідне припущення наївного байєсівського класифікатора стверджує, що повідомлення зі спамом складається з незалежно обраних слів, при цьому кожне слово w може бути вибрано з імовірністю $p_{w,spam}$. Аналогічне вихідне припущення робиться для звичайних, «правильних» повідомлень. Таке вихідне припущення абсолютно абсурдне, оскільки насамперед воно повністю ігнорує порядок слів у тексті. Але, незважаючи на цю очевидну безглуздість вихідного припущення, наївні байєсівські класифікатори

продемонстрували досить високу ефективність під час розв'язання таких завдань, як класифікація спаму [23].

Основа ідея NBC – взявши точку(елемент) даних із набору ознак

$X = x_1 \dots, x_n$, необхідно визначити вірогідність того, що мітка Y для цієї точки представляє клас C. Нижче ця концепція виражена у вигляді формули вірогідності:

$$\Pr[Y = C \mid X = (x_1 \dots, x_n)] \tag{2.6}$$

Використовуючи формулу Байєса, цю вірогідність можна представити у наступному вигляді:

$$\frac{\Pr\left[X = (x_1 \dots, x_n) \middle| Y = C\right] \times \Pr[Y=C]}{\Pr[X=(x_1 \dots, x_n)]} \tag{2.7}$$

Якщо для вибірки даних у кожному класі ознаки елементів обираються незалежно один від одного, формула матиме вигляд:

$$\Pr[X = (x_1 \dots, x_n) | Y = C] = \prod_{i=1}^{n} \Pr[X_i = x_i | Y = C] \tag{2.8}$$

Можна обчислити оцінку чисельника у формулі 7 за даними з мітками: $\Pr[X_i = x_i | Y = C]$- це частина елементів вибірки із і-ознакою, рівна x_i із всіх елементів вибірки класу C, при цьому $\Pr[Y = C]$-частина елементів вибірки класу C із усіх помічених елементів вибірки.

Розглядаючи знаменник формули 7, немає необхідності його обраховувати, бо при класифікації за двома класами достатньо обрахувати відношення оцінок вірогідності для двох класів C_1 та C_2. Це відношення дає оцінку у формі позитивного цілого числа:

$$\theta = \frac{\Pr[Y=C_1|X=(x_1 \ldots, x_n)]}{\Pr[Y=C_2|X=(x_1 \ldots, x_n)} \approx \frac{\Pr[Y=C_1]\prod_{i=1}^{n} X_i=x_i|Y=C_1]}{\Pr[Y=C_2]\prod_{i=1}^{n} X_i=x_i|Y=C_2]} \quad (2.9)$$

Оцінка $\theta > 1$ говорить про те, що C_1 є більш вірогідним класом, тоді як оцінка $\theta < 1$ говорить що клас C_2 більш вірогідний.

Треба зауважити, що оцінка θ обрахована без функції втрат або алгоритму оптимізації. Насправді алгоритм оптимізації прихований у виразі оцінки вірогідності $\Pr[X_i = x_i|Y = C]$. Використання частини елементів вибірки з набору тренувальних даних дає оцінку за методом максимальної правдоподібності, тобто ту саму функцію втрат, яка застосовується для логістичної регресії. На цьому схожість із логістичною регресією не закінчується: якщо взяти логарифми виразів у формулі 9, то права частина стає лінійною функцією досліджуваних ознак. На цій підставі наївний байєсівський метод можна розглядати ще й як лінійний класифікатор.

При використанні NBC треба враховувати декілька тонкощів:

– коли оцінка θ стає рівною нулю або нескінченності використовується згладжування, при якому додаються фіктивні(фантомні) елементи в позначені дані для кожної ознаки. Наприклад, якщо згладжування виконується за допомогою множника α, то значення для будь-якої ознаки слід обчислювати наступним чином:

$$\Pr[X_i = x_i|Y = C] = \frac{(\text{кількість елементів у класі C із ознакою } x_i)+\alpha}{(\text{кількість елементів у класі C})+\alpha\times(\text{кількість ознак})} \quad (2.10)$$

При $\alpha = 1$, метод називається згладжуванням Лапласа. При $\alpha < 1$, то це згладжування Лідстоуна.

– коли ознака x_i з'являється у валідаційному наборі але не з'являлась у тренувальному наборі, не отримується оцінка для $\Pr[X_i = x_i|Y = C]$. Найпростіша оцінка визначається як вірогідність для $\Pr[Y = C]$.

2.4 Дерева рішень

Дерева рішень є універсальними та гнучкими моделями навчання з вчителем, які мають важливу властивість – простота інтерпретації. Цей алгоритм є структурою у вигляді бінарного дерева для прийняття рішень. Дерева рішень надають можливість прогнозування як категоріальних значень(дерева класифікації), так і значення дійсних чисел(дерева регресії), а також можуть містити і числові, і категоріальні дані без операцій нормалізації або створенні фіктивних змінних.

Нижче наведено детальну процедуру створення дерев рішень, структури яка формується зверху вниз.

Починаючи з кореня дерева увесь набір ділиться на дві підмножини на основі бінарної умови. Для усіх даних умова є істиною – направляються у лівій набір-потомок, в іншому випадку у правий набор-потомок.

Далі підмножини-потомки продовжують рекурсивно розділятися на основі інших умов. Умови розділення автоматично обираються на кожному кроці в залежності від того, яка умова найкращим чином розділює набір елементів. Існують декілька метрик за допомогою яких кількісно оцінюється якість розділення:

– міра неоднорідності Джині: якщо елементи в підмножині були довільним чином позначені відповідно до розподілу міток у наборі, то відносна частка неправильно позначених елементів має бути визначена як міра неоднорідності Джині. Наприклад, якщо в підмножині 25 % елементів мають мітку 0 (відповідно 75 % елементів із міткою 1), то присвоєння мітки 0 випадково обраним 25 % від усіх елементів (решті присвоюється мітка 1) має давати 37,5 % неправильних міток: 75 % елементів з міткою 0 і 25 % елементів з міткою 1 мають бути некоректними. Операція поділу дерева рішень вищої якості має розділяти набір даних на підмножини, які чітко визначаються за їхніми мітками, отже, у результаті міра неоднорідності Джині виходить нижчою. Таким чином, коефіцієнт неправильної

класифікації має бути низьким, якщо більшість елементів у наборі належить до одного й того самого класу;

– зменшення дисперсії часто використовують у деревах регресії з постійно присутньою (такою, що поширюється по дереву) залежною змінною. Зменшення дисперсії визначається як сумарне зменшення дисперсії в наборі, розділеному на дві підмножини нащадків. Найкращим варіантом поділу у вузлі дерева рішень має вважатися поділ, результатом якого є найбільше зниження дисперсії;

– приріст інформації це міра однорідності підмножин, отриманих у результаті поділу. Приріст інформації обчислюється за допомогою віднімання зваженої суми ентропії кожного вузла нащадка дерева рішень з ентропії батьківського вузла. Що менша ентропія нащадків, то більший приріст інформації, отже, якісніше виконано поділ.

Існує декілька методів для визначення моменту зупинки процедури розділення вузлів:

– коли всі листи дерева є однорідними, кожен лист вузла містить тільки елементи, які належать одному й тому самому класу, тоді поділ припиняється;

– коли чергова гілка дерева досягає деякої попередньо визначеної максимальної глибини, подальший поділ гілок припиняється;

– коли один будь-який із вузлів-нащадків містить кількість елементів, меншу, ніж мінімальна кількість елементів вибірки, такий вузол більше не підрозділяється.

В кінці алгоритм виводить структуру дерева, в якій кожен вузол представляє бінарне рішення, нащадки кожного вузла представляють два можливих вихідних результати цього рішення, а кожен лист представляє класифікацію елементів даних відповідно до шляху від кореня дерева до цього аркуша. Для неоднорідних листків рішення визначається більшістю голосів у тренувальній вибірці даних у відповідному аркуші.

Однією із властивістю дерев рішень є відносна простота пояснення класифікацій або результаті регресії, бо кожен прогноз може бути виражено у вигляді послідовності логічних умов, що дозволяє відслідкувати шлях від кореня до дерева із будь-якого вузла-листа. Наприклад, якщо модель дерева рішень передбачає, що розглянутий зразок шкідливого ПЗ належить до сімейства А шкідливих програм, то нам точно відомі причини: він не пов’язаний із конкретно визначеним вікном у робочому середовищі, яке управляє, виконує численні вихідні мережеві виклики, спрямовані на IP-адреси, які знаходяться на території визначеної країни. Оскільки кожному зразку вибірки відповідає повна висота бінарного дерева в граничному випадку (часова складність O(log n)), дерева рішень також ефективні для тренування і подальшого прогнозування. Таким чином, дерева рішень можна вважати найкращим варіантом для великих наборів даних.

В свою чергу деревам рішень властиві обмеження:

– у деревах рішень часто виникає проблема перепідгонки, коли дерева стають надмірно складними і не здатні правильно узагальнити тренувальний набір даних. Для зменшення складності дерев як регулювальна методика застосовується відсікання малоперспективних гілок);

– дерева рішень іноді менш точні та стійкі, ніж інші методики навчання з учителем. Незначні зміни в тренувальному наборі даних можуть призводити до досить істотних змін у дереві, що у свою чергу спричиняє зміни в моделі прогнозів. Це означає, що дерева рішень (і більшість інших подібних моделей) незастосовні для онлайн навчання або інкрементального навчання;

– метрики якості поділу для категоріальних змінних у деревах рішень мають тенденцію зміщуватися в бік змінних із більш імовірними значеннями, тобто поділ за безперервними змінними або за категоріальними змінними з трьома і більше категоріями призведе до вибору з вищою ймовірністю, порівняно з бінарними змінними;

– жадібний алгоритм тренування дерев рішень, який виконується майже у всіх випадках не гарантує формування оптимального дерева рішень, тому що в кожному пункті поділу створюється локально оптимальне, а не глобальне оптимальне рішення. Насправді процедура тренування глобального оптимального дерева рішень являє собою NP повну задачу;

– дерева рішень можуть втрачати свою ефективність при використанні деяких відношень. На рисунку 3 проілюстровано відношення AND,OR. А на рисунку 4 продемонстровано відношення XOR, для якого потрібен додатковий проміжний вузол і процедура для коректного представлення цієї операції. Це тільки простий приклад, у реальному наборі даних це призводить до швидкого росту складності моделі.

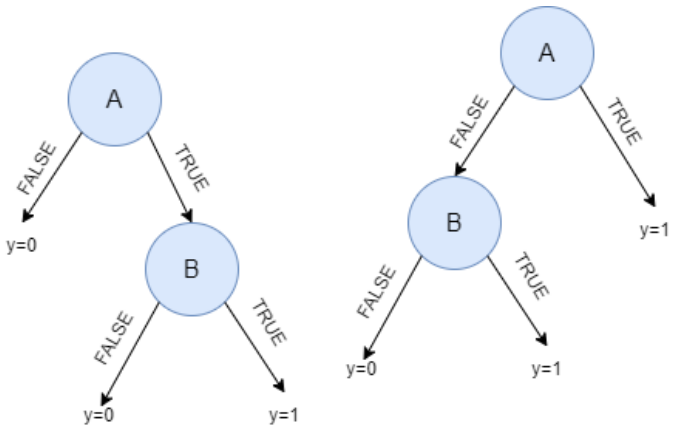

Рисунок 2.2 – Відношення AND,OR

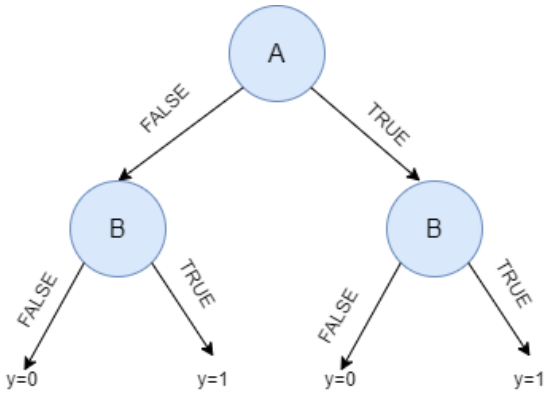

Рисунок 2.3 – Відношення XOR

2.5 Оцінка якості моделей

Для оцінки моделей використовують наступні метрики.

Середня абсолютна похибка(MAE) це різниця між фактичними та прогнозованими значеннями:

$$MAE = 1 \div n(|y - \hat{y}|) \quad (2.11)$$

де n- кількість точок даних;

y- фактичне значення;

$\hat{y}$- прогнозоване значення.

Середня квадратична похибка(MSE) є більш точною версією середньо абсолютної похибки, бо знаходить похибку за допомогою квадратичної різниці між значеннями:

$$MSE = 1 \div n\,((y - \hat{y}))^2 \quad (2.12)$$

Корінь квадратний із середньої квадратичної похибки(RMSE):

$$RMSE = \sqrt{MSE} \quad (2.13)$$

R-квадрат(R^2) отримають визначенням частки варіацій залежної змінної, яка прогнозується незалежною змінною:

$$R^2 = 1 - RSS \div TSS \quad (2.14)$$

де R^2-коефіцієнт детермінації;

RSS-сума залишків квадратів;

TSS-повна сума квадратів.

Крім вище зазначених метрик для оцінки якості класифікації використовують матрицю помилок. У процесі навчання класифікатор робить передбачення на навчальних прикладах, для яких мітка класу відома. Якщо передбачений клас відповідає фактичному, то результат класифікації вважається істинним, а в іншому випадку - хибним. Приклади позитивного і негативного класів, для яких результат передбачення істинний, називаються істинно позитивними (true-positive, TP) і істинно негативними (true-negative, TN) відповідно. Що відповідає правильно класифікованим прикладам. Приклади позитивного і негативного класів, для яких результат передбачення є хибним, називаються хибно позитивними (false-positive, FP) і хибно негативними (false-negative, FN) відповідно. На цих прикладах класифікатор припустився помилки.

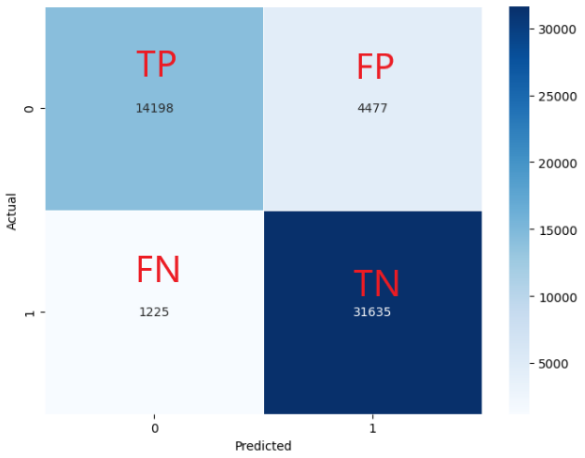

Рисунок 2.4 – Приклад матриці помилок для бінарної класифікації

Матрицю помилок можна застосовувати і для багатокласової класифікації. У цьому разі кількість рядків і стовпців у ній дорівнюватиме числу класів. Однак у цьому разі поняття від'ємного і позитивного класів у тому вигляді, в якому вони були сформульовані для бінарної моделі, не працюють. Тому в комірках матриці

ставляться не величини TP, FP, TN і FN, а кількості класифікованих відповідним чином прикладів(рис. 2.5).

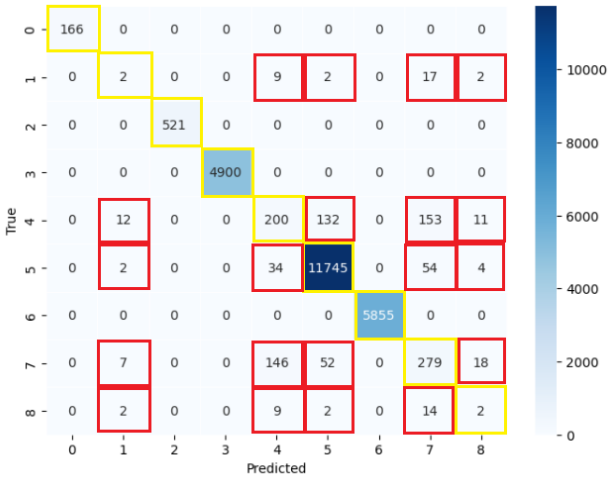

Рисунок 2.5 – Приклад матриці помилок для багатокласової класифікації

Числа, що стоять у комірках на перетині рядків і стовпчиків для однойменних класів (коли передбачений клас відповідає фактичному), визначають число правильно класифікованих прикладів. Такі комірки будуть розташовуватися на головній діагоналі матриці (помічені жовтим). Комірки, розташовані поза нею, міститимуть кількості помилково класифікованих прикладів (помічені червоним). Якщо деякі комірки матриці залишилися порожніми, це вказує на відсутність помилок моделі у відповідних комбінаціях фактичних і передбачених класів і у них записуються нулі.

За результатами матриці помилок можна обрахувати метрики які характеризують якість моделі. Дані метрики описані нижче.

Accuracy (ACC або OCR) – частка правильно класифікованих прикладів:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.15)$$

Precision (точність) – відношення числа істинно позитивних класифікацій до загального числа позитивних класифікацій. Ця величина також відома як positive predictive value (PPV) або позитивне прогностичне значення:

$$Precision = PPV = \frac{TP}{TP+FP} \quad (2.16)$$

Recall (повнота) – частка істинно позитивних прикладів (TPR) або чутливість. Визначається як число істинно позитивних класифікацій відносно загального числа позитивних прикладів:

$$Recall = TPR = \frac{TP}{TP+FN} \quad (2.17)$$

F1-score – об'єднує в собі інформацію про точність і повноту, тому дає змогу знаходити баланс між ними:

$$F1 = \frac{2*TP}{2*TP+FP+FN} \quad (2.18)$$

Висновки до розділу 2

У цьому розділі було проведено огляд методів класифікації та метрик для оцінки якості моделей у контексті виявлення кіберзагроз. Кожен з розглянутих методів та метрик відігріває важливу роль у розробці та оцінці системи виявлення кіберзагроз.

Оглянуті алгоритми класифікації виявилися потужними інструментами для вирішення завдань виявлення кіберзагроз. Кожен з них має свої переваги та обмеження.

Розглянуті метрики, дозволяють об'єктивно оцінити результати класифікації. Вони надають важливу інформацію про ефективність моделі та дозволяють зробити зважене рішення при аналізі результатів навчання моделі.

Загалом, виявлення кіберзагроз вимагає не тільки високопродуктивних алгоритмів, але й уважного вивчення та аналізу метрик для визначення ефективності та придатності моделі до реального застосування. Важливо враховувати особливості даних, щоб вибрати оптимальний метод та метрики для конкретного завдання виявлення кіберзагроз.

3 СТРУТКУРА ТА ПІДГОТОВКА НАБОРУ ДАНИХ

3.1 Структура набору даних

Набір даних UNSW-NB 15 був створений у лабораторії кіберполігону Австралійського центру безпеки із використанням інструменту IXIA PerfectStorm [1].

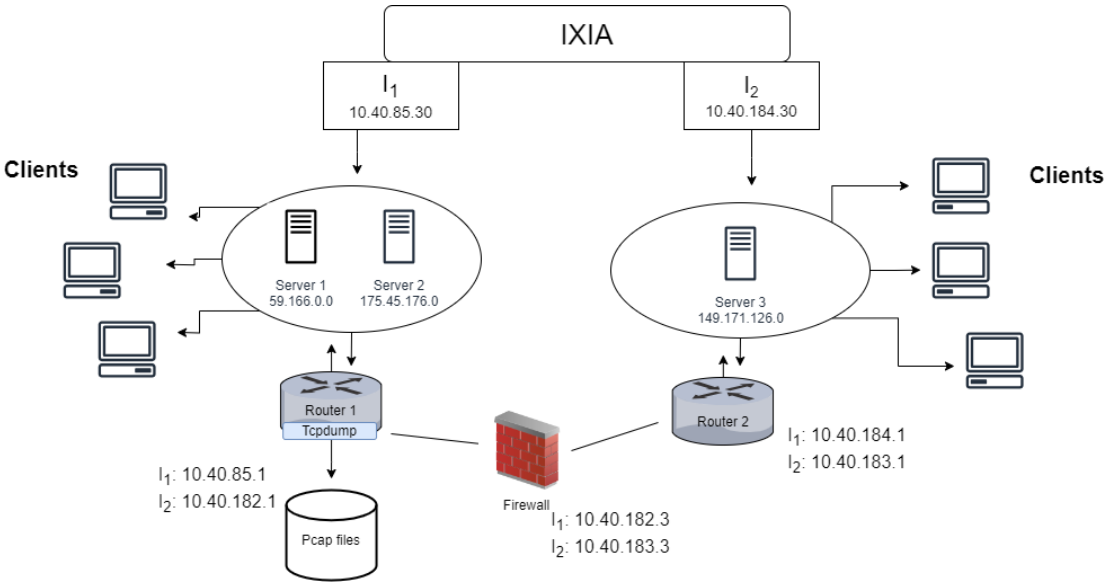

Рисунок 3.1 – Схема формування набору

Згідно із рисунком 3.1 генератор трафіку IXIA конфігурований із трьома віртуальними серверами. Сервери 1 та 3 налаштовані на нормальне розповсюдження трафіку, в той час як сервер 2 формує аномальну активність в мережевому трафіку. Для налагодження взаємодії між серверами та отримання публічного та приватного мережевого трафіку існують два віртуальних інтерфейси з IP-адресами 10.40.85.30 та 10.40.184.30. Сервери підключені до хостів через два маршрутизатори. Маршрутизатор 1 має IP-адреси 10.40.85.1 та 10.40.182.1 тоді як маршрутизатор 2 має IP-адреси 10.40.184.1 та 10.40.183.1. Ці маршрутизатори підключені до брандмауера, який налаштований пропускати весь трафік або нормальний або аномальний. Інструмент tcpdump встановлено на маршрутизаторі

1 для перехоплення Pcap-файлів під час роботи симуляції. Основним наміром всього цього процесу було зафіксувати нормальний або аномальний трафік, який походить від IXIA і розсіюється між вузлами мережі (між серверами і клієнтами). Важливо, що інструмент IXIA використовується як генератор атакуючого трафіку нарівні зі звичайним трафіком, причому поведінка атаки підживлюється з сайту CVE з метою реальної репрезентації сучасного середовища загроз.

Всього було перехоплено 100 ГБ трафіку під час двох симуляцій. IXIA був налаштований на генерацію однієї атаки в секунду під час першої симуляції, щоб захопити перші 50 ГБ. Друга симуляція налаштована на десять атак в секунду для вилучення інших 50 ГБ [2].

При виконанні симуляції на рисунку 3.1 pcap-файли генеруються за допомогою інструменту tcpdump. Далі ці дані витягуються за допомогою Argus, Bro-IDS та алгоритмів, які написані на мові програмування C#. Крім того, ці ознаки підібрані відповідно до ознак рівномірності потоку, як зазначено в таблиці 3.1.

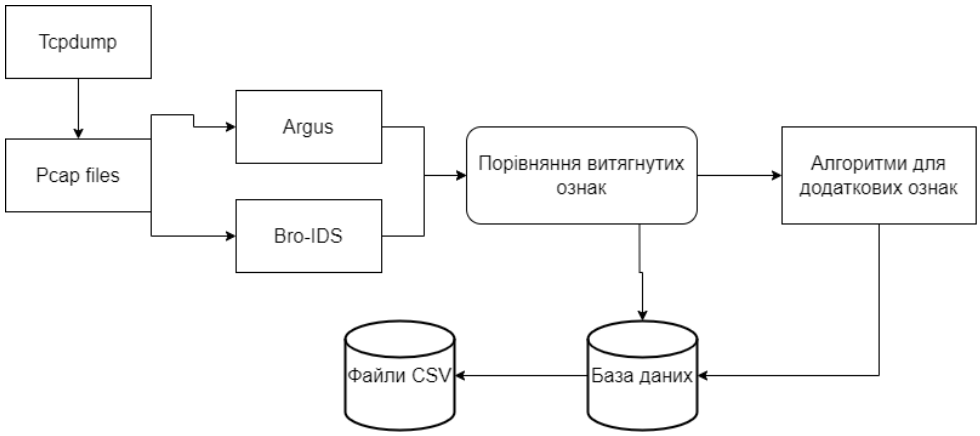

Рисунок 3.2 – Архітектура фреймоворку для створення набору даних

Інструмент Argus обробляє необроблені мережеві пакети (pcap-файли) і генерує атрибути/характеристики пакетів мережевого потоку. Інструмент Argus складається з Argus-сервера та Argus-клієнтів. Аргус-сервер записує pcap-файли

отриманих пакетів у Argus у бінарному форматі. Клієнти Argus витягують функції з Argus-файлів.

Інструмент Bro-IDS - це аналізатор мережевого трафіку з відкритим вихідним кодом. Це переважно монітор безпеки, який перевіряє весь мережевий трафік на наявність зловмисних дій. Інструмент Bro-IDS налаштований на створення трьох лог-файлів з файлів pcap: файл conn записує всю інформацію про з'єднання, що міститься в pcap-файлах; файл http включає всі HTTP-запити і відповіді; файл ftp записує всі дії сервісу FTP.

Нарешті, вихідні файли двох різних інструментів, Argus і Bro-IDS зберігаються в базі даних SQL Server, щоб відповідати ознаками, згенерованими Argus та Bro-IDS, за допомогою ознак потоку, як показано в таблиці 3.1.

Таблиця 3.1 – Ознаки потоку

№	Назва	Тип	Опис
1	srcip	N	IP-адреса джерела
2	sport	I	Номер порту джерела
3	dstip	N	IP-адреса призначення
4	dsport	I	Номер порту призначення
5	proto	N	Протокол транзакції

Ці ознаки включають різноманітні ознаки на основі пакетів. Останні допомагають при перевірці корисного навантаження, окрім заголовків пакетів. З іншого боку, для підтримання низького рівня обчислювального аналізу, замість того, щоб спостерігати за всіма пакетами, що проходять через мережеве з'єднання, розглядаються лише з'єднані пакети мережевого трафіку. Більше того, характеристики на основі потоку базуються на напрямку, часі між прибуттями та між пакетної довжини (згадані в таблицях 3.2 та 3.3). Відповідні ознаки поділяються на три групи: Основні, Змістовні та Часові, які були описані в Таблицях 3.2, 3.3 та 3.4, відповідно.

Таблиця 3.2 – Базові ознаки

№	Назва	Тип	Опис
6	state	N	Стан і залежний від нього протокол, наприклад ACC, CLO та інші
7	dur	F	Загальна тривалість запису
8	sbytes	I	Байти від джерела до місця
9	dbytes	I	Байти від місця призначення до джерела
10	sttl	I	Час життя від джерела до призначення
11	dttl	I	Час життя від місця призначення до джерела
12	sloss	I	Пакети джерела повторно передано або втрачено
13	dloss	I	Повторно передані або втрачені пакети призначення
14	service	N	http, ftp, ssh, dns
15	sload	F	Кількість вихідних бітів за секунду
16	dloat	F	Кількість біт в секунду для приймача
17	spkts	I	Кількість пакетів від джерела до призначення
18	dpkts	I	Кількість пакетів від отримувача до джерела

Важливо зазначити, що ознаки з №1 по №35 представляють інтегровану зібрану інформацію з пакетів даних. Більшість ознак генеруються із заголовків пакетів, як показано в Таблицях 3.1-3.4.

Таблиця 3.3 – Змістовні ознаки

№	Назва	Тип	Опис
19	swin	I	Джерело інформації TCP-вікна
20	dwin	I	Адресат TCP-вікна з інформацією
21	stcpb	I	Порядковий номер джерела TCP
22	dtcpb	I	Порядковий номер TCP-вікна призначення
23	smeansz	I	Середнє значення розміру пакету даних, переданого потоком src
24	dmeansz	I	Середнє значення розміру пакету даних, переданого потоком dst
25	trans_depth	I	Глибина з'єднання http транзакції запиту/відповіді
26	res_bdy_len	I	Розмір вмісту даних, переданих з http-сервісу сервера

Таблиця 3.4 – Часові ознаки

№	Назва	Тип	Опис
27	sjit	F	Джиттер джерела (мсек)
28	djit	F	Джиттер призначення (мсек)
29	stime	T	Час початку запису
30	ltime	T	Час закінчення запису
31	sintpkt	F	Час прибуття між пакетами джерела (мсек)
32	dintpkt	F	Час прибуття між пакетами до пункту призначення (мсек)
33	tcprtt	F	Сума "synack" і "ackdat" TCP.
34	synack	F	Час між пакетами SYN та SYN_ACK пакетами TCP
35	ackdat	F	Час між пакетами SYN_ACK та ACK пакетами TCP.

Таблиця 3.5 розділена на дві частини відповідно до характеру та призначення додаткових згенерованих ознак. Ознаки з 36-40 розглядаються як ознаки загального призначення, тоді як ознаки з 41-47, позначені як ознаки з'єднання. В елементах загального призначення, кожен елемент має своє призначення, відповідно до точки зору захисту, в той час як елементи з'єднання створені виключно для забезпечення захисту під час спроб з'єднання сценаріїв. Зловмисники можуть сканувати хости у примхливий спосіб: проводячи сканування раз у одну хвилину або годину. Для того, щоб ідентифікувати таких зловмисників, ознаки 36-47 призначені для сортування відповідно до ознаки останнього часу, щоб схожі характеристики записів з'єднань для кожних 100 послідовно впорядкованих з'єднань.

Таблиця 3.5 – Додаткові згенеровані ознаки

№	Назва	Тип	Опис
Ознаки загального призначення			
36	is_sm_ips_ports	B	Якщо джерело №1 тотожне одержувачу №3 IP адреси та номери портів №2, №4 рівні, ця змінна приймає значення 1, інакше 0
37	ct_state_ttl	I	Кількість для кожного стану №6 відповідно до конкретного діапазону значень для часу, який залишився до кінця життя №10, №11
38	ct_flw_http_mthd	I	Кількість потоків, які мають методи Get та Post у http-сервісі.
39	is_ftp_login	B	Якщо доступ до ftp-сесії здійснюється під користувачем і паролем то 1, інакше 0.
40	ct_ftp_cmd	I	Кількість потоків, які мають команду у сесії ftp.
Ознаки з'єднання			
41	ct_srv_src	I	Кількість з'єднань, що містять однаковий сервіс №14 та адресу джерела №1 у 100 з'єднань за останній час №26
42	ct_srv_dst	I	Кількість з'єднань, що містять однакові послуги №14 та адресу призначення №3 у 100 з'єднаннях за останній час №26
43	ct_dst_ltm	I	Кількість з'єднань з однаковою адресою призначення №3 у 100 з'єднаннях за останній час №26
44	ct_src_ ltm	I	Кількість з'єднань одного джерела адреси №1 у 100 з'єднаннях за останній час №26
45	ct_src_dport_ltm	I	Кількість з'єднань з однаковою вихідною адресою №1 та портом призначення №4 за 100 з'єднаннях за останній час №26
46	ct_dst_sport_ltm	I	Кількість з'єднань з однаковою адресою призначення №3 та портом джерела №2 за 100 з'єднань за останній час №26
47	ct_dst_src_ltm	I	Кількість з'єднань з однією і тією ж адресою джерела №1 та адресою призначення №3 за 100 з'єднань за останній час №26

Останні ознаки у наборі відповідають за тип атаки та мітку, більш детальна інформація відображена у таблиці 3.6.

Таблиця 3.6 – Мітки

№	Назва	Тип	Опис
48	attack_cat	N	Назва категорії атаки
49	label	B	0 для нормальних, 1 для атак

У таблиці 3.7 надана інформація про всі типи атак які зустрічаються у наборі.

Таблиця 3.7 – Типи атак

Тип атаки	Опис
Normal	Нормальні дані
Fuzzers	Спроба викликати призупинення роботи програми або мережі шляхом подачі їй випадково згенерованих даних
Analysis	Містить різні атаки на порти сканування, проникнення через спам і html-файли
Backdoors	Атака при якій механізм безпеки системи приховано обходиться задля тримання доступу до комп’ютера чи його даних
DoS	Зловмисна спроба зробити сервер або мережевий ресурс недоступним для користувачів, зазвичай шляхом тимчасового переривання або призупинення послуг хоста, підключеного до Інтернету
Exploits	Зловмисник знає про проблему безпеки в операційній системі або програмного забезпечення і використовує ці знання для доступу к даним
Generic	Метод працює проти всіх блокових шифрів (з заданим розміром блоку і ключа), без урахування структуру блок-шифру
Reconnaissance	Містить всі атаки, які можуть імітувати атаки, що збирають інформацію
Shellcode	Невеликий шматок коду, який використовується в якості корисного навантаження при використанні програмної вразливості
Worms	Зловмисник реплікує себе, щоб поширюватися на інших комп'ютерах. Зазвичай використовується комп'ютерна мережа для розповсюдження себе, покладаючись на збої в системі безпеки на цільовому комп'ютері, щоб отримати доступ до нього

3.2 Підготовка набору даних

Завантажимо та переглянемо набір даних.

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	0	
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	0	
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	0	
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	1	
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
175336	175337	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	13	24	0	0	
175337	175338	0.505762	tcp	-	FIN	10	8	620	354	33.612649	...	1	2	0	0	
175338	175339	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	3	13	0	0	
175339	175340	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	14	30	0	0	
175340	175341	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	16	30	0	0	

175341 rows × 45 columns

Рисунок 3.3 – Вміст набору даних

Як видно набір даних містить 175341 записів та 45 колонок. Але видно що у колонці «service» є пропуски, перетворимо ці записи на пусті та переглянемо скільки всього пустих значень у наборі.

```
id                    0
dur                   0
proto                 0
service           94168
state                 0
spkts                 0
dpkts                 0
sbytes                0
dbytes                0
rate                  0
sttl                  0
dttl                  0
sload                 0
dload                 0
sloss                 0
dloss                 0
sinpkt                0
dinpkt                0
sjit                  0
djit                  0
swin                  0
stcpb                 0
dtcpb                 0
dwin                  0
tcprtt                0
synack                0
ackdat                0
smean                 0
dmean                 0
trans_depth           0
response_body_len     0
ct_srv_src            0
ct_state_ttl          0
ct_dst_ltm            0
ct_src_dport_ltm      0
ct_dst_sport_ltm      0
ct_dst_src_ltm        0
is_ftp_login          0
ct_ftp_cmd            0
ct_flw_http_mthd      0
ct_src_ltm            0
ct_srv_dst            0
is_sm_ips_ports       0
attack_cat            0
label                 0
dtype: int64
```

Рисунок 3.4 – Кількість пустих значень

Пусті значення є тільки у колонці «service», тому видалимо їх.

Для кращого розуміння зобразимо кількість атак, та їх типів(див. рис. 3.5-3.6).

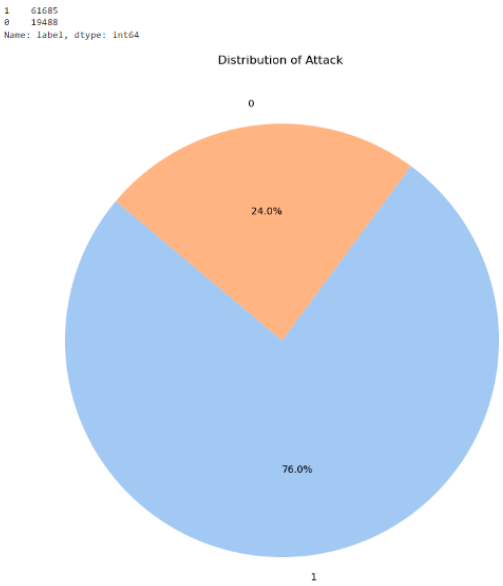

Рисунок 3.5 – Кругова діаграма кількості атак

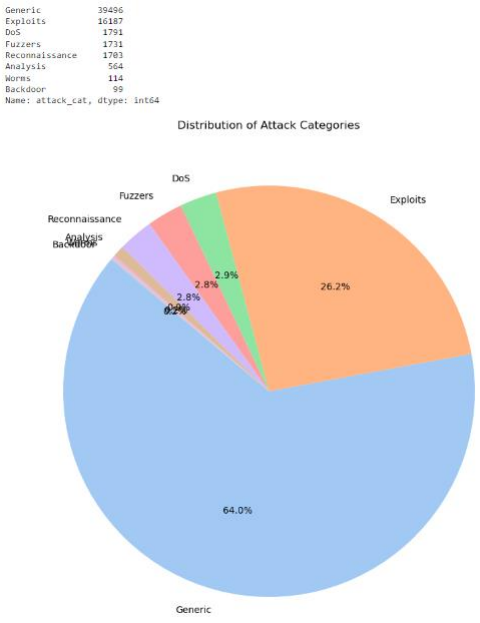

Рисунок 3.6 – Кругова діаграма кількості типів атак

З попередніх рисунків видно, що із всіх даних тільки 24% є атаками або 61685 атак. Із них найбільше «Generic» (64% або 39496), а найменше «Backdoor» (0.2% або 99).

Наступним етапом є кодування категоріальних ознак. Основна ідея - розширити кожну унікальну категорію у вектор, де кожен елемент відповідає одній категорії, і встановити значення 1 для категорії, яку спостерігаємо, і 0 для всіх інших категорій. Задля цього виокремимо категоріальні ознаки у окремий набір даних.

	proto	service	state
3	tcp	ftp	FIN
11	tcp	smtp	FIN
15	udp	snmp	INT
17	tcp	http	FIN
21	tcp	http	FIN

Рисунок 3.7 – Категоріальні ознаки

Тепер за допомогою функції get_dummies() виконаємо кодування, після чого додамо нові стовпці до основного набору даних, при цьому старі стовпці обов’язково видалимо.

	proto_tcp	proto_udp	service_dhcp	service_dns	service_ftp	service_ftp-data	service_http	service_irc	service_pop3	service_radius	service_smtp	service_sn
3	1	0	0	0	1	0	0	0	0	0	0	
11	1	0	0	0	0	0	0	0	0	0	1	
15	0	1	0	0	0	0	0	0	0	0	0	
17	1	0	0	0	0	0	1	0	0	0	0	
21	1	0	0	0	0	0	1	0	0	0	0	

Рисунок 3.8 – Результат кодування

Нормалізація даних є важливим етапом в підготовці даних для алгоритмів машинного навчання. Основна мета нормалізації - привести дані до стандартного масштабу або діапазону значень, щоб уникнути проблем в роботі з алгоритмами,

які чутливі до різниць в масштабі чи величині ознак. Виокремимо числові ознаки та за допомогою MinMax Scaler виконаємо нормалізацію.

```
num_col = list(data.select_dtypes(include='number').columns)
num_col.remove('id')
num_col.remove('label')
print(num_col)
```

```
['dur', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'rate', 'sttl', 'dttl', 'sload', 'dload', 'sloss', 'dloss', 'sinpkt', 'dinpkt',
'sjit', 'djit', 'swin', 'stcpb', 'dtcpb', 'dwin', 'tcprtt', 'synack', 'ackdat', 'smean', 'dmean', 'trans_depth', 'response_body
_len', 'ct_srv_src', 'ct_state_ttl', 'ct_dst_ltm', 'ct_src_dport_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'is_ftp_login', 'c
t_ftp_cmd', 'ct_flw_http_mthd', 'ct_src_ltm', 'ct_srv_dst', 'is_sm_ips_ports', 'proto_tcp', 'proto_udp', 'service_dhcp', 'servi
ce_dns', 'service_ftp', 'service_ftp-data', 'service_http', 'service_irc', 'service_pop3', 'service_radius', 'service_smtp', 's
ervice_snmp', 'service_ssh', 'service_ssl', 'state_CON', 'state_FIN', 'state_INT', 'state_REQ', 'state_RST']
```

```
minmax_scale = MinMaxScaler(feature_range=(0, 1))
def normalization(df,col):
    for i in col:
        arr = df[i]
        arr = np.array(arr)
        df[i] = minmax_scale.fit_transform(arr.reshape(len(arr),1))
    return df
```

Рисунок 3.9 – Виокремлення числових ознак та функція для нормалізації

	id	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	...	service_radius	service_smtp	service_snmp	service_ssh	service_s
3	4	1.681642	12	12	628	770	13.677108	62	252	2.740179e+03	...	0	0	0	0	
11	12	2.093085	62	28	56329	2212	42.520967	62	252	2.118251e+05	...	0	1	0	0	
15	16	0.000002	2	0	138	0	500000.001300	254	0	2.760000e+08	...	0	0	1	0	
17	18	0.393556	10	8	860	1096	43.195886	62	252	1.573347e+04	...	0	0	0	0	
21	22	0.338017	10	6	998	268	44.376468	254	252	2.127704e+04	...	0	0	0	0	

5 rows × 61 columns

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

```
data.head()
```

	id	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	...	service_radius	service_smtp	service_snmp	service_ssl
3	4	2.802780e-02	0.001144	0.001093	0.000041	0.000053	0.000014	0.138393	0.992126	0.000001	...	0.0	0.0	0.0	0.0
11	12	3.488529e-02	0.006344	0.002551	0.004337	0.000151	0.000043	0.138393	0.992126	0.000092	...	0.0	1.0	0.0	0.0
15	16	1.666693e-08	0.000104	0.000000	0.000003	0.000000	0.500000	0.995536	0.000000	0.119792	...	0.0	0.0	1.0	0.0
17	18	6.559354e-03	0.000936	0.000729	0.000059	0.000075	0.000043	0.138393	0.992126	0.000007	...	0.0	0.0	0.0	0.0
21	22	5.633690e-03	0.000936	0.000547	0.000069	0.000018	0.000044	0.995536	0.992126	0.000009	...	0.0	0.0	0.0	0.0

5 rows × 61 columns

Рисунок 3.10 – Дані до та після нормалізації

Так як моделі будуть навчатися для вирішення бінарної та мультикласової класифікації, треба створити два окремих набори. Для бінарної класифікації у якості мітки буде використовуватися ознака «label», а для мультикласової «attack_cat».

У бінарній класифікації все просто – якщо атаки немає то позначаємо цифрою 0, у іншому випадку 1. При мультикласовій класифікації у якості мітки використовується ознака «attack_cat», яка містить у собі 9 унікальних значень (видів атак), тому кожне значення кодуємо цифрами від 0 до 8.

```
binary_data['label']
3         1
11        1
15        1
17        1
21        1
         ..
175335    0
175336    0
175338    0
175339    0
175340    0
Name: label, Length: 81173, dtype: int32
```

```
multiclass_data['label']
3         6
11        6
15        6
17        6
21        6
         ..
175335    5
175336    5
175338    5
175339    5
175340    5
Name: label, Length: 81173, dtype: int32
```

Рисунок 3.11 – Результат кодування

У наборах даних присутня велика кількість ознак, що може негативно вплинути на результати навчання моделей. Для вирішення цієї проблеми треба створити матрицю кореляцій для наборів та відібрати ознаки які мають найбільшу кореляцію із «label».

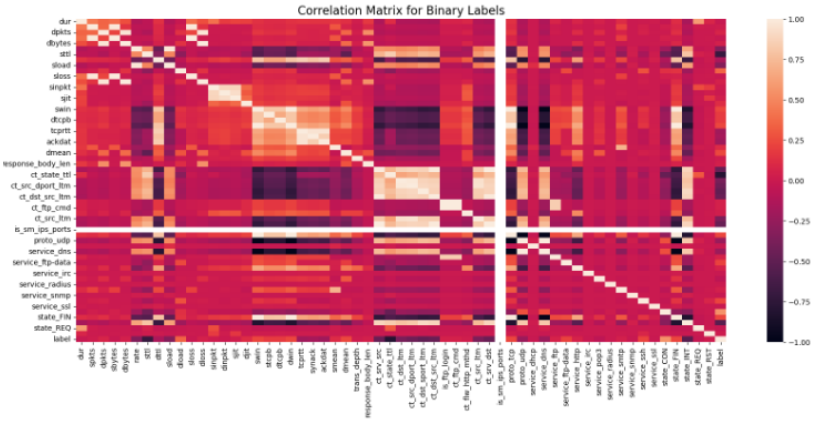

Рисунок 3.12 – Матриця кореляції для бінарного набору

Відберемо ознаки у яких кореляція більше за 0.3.

```
sload              0.334562
dload              0.343910
rate               0.344535
ct_src_ltm         0.368486
ct_dst_ltm         0.387358
ct_src_dport_ltm   0.444874
ct_srv_dst         0.459984
ct_srv_src         0.463153
ct_dst_src_ltm     0.463735
ct_dst_sport_ltm   0.497234
state_INT          0.546631
state_CON          0.552505
sttl               0.707337
ct_state_ttl       0.801403
label              1.000000
Name: label, dtype: float64
```

Рисунок 3.13 – Відібрані ознаки

Тож набір для бінарної класифікації матимете 15 ознак.

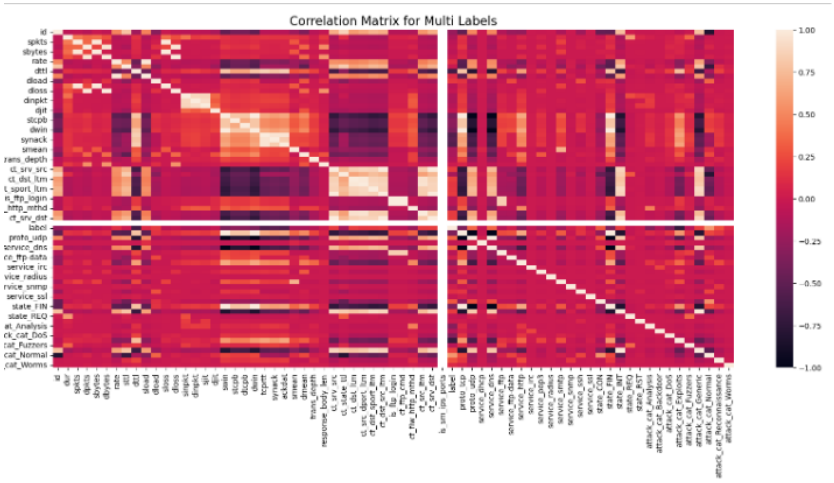

Рисунок 3.14 – Матриця кореляції для мультикласового набору

```
state_CON              0.302853
attack_cat_Analysis    0.326209
attack_cat_DoS         0.339669
state_FIN              0.361750
proto_udp              0.364393
swin                   0.364393
dwin                   0.364393
proto_tcp              0.364393
service_dns            0.365346
synack                 0.524027
ackdat                 0.570098
tcprtt                 0.570205
attack_cat_Normal      0.570858
dttl                   0.646589
attack_cat_Exploits    0.719733
label                  1.000000
Name: label, dtype: float64
```

Рисунок 3.15 – Відібрані ознаки

Із рисунку 3.15 бачимо, що у мультикласовому наборі 16 ознак

Висновки до розділу 3

У даному розділі була детально розглянута структура та процес підготовки набору даних для подальшого навчання моделей виявлення кіберзагроз.

Описано процес та інструменти для створення набору, розглянуто кожну ознаку у наборі. Це надає зрозуміння основних характеристик даних та визначає контекст, у якому модель буде застосовуватися.

Проведено огляд кроків, необхідних для створення та підготовки даних. Це включає в себе обробку відсутніх значень, кодування категоріальних ознак та міток для бінарної та багатокласової класифікації, нормалізацію даних та обрання ознак орієнтуючись на матрицю кореляцій. У результаті чого було сформовано два набори даних.

Підсумовуючи цей розділ, можна стверджувати, що якісно підготовлений та структурований набір даних є ключовим фактором для досягнення високої ефективності моделей машинного навчання. Описані методи підготовки набору даних не лише дозволяють ефективно використовувати алгоритми класифікації, але і підвищують точність та узагальнюють їх придатність до широкого спектру сценаріїв виявлення кіберзагроз.

4 СТВОРЕННЯ ТА ТЕСТУВАННЯ МОДЕЛЕЙ

Всього буде проводитися навчання чотирьох моделей для бінарної і чотири для багатокласової класифікації. Будуть використовуватися наступні алгоритми:

– логістична регресія;
– метод опорних векторів;
– дерево рішень;
– наївний байєсівський класифікатор.

4.1 Бінарна класифікація

Для початку треба розділити набір на навчальну та тестову вибірки. На тестову вибірку припадає 20% набору, на навчальну останні 80% (див. рис. 4.1).

```
X = bin_data.drop(columns=['label'],axis=1)
Y = bin_data['label']
```

```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.20, random_state=50)
```

Рисунок 4.1 – Поділ на навчальну та тестову вибірки

Логістична регресія.

На рисунку 4.2 продемонстровано створення та навчання моделі логістичної регресії.

```
logr_bin = LogisticRegression(random_state=123, max_iter=5000)
logr_bin
```

```
LogisticRegression
LogisticRegression(max_iter=5000, random_state=123)
```

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

```
LogisticRegression
LogisticRegression(max_iter=5000, random_state=123)
```

```
y_pred = logr_bin.predict(X_test)
```

Рисунок 4.2 – Створення та навчання моделі логістичної регресії

На рисунку 4.3 наведена матриця помилок для моделі логістичної регресії. А на рисунку 4.4 наведено розрахунок метрик для логістичної регресії.

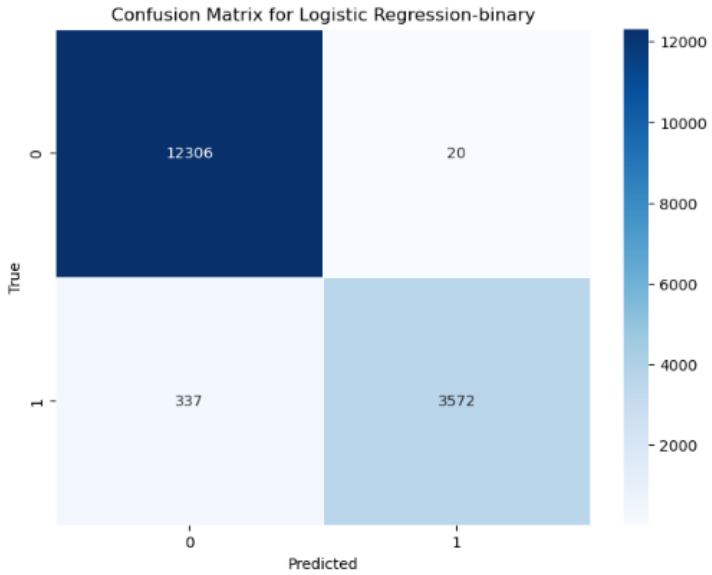

Рисунок 4.3 – Матриця помилок логістичної регресії

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.02198952879581152
Mean Squared Error -  0.02198952879581152
Root Mean Squared Error -  0.1482886671186019
R2 Score -  88.17947258428785
Accuracy -  97.80104712041884
```

```
cls_report= classification_report(y_true=y_test, y_pred=y_pred,target_names=le1.classes_)
print(cls_report)
```

```
              precision    recall  f1-score   support

    abnormal       0.97      1.00      0.99     12326
      normal       0.99      0.91      0.95      3909

    accuracy                           0.98     16235
   macro avg       0.98      0.96      0.97     16235
weighted avg       0.98      0.98      0.98     16235
```

Рисунок 4.4 – Розрахунок метрик для логістичної регресії

Проаналізувавши рисунки 4.3 та 4.4 можна сказати що модель добре класифікує атаки, із 12326 атак не правильно було класифіковано 20 атак. З приводу не атак ситуація трохи гірше: із 3909 не атак було класифіковано як атаки 337 записів. Частка правильно класифікованих прикладів 97.8%.

Метод опорних векторів.

На рисунку 4.5 продемонстровано створення та навчання моделі методу опорних векторів.

```
lsvm_bin = LinearSVC(random_state=42)
lsvm_bin.fit(X_train,y_train)
```

```
C:\Users\ILSYS\anaconda3\Lib\site-packa
rom `True` to `'auto'` in 1.5. Set the
  warnings.warn(
```

```
LinearSVC
LinearSVC(random_state=42)
```

```
y_pred = lsvm_bin.predict(X_test)
```

Рисунок 4.5 – Створення та навчання моделі методу опорних векторів

На рисунку 4.6 наведена матриця помилок для моделі методу опорних векторів. А на рисунку 4.7 наведено розрахунок метрик для методу опорних векторів.

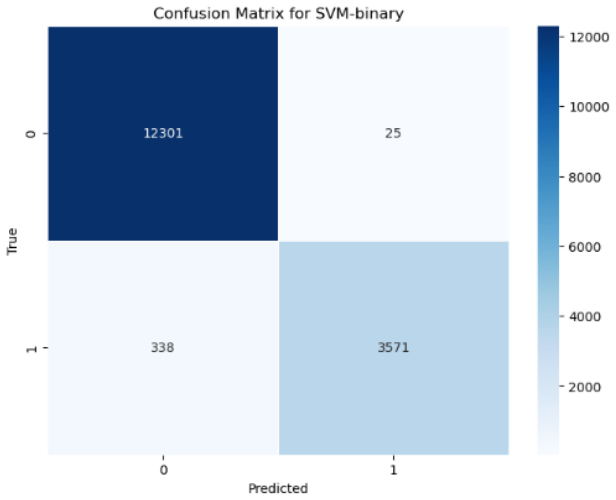

Рисунок 4.6 – Матриця помилок методу опорних векторів

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.022359100708346166
Mean Squared Error -  0.022359100708346166
Root Mean Squared Error -  0.14952959810133298
R2 Score -  87.97207291784578
Accuracy -  97.7640899291654
```

```
cls_report= classification_report(y_true=y_test, y_pred=y_pred,target_names=le1.classes_)
print(cls_report)
```

```
              precision    recall  f1-score   support

    abnormal       0.97      1.00      0.99     12326
      normal       0.99      0.91      0.95      3909

    accuracy                           0.98     16235
   macro avg       0.98      0.96      0.97     16235
weighted avg       0.98      0.98      0.98     16235
```

Рисунок 4.7 – Розрахунок метрик для методу опорних векторів

Тут результати трохи гірше ніж у логістичній регресії. Метод опорних векторів неправильно класифікував атаки на п’ять більше, а не атаки на одну більше порівнюючи із логістичною регресією. Тому і частка правильно класифікованих прикладів менша, а саме 97.7%.

Дерево рішень.

На рисунку 4.8 продемонстровано створення та навчання моделі дерева рішень.

```
dt_bin = DecisionTreeClassifier(random_state=123)
dt_bin.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=123)
```

```
y_pred = dt_bin.predict(X_test)
```

Рисунок 4.8 – Створення та навчання моделі дерева рішень

На рисунку 4.9 наведена матриця помилок для моделі методу дерева рішень. А на рисунку 4.10 наведено розрахунок метрик для методу дерева рішень.

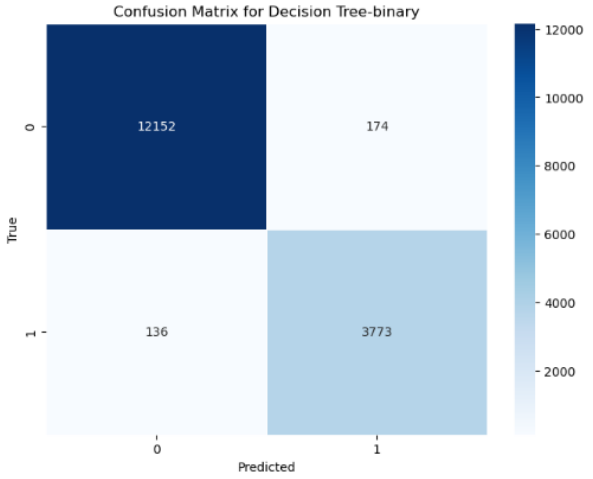

Рисунок 4.9 – Матриця помилок моделі дерева рішень

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.019094548814290114
Mean Squared Error -  0.019094548814290114
Root Mean Squared Error -  0.13818302650575473
R2 Score -  89.55757103838098
Accuracy -  98.09054511857099
```

```
cls_report= classification_report(y_true=y_test, y_pred=y_pred,target_names=le1.classes_)
print(cls_report)
```

```
              precision    recall  f1-score   support

    abnormal       0.99      0.99      0.99     12326
      normal       0.96      0.97      0.96      3909

    accuracy                           0.98     16235
   macro avg       0.97      0.98      0.97     16235
weighted avg       0.98      0.98      0.98     16235
```

Рисунок 4.10 – Розрахунок метрик для дерева рішень

Тут ситуація кардинально інша. Дерево рішень краще впоралось із класифікацією не атак, тільки 136 неправильно класифікованих записів. Але атаки вдалося класифікувати гірше, 174 неправильно класифікованих записів. Попри це частка правильно класифікованих прикладів вище ніж у двох попередніх моделей і складає 98.1%.

Наївний байєсівський класифікатор.

На рисунку 4.11 продемонстровано створення та навчання моделі наївного байєсівського класифікатора.

```
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train,y_train)
```

```
GaussianNB()
```

```
y_pred = naive_bayes_model.predict(X_test)
```

Рисунок 4.11 – Створення та навчання моделі *наївного баєсівського класифікатора*

На рисунку 4.12 наведена матриця помилок для моделі наївного байєсівського класифікатора. А на рисунку 4.13 наведено розрахунок метрик для наївного байєсівського класифікатора.

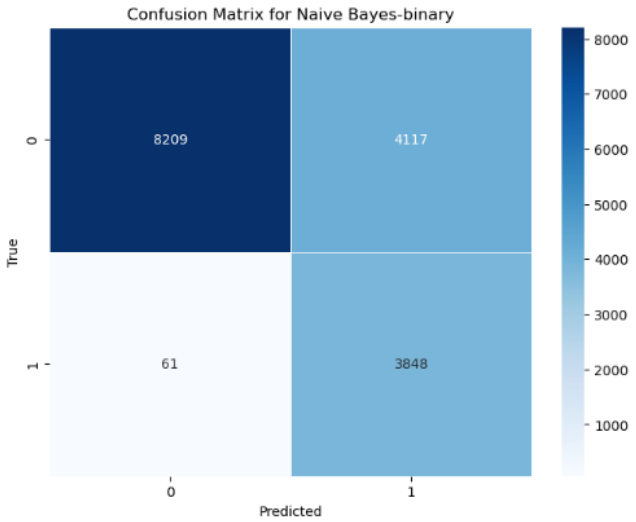

Рисунок 4.12 – Матриця помилок моделі *наївного баєсівського класифікатора*

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.2573452417616261
Mean Squared Error -  0.2573452417616261
Root Mean Squared Error -  0.5072920675130118
R2 Score -  -6.633883696875298
Accuracy -  74.26547582383739
```

```
cls_report= classification_report(y_true=y_test, y_pred=y_pred,target_names=le1.classes_)
print(cls_report)
```

```
              precision    recall  f1-score   support

    abnormal       0.99      0.67      0.80     12326
      normal       0.48      0.98      0.65      3909

    accuracy                           0.74     16235
   macro avg       0.74      0.83      0.72     16235
weighted avg       0.87      0.74      0.76     16235
```

Рисунок 4.13 – Розрахунок метрик для наївного баєсівського класифікатора

У цій моделі погано класифікуються атаки, 4117 записів атак було класифіковано не правильно. Але не атаки були класифіковані краще, тільки 61 запис був класифікований не правильно. Це краще ніж у інших моделей. Але через багато помилок при класифікації атак, частка правильно класифікованих прикладів моделі наївного баєсівського класифікатора найменша і складає всього 74.2%.

4.2 Багатокласова класифікація

Тут також розділимо набір, але для тестової вибірки виділимо вже 30%(див. рис. 4.14). Також створення моделей буде таке саме як і при бінарній класифікації, тож наводити рисунки їх створення не має сенсу.

```
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)
```

Рисунок 4.14 – Поділ на навчальну та тестову вибірки

Логістична регресія.

На рисунку 4.15 наведена матриця помилок для моделі логістичної регресії. А на рисунку 4.16 наведено розрахунок метрик для логістичної регресії.

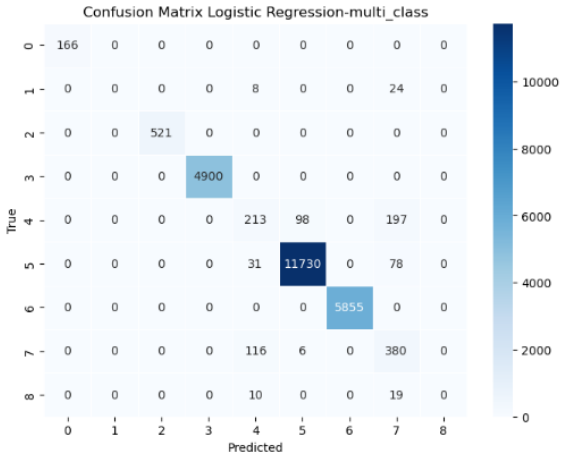

Рисунок 4.15 – Матриця помилок моделі логістичної регресії

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.060077201051248356
Mean Squared Error -  0.18056011826544022
Root Mean Squared Error -  0.42492366169165047
R2 Score -  87.87674567880146
Accuracy -  97.58952036793693
```

```
print(classification_report(y_test, y_pred,target_names=le2.classes_))
```

```
              precision    recall  f1-score   support

    Analysis       1.00      1.00      1.00       166
    Backdoor       0.00      0.00      0.00        32
         DoS       1.00      1.00      1.00       521
    Exploits       1.00      1.00      1.00      4900
     Fuzzers       0.56      0.42      0.48       508
     Generic       0.99      0.99      0.99     11839
      Normal       1.00      1.00      1.00      5855
Reconnaissance       0.54      0.76      0.63       502
       Worms       0.00      0.00      0.00        29

    accuracy                           0.98     24352
   macro avg       0.68      0.69      0.68     24352
weighted avg       0.97      0.98      0.97     24352
```

Рисунок 4.16 – Розрахунок метрик для логістичної регресії

В цій моделі класифіковані повністю правильно три атаки, а саме «Analysis», «DoS», «Exploits». Не були класифіковані атаки «Backdoor», «Worms». Усі записи без атак були класифіковані правильно. У атаці «Fuzzers» було не правильно класифіковано 295 записів; у «Generic» 109 записів; у «Reconnaissance» 122 записи. Частка правильно класифікованих прикладів складає 97.5%.

Метод опорних векторів.

На рисунку 4.17 наведена матриця помилок для моделі методу опорних векторів. А на рисунку 4.18 наведено розрахунок метрик для методу опорних векторів.

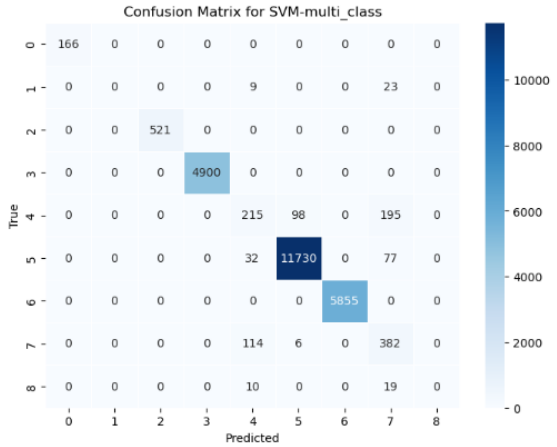

Рисунок 4.17 – Матриця помилок моделі методу опорних векторів

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.05942017082785808
Mean Squared Error -  0.17784986859395532
Root Mean Squared Error -  0.4217225018824053
R2 Score -  88.05849574122144
Accuracy -  97.60594612352168
```

```
print(classification_report(y_test, y_pred,target_names=le2.classes_))
```

```
                precision    recall  f1-score   support

      Analysis       1.00      1.00      1.00       166
      Backdoor       0.00      0.00      0.00        32
           DoS       1.00      1.00      1.00       521
      Exploits       1.00      1.00      1.00      4900
       Fuzzers       0.57      0.42      0.48       508
       Generic       0.99      0.99      0.99     11839
        Normal       1.00      1.00      1.00      5855
Reconnaissance       0.55      0.76      0.64       502
         Worms       0.00      0.00      0.00        29

      accuracy                           0.98     24352
     macro avg       0.68      0.69      0.68     24352
  weighted avg       0.97      0.98      0.97     24352
```

Рисунок 4.18 – Розрахунок метрик для методу опорних векторів

Тут також повністю правильно класифіковані три атаки («Analysis», «DoS», «Exploits»). Не були класифіковані атаки «Backdoor», «Worms». Усі записи без атак були класифіковані правильно. У атаці «Fuzzers» було не правильно класифіковано 293 записів; у «Generic» 109 записів; у «Reconnaissance» 120 записів. Частка правильно класифікованих прикладів складає 97.6%.

Дерево рішень.

На рисунку 4.19 наведена матриця помилок для моделі дерева рішень. А на рисунку 4.20 наведено розрахунок метрик для дерева рішень.

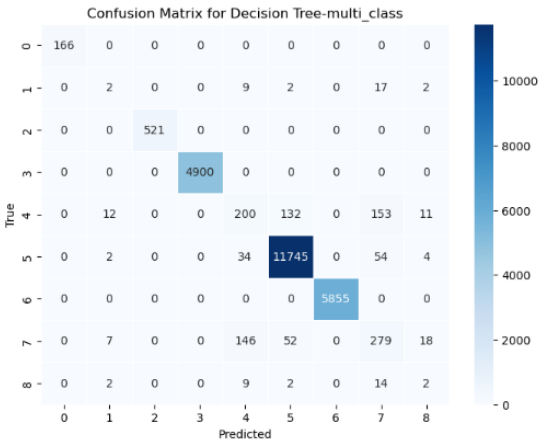

Рисунок 4.19 – Матриця помилок моделі дерева рішень

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.06800262812089355
Mean Squared Error -  0.20532194480946123
Root Mean Squared Error -  0.4531246459965086
R2 Score -  86.17743099336013
Accuracy -  97.19940867279895
```

```
print(classification_report(y_test, y_pred,target_names=le2.classes_))
```

```
              precision    recall  f1-score   support

    Analysis       1.00      1.00      1.00       166
    Backdoor       0.08      0.06      0.07        32
         DoS       1.00      1.00      1.00       521
    Exploits       1.00      1.00      1.00      4900
     Fuzzers       0.50      0.39      0.44       508
     Generic       0.98      0.99      0.99     11839
      Normal       1.00      1.00      1.00      5855
Reconnaissance       0.54      0.56      0.55       502
       Worms       0.05      0.07      0.06        29

    accuracy                           0.97     24352
   macro avg       0.68      0.67      0.68     24352
weighted avg       0.97      0.97      0.97     24352
```

Рисунок 4.20 – Розрахунок метрик для дерева рішень

В цьому методі ситуація трохи інша. Як і в минулих методах повністю правильно класифіковані три атаки («Analysis», «DoS», «Exploits») та усі записи не атак. Але вже правильно класифікувалася атака «Backdoor»: тільки 2 записи із 32 були класифіковані правильно; «Worms»: теж 2 записи із 29 записів були класифіковано правильно. У атаці «Fuzzers» було не правильно класифіковано 296 записів; у «Generic» 92 записів; у «Reconnaissance» 261 записів. Частка правильно класифікованих прикладів складає 97.1%.

Наївний баєсівський класифікатор.

На рисунку 4.21 наведена матриця помилок для моделі наївного байєсівського класифікатора. А на рисунку 4.22 наведено розрахунок метрик для наївного байєсівського класифікатора.

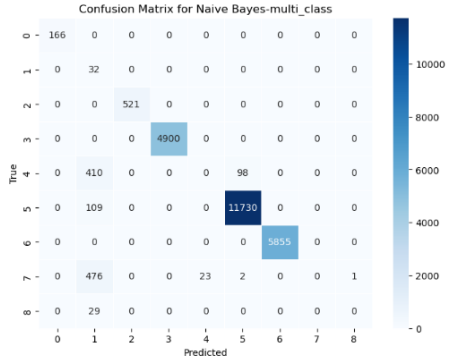

Рисунок 4.21 – Матриця помилок моделі наївного баєсівського класифікатора

```
print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

```
Mean Absolute Error -  0.20109231274638634
Mean Squared Error -  0.998069973718791
Root Mean Squared Error -  0.9990345207843375
R2 Score -  35.29555978660272
Accuracy -  95.28580814717476
```

```
print(classification_report(y_test, y_pred,target_names=le2.classes_))
```

```
              precision    recall  f1-score   support

    Analysis       1.00      1.00      1.00       166
    Backdoor       0.03      1.00      0.06        32
         DoS       1.00      1.00      1.00       521
    Exploits       1.00      1.00      1.00      4900
     Fuzzers       0.00      0.00      0.00       508
     Generic       0.99      0.99      0.99     11839
      Normal       1.00      1.00      1.00      5855
Reconnaissance       0.00      0.00      0.00       502
       Worms       0.00      0.00      0.00        29

    accuracy                           0.95     24352
   macro avg       0.56      0.67      0.56     24352
weighted avg       0.95      0.95      0.95     24352
```

Рисунок 4.21 – Розрахунок метрик для наївного баєсівського класифікатора

В цьому методі правильно класифіковані атаки «Analysis», «Backdoor», «DoS», «Exploits». Також усі не атаки були правильно класифіковані. Але зовсім не були класифіковані «Fuzzers», «Reconnaissance», «Worms». У атаці «Generic» було не правильно класифіковано 109 записів. Частка правильно класифікованих прикладів складає 95.2%.

4.3 Систематизація отриманих результатів

На рисунку 4.22 наведено гістограму результатів навчання моделей.

Говорячи про бінарну класифікацію, найкраще справився метод дерева рішень із часткою правильно класифікованих прикладів 98.1%. Найгірше справився наївний байєсівський класифікатор із результатом 74.1%.

У багатокласовій класифікації найкраще справився метод опорних векторів, із результатом 97.6%. Найгірше знову справився наївний байєсівський класифікатор із результатом 95.2%, але тут частка правильно класифікованих прикладів набагато вище. Також хочеться зауважити, що усі алгоритмі у багатокласовій класифікації погано класифіковані атаки «Backdoor», «Worms» б або були взагалі не класифіковані. Це пов’язано із малою кількістю записів із цими атаками.

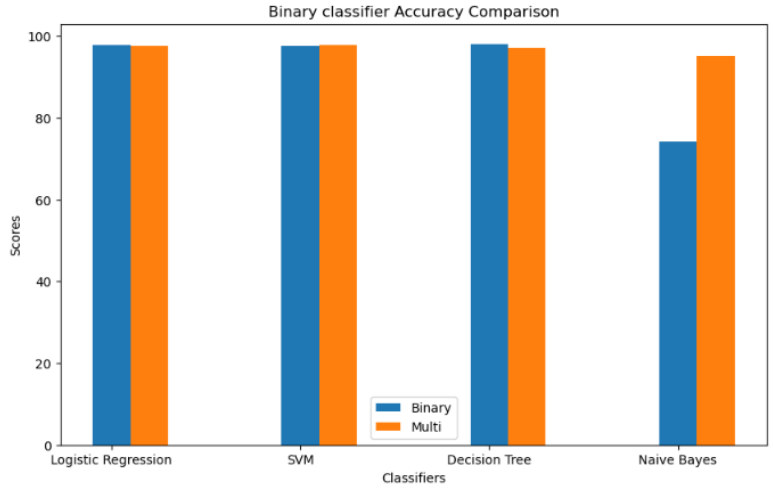

Рисунок 4.22 – Порівняння результатів

Висновки до 4 розділу

Результати аналізу використання різних класифікаційних моделей у бінарній та багатокласовій класифікації вказують на різні переваги та недоліки кожної з них.

У бінарній класифікації логістична регресія показала високу точність, класифікуючи атаки і не атаки з часткою правильно класифікованих прикладів на рівні 97.8%. Метод опорних векторів та дерево рішень також показали хороші результати, хоча метод опорних векторів мав трошки меншу частку правильно класифікованих прикладів (97.7%). Наївний байєсівський класифікатор показав менш задовільні результати з часткою правильно класифікованих прикладів на рівні 74.2%.

У багатокласовій класифікації логістична регресія та метод опорних векторів показали схожі результати, повністю правильно класифікуючи три атаки та усі не атаки, але маючи проблеми з іншими категоріями атак. Дерево рішень показало найвищу частку правильно класифікованих прикладів (98.1%), але мало проблеми з класифікацією деяких атак. Наївний баєсівський класифікатор виявився менш ефективним у багатокласовій класифікації з часткою правильно класифікованих прикладів на рівні 95.2%.

ВИСНОВКИ

Із кожним днем людство все більше інтегрує своє життя у інтернет, залишаючи у мережі свої особисті дані. З кожним днем даних у інтернеті все більше і стандартні методи виявлення кіберзагроз стають менш актуальними, не справляючись із великим потоком даних. Тому використання машинного навчання у кібербезпеці буде актуальним ще довгий час.

Здійснений аналіз показав, що все частіше у системах захисту від атак використовуються елементи машинного навчання в поєднанні із класичними методами виявлення кіберзагроз. Це дозволяє полегшати та оптимізувати роботу людей які відповідають за безпеку на підприємствах.

Було встановлено що для виявлення аномального трафіку краще за все підходять класифікаційні алгоритми, які надають можливість класифікувати великий обсяг трафіку та відносити його до нормального або аномального. Також крім цього є можливість класифікувати кожну атаку окремо.

Для навчання моделей був підібраний набір даних, та проаналізований. На основі отриманих знань відбулася його підготовка. Підготовка набору є невід'ємною і важливою частиною у машинному навчанні. Підготовлений набір дозволяє досягти великої точності моделей. Також набір був підготовлений як для бінарної так і для багатокласової класифікації. Для відбору ознак використовувалась матриця кореляції, яка дозволила відсіяти ознаки які не впливають на мітку. Сам набір даних не є ідеальним. Більшість записів є нормальним трафіком, а на атаки припадає всього 24%. Також мало записів із атаками «Backdoor», «Worms». Ці недоліки впливають на результат.

У результаті для двох видів класифікації були навчені моделі. Їх результати були проаналізовані. У двох випадках найгірше себе показав наївний байєсівський класифікатор. У бінарній класифікації найкращим виявився метод дерева рішень, а у багатокласовій – метод опорних векторів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 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.

2. Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." *Information Security Journal: A Global Perspective* (2016): 1-14.

3. Moustafa, Nour, et al. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." *IEEE Transactions on Big Data (2017).*

4. Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." *Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017.* 127-156.

5. Sarhan, Mohanad, Siamak Layeghy, Nour Moustafa, and Marius Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings (p. 117). Springer Nature.

6. I. Butun, P. Osterberg, H. Song, Security of the internet of things: Vulnerabilities, attacks, and countermeasures, IEEE Commun. Surv. Tutor. 22 (1) (2020) 616–644.

7. M. Aladag, F.O. Catak, E. Gul, Preventing data poisoning attacks by using generative models, in: 1st International Informatics and Software Engineering Conference, UBMYK, Ankara, Turkey, 2019, pp. 1–5.

8. N. Papernot, A marauder’s map of security and privacy in machine learning, in: 11th ACM Workshop on Artificial Intelligence and Security, Toronto, Canada, 2018.

9. J. Chen, X. Zhang, R. Zhang, C. Wang, L. Liu, De-pois: An attackagnostic defense against data poisoning attacks, 2021, CoRR.

10. P. Mohassel, Y. Zhang, SecureML: A system for scalable privacypreserving machine learning, in: IEEE Symposium on Security and Privacy, S&P, San Jose, USA, 2017, pp. 19–38.

11. V. Ambalavanan, “Cyber threats detection and mitigation using machine learning,” in Handbook of Research on Machine and Deep Learning Applications for Cyber Security. Hershey, PA, USA: IGI Global, 2020, pp. 132-149.

12. T. Thomas, A. P. Vijayaraghavan, and S. Emmanuel, “Machine learning and cybersecurity,” in Machine Learning Approaches in Cyber Security Analytics. Singapore: Springer, 2020, pp. 37-47. [Online]. Available: https://link.springer.com/chapter/ 10.1007/978-981-15-1706-8 3.

13. A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Commun. Surveys Tuts. vol. 18, no. 2, pp. 1153-1176, 2nd Quart., 2016.

14. R. M. Mohammad, F. Thabtah, and L. McCluskey, “Tutorial and critical analysis of phishing Websites methods,” Comput. Sci. Rev., vol. 17, pp. 1-24, Aug. 2015.

15. N. Jindal and B. Liu, “Review spam detection,” in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 1189-1190. S. M. Abdulhamid, M. S. Abd Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan, “A review on mobile SMS spam filtering techniques,” IEEE Access, vol. 5, pp. 15650-15666, 2017.

16. D. Michie, D. J. Spiegelhalter, and C. Taylor, “Machine learning,” Neural Stat. Classification, vol. 13, 1994. S. Dua and X. Du, Data Mining and Machine Learning in Cybersecurity. New York, NY, USA: Auerbach, 2016.

17. A. Abeshu and N. Chilamkurti, “Deep learning: The frontier for dis- tributed attack detection in fog-to-things computing,” IEEE Commun. Mag., vol. 56, no. 2, pp. 169- 175, Feb. 2018.

18. S. Purushotham, C. Meng, Z. Che, and Y. Liu, “Benchmarking deep learning models on large healthcare datasets,” J. Biomed. Informat., vol. 83, pp. 112-134, Jul. 2018.

19. B. Sch¨olkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, “Support vector method for novelty detection,” in Proc. Adv. Neural Inf. Process. Syst., 2000, pp. 582-588.

20. A. L. Prodromidis and S. J. Stolfo, “Cost complexity-based pruning of ensemble classifiers,” Knowl. Inf. Syst., vol. 3, no. 4, pp. 449-469, Nov. 2001.

21. S. He, G. M. Lee, S. Han, and A. B. Whinston, “How would information disclosure influence organizations’ outbound spam volume? Evidence from a field experiment,” J. Cybersecurity, vol. 2, no. 1, pp. 99-118, Dec. 2016.

22. S. T. Miller and C. Busby-Earle, “Multi-perspective machine learning a classifier ensemble method for intrusion detection,” in Proc. Int. Conf. Mach. Learn. Soft Comput. (ICMLSC), 2017, pp. 7-12.

23. L. Jiang, H. Zhang, and Z. Cai, “A novel Bayes model: Hidden naive Bayes,” IEEE Trans. Knowl. Data Eng., vol. 21, no. 10, pp. 1361-1371, Oct. 2009.

24. X. Xu, Y. Sun, and Z. Huang, “Defending DDoS attacks using hid- den Markov models and cooperative reinforcement learning,” in Proc. Pacific-Asia Workshop Intell. Secur. Inform. Berlin, Germany: Springer, 2007, pp. 196-207.

25. . I. Alsmadi and I. Alhami, “Clustering and classification of email con- tents,” J. King Saud Univ. Comput. Inf. Sci., vol. 27, no. 1, pp. 46-57, Jan. 2015.

26. D. C. Chandrasekar, “Classification techniques using spam filtering email,” Int. J. Adv. Res. Comput. Sci., vol. 9, no. 2, pp. 402-410, Apr. 2018.

27. A. Sharaff, N. K. Nagwani, and A. Dhadse, “Comparative study of classification algorithms for spam email detection,” in Emerging Research in Computing, Information, Communication and Applications. New Delhi, India: Springer, 2016, pp. 237-244.

28. I. Ahmed, D. Guan, and T. C. Chung, “SMS classification based on naive Bayes classifier and Apriori algorithm frequent itemset,” Int. J. Mach. Learn. Comput., vol. 4, no. 2, p. 183, 2014.

29. H. Li, X. Xu, C. Liu, T. Ren, K. Wu, X. Cao, W. Zhang, Y. Yu, and D. Song, “A machine learning approach to prevent malicious calls over telephony networks,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2018, pp. 53-69.

30. N. B. Amor, S. Benferhat, and Z. Elouedi, “Naive bayes vs decision trees in intrusion detection systems,” in Proc. ACM Symp. Appl. Comput. (SAC), 2004, pp. 420-424.

31. Y. Bouzida and F. Cuppens, “Neural networks vs. decision trees for intrusion detection,” in Proc. IEEE/IST Workshop Monitor., Attack Detection Mitigation (MonAM), vol. 28. Citeseer, 2006, p. 29.

32. B. Sezari, D. P. F. Moller, and A. Deutschmann, “Anomaly-based network intrusion detection model using deep learning in airports,” in Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE), Aug. 2018, pp. 1725-1729.

33. C. Chen, J. Zhang, Y. Xie, Y. Xiang, W. Zhou, M. M. Hassan, A. AlElaiwi, and M. Alrubaian, “A performance evaluation of machine learning-based streaming spam tweets detection,” IEEE Trans. Comput. Social Syst., vol. 2, no. 3, pp. 65-76, 2015.

34. Kamran Shaukat, Suhuai Luo, Vijay Varadharajan, Ibrahim A. Hameed, Min Xu, “ A Survey on Machine Learning Techniques for Cyber Security in the Last Decade”, IEEE Access, vol. 8, pp. 222310 – 222354, 2020.

35. M. Weber, X. Xu, B. Karlas, C. Zhang, B. Li, RAB: Provable robustness against backdoor attacks, 2020.

36. Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. 2012. Machine learning strategies for time series forecasting. In European Business Intelligence Summer School. 62–77.

37. Emilie Bout, Valeria Loscri, and Antoine Gallais. 2021. How machine learning changes the nature of cyberattacks on IoT networks: A survey. IEEE Commun. Surv. Tutor. (2021).

38. Anna L. Buczak and Erhan Guven. 2015. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutor. 18, 2 (2015), 1153–1176.

39. Howard Chivers, John A. Clark, Philip Nobles, Siraj A. Shaikh, and Hao Chen. 2013. Knowing who to watch: Identifying attackers whose actions are hidden within false alarms and background noise. Inf. Syst. Front. 15, 1 (2013), 17–34.

40. Andrea Corsini, Shanchieh Yang, and Giovanni Apruzzese. 2021. On the evaluation of sequential machine learning for network intrusion detection. In Proceedings of the International Conference Availability, Reliability, Security.

41. Arnaldo Gouveia and Miguel Correia. 2020. Towards quantum-enhanced machine learning for network intrusion detection. In Proceedings of the IEEE 19th International Symposium on Network Computing and Applications (NCA’20). 1–8.

42. Ahsan Al Zaki Khan. 2019. Misuse intrusion detection using machine learning for gas pipeline scada networks. In Proceedings of the International Conference Security and Management. 84–90.

43. Nuno Martins, José Magalhães Cruz, Tiago Cruz, and Pedro Henriques Abreu. 2020. Adversarial machine learning applied to intrusion and malware scenarios: A systematic review. IEEE Access 8 (2020), 35403–35419.

44. Steven McElwee, Jeffrey Heaton, James Fraley, and James Cannady. 2017. Deep learning for prioritizing and responding to intrusion detection alerts. In Proceedings of the IEEE Military Communications Conference. 1–5.

45. Amirreza Niakanlahiji, Bei-Tseng Chu, and Ehab Al-Shaer. 2018. PhishMon: A machine learning framework for detecting phishing webpages. In Proceedings of the IEEE International Conference Intelligent Security Informatics. 220–225.