НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ Дмитро ЛАНДЕ
“___”_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»

зі спеціальності 125 «Кібербезпека»

на тему: Виявлення DoS/DDoS атак в IoT за допомогою машинного навчання

Виконав здобувач ступеня магістра _2_ курсу, групи ____ФБ-01мп__
(шифр групи)

Соловей Богдан Вадимович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент кафедри інформаційної безпеки, канд. техн. наук, Гальчинський Л.Ю. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Здобувач ступеня магістра _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Дмитро ЛАНДЕ
(підпис)
«___»_____2021 р.

ЗАВДАННЯ
на магістерську дисертацію здобувачу ступеня магістра

Солов'ю Богдану Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Виявлення DoS/DDoS атак в IoT за допомогою машинного навчання

науковий керівник дисертації Гальчинський Леонід Юрійович,
канд. техн. наук, доцент, доцент кафедри інформаційної безпеки
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «05 » листопада 2021 р. № 3683-с

2. Термін подання здобувачем дисертації 06.12.2021 р.

3. Об'єкт дослідження DDoS/DoS атаки в мережі пристроїв IoT

4. Вихідні дані: технічна документація, теоретичні та статистичні дані;

5. Перелік завдань, які потрібно розробити 1.Проаналізувати предметну область. 2. Проаналізувати загрози кібербезпеці пристроїв IoT. 3. Вивчити можливість використання алгоритмів машинного навчання для розв'язання проблеми виявлення DoS/DDoS атак. 4. Розробити проект для оцінки різних алгоритмів. 5. Оцінити алгоритми за кількома критеріями.

6. Орієнтовний перелік ілюстративного матеріалу 13 ілюстрацій

_____

_____

7. Орієнтовний перелік публікацій: Соловей Б., Гальчинський Л. (2021). ВИЯВЛЕННЯ DOS/DDOS АТАК В ІОТ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ. *Матеріали конференцій Молодіжної наукової ліги*. вилучено із https://ojs.ukrlogos.in.ua/index.php/liga/article/view/15495. Індексовано в Google Scholar

8. Дата видачі завдання 01.09.2021

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Аналіз проблем кібербезпеки в IoT	01.07.2021	
2.	Розгляд можливості використання машинного навчання для виявлення DoS/DDoS атак в мережах IoT	15.09.2021	
3.	Вибір та аналіз набору даних	01.10.2021	
4.	Реалізація тестового проекту	18.10.2021	
5.	Отримання інтегральної оцінки	28.11.2021	
6.	Оформлення пояснювальної записки	28.11.2021	
7.	Підготовка до захисту	13.12.2021	

Здобувач ступеня магістра _____ Богдан СОЛОВЕЙ
(підпис) (власне ім'я, ПРІЗВИЩЕ)

Науковий керівник _____ Леонід ГАЛЬЧИНСЬКИЙ
(підпис) (власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Робота обсягом 109 сторінки включає 13 ілюстрацій, 33 таблиці, 35 джерела літератури та 1 додаток.

Об’єктом дослідження є DDoS/DoS атаки в мережі пристроїв IoT.

Предметом дослідження є виявлення DoS/DDoS атак в мережі пристроїв Інтернету речей.

Методи дослідження – поєднання існуючих методів і технологій виявлення аномалій в мережевому трафіку та методів оцінювання алгоритмів.

Метою роботи є вирішення проблеми підвищення ефективності виявлення DoS/DDoS атак в мережах IoT.

Результати роботи можуть використовуватися для побудови системи виявлення вторгнень в мережі пристроїв IoT.

Ключові слова: кібербезпека, машинне навчання, загрози, пристрої Інтернету речей, DoS/DDoS атаки, оцінка

ABSTRACT

The work includes 109 pages, 13 illustrations, 33 tables, 35 bibliography references and appendices.

The object of the study is DDoS / DoS attacks in the network of IoT devices.

The subject of the study is the detection of DoS / DDoS attacks in the network of Internet of Things devices.

Research methods - a combination of existing methods and technologies for detecting anomalies in network traffic and methods for estimating algorithms.

The purpose of this work is a practical solution to the problem of improving the detection of DoS / DDoS attacks in IoT networks through the use of machine learning algorithms.

The results can be used to build a system for detecting intrusions in the network of IoT devices.

Keywords:

cybersecurity, machine learning, threats, IoT devices, DoS / DDoS attacks, evaluation

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT – Інтернет речей;

ПЗ – програмне забезпечення;

IDS – система виявлення вторгнень;

ML – машинне навчання;

Random Forest – алгоритм машинного навчання випадковий ліс;

DoS – атака на відмову в обслуговуванні;

DDoS – розподілена атака на відмову в обслуговуванні;

ВСТУП

Швидкий розвиток Інтернету спричинив появу Інтернету речей (англ. Internet of Things, IoT) з такими прикладами: розумні будинки та міста, системи охорони здоров'я та кіберфізичні системи. IoT - це сукупність взаємопов'язаних маленьких пристроїв, якими можна керувати за допомогою веб-сервісів або інших типів інтерфейсів [1]. Очікувано, що будь-яка нова популярна технологія, привертає інтерес кіберзловмисників до зловживання нею з використанням різних прийомів злому. Ця проблема посилюється відсутністю стандартизації в системах Інтернету речей, а також тим що в них використовуються дешеві, та малопотужні пристрої [2], що робить їх легкими цілями для таких атак як: відмова в обслуговуванні (DoS) та розподілена відмова в обслуговуванні (DDoS) [2]. Такі атаки можуть завдавати великі збитки компаніям, вони популярні своєю відносною простотою, а також великою кількістю відкритих відомостей про її реалізації. Головна ідея схожих атак полягає в тому, що кіберзловмисник здійснює спробу унеможливити коректне обслуговування системи. Проблема в тому що традиційні високоякісні рішення безпеки не підходять для захисту IoT-систем і для досягнення безпечного розвитку галузі необхідно запровадити практичні заходи протидії атак в мережі Інтернету речей, оптимізовані під IoT. Будь-який пристрій Інтернету речей є вразливим, а дані, які збираються і обробляються ним є цінними. Кібератаки на ці пристрої потенційно можуть завдати шкоду об'єктам критичної інфраструктури та фізичним приладам. Тому дуже важливо вміти вчасно виявляти різні вразливості й атаки на пристрої IoT. Пристрої в мережі інтернету речей є досить автономними, тобто не вимагають втручання користувача для коректної роботи, це означає, що розумні мережеві рішення безпеки, такі як рішення з машинним навчанням (ML) необхідно розвивати. Застосування машинного навчання дозволяє створювати ефективні адаптивні системи виявлення мережевих вторгнень і дозволяє підвищити рівень захисту систем пристроїв Інтернету речей від зовнішніх атак. Хоча багато

досліджень останніх років [3][4][5] мають місце, але в них недостатньо уваги приділяється виявленню атак саме в мережах IoT.

Об’єктом дослідження є DDoS/DoS атаки в мережі пристроїв IoT.

Предметом дослідження є виявлення DoS/DDoS атак в мережі пристроїв Інтернету речей.

Методи дослідження – поєднання існуючих методів і технологій виявлення аномалій в мережевому трафіку.

Актуальність дослідження: в сучасному світі спостерігається безперервний ріст кількості пристроїв IoT, а також сфер їх застосування і це сприяє росту кількості та складності потенційних загроз та ризиків. Також сфера безпеки IoT є недостатньо дослідженою, особливо у випадку DoS/DDoS атак які можуть виглядати як нормальна поведінка системи

У даній роботі за мету ставиться є вирішення проблеми підвищення ефективності виявлення DoS/DDoS атак в мережах IoT за рахунок використання алгоритмів машинного навчання. Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі методи виявлення DoS/DDoS атак в мережах звичайних пристроїв та пристроїв Інтернету речей;

– розробити програмне забезпечення;

– провести тестування засобу на наборі даних зібраному в мережі пристроїв IoT.

– провести оцінку алгоритмів та вибрати найкращий з них

Практичне значення роботи полягає в тому що результати можуть використовуватися для побудови системи виявлення вторгнень в мережі пристроїв IoT.

Наукова новизна дослідження: підвищення ефективності виявлення DDoS/DoS атак у мережах IoT, а також оцінка якості роботи алгоритмів за допомогою методу прийняття рішень в багатокритеріальних задачах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інтернет речей

Інтернет речей (IoT) – сучасна технологія, що зосереджується на взаємодії між пристроями або речами а також користувачами або людьми для досягнення певних цілей. IoT забезпечується багатьма існуючими технологіями, такими як Wireless sensor and actuator networks (WSAN) і радіочастотна ідентифікація (RFID).

Інтернет речей (IoT) – це новий, але водночас старий термін. Про ще згадував Кевін Ештон у 1999 році під час проведення презентації в Proctor & Gamble. Він використав цей термін для зв'язку Ідея радіочастотної ідентифікації (RFID) до нової на той час теми Інтернету [6]. З тих пір використання цього терміну набувало все більшої популярності, а великі компанії передбачили зростання IoT [6,7]. Один з прогнозів полягав у тому, що кількість пов'язаних речей у світі збільшиться в тридцять разів між 2009 і 2022 роками, таким чином, до 2022 року буде 26 мільярдів речей, з доступом у Інтернет [7]. Причина того, що IoT став таким величезним, частково залежить від двох речей: закону Мура та Закон Кумі. Закон Мура стверджує, що кількість транзисторів на мікросхемі збільшується приблизно вдвічі кожні два роки. Це дозволило людям розробляти більш потужні комп'ютери при розмірі чіпу такого ж розміру.

Різні технологічні досягнення внесли свій внесок у наш спосіб життя. Зі швидким зростанням технологій, люди рухаються до автоматизованих технологій, щоб робити життя легшим. Серед усіх цих технологій Інтернет речей потенційно вплинув на нас найбільше. Технології IoT використовують значну кількість даних за допомогою мереж датчиків. Очікується, що галузь Інтернету речей виготовить 75,44 мільярдів пристроїв і генеруватиме 79,4 зеттабайт даних до кінця 2025 року [7]. Пристрої IoT розвиваються надзвичайно швидко і продовжать ріст в найближчі роки. Однак збільшене використання IoT пристроїв

призводить до обміну особистою інформацією, що призводить до потенційного порушення безпеки та конфіденційності.

IoT — це інтелектуальна мережа, яка підключається до багатьох інших пристроїв і центрів обробки даних. Пристрій IoT безперервно використовує та аналізує дані за допомогою датчиків з навколишнього середовища. Більшість таких пристроїв є автономними та функціонують з мінімальним втручанням людини. Деякі пристрої IoT мають унікальні ідентифікатори для аутентифікації. Низьке споживання енергії – головна особливість пристроїв IoT. Як результат - пристрої IoT можуть бути розміщені у віддалених районах для збору та передачі даних. Інтернет речей має широкий спектр застосувань, таких як розумні будинки, підключені транспортні засоби, міста, системи охорони здоров'я, світлофори, середовища для моніторингу в промисловості, розумні лічильники, моніторинг мереж водопроводу та розумної логістики [3, 5] та решта. Область застосування IoT не обмежується вищезгаданими прикладами. Система IoT зазвичай складається з трьох основних етапів: збір даних, передача даних та аналіз даних. Перший етап відповідає за збір і передачу даних, де використовуються сенсори і мікроконтролери. Цей етап також відомий як фізичний рівень. Другий етап - для передачі даних, де використовуються концентратор і шлюз або мережа IoT. Останній етап потрібен для аналізу даних, він включає інтерфейси користувача та серверну систему, таку як хмара. Будь-яке порушення кібербезпеки на цих етапах призведе до витоку важливої інформації з пристроїв.

1.2 Проблеми кібербезпеки Інтернету речей

Ідея Інтернету речей була не новою, проте не реалізовувалася повноцінно протягом тривалого часу. Значне зростання технології стало можливим завдяки такій головній особливості, як менше споживання енергії, ніж у звичайних вбудованих систем. Підключення через мережу інтернет зробило передачу даних від кінцевого користувача до серверу зручним та ефективним. Зі збільшенням підключення розумних пристроїв, споживчий цифровий сектор економіки

створив великий потік доходів і зростання. Розумне виробництво, розумний ланцюжок поставок, інтелектуальні електромережі та розумні міста стали багатообіцяюче зростати завдяки автоматизації та контролю за допомогою пристроїв IoT.

Як правило, розслідування у сфері кібербезпеки базується на точній моделі атаки. Аналіз загроз безпеці та методи виявлення/пом'якшення атак обмежуються одним рівнем мережі. Однак різні реальних атаки можуть бути комбінацією зловмисних зусиль з кількох мережевих рівнів. Техніка захисту, розроблена для певного рівня IoT і мереж датчиків, може не запобігти атакам з інших рівнів. Таким чином, для ретельної оцінки необхідна повноцінна мережа. Більше того, оцінка на основі моделювання не може точно виміряти затримку, споживання енергії та витоки в побічних каналах методів захисту. Потенціал девайсів IoT, як вектора атаки, без будь-яких втручань, буде постійно зростати. Основними чинниками зростання IoT можна вважати зниження технологічних витрат, розвитку партнерських відносин і бізнес-моделей, а також збільшення можливостей підключення та обчислювальної потужності. Пристрої IoT розглядаються як найслабша ланка в ланцюжку безпеки. Їх виробники часто керуються ринковим принципом «переможець отримує все» що означає, що поспішати на ринок краще, ніж пропонувати безпечні продукти.

Забезпечення кібербезпеки залишається викликом для спеціалістів через особливості мереж пристроїв Інтернету речей, в них поєднується використання різноманітних технологій, частина з яких є застарілими. Ці технології мають традиційні вади в забезпеченні конфіденційності та безпеки даних, і вони повинні виправлятися з огляду на специфіку IoT. Безліч дослідників прикладають свої зусилля для вирішення різних проблем безпеки в IoT, але все ж безпека пристроїв Інтернету речей досі не забезпечена в повній мірі. Мережі IoT вразливі до відомих кібератак, серед яких відмова в обслуговуванні (DoS), розподілена відмова в обслуговуванні, атаки повторення, людина посередині атаки маршрутизації та підслуховування [8].

Пристрої IoT дуже обмежені в потужності обчислювальних ресурсів, через що існуючі криптографічної рішення для кібербезпеки підходять для цих пристроїв, що робить їх не захищеними в питаннях цілісності та конфіденційності інформації. Аутентифікація пристроїв та механізми контролю доступу також є однією з головних проблем безпеки в IoT. Вона пояснюється численною кількістю пристроїв і їх зв'язком одне з одним.

Крім надзвичайної важливості та широкого застосування IoT, його все ж досить важко використати в критичних областях, де конфіденційність та доступність є критично важливими. Наприклад: успішно проведена атака на розумну систему для охорони здоров'я може вилитися у втрати людських життів, а також вона може спричинити матеріальні втрати та смерть учасників руху якщо її проведуть на розумній транспортній системі. Кіберезпека IoT – це не звичайна область і потребує подальшої науково-дослідної роботи, щоб впоратися з цими викликами.

Через обмежування енергоспоживання пристроїв IoT система керування ключами може бути скомпрометована. За словами автора Сімплісіо, спрощені системи аутентифікації ключів мають недостатню надійність у вискошомасштабованих реалізаціях [9]. Такі мережеві протоколи, як IPv6 і IPv4, є цілями для спроби встановлення віддаленого доступу. Автор Czyz продемонстрував що до пристроїв IoT можна отримати віддалений доступ через інтерфейс командного рядка, через використання протоколу Telnet [10]. Мережі IoT часто безперервно обробляють великі дані. Втрата даних або відмова в обслуговуванні може призвести до появи величесного обсягу трафіку в мережі та втрати контролю над роботою системи. За словами автора Ангріші [11], шкідливе програмне забезпечення може залишатися бездіяльним у пристроях IoT, а після запуску воно перетворює пристрій IoT у бота та використовує його щоб проводити DDoS-атаку.

У 2013 році австрійська та німецька електромережі почали виходити з ладу через самозавдану DDoS-атаку і заповнили центральний командний центр трафіком [11]. DDoS-атака може тимчасово відключити комунікаційну мережу

та протокол пристроїв IoT. Автоматизована система інтелектуальних мереж значною мірою покладається на мережу моніторингу IoT. Сенсорна мережа збирає інформаційні дані в режимі реального часу для моніторингу стану обладнання та керуючих установок. Як наслідок, в разі атаки станція втратить можливість моніторингу передаючої та розподільчої мережі.

1.3 Протоколи в IoT

Цей підрозділ здебільшого зосереджений на різних протоколах для Інтернету речей для клієнтів і користувачів. Стек протоколів IoT має чотири основні рівні, показані на рис. 1.1. Більш детально про них шари буде розказано в цьому підрозділі.

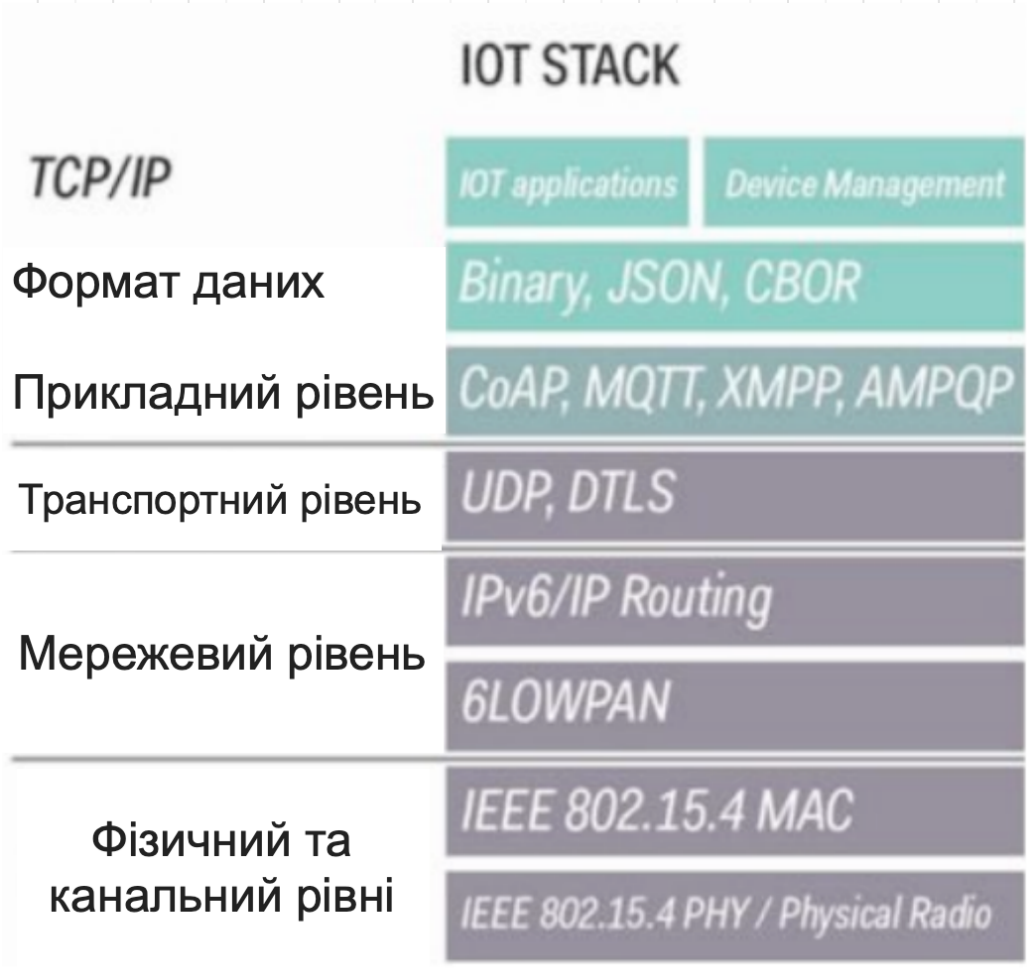

Рисунок 1.1 - Стек протоколів IoT

1.3.1 Фізичний та канальний рівні

На канальному і фізичному рівні передаються цифрові дані від пристрою до серверної частини системи IoT. Цей рівень зазвичай складається з бездротових пристроїв, які збирають дані та передають їх через декілька шлюзів до користувача або до хмари. Передача даних зазвичай визначає низькошвидкісні бездротові персональні мережі (LR-WPAN). Фізичний рівень складається з глобальної мережі низької потужності (LPWAN), Bluetooth Low Energy (BLE), Zigbee, RFID і бездротова локальна мережа IEEE 802.11 є основними протоколами зв’язку.

1.3.2 Мережевий рівень

Мережевий рівень відповідає як шлюз за пересилання даних через вузли мережі. Основна функція цього рівня полягає в тому, щоб направляти пакет від транспортного рівня до канального рівня. Стандартними протоколами є Internet Protocol (IPv6), Internetwork Packet Exchange (IPX), 6LoWPAN and Internet Protocol Security (IPsec) і User Datagram Protocol. Серед них найпопулярнішим протоколом є IPv6 і включає в себе такі функції, як конфігурація адреси та маршрутизація мережі. IPv6 забезпечує повну передачу даних і комутацію мережевих вузлів

1.3.3 Транспортний рівень

Транспортний рівень забезпечує зв'язок між клієнтами. Цей рівень керує всім масовим транзитом пакетів даних від вихідного хоста до кінцевого. Головною особливістю транспортного рівня є керування потоком даних і те що він достатньо надійний для відновлення помилок. Найпоширенішим прикладом протоколу з цього рівню є протоколи TCP і UDP. TCP відповідає за встановлення безпечного з'єднання між клієнтом і сервером. TCP також може надавати кілька кінцевих точок для окремих хостів і керує потоком для уникнення колапсу мережі. TCP забезпечує передачу даних до фактичного адресного вузла та надійність мережі. UDP несумісний із чутливою до часу програмою, оскільки він може втрачати пакети. UPD не має протоколів рукостискання і може спричиняти ненадійність мережі.

1.3.4 Прикладний рівень

Прикладний рівень — це останній рівень протоколів стеку IoT. Прикладний рівень відповідає за обробку даних інтерфейсу користувача. Найпоширенішим прикладом є протокол Constrained Application Protocol (CoAP), полегшена версія HTTP. Іншими протоколами обміну повідомленнями є AMQP і XMPP, які також часто використовуються в додатках IoT. Інтернет речей створює масову передачу даних між пристроями та мережами. Усі пристрої та датчики, системи та виконавчі механізми відповідають за генерацію великого обсягу пакетних даних. Отже, протокол зв'язку є надзвичайно важливим. Одним із цих протоколів є телеметричний транспорт черги повідомлень — це протокол керування передачею на основі підписки та протоколу публікації повідомлень, розроблений для зв'язку між легкими пристроями. Цей протокол не тільки обробляє взаємодію між людьми та пристроями з датчиками, але й між самими машинами.

1.4 Вразливості IoT

Безпека є серйозною проблемою в епоху великих даних, Інтернету речей і штучного інтелекту. Зростаюча кількість розумних пристроїв викликає нагальну потребу розуміння нових загроз кібербезпеці в Інтернеті речей і мережах датчиків і відповідно розробити ефективні заходи протидії цим атакам.

1.4.1 Програмні вразливості

Вразливості програмного забезпечення можуть мати катастрофічний вплив на Інтернет речей. Зловмисники можуть отримати контроль пристроючерез вразливість програмного забезпечення. Наявні науково-дослідні роботи показують що відсутність механізмів захисту безпосередньо спричиняють проблеми безпеки пристроїв Інтернету речей. Чепмен [12] і Родрігес [13] атакували пристрої Інтернету речей із не правильною конфігурацією або слабкою автентифікацією. Макс [14] оцінив безпеку розумних замків і виявив, що аутентифікація та конфігурація за замовчуванням є небезпечними та слабкими. Фернандес та ін. [15] продемонстрували, як неявна довіра завдає шкоди безпеці

пристроїв розумного дому з боку сторонніх програм. Дослідження Костіна базується на оновленнях прошивки та розказує про низку недоліків [16]. Кібератака зазвичай здійснюється зловмисною програмою для того щоб спричинити виток інформації і щоб порушити нормальний процес роботи. Автоматизована розумна електрична мережа значною мірою покладається на мережу моніторингу IoT. Сенсорна мережа збирає інформаційні дані в режимі реального часу для моніторингу стану обладнання та керуючих установок. Деякі з електростанції все ще працюють під управлінням старих операційних систем, таких як Windows XP або Windows7. Вони схильні до атак хробаків через інтернет-серфінг, оскільки у них немає брандмаузера на хості [16]. Як тільки мережа буде скомпрометована, підключений сервер буде скомпрометований і вплине на інші операційні системи в мережі заводу та мережі керування. Будь-які несправності в лінії передачі та розподілу будуть невидимі для центр керування, що призводить до відключення електрики. Зловмисник може отримати доступ і взяти під контроль IoT через вразливість програмного забезпечення. Пристрій IoT – це пристрій з низьким споживанням енергії, а також проводиться оптимізація програмного забезпечення яка зменшує споживання енергії через те що такі девайси розміщуються у віддалених районах. Уразливість прошивки є однією з найважливіших проблем безпеки пристроїв IoT, оскільки вона може порушити регулярну роботу пристрою. Автор Костін [16] зміг отримати пароль у вигляді відкритого тексту та 109 приватних ключів RSA з 428 вбудованого програмного забезпечення та ще 56 сертифікатів SSL із 428 прошивок.

1.4.2 Вразливості мережі

Засоби кібербезпеки для мереж IoT не встигають за швидкою швидкістю підключення нижчих граничних вузлів. Будь-яка лазівка в безпеці створить можливість для супротивника здійснити атаки та спричинити витік конфіденційної інформації. Останні звіти показують, що хакери змогли зламати пристрої Amazon IoT, такі як дверний замок та домашня камера відеонагляду. Зловмисник зміг перехопити імена користувачів і паролі через незашифрований

протокол передачі гіпертексту (HTTP) у локальній мережі користувача, таким чином отримавши доступ до пристроїв IoT. Експерти галузі вважають, що легко порушити безпеку камери IoT і голосових помічників, таких як Alexa і Google Home [17].

Багато пристроїв IoT покладаються на протоколи UPnP, щоб забезпечити такі функції, як легке налаштування і контроль. Однак UPnP використовує протокол HTTP, який не забезпечує жодного захисту конфіденційності та цілісності. Існують різні методи зламу UPnP протоколу через брак аутентифікації, перевірки та реєстрації. Таким чином, що пристрої IoT в середовищі розумних будинків є вразливими, якщо ні захищений якимись додатковими вдосконаленими захисними засобами.

1.4.3 Вразливості апаратного забезпечення

Для отримання даних у реальному часі датчики дистанційно вимірють та підключаються до пристрою IoT. Ці датчики можуть обробляти дані в реальному часі на вузлі і зменшувати навантаження на центральну мережу. Цей тип датчика також може бути імплантований на стороні електростанції. Датчик контролює навантаження та керує турбінами генераторів. Більшість датчиків є низькоенергетичними пристроями IoT, і дані передаються в систему моніторингу зашифрованими. Противник може застосувати атаку людини по середині (MITM) для витоку конфіденційної інформації або відправити власний маніпулятивний сигнал до центру управління. Атаки на розширену інфраструктуру інтелектуального вимірювання (AMI) створюють нову точку входу для зловмисників для злому інформації з електромережі. Основними наслідками атак AMI є крадіжка даних, викрадення електроенергії, локалізована або масова відмова в подачі електроенергії та порушення роботи електромережі. Приблизно 43% домогосподарств у США зараз мають інстальовані розумні лічильники. Такі пристрої та багато інших збирачів даних для електромережі зазвичай спілкуються за допомогою радіочастотних методів, спектр частот яких становить 900 МГц у ISM діапазоні (промисловий, науково-медичний). У діапазоні ISM багато

неліцензованих пристроїв змагаються за використання цього спектру. В результаті спуфінг атаки на діапазон ISM легко досягають успіху.

1.4.4 Вразливості на рівні процесорів

Апаратний троян (HT) – це зловмисне вставлення в існуючу схему, що спричиняє зміну функціональності після активації. Зловмисники вставляють певне корисне навантаження HT в пристрій IoT, щоб спричинити витік інформацій або щоб маніпулювати нею чи обійти безпеку системи. Інтегровані мікросхеми, такі як специфічні для застосунків системи на чіпі та пристрої, що програмуються, є вразливими до цифрових та аналогових HT. Через особливості малої потужності пристроїв криптографічні блоки є більш незахищеними, ніж високопродуктивні інтегральні схеми. Помилка програмного забезпечення може бути легко виправлена оновленням прошивки, і це економічно вигідно. Однак апаратні вразливості, такі як HT, не можуть бути усунені простим програмним забезпеченням і є більш дорогими при виправленні. Через складну природу відновлення, HT може спричинити постійні пошкодження та погіршення якості обслуговування. Цей троян може бути легко реалізований ненадійними сторонніми постачальниками IP-адресів, ненадійним виробником та стороннім інструментом автоматизації електронного дизайну для виготовлення мікрочіпів пристроїв Інтернету речей. Автор Шпрейцер продемонстрував, що супротивник спостерігає за поведінкою пристрою IoT, не порушуючи його, використовуючи пасивний електромагнітний аналіз бічного каналу [18]. Атаки по бічних каналах є неінвазивними типами, які можуть спричинити фізичну реалізацію техніки вилучення даних. Ціллю таких атак є секретний ключ вбудованих пристроїв IoT. Вони можуть вилучати зашифровану інформацію з криптографічного модуля. Критичні дані такі як паролі та ключі можна вгадати шляхом моніторингу та вимірювання споживання електроенергії та електромагнітного випромінювання криптографічних операцій. Зловмисник може аналізувати інформацію побічного каналу, таку як напруга, струм, теплове випромінювання та інформацію про час, щоб розшифрувати важливу інформацію пристрою. Автор Памму запропонував

атаку кореляційного електромагнітного аналізу, щоб перехопити модуль шифрування AES-128 і успішно отримати секретний ключ [19]. Автор Генкін проілюстрував, що вилучення повних 4096-бітних ключів дешифрування RSA можна здійснити за допомогою чутливого мікрофона, розміщеного на відстані 4 м від пристрою IoT [20].

1.5 DoS та DDoS атаки

1.5.1 Опис атак та основні етапи

DoS (Denial of Service – атака «відмови в обслуговуванні») – це кібератака направлена на систему з завданням вивести її з ладу, спричинити відмову. Мається на увазі створення умов, при яких звичайні користувачі можуть отримати відмову в отриманні доступу до конкретних сервісів, або ж цей доступ стає ускладненим.

DDoS (Distributed Denial of Service) – розподілена атака на відмову в обслуговуванні, це атаки DoS що виконуються водночас з багатьох пристроїв, це можуть бути боти. Метою таких атак є руйнування системи таким чином, щоб її служби стали недоступні. Хоча існує багато різних варіантів і типів атак, по суті зловмисник використовує вразливості, щоб взяти під свій контроль кілька машин. Різниця між атакою DoS і DDoS полягає в тому, що це завдання виконує кілька машин. Ці скомпрометовані хости продовжують надсилати трафік до перевантажених служб, щоб звтчайні користувачі не мали змоги працювати з певними сервісами системи. Процес зображений на рис. 1.2. Для обговорення різних типів DDoS-атак у існуючій літературі наведено кілька різних категорій або так званих таксономій. Модель OSI зазвичай використовується для обговорення комп'ютерних мереж. Це актуально, оскільки DDoS-атаки є атаками, спрямованими на порушення доступу до мережевих послуг. Крім того, існують таксономії, спеціально пов’язані з DDoS-атаками а також їх операцій.

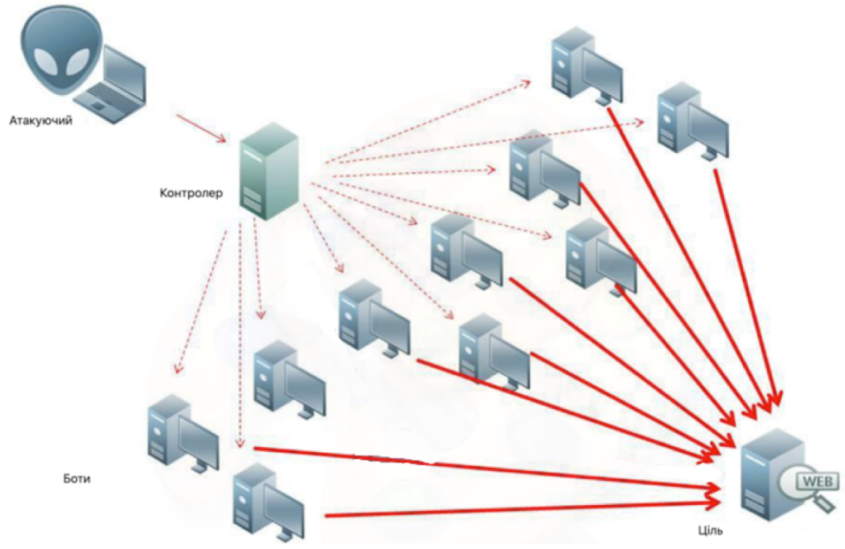

Рисунок 1.2 – Спрощена ілюстрація як працює DDoS атака

Таксономії атак, згадані вище класифікуються на рис 1.3.

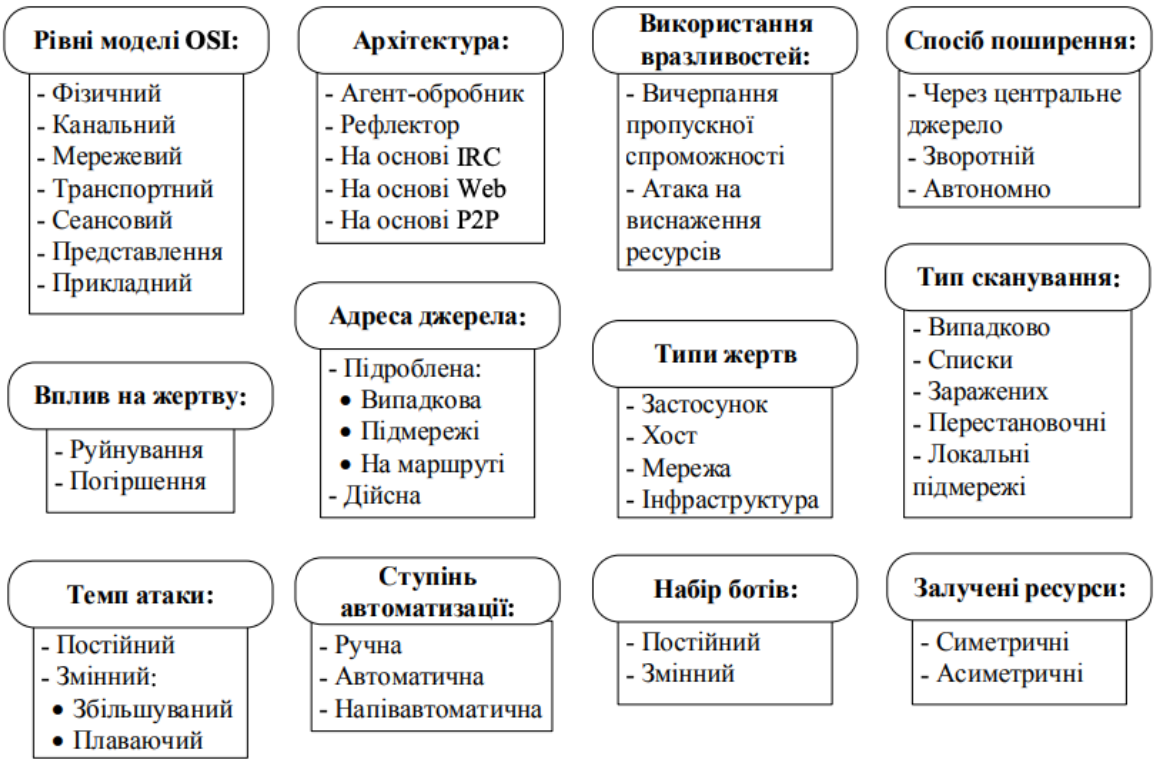

Рисунок 1.3 – Класифікація (таксономія) DDoS-атак

DDoS зазвичай виконується в такі етапи:

- Сканування: атакуючий досліджує мережу на предмет наявності вразливих пристроїв, які пізніше будуть використовуватися як боти для здійснення нападу на справжню ціль. Раніше цей етап виконувався в ручному режимі, проте в наш час для цього існують автоматизовані інструменти.

- Інфікування та експлуатація: на цьому етапі знайдені вразливі машини інфіковуються та додаються до ботнету.
- Зв’язок: хакер використовує інфраструктуру для видачі команд та керування, для того аби в режимі реального часу взаємодіяти з ботнетом, та знати які із ботів доступні.
- Атака: кіберзловмисник надає команду розпочати атаку, а інфіковані машини розпочинають відправку запитів на жертву. Властивості атаки (адреса жертви, тривалість атаки та властивості пакета) зазвичай налаштовуються під час цього етапу. Не дивлячись на те, що підміна IP-адрес не обов’язкова для успішного проведення DDoS-атаки, зловмисники часто підробляють IP-адресу задля приховання реальних інфікованих машини під час атаки.

1.5.2 OSI

Еталонна модель OSI — це концептуальна модель, яка представляє, як комп'ютерні системи взаємодіють одна з одною. Кожен рівень представляє різні функції для цієї комунікаційної мети. Важливо зазначити, що це скоріше концептуалізація мережі, ніж точне визначення. Кожен рівень потенційно може стати причиною збою. Це означає, що послуга може бути порушена, коли один рівень повністю заповнено.

Cloudflare, як одна з найбільших у світі мережевих і таких що протидіють DDoS, посилається на модель OSI, щоб пояснити три різні категорії атак [21]. А саме об’ємні атаки, атаки прикладного рівня та протокольні атаки. Об’ємні атаки ефективні, як і інші атаки, оскільки вони надсилають великі обсяги шкідливого трафіку, щоб споживати пропускну здатність жертви. Атаки прикладного рівня відносяться до застосування, зокрема рівня кінцевих користувачів в моделі OSI. Вони не тільки використовують пропускну здатність, але й системні ресурси. Атаки протоколів на транспортному та мережевих рівнях або атаки вичерпання стану атакують базову мережеву інфраструктуру.

1.5.3 Атаки відмови в облуговуванні через пристрої IoT

Атаки відмови в обслуговуванні здійснюються через пристрої IoT, декілька механізмів роблять їх цікавим вектором атаки. Хоча пристрої IoT часто не мають високої обчислювальної потужності або можливостей пропускної здатності в порівнянні з традиційними зараженими комп'ютерами, проте вони стають доступні у великій кількості за відносно невеликих зусиль. У 2020 році кількість пристроїв IoT вперше перевищила кількість пристроїв без Інтернету речей: їх є 11,7 мільярдів з усіх 21,7 мільярдів. Було визначено п'ять особливих причин, чому ботнети IoT стають все більш популярними: постійна та ненав'язлива робота, слабкий захист, погане технічне обслуговування, значний трафік атак і мінімальні інтерфейси користувача.

Зростання вразливих пристроїв IoT також призвело до збільшення сімейств шкідливих програм із розширенням можливостей DDoS та більш широким діапазоном пристроїв, на які вони націлені. Деякі поточні активні ботнети — це Hajime та IoT_Reaper [22]: Hajime не має жодних можливостей для DDoS-атаки, а IoT_Reaper поки що, як відомо автору, не має жодної активності. Хоча вони бачили високу кількість заражених пристроїв, на даний момент вони не були виправлені. Тому вони також менш популярні, ніж варіанти Mirai, останній вважається найбільш домінуючою формою за останні кілька років через її високу ефективність і резонансні випадки атак. Це робить Mirai кращим прикладом для розробки IoT DDoS-атак. Mirai — популярне зловмисне програмне забезпечення, яке націлено на пристрої IoT і використовує їх для DDoS-атак. Деякі відомі DDoS-атаки були здійснені цим ПЗ, так у 2016 велика кількість атак була здійснена на Dyn (постачальник DNS), «Krebs on Security» (відомий блог про кібербезпеку) та на OVH (провайдер) [23]. Вихідний код Mirai опублікований розробником, що дозволяє глибоко аналізувати роботу цього ботнету. Для атаки використовуються чотири компоненти: (1) бот, який є пристроєм IoT, зараженим шкідливим програмним забезпеченням, що виконує атаку, (2) сервер керування та контролю (C&C), який використовується зловмисником для зв'язку зі своїм ботнетом, (3) сервер завантажувача, який з'єднується з пристроєм IoT для впровадження

шкідливого програмного забезпечення, і (4) сервер звітів, який збирає всю інформацію про активні заражені пристрої. Схема роботи зображена на рис. 1.4. Використовуючи ці чотири компоненти, Mirai може підтримувати, розвивати та створювати атаки. Вони можуть підтримувати та контролювати програмне забезпечення за допомогою модуля ki11er, який вимикає порти 22, 23 і 80, тому інші шкідливі програми не можуть контролювати ці пристрої. Крім того, він сканує та знищує подібні шкідливі програми, створені іншими атаками. Ботнет може продовжувати рости, оскільки модуль сканера використовує telnet і випадково згенеровані IP-адреси для пошуку інших вразливих пристроїв IoT. Вони перевіряють свій шлях для доступу за допомогою визначеного списку комбінацій облікових записів і паролів за замовчуванням і звітують назад на сервер звітів. Mirai використовує мінімальне використання інтерфейсів пристроїв IoT як люди, намагаючись вгадати інформацію про доступ, оскільки користувачі часто не змінюють комбінацію облікового запису та пароля за замовчуванням.

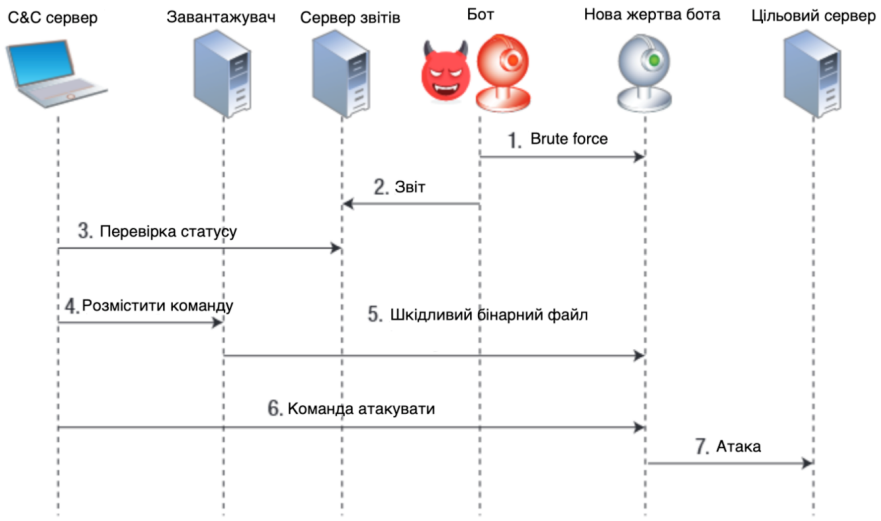

Рисунок 1.4 – Ботнет Mirai, експлуатація та зв'язок

1.6 Методи виявлення та захисту від DoS/DDoS-атак

Задачею виявлення DoS/DDoS атак є розпізнавання який трафік є нормальним а який відноситься до атаки. Для ефективного виявлення необхідно дотримуватися наступних критеріїв успіху: швидкість та точність виявлення. Щоб успішно виявляти атаки, спершу потрібно зібрати достатньо інформації про

трафік мережі, а після цього аналізувати дані для того, щоб дізнатися, чи є запит аномальним. Методи виявлення є одним із ключовим факторів при створенні надійного захисту від DDoS. Відомі два способи виявлення таких атак: (1) вбудовані перевірки усіх запитів та пакетів та (2) аналіз мережевого трафіку. До вбудованих перевірок відносять різного типу брандмаузери та системи запобігання проникненню (IPS) а також балансувальники навантажень, вони можуть підходити для виявлення DoS атак в звичайних пристроях але не в реаліях IoT, а от для великих за обсягом даних і запитів DDoS атак вони не підходять бо це перевантажить ці засоби в будь-якому випадку. Системи виявлення які знаходяться на окремій машині є більш ефективними, адже вони можуть бути більш потужними, більш захищеними, а також добре вписуються в сучасну мікросервісну архітектуру, в якій вони виконують лише одну задачу. На таких машинах виявлення аномалій виконується системою що спочатку отримує дані потоку пристроїв, а потім аналізує цей потік для виявлення підозрілих даних. Після цього запускається механізм пом'якшення атак вручну або автоматично за допомогою маршрутизації.

Головна небезпека переважної кількості DoS/DDoS атак – в їх абсолютній очевидності і «нормальності». Адже помилки в ПЗ завжди можуть бути виправлені, а повна витрата ресурсів – це майже нормальна поведінка для сучасних інформаційних систем. Сервери налаштовують таким чином щоб вони не простоювали тому із закінченням ресурсів щоденно стикаються багато адміністраторів. Якщо ж вводити обмеження на трафік і ресурси неправильно, то можна легко втратити частину клієнтів, яким буде відмовлено в обслуговуванні. Захист від таких атак можна розділити на 3 напрямки, а саме : виявлення, запобігання, реакція. Основною метою запобігання атакам є її припинення до нанесення збитків. Виявлення має перед собою мету контролювати та досліджувати систему в якій відбуваються аномальні події. Способи реакції це дії які жертва виконує вже після того як було виявлено проведену атаки. У таких методах за виявлення і протидію нападу відповідають жертви.

Як показує аналіз: найефективнішими засобами для запобігання і виявлення DoS/DDoS атак є такі що використовують інтелектуальні методи (IDMS), поміж них можна виділити: експертні системи, статистичний метод, методи на основі використання машинного навчання а також нейронних мереж. Методи які базуються на статистиці спроможні змінювати свої оцінки залежно від поведінки суб'єкта, вони не вимагають знань про всі можливі атаки. Аналізатори які звертаються до нейронних мереж можуть спочатку вивчати особливості атак а потім вже класифікувати запити, але точність і швидкість таких прогнозів залежить від якості навчання цих нейромереж. У випадку використання ML витрачається набагато менше часу для виявлення, але вимагається висока обчислювальна потужність на машині де аналізується поведінка. [1, 2] Після аналізу літератури були виявлені деякі недоліки в знайдених дослідженнях, наприклад в [2] успішністю алгоритму вважали лише показник точності, хоча він не підходить для незбалансованих наборів даних.

Методи виявлення DDoS атак мають наступне завдання: потрібно зробити найбільш можливий точний прогноз який вкаже атакують пристрої в даний момент часу, чи не атакують. Одним із найбільш важливих факторів для цієї проблеми є швидкодія роботи методу класифікації, тому що найменші затримки приводять до великих втрат.

Стандартні методи аналізування статистик не дозволяють виявляти невідомі до аналізу атаки, тому в якості інструменту вирішення цієї проблеми часто використовують і досліджують алгоритми машинного навчання [3]. В даній роботі ми також будемо використовувати саме цей підхід. В останні роки проводять багато досліджень по застосуванню машинного навчання як інструмент виявлення DoS атак. Проте вони часто не достатньо обгрунтовані, у них мало деталей отриманого результату, і також вони рідко спеціалізуються на мережах IoT, та оцінють результати лише за точністю, або не чітко описують підхід за яким вибиралось рішення. Саме тому метою даної роботи є підвищення обґрунтованості вибраної методики та точності виявлення DoS та DDoS атак в IoT з використанням машинного навчання.

Безпека IoT є серйозною проблемою в епоху великих даних і штучного інтелекту. Зростаюча кількість розумних пристроїв викликає нагальну потребу зрозуміти загрози безпеці IoT і мереж датчиків і відповідно розробити ефективні заходи протидії цим атакам. Існуючі механізми захисту зазвичай оцінюються в умовах штучної атаки, визначеної конкретною моделлю атаки, тому, можливо, їм не вистачає стійкості проти інших атак. Окрім кібератак, фізична атака також може мати катастрофічний вплив на Інтернет речей. Фізична атака, яка зазвичай здійснюється з використанням апаратних троянів може використовуватися щоб порушити нормальний потік операцій.

1.7 Ботнети

1.7.1 Передісторія мереж ботів

Ботнети мали багату історію та розвивалися протягом багатьох років, руйнуючи та руйнуючи комп'ютерні та мережеві системи. Спочатку ботнети створювалися для благодійних цілей, їх основна функція полягала в тому, щоб надавати адміністративну допомогу Інтернет-ретрансляційним чатам (IRC), формі спілкування, досить популярній у 90-х. Перший IRC-бот з'явився в 1993 році, отримав назву Eggdrop і надавав допомогу в комунікації каналу IRC. Після Eggdrop з'явилися перші шкідливі боти, причому GTbot в 1998 році став першим у своєму роді, який зміг виконувати сценарії за запитом через IRC-канал командування та керування (C&C). У 2003 році з'явилися нові, складніші боти, такі як Agobot, який був більш надійним і гнучким, ніж ті які були згадані раніше, а також він включав постійний канал C&C. У 2007 році з'явився Storm, ботнет, який був охарактеризований як один з найпотужніших ботнетів свого часу. Він використовував інфраструктуру P2P C&C, основною функцією якої були спам-повідомлення, DDoS-атаки і мав можливість порушити роботу Інтернету в масштабах цілих країн. У тому ж році з'явився, мабуть, один з найбільш сумнозвісних ботнетів, Zbot або Zeus. Маючи на той час під своїм контролем майже 3,6 мільйона ботів, пізніше з'явилися інші варіанти, включаючи версію P2P у 2011 році під назвою Gameover Zeus, яка була здатна виконувати широкий

спектр шкідливих дій, включаючи крадіжку банківського рахунку, DDoS та спам. Зрештою, він був знищений завдяки співпраці ФБР, службою кібербезпеки Великої Британії, Shadowserver Foundation та CTU IN Dell у червні 2014 року. Kraken входив до сімейства спамерських бот-мереж і, як повідомлялося, у квітні 2008 р. включав 400 000 ботів до своєї армії зомбі. Torpig був ботнетом для вилучення даних, який виконував атаки «людина в браузері» та використовував централізовану інфраструктуру C&C на основі HTTP.

Наступник ботнету Storm, Waldec, був виявлений у 2009 році, він мав інфраструктуру керування та керування P2P, його основна функція полягала в розсилці спам-повідомлень, що досягали близько 7000 повідомлень на день, хоча він також здійснював крадіжку облікових даних та DDoS-атаки. Зрештою, у 2010 році його знешкодили. Іншою помітною віхою для бот-мереж у 2009 році стала поява попередника мобільних бот-мереж, де бот-мережі використовують мобільні телефони як своїх ботів (зомбі), під назвою SymbOS\Yxes, які були націлені на пристрої Symbian і використовували SMS-повідомлення для саморозповсюдження.

Після появи SymbOS наприкінці 2010 року було помічено перший ботнет, орієнтований на пристрої Android під назвою Geinimi. В основному він розповсюджувався у Китаї, він використовував просту інфраструктуру C&C на основі HTTP і міг надсилати SMS, електронні листи, знати місцезнаходження зараженого пристрою.

Останнім часом розробники ботнетів скористалися перевагами стрімкого поширення та постійного зростання IoT, і ми вже бачили приклади IoT ботнетів і на що вони здатні. Наступним еволюційним кроком бот-мереж були ботнети, що складалися з пристроїв IoT. Найвідоміший з них вперше з'явився у вересні 2016 року під псевдонімом Mirai, ми вже згадували його раніше. Mirai здійснила одні з найпотужніших DDoS-атак в історії Інтернету, а саме: 620 Гбіт/с направлені на веб-сайт Браяна Креба, 1,1 Тбіт/с проти французького постачальника хмарних послуг OVH, а в жовтні 2016 року атакувала постачальника послуг Dyn та забрала частини Інтернету, як-от Twitter, Netflix та GitHub. Після випуску

вихідного коду Mirai з'явилися різні варіанти, як-от Persirai, який діє з квітня 2017 року, це більш досконала версія Mirai, яка орієнтована на конкретні пристрої вибраних виробників. Інші ботнети IoT включають Hajime, який з'явився в жовтні 2016 року і використовував децентралізовану інфраструктуру C&C, яка, здавалося, «захищає» пристрої від зараження Mirai. Нарешті, BrickerBot був помічений у квітні 2017 року, і, як випливає з назви, він намагався «замурувати» пристрої IoT, що можна вважати постійною DoS-атакою.

1.7.2. Активності ботнетів

Основна причина, чому ботнети привертають таку велику увагу, не через майстерні способи, які бот-майстри використовують, щоб заховати своїх ботів від правоохоронних органів, а скоріше через практичні можливості, якими володіють ботнети, і послуги, які вони надають бот-майстрам та їхнім клієнтам. Існують різні методи злому, які використовуються бот-мережами, включаючи атаки розподіленої відмови в обслуговуванні, кейлогінг, фішинг, розсилку спаму, шахрайство з крадіжкою кліків, крадіжку особистих даних і навіть розповсюдження інших шкідливих програм-ботів.

Ботнети, як правило, спеціалізуються на виконанні невеликої підмножини вищезгаданих методів злому, хоча були випадки, коли варіанти ботнетів були здатні до різних типів шкідливих дій, наприклад, Gameover Zeus, який міг виконувати DDoS атаки, надсилати спамові листи. та викрасти інформацію про банківські рахунки. Існує багато способів, через які ботнет може здійснити DDoS-атаку, і, виходячи з техніки та протоколів, які використовуються, є кілька прикладів таких атак, деякі з яких: «Відмова у сплячому режимі», «UDP flood», «TCP SYN flood», ICMP ping flood , Ping of Death, атака Smurf, посилення DNS, HTTP flood. Існують різні засоби контролю кібербезпеки, наприклад, виявлення загроз і розвідка, а також методи виявлення вторгнень, використовувалися для розпізнавання та запобігання бот-мережам у мережах IoT. Ці методи певною мірою корисні, оскільки вони можуть виявляти лише відомі кібератаки, але вони не можуть ідентифікувати атаки нульового дня (тобто нові/майбутні атаки),

оскільки немає сигнатур цих атак, що зберігаються в чорних списках. Це мотивує дослідників зосередитися на механізмах форензики, щоб відстежувати та визначати походження кібератак, а також допомагати в дослідженні того, як структури ботнетів виникають у системах IoT; таким чином можна покращити контроль безпеки під час виявлення відомих і нових бот-мереж.

1.8 Система виявлення вторгнень

Системи виявлення вторгнень (IDS) - класифікуються на системи виявлення вторгнень в мережі та на хостах. Мережеві системи виявлення вторгнень (NIDS) розміщуються в стратегічній точці мережі, як правило, вузлі, який з'єднує внутрішню мережу з Інтернетом, їм доручено сканувати вхідний і вихідний трафік на предмет відомих моделей атак [24]. У міру виявлення відомих зловмисних шаблонів будуть відображуватися сповіщення та можуть бути запущені подальші механізми криміналістичної експертизи, що зменшує час відповіді та потенційно підвищує точність розслідування. Такі IDS можуть включати ML у свій процес виявлення аномалій з класифікацією, кластеризацією та іншими методами, що використовуються для виявлення аномального трафіку.

Висновки до першого розділу.

Отже в цьому розділі було досліджено існуючі дослідження в сфері пристроїв Інтернету речей, було виявлено головні загрози притаманні їм на різних рівнях. Цей розділ показав, що пристрої IoT продовжують зростати, проте існують структурні проблеми для їх захисту. Хоча зловмисне програмне забезпечення IoT існувало деякий час, лише з Mirai воно стало серйозною загрозою. Mirai може підтримувати, розвивати та створювати атаки та нападати на вразливі пристрої з логіном користувача за замовчуванням. Ці атаки також стали більш популярними, оскільки вихідний код зловмисного програмного забезпечення є загальнодоступним, що дозволяє кожному створити свій власний ботнет. DDoS-атаки на основі IoT залишаються проблемою, і їх потенціал, тільки зростає.

Проблема виявлення та захисту від DDoS/DoS атак є не тривіальною а також досі актуальною та не вирішеною через те що її не можна вирішити задовільно за допомогою стандартних методів, адже хакери комбінують різні типи атак так щоб їх було важко відрізнити від нормальних запитів. Також в існуючих дослідженням на тему захисту від атак DDoS не достатньо уваги приділяють мережам IoT, тому було проаналізовано основні методи та методики виявлення таких атак, з чого зроблено висновок що побудова системи виявлення вторгнень з використанням алгоритмів машинного навчання може покращити результати виявлення та дозволяє запобігти тяжким наслідкам від спроб проведення атаки на відмову в обслуговуванні. Було виявлено що в багатьох дослідженнях вибір кращих моделей машинного навчання здійснюється за точністю, або часом, а сама методика вибору не описується чітко. За мету роботи було покладено реалізувати деякі із алгоритмів ML задля вивчення можливості їх використання в мережі Інтернету речей та вибрати найкращий для побудови IDS.

2 ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ

У наш час системи виявлення вторгнень – це обов'язковий елемент для забезпечення надійного захисту від атак в мережі. Головною задачею таких систем є виявлення випадків не дозволеного доступу в закриту мережу або несанкціонованого керування. Для розв'язання цієї задачі IDS виконують такі основні дії:

- проводять моніторинг усіх подій в системі із метою виявлення аномальної поведінки;
- інформація про інциденти збирається та записується локально, а потім відправляється в визначену централізовану систему збору подій;
- адміністратори інформаційної безпеки або адміністратори системи отримують повідомлення про можливу атаку;
- створюються звіти, у яких уточнюють або, навпаки, узагальнюють дані по одній або декільком подіям.

2.1 Задача виявлення DoS/DDoS атак

Для розробки ефективного алгоритму виявлення DoS/DDoS атак на основі машинного навчання спочатку потрібно розглянути математичну модель виявлення із якою потрібно буде працювати.

Формальна постановка задачі виявлення DDoS атак:

Нехай існує деяка множина вхідних даних X і є множина відповідей Y. Y визначає приналежність конкретного прикладу до атаки. Існує певна тренувальна вибірка $\{x_1, \ldots, x_n\} \subset$ X та вибірка відповідей $\{y_1, \ldots, y_n\} \subset$ Y.

Задачею виявлення атаки в такому разі називають визначення такого правила(алгоритму або моделі), який дасть значення найближче до правильних значень на всій множині X:

a: X → Y

$x_i \subset$ X набір ознак даних про трафік, ознаки поділяються на типи:

- Кількісні (значення із множини дійсних чисел)
- Булеві {0, 1};
- Номінальні (значення із скінченної підмножини N)
- порядкові, що представляють собою номінальні ознаки, із лінійним порядком;

$y_i \subset Y$ показує чи є даний приклад атакою.

Булевою ознакою може бути певний стовпчик даних який вказує що це приклад атаки або не атаки. Номінальною може бути протокол, до якого належить цей пакет. Порядковою ознакою можна назвати кількість прапорців у заголовках пакета. Прикладом кількісних ознак є розмір пакету.

Тренувальна вибірка створюється із певного набору даних у вигляді матриці ознак, у якій рядки це приклади трафіку а також це відповідні відомі стани.

Y також може складатися із ознак різних типів: булевих, номінальних. номінальних, що перетинаються, також приклад даних може належати одразу до декількох типів атак. Особливістю двох останніх варіантів є те, що повинно бути вказано, чи є приклад атакою, а вже після цього буде визначено її тип.

2.2 Машинне навчання

2.2.1 Опис

Машинне навчання описують як множину обчислювальних методів, які використовують досвід задля покращення продуктивності або з метою точного прогнозування [25]. В цьому випадку досвід це певна попередня інформація, тобто тренувальні дані, які, як правило, зібрані у формі електронних даних та надані для аналізу. Прогнози в такому випадку роблять без явного програмування, лише базуючись на даних що були надані. Для успіху прогнозування вирішальну роль грають розмір та якість наданих датасетів. ML полягає у розробці точних та ефективних алгоритмів передбачення. Так само як і в інших областях інформаційних технологій, критичним показником якості алгоритмів є їх складність а також додатково складність датасету, який потрібно аналізувати.

Одним із методів штучного інтелекту є машинне навчання, його головною перевагою називають не пряме вирішення поставлених завдань, а тренування в процесі прийняття великої кількості схожих рішень для задач. При використанні таких алгоритмів використовують засоби чисельних методів, статистики, теорію графів, теорію ймовірності, різноманітні оптимізаційні методи, а також методики для роботи цифровими даними. Навчання розділяють на такі типи:

– дедуктивне, базується на формалізації та оцінці знань експертів та внесення цієї інформації до бази даних.

– індуктивне, основується на виявленні певних закономірностей у наборі даних;

2.2.2 Типові завдання машинного навчання

Зазвичай завданнями ML є:

- класифікація – розбиття на категорії
- кластеризація - розподіл на однорідні підмножини
- рангування – впорядковування даних
- зменшення розмірності з збереженням певних властивостей
- регресія – для кожного елементу прогнозується значення

2.2.3 Термінологія при роботі з машинним навчанням

Введемо важливі терміни для роботи з машинним навчанням:

- приклади: це екземпляри з набору даних, які використовують для тренування або оцінювання;
- мітки: це категорії або значення, присвоєні прикладам. Мітки це результат функцій прогнозування;
- параметри: це набори атрибутів, які зазвичай представляються у векторному вигляді, який представляє собою приклад;
- тренувальні дані: це множина прикладів, які необхідні для навчання алгоритму;

- приклад перевірки: це приклади, які використовують для того щоб здійснити налаштування параметрів алгоритму;
- тестовий зразок: це приклади, які використовують при оцінці ефективності алгоритму;
- функція збитків: це функція, що вимірює втрати або різницю між передбачуваною та реальною міткою

2.2.4 Побудова моделей машинного навчання

Представлені нижче пункти описують етапи побудови моделей ML, їх також можна побачити на рисунку 2.1:

1. Підготовка та збирання інформації.
2. Підбір ознак (параметрів)
3. Вибір алгоритмів
4. Вибір параметрів і моделей
5. Тренування (навчання)
6. Оцінка ефективності отриманих результатів

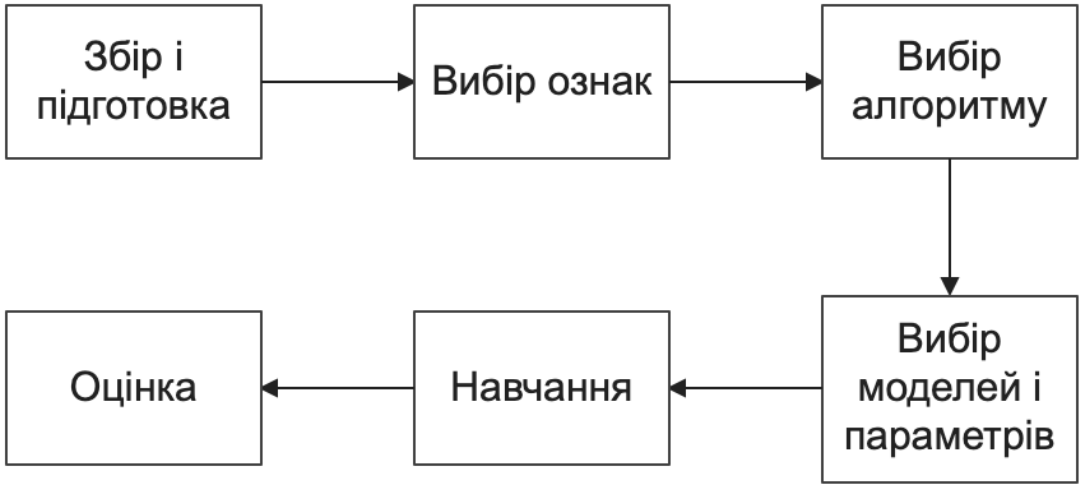

Рисунок 2.1 – Етапи побудови моделі ML

На першому етапі інформація збирається у форматі який може бути наданий алгоритмові в якості вхідних даних. Дані можуть містити шум, можуть не мати чіткої структури тому їх треба попередньо обробити. Хороша підготовка даних

дозволяє проводити ефективний аналіз, обмежує помилки та неточності, які можуть виникати з даними під час обробки, і робить усі оброблені дані зручними у користуванні. Це також стало простіше з новими інструментами, які дозволяють будь-якому досліднику самостійно очищати та кваліфікувати дані. Очищення даних часто є найбільш трудомісткою частиною процесу підготовки даних, але це дуже важливо для видалення несправних даних і заповнення прогалин. Серед важливих завдань тут:

- Видалення сторонніх даних і викидів.
- Заповнення пропущених значень.
- Відповідність даних стандартизованому шаблону.
- Маскування конфіденційних або чутливих даних.

На другому кроці з поміж усіх існуючих ознак відбираються ті які мають значення для процесу навчання. Часто набори даних містять велику кількість ознак, число яких може вимірюватися в сотнях або навіть досягати тисячі. При побудові моделей машинного навчання не завжди зрозуміло, які з цих ознак дійсно є важливими (тобто мають зв'язок з цільовою змінною), а які є надлишковими (або шумовими). Видалення таких параметрів допомагає досліднику краще зрозуміти дані, а також скорочує час налаштування моделі і правильність її прогнозів. Іноді ця задача і може бути найважливішою, наприклад, досягнення оптимального набору ознак може допомогти розшифрувати механізми, які лежать в основі дослідженої проблеми. Це стає корисним для розробки методик. Методи відбору ознак зазвичай поділяють на 3 наступні категорії: фільтри, вбудовані методи, та обгортки.

На наступному етапі вибираються кращі алгоритми для вирішення конкретної проблеми. Для будь-якої задачі машинного навчання можна застосувати численні алгоритми та створити декілька моделей. Наприклад, проблему класифікації для виявлення спаму можна вирішити за допомогою різноманітних моделей, включаючи наївний баєс, логістичну регресію та методи глибокого навчання, такі як BiLSTM. Наявність великої кількості варіантів — це добре, але вирішальне значення має рішення, яку модель впровадити у

виробництво. Хоча у нас є ряд показників продуктивності для оцінки моделі, нерозумно реалізовувати кожен алгоритм для кожної проблеми. Це вимагає багато часу і багато зайвої роботи. Тому важливо знати, як вибрати правильний алгоритм для конкретного завдання. Далі представлений список факторів які потрібно враховувати при реалізації алгоритму, його було сформовано в ході дослідження літератури: інтерпретація, кількість точок даних і ознак, формат даних, лінійність даних, час навчання, час передбачення, вимоги до пам'яті, останні 3 дуже важливі в контексті нашої задачі для пристроїв IoT.

На четвертому етапі коректуються параметри алгоритмів машинного навчання, також цей крок називають тюнінгом. Точне налаштування прогнозної моделі машинного навчання є важливим кроком для підвищення точності прогнозованих результатів. Для цього підбираються оптимальні параметри для алгоритмів за яких вони мають найкраще співвідношення точності до затраченого часу.

Після вибору алгоритму та значень параметрів модель навчають, використовуючи частину датасету як навчальні дані. Частина даних береться з всього набору і дається алгоритму для виявлення шаблонів, потім він може робити припущення на основі отриманих даних.

Наприкінці, перед впровадженням системи модель тестують на інших даних (не тренувальних) задля того щоб дізнатися якість прогнозу. Якість прогнозу часто помилково оцінюють лише за такою метрикою як точність, хоча вона не підходить для оцінки незбалансованих наборів даних, де кількість атак значно перевищує кількість нормальних даних. Тому в даному дослідженні буде використано інші показники якості прогнозів, які будуть описані в третьому розділі.

2.3 Методи прийняття рішень

Раніше ми розглядали завдання, в яких була єдина ціль (критерій) і тим самим єдина цільова функція. Тепер припустимо, що якість рішення оцінюється за багатьма критеріями. Тоді вибір найкращого рішення є нетривіальним завданням,

виникає нова проблема: що слід розуміти під оптимальним рішенням? Справа в тому, що об'єктивно невідомо, яке рішення краще, якщо критеріїв багато, і вони, можливо, «конфліктуючі». Зокрема, незрозуміло, яке рішення краще, якщо дослідник одночасно прагне максимізувати точність та мінімізувати витрати часу. Тому потрібно займатися пошуком компромісного рішення, яке враховує важливість усіх цільових функцій. Таким чином, дослідники прийшли до поняття ефективного (оптимального щодо Парето) рішення. Властивість ефективності (у крайньому випадку слабку ефективность) має мати будь-яке рішення, що претендує на те, щоб його назвали оптимальним. Однак за такого підходу залишається проблема вибору єдиного рішення, оскільки парето-оптимальних рішень, як правило, багато (часто нескінченно багато).

Проблему вибору рішення з парето-оптимальних можна вирішити, наприклад, використовуючи метод цільового програмування. За такого підходу вдається побудувати одне рішення, яке є парето-оптимальним. Слід зазначити, що цільове програмування – це єдиний метод знаходження одного парето-оптимального рішення. У разі лінійного багатокритеріального завдання є сенс говорити про екстремальні ефективні (парето-оптимальні) рішення. Таких рішень кінцева кількість, що суттєво спрощує вирішення проблеми вибору

Для відбору рішення важливо використовувати методи багатокритеріальної оптимізації, щоб знайти відповідну оцінку, якщо вона включає в себе кілька критеріїв. Ці методи дозволяють ранжувати різні альтернативи за якісними критеріями.

У літературі багато методів опитмізації. Чотири з цих методів: Метод організації ранжування переваг для оцінки збагачення II (PROMETHEE II), VIseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR), Техніка порядку переваги за схожістю на ідеальне рішення (TOPSIS) та ELImination Et Choix Traduisant la Re´alite´ (ELECTRE). PROMETHEE II можна використовувати, коли необхідна повна класифікація за наявності кінцевого набору альтернатив. VIKOR вирішує проблеми прийняття рішень за критеріями однакового

пріоритету, визначаючи альтернативу, найближчу до ідеалу (дозволяє визначити рейтинги).

Методи ELECTRE актуальні, коли стикаються з ситуаціями прийняття рішень з наступними характеристиками:

1. Особа, котра приймає рішення (DM), хоче включити в модель принаймні три критерії. Однак процедури агрегування більш адаптовані в ситуаціях, коли моделі прийняття рішень включають не менше п’яти критеріїв (до дванадцяти або тринадцяти). принаймні одна з наведених нижче ситуацій має бути перевірена.

2. Дії оцінюються (принаймні за одним критерієм) за порядковою шкалою або за слабко інтервальною шкалою. Ці шкали не придатні для порівняння відмінностей. Отже, важко та/або штучно визначити кодування, яке має сенс з точки зору переваги відмінності.

3. Серед критеріїв (наприклад, тривалість, шум, відстань, безпека, культурні об'єкти, пам'ятники, існує сильна неоднорідність, пов'язана з природою оцінок. Це ускладнює агрегування всіх критеріїв в єдиному та загальному масштабі.

4. Компенсація збитку за даним критерієм виграшем за іншим може бути неприйнятною для особи, що приймає рішення.. Тому такі ситуації вимагають використання некомпенсаційних процедур агрегування.

5. Принаймні для одного критерію справедливо: невеликі відмінності в оцінках не є значущими з точки зору уподобань, тоді як накопичення кількох невеликих відмінностей може стати суттєвим. Це вимагає введення порогових значень дискримінації (байдужості та переваги), що призводить до структури переваг із всеосяжним неперехідним бінарним відношенням байдужості.

TOPSIS оцінює ефективність альтернатив за кількома критеріями порівняння. Мета застосування методів ЕЛЕКТРА - звуження паретівської множини альтернативних рішень. Робиться це наступним чином: до кожного з критеріїв (передбачається, що вони - числові) визначається «вага» за результатами опитування - число, що характеризує важливість відповідного критерію. У всіх модифікаціях методу ЕЛЕКТРА робиться спроба отримання інформації якісного

характеру щодо відносної важливості критеріїв (висловлювання типу «критерії 3 і 4 мають однакову важливість і розглядаються спільно мають більшу важливість, ніж критерій 1») і перетворення її в кількісну, числову. Проблема тут у тому, що це можна у загальному випадку безліччю способів.

Для того, щоб визначити чи перевершує альтернативний варіант $x = (x_1, x_2, \dots x_n)$ варіант $y = (y_1, y_2, \dots y)$(де x_1, y_1- значення і-того критерію, що повідомляються йому варіантами *x* і *y* відповідно), проводять такі дії:

- Множина I критеріїв розділяється на три підмножини:
  - $I^+(x, y)$ – критерії в яких *x* переважає *y*.
  - $I^=(x, y)$ – критерії в яких *x* та *y* мають однакові оцінки
  - $I^-(x, y)$ - критерії в яких *y* переважає *x*.
- Далі визначається відносна важливість кожної з цих підмножин: P_{xy}^+ , $P_{xy}^-, P_{xy}^=$
- Встановлюється також деякий поріг і вважається, що варіант *x* перевершує варіант *y* тільки в тому випадку, коли деяка функція називається індексом згоди, задовольняє умові:

  f(P_{xy}^+ , $P_{xy}^-, P_{xy}^=$) $\geq C$. Вигляд функції f визначається за своїм для кожної модифікації методу ЕЛЕКТРА. Ця умова є необхідною, але не достатньою умовою переваги x над y. У методах ЕЛЕКТРА формулюються додаткові умови, призначені враховувати не тільки порядок проходження оцінок x і y за критеріями, а й значення модулів різниць $x_i - y_i$. Ці умови, які ще називають індексом незгоди, можуть бути записані у вигляді $d_{xy} \leq d$, де d – граничне значення індексу незгоди кожної модифікації методу ЕЛЕКТРА визначається по-своєму.

У всіх модифікаціях методу ЕЛЕКТРА першому етапі з допомогою людини яка приймає рішення(DM) визначаються ваги критеріїв – позитивні дійсні числа, які тим більше, чим важливіше для ЛПР відповідний критерій. Такий підхід, звісно, має істотний недолік – неоднозначність визначення вагових коефіцієнтів.

Однак, повністю уникнути суб'єктивних оцінок у процедурі прийняття рішень неможливо, слід лише з великою ретельністю підходити визначенню ваги.

Існує чотири методи ELECTRE (англ. Elimination and Choice Translating Reality) (I, II, III та IV). Вони мають різні застосування [25]:

- ELECTRE I для відбору, але без рейтингу;
- ELECTRE II, III та IV для задач про ранжирування;
- ELECTRE II і III, коли можливо і бажано кількісно визначити відносну важливість;
- ELECTRE III включає нечіткий характер прийняття рішень;
- ELECTRE IV, коли кількісна оцінка неможлива.

Усі методи засновані на тих самих фундаментальних концепціях, як пояснюється згодом, але відрізняються як оперативно, так і за типом проблеми вирішення. Зокрема, ELECTRE I призначений для задач вибору, а ELECTRE II, III та IV для задач ранжування.

Ряд факторів вплинув на конкретний вибір методу ELECTRE III для проблеми рейтингу алгоритмів. По-друге, ELECTRE спочатку був розроблений Роєм для включати нечіткий (неточний і невизначений) характер прийняття рішень, використовуючи пороги байдужісті і перевага. Ще одна особливість ELECTRE, яка відрізняє його від багатьох критеріїв методів вирішення, полягає в тому, що воно принципово некомпенсаційне. Це означає, зокрема, що дуже погана оцінка за критерієм не може бути компенсована хорошими оцінками за іншими критеріями. А оригінальна особливість полягає в тому, що моделі ELECTRE допускають непорівнянність. Непорівнянність, яка не слід плутати з байдужістю, виникає між будь-якими альтернативами a і b, коли є немає чітких доказів на користь a чи b.

Нарешті, на вибір ELECTRE III також вплинуло успішне застосування підходу [26] [27]. Слід зазначити, що існують інші підходи до моделювання рішень, усі з різними перевагами. Наприклад, Сімпсон [28] порівняв SMART і ELECTRE і прийшов до висновку, що «між методами є очевидні відмінності, але не очевидно, що один метод сильніший за інший».

2.4 Вибір датасету

Програмам для забезпечення безпеки мережі з використанням методів ML потрібні великі набори даних для того, щоб аналізувати мережу та розрізняти звичайні та аномальні дії. За роки досліджень було проведено багато експериментів для створення наборів даних. Але все ж значна частина з датасетів залишається закритими, переважно з міркувань безпеки. Деякі дані стали доступними: DARPA 98, KDD99, UNSW-NB15, CICIDS2017 і N-BaIoT [29], але реалістичних даних зібраних для пристроїв IoT, які включають приклади DoS та DDoS досі небагато. Ще важливіше те, що в деяких наборах даних відсутній трафік, створений Інтернетом речей, тоді як інші нехтують створенням будь-яких нових ознак. У деяких випадках використовуваний тестове налаштування було нереалістичним, тоді як в інших випадках сценарії атак були недостатньо різноманітними. Наприклад, у [29] Медін створив загальнодоступний набір даних IoT під назвою N-BaIoT, і багато пізніших досліджень використовували цей набір даних для навчання та тестування своїх моделей. Хоча цей набір даних є відносно великим і чистим, він незбалансований, співвідношення звичайних даних набагато нижче порівняно з даними атаки. Мустафа намагався усунути недоліки, розробивши набір даних Bot-IoT, який ми використовували для наших експериментів. Набір даних включає реальний і змодельований мережевий трафік IoT разом із різними типами атак[24]. Атаки в ньому поділяються на три типи: атаки зондування, DoS та викрадення інформації.

2.4.1 IoT-23

Набір даних IoT-23 добре підходить для експериментів, але в ньому мало прикладів атак на відмову в обслуговуванні. IoT-23 — це новий набір даних мережевого трафіку з пристроїв Інтернету речей. Він містить всередині 20 прикладів шкідливих програм на пристроях IoT та 3 захоплення трафіку звичайних пристроїв IoT. Він був вперше опублікований у січні 2020 року, із захопленнями в діапазоні від 2018 до 2019 року. Цей мережевий трафік IoT був захоплений в лабораторії Stratosphere, в національному університеті Чеської

Республіки [30]. Мета його створення — запропонувати дослідникам великий набір даних про реальні та позначені шкідливі програми в IoT і небезпечний трафік для тренування алгоритмів машинного навчання. Цей набір даних та його дослідження фінансує Avast Software.

Набір даних IoT-23 складається з двадцяти трьох записів (так звані сценарії) різного мережевого трафіку IoT. Ці сценарії поділені на двадцять мережевих захоплень (файлів pcap) із заражених пристроїв (які матимуть назву зразка шкідливого програмного забезпечення, що виконується в кожному сценарії) і три мережеві захоплення мережевого трафіку реальних пристроїв IoT (які також мають назви пристроїв, де трафік було зафіксовано). Для кожного шкідливого сценарію творці запускали певний зразок шкідливого програмного забезпечення в Raspberry Pi, який використовував кілька протоколів і виконував різні дії. Мережевий трафік, зібраний для безпечних сценаріїв, був отриманий шляхом захоплення мережевого трафіку трьох різних пристроїв IoT: розумної світлодіодної лампи Philips HUE, домашнього інтелектуального персонального помічника Amazon Echo та розумного дверного замка Somfy. Важливо зазначити, що ці три пристрої IoT є реальним апаратним забезпеченням, а не імітованим. Це дозволило авторам фіксувати та аналізувати реальну поведінку мережі. Як шкідливі, так і небезпечні сценарії працюють у контрольованому мережевому середовищі з необмеженим підключенням до Інтернету, як і будь-який інший справжній пристрій IoT.

Мета цього набору даних — зробити два типи наборів даних доступними для спільноти: перший тип містить шкідливий мережевий трафік, а другий — лише безпечний трафік Інтернету речей. Як доброякісні, так і шкідливі потоки трафіку мають два нові стовпці для міток опису поведінки мережі. Ці мітки призначаються наступним чином:

- Оригінальний файл .pcap аналізується вручну. Підозрілі потоки виявляються і мітки призначаються на інформаційній панелі аналізу.
- Файл labels.csv створюється аналітиком. Він зберігається в папці bro кожного захоплення. Цей файл labels.csv містить набір правил, які

використовуються скриптом написаним на Python для додавання міток до кожного потоку. Файл labels.csv буде пояснено далі в цьому документі.

- Скрипт написаний на Python зчитує дані кожного потоку у файлі conn.log і порівнює ці дані з правилами, знайденими у файлі labels.csv. Сценарій порівнює кожен потік з даними у файлі csv, і якщо дані потоку відповідають критеріям маркування, додається відповідна мітка.

2.4.1.1 Опис ознак

Кожен запис у цьому наборі даних являє собою образ мережевого з'єднання і включає 23 інформаційних ознаки - індивідуальне вимірюване властивість або характеристику явища, яке спостерігається - і промаркована як «атака» або «не атака».

У досліджуваному наборі атаки поділяються на дев'ять основних категорій:

Attack: ця мітка вказує на те, що був певний тип атаки від зараженого пристрою до іншого хоста. Тут ми позначаємо як атаку на будь-який потік, який, аналізуючи його корисне навантаження та поведінку, намагається скористатися деякими вразливими послугами. Наприклад, атака грубої сили для деякого логіну через telnet, ін'єкція команди в заголовок запиту GET тощо.

C&C: ця мітка показує що заражений вузол було підключено до серверу C&C. Така поведінка була виявлена при аналізі захоплення шкідливих програм у мережі, тому що з'єднання з підозрілим сервером є періодичними, або наш заражений пристрій завантажує з нього деякі двійкові файли, або деякі декодовані або IRC-подібні запити надходять і залишають його.

DDoS: мітка показує що пристрої виконують розподілені атаки на відмову в обслуговуванні. Такі потоки трафіку виявляються як частина DDoS-атаки через кількість потоків, спрямованих на одну і ту ж IP-адресу.

FileDownload - показує, що файл завантажувався на заражений пристрій. Це можна виявити шляхом фільтрування з'єднань розміром відповіді більше 5 КБ,

зазвичай це також поєднують з відомим потенційно підозрілим портом призначення або IP, відомим як C&C сервер.

HeartBeat: відображає те, що пакети, надіслані через це з'єднання, були використані для відстеження зараженого хоста C&C сервером. Це було виявлено шляхом фільтрації з'єднань з байтами відповіді нижче 1B і з періодичними аналогічними з'єднання, зазвичай це поєднується з деяким відомим підозрілим портом призначення або IP адресою призначення, відомим як сервер C&C.

Mirai: ця мітка вказує, що з'єднання мають властивості ботнету Mirai. Ця мітка додається, коли потоки мають аналогічні шаблони як найбільш поширені відомі атаки Mirai.

Okiru: ця мітка вказує, що з'єднання мають характеристики ботнета Okiru. Це рішення про маркування було зроблено з тими ж параметрами, що і з Mirai, але з тією різницею, що ця родина ботнетів менш поширена.

PartOfAHorizontalPortScan: ця мітка вказує, що з'єднання використовуються для виконання горизонтального сканування портів для збору інформації для виконання подальших атак. Для розміщення цих міток ми покладаємося на шаблон, в якому з'єднання спільно використовують один і той же порт, однакову кількість байтів, що передаються, і кілька різних IP-адрес призначення.

Torii: ця мітка вказує, що з'єднання мають характеристики ботнету Torii. Це рішення про маркування було зроблено з тими ж параметрами, що і з Mirai, але з тією різницею, що ця родина ботнетів менш поширена.

Мітка вказує на те, що у сполуках не було виявлено жодних підозрілих шкідливих дій: Benign.

Три найбільш поширених шкідливих (недоброякісних потоків) мітки є: PartOfAHorizontalPortScan (213 852 920 потоків), Okiru (60 990 707 потоків) та DDoS (19 472 910 потоків). У той час як три найменш поширені шкідливі (недоброякісні потоки) мітки: C&C-Mirai (2 потоки), Okiru-Attack (3 потоки) та PartOfAHorizontalPortScan-Attack (5 потоків). Розподіл цільових ознак можна побачити на рис. 2.2. У вигляді таблиці (Табл. 2.1) представлено всі типи інформації про потоки які присутні в даному датасеті.

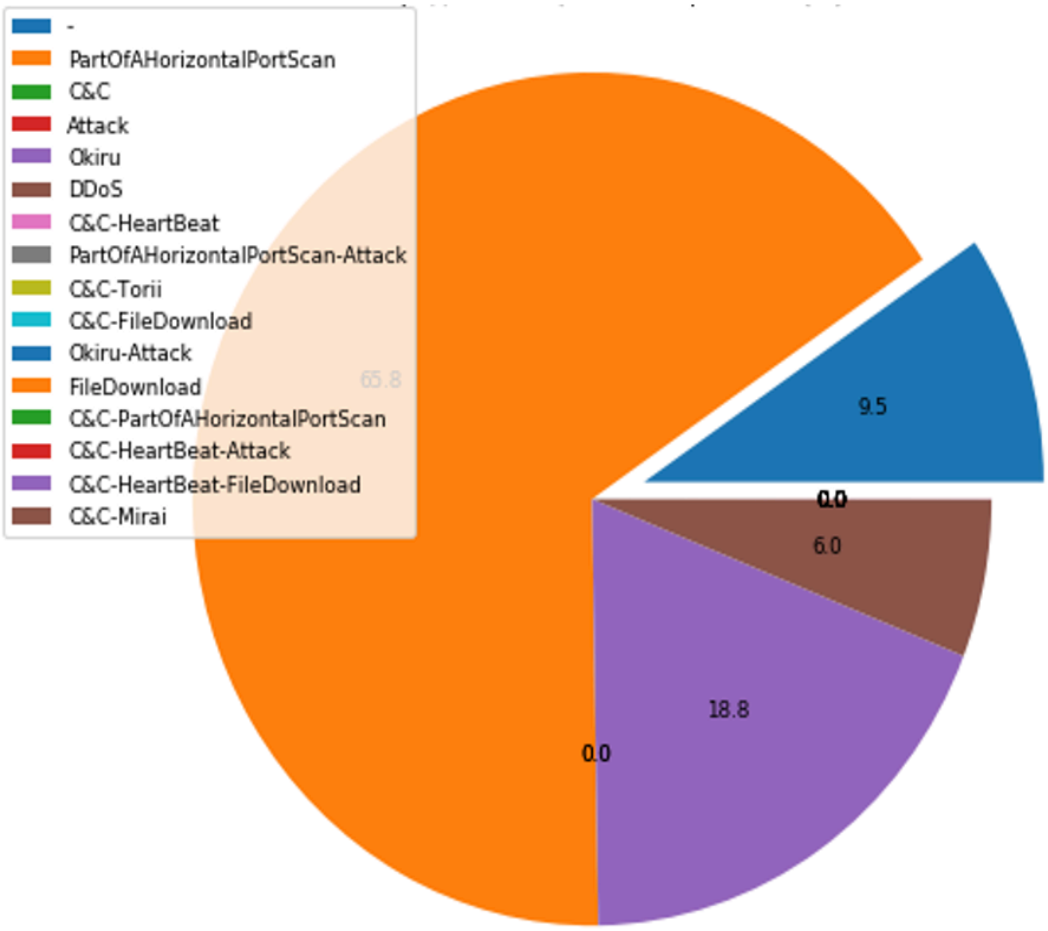

Рисунок 2.2 – Розподіл потоків в IoT-23

Таблиця 2.1 – Типи інформації в стовпчиках

Колонка	Опис	Тип даних
ts	час, коли захоплення було зроблено, виражене в Unix часі	int
uid	ідентифікатор захоплення	str
id_orig.h	IP адреса місця де сталась атака (IPv4 чи IPv6)	str
id_orig.p	порт, який використовується відповідачем	int
id_resp.h	IP-адреса пристрою, на якому відбулося захоплення	str
id_resp. p	порт, який використовується для відповіді з пристрою, де відбулося захоплення	int
proto	мережевий протокол, який використовується для пакета даних	str
service	протокол заявки	str
duration	кількість часу, протягом яких дані переміщуються між пристроєм і зловмисником	float
orig—bytes	обсяг даних, відправлених на пристрій	int
resp_bytes	обсяг даних, відправлених пристроєм	int

Продовження таблиці 2.1

Колонка	Опис	Тип даних
conn_state	стан з'єднання	str
local_orig	чи виникло з'єднання локально	bool
local_resp	чи виникла відповідь локально	bool
missed_bytes	кількість втрачених байтів у повідомленні	int
history	історія стану зв'язку	str
orig_pkts	кількість пакетів, що надсилаються на пристрій	int
orig_ip_bytes	кількість байтів, що надсилаються на пристрій	int
resp_pkts	кількість пакетів, що надсилаються з пристрою	int
resp_ip_bytes	кількість байтів, що надсилаються з пристрою	int
tunnel_parents	ідентифікатор з'єднання, якщо воно йшло через тунель	str
label	тип захоплення, доброякісний або зловмисний	str
detailed_label	якщо захоплення шкідливе, тип захоплення, як описано вище	str

2.4.2 Bot-IoT

Набір даних BoT-IoT був створений шляхом проектування реалістичного мережевого середовища в австралійській лабораторії Cyber Range Lab UNSW Canberra. Середовище включало комбінації реального та бот-трафіку. Вихідні файли цього набору даних доступні в різних форматах, включаючи оригінальні файли .pcap, згенеровані файли .argus і файли csv. Всі файли були поділені на кілька категорій та підкатегорій атак для того, щоб полегшити процес маркування. Розмір опублікованих записаних файлів .pcap становить 69,3 ГБ і містить понад 72 000 000 записів. Розмір видобутого потоку у форматі .csv становить 16,7 ГБ. Даний набір даних включає атаки DDoS, DoS, сканування сервісів, кейлогінгу і також в ньому є класифікація цих категорій на основі протоколу який використовувався при атаці. Для того щоб полегшити роботу з датасетом розробники витягли 5% вихідного набору даних за допомогою запитів MySQL. Ці вилучені 5% складаються із чотирьох файлів сумарним розміром приблизно 1,07 ГБ і включають близько 3 мільйонів записів. Щодо DDoS та DoS атак: автори використовували інструмент Hping3 увипадку використання

протоколів TCP і UDP, f. Для атак HTTP автори використовували інструмент Goldeneye.

Статистика атак, які є у наборі даних, та ПЗ яке використовувалося для їх проведення, описана в (табл. 2.2). В нашій роботі нас цікавлять атаки відмови в обслуговуванні. Такі атаки можуть бути класифікованими на основі використаної методології: об'ємні та ті що базуються на протоколах [31]. У вибраному датасеті присутні приклади DDoS і DoS атак, і використовувалися такі протоколи: TCP, UDP і HTTP.

Таблиця 2.2 – Статистика атак

Тип	Атака	Протокол	ПЗ	К-сть
Збір інформації	Сканування сервісів Збір інформації про ОС		nmap, hping3 nmap, xprobe2	1 463 364 358 275
Відмова в обслуговуванні	DDoS	TCP	hping3	19 547 603
		UDP	hping3	18 965 100
		HTTP	golden-eye	19 771
	DOS	TCP	hping3	12 315 997
		UDP	hping3	20 659 491
		HTTP	golden-eye	29 706
Витік інформації	Кейлогінг		Metasploit	1469
	Крадіжки даних		Metasploit	118
Загалом				73 360 900

У тестовій мережі автори спроєктували типову конфігурацію розумного дому. Спочатку було змодельовано п'ять розумних пристроїв, які працювали локально. Було використано Node-Red для підключення розумних пристроїв і відповідної хмарної інфраструктури для генерування нормального та зловмисного мережевого трафіку. Крім того, інструмент Ostinato використовувався для створення величезної кількості нормального трафіку між віртуальними машинами, наприклад, мережевими виробничими системами. Конфігурація віртуальних машин і використовуваних платформ являла собою реалістичну мережу розумного дому і містила п'ять пристроїв IoT (Рис 2.3):

1) розумний холодильник,
2) розумні двері гаража,
3) система моніторингу погоди,

4) розумне освітлення та

5) розумний термостат, розгорнуті в розумних будинках.

Потім згенеровані повідомлення були передані постачальнику хмарних послуг (AWS) за допомогою протоколу MQTT.

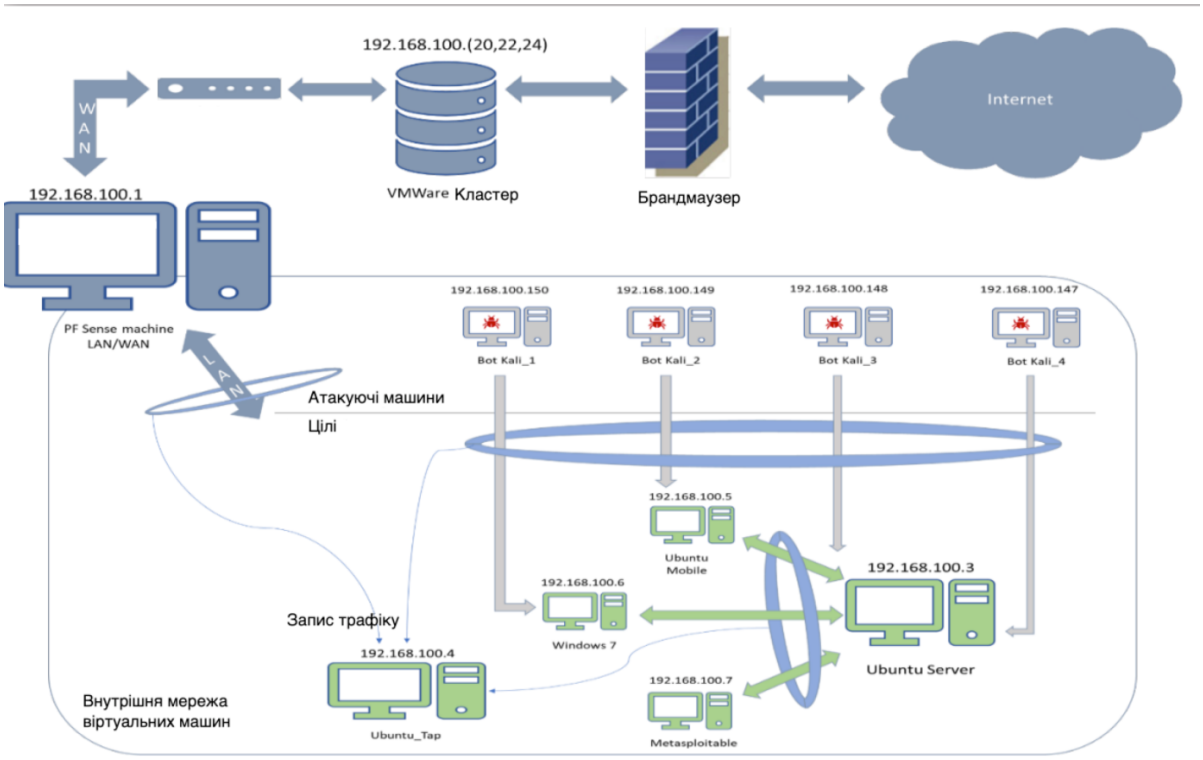

Рисунок 2.3 – Тестове оточення використане для генерування Bot-IoT

Для генерації шкідливого трафіку було використано 4 віртуальні машини під керуванням Kali Linux, які паралельно генерували різні ботнет сценарії які були представлені в таблиці 1.

Для імітації поведінки пристроїв IoT в мережі було використано інструмент Node-red. Node-red [24] — це популярне проміжне програмне забезпечення, яке використовується для з'єднання фізичних пристроїв Інтернету речей з їхнім серверним хмарним сервером і застосунками, покращуючи та прискорюючи зв'язок між різними частинами розгортання Інтернету речей. На інструменті Node-Red було розроблено код написаний на JavaScript, який імітував датчики IoT, такі як датчики температури, тиску та вологості. Протокол MQTT [24] використовувався як полегшений протокол зв'язку, який забезпечує комунікацію між машинами, створюючи підходящий вибір для рішень IoT. Він працює за

схемою опублікування/підписки, коли пристрій публікує дані брокеру MQTT (на стороні сервера) за темою, яка використовується для організації цих даних і дозволяє клієнтам підключатися до брокера та отримувати інформацію з теми на яку вони підписані із пристрій, з яким вони хочуть взаємодіяти.

Щоб генерувати величезний обсяг трафіку, був використаний інструмент Ostinato, завдяки його гнучкості генерування реалістичного доброякісного трафіку з заданими IP-адресами та портами [24]. Автори також періодично підтримували нормальні з'єднання між віртуальними машинами, виконуючи звичайні функції служб, встановлених на сервері Ubuntu, такі приклади включають сервер DNS, який розв'язував імена VMS на їхні IP-адреси та FTP-сервер, який використовується для передачі певних файлів між віртуальні машини. Щоб зібрати весь звичайний і атакуючий обсяг необроблених пакетів, якими обмінюються в налаштованій мережі, інструмент tshark був використаний на машині Ubuntu Tap, встановивши її NIC у безладний режим, це забезпечувало масштабованість тестової мережі.

Автори застосували такі сценарії IoT на в тестовій мережі:

1. Метеостанція, яка генерує інформація про тиск, вологість і температуру повітря.

2. Розумний холодильник, який вимірює температуру холодильника і за потреби регулює її нижче порогового значення.

3. Ліхтарі, активовані рухом, які вмикаються або вимикаються на основі псевдовипадкового сформованого сигналу.

4. Дистанційно активовані гаражні, які відкриваються або закривається на основі імовірнісного введення.

5. Розумний термостат, який регулює температуру в будинках шляхом запуску системи кондиціонування повітря.

2.4.3 N-BaIoT датасет

Набір даних N-BaIoT [29] найстаріший і менш збалансований за решту розглянутих наборів даних. Дані з мережі складаються з 155 ознак, зібраних за допомогою дзеркального відображення портів комутаційних пристроїв у середовищах IoT. Набір даних був згенерований із реального мережевого трафіку, включаючи дев'ять комерційних пристроїв IoT; 23 основні ознаки були виділені через різні часові інтервали (100 мс, 500 мс, 10 с, 10 хв і 1 хв). Набір даних містить дві основні атаки, а саме Mirai та BASHLITE. Пристрої були підключені до Wi-Fi за допомогою багатьох точок доступу. На комутаційних пристроях налаштовано дзеркальне відображення портів для отримання та аналізу реального мережевого трафіку. Набори даних були записані за допомогою програмного забезпечення Wireshark. Атаки BASHLITE - один із типів DDoS атаки через ботнет, були розроблені з використанням програмування на мові C для зараження систем Linux. Ця атака є найпоширенішою атакою через ботнет, яка заражає пристрої Інтернету речей, такі як камери. Для ботнет атак Mirai, виявлені Paras у 2016 році, використовують шкідливе ПЗ, запущене на процесорах ARC, для зараження великомасштабних мереж IoT. До недоліків цього набору даних можна віднести те що він найменш збалансований з представлених, в ньому дуже мало нормальних даних.

2.4.4 Вибір датасету

В даній роботі вибрано набір даних BotIoT для експериментів, серед причин: він відкритий і може бути використаний в дослідженнях, мітить захоплення трафіку саме з мережі IoT, в ньому велика різноманітність атак, включаючи великий обсяг прикладів DoS/DDoS атак з використанням різних протоколів, він включає реальний трафік, та дає можливість генерувати нові ознаки з необробленого набору даних. Його було створено у лабораторії Австралійського центру кібербезпеки (ACCS) спеціально для того щоб дослідники вивчали можливості машинного навчання у виявленнні атак в мережах IoT.

2.5 Вибір інструментів для програмної реалізації

Проекти з використанням машинного навчання відрізняються від традиційного програмного забезпечення. Головні відмінності це набори технологій, навички, необхідні для проекту та необхідні дослідження. Щоб реалізувати використання цих алгоритмів потрібно обрати мову програмування, котра буде стабільною, гнучкою та матиме доступні інструменти з зрозумілими інтерфейсами.

2.5.1 Python

Мова програмування Python була розроблена наприкінці 80-х і відіграє вирішальну роль у забезпеченні внутрішньої інфраструктури Google. Python складається з розробників-ентузіастів, і тепер його використовують у широко використовуваних програмах YouTube, Instagram, Quora та Dropbox. Python широко використовується в IT-бізнесі і дозволяє легко співпрацювати всередині груп розробників. Таким чином, якщо вам потрібна адаптивна та багатофункціональна мова програмування з підтримкою величезної мережі інженерів поряд із розширюваними пакетами ML, то Python є, мабуть, найкращим вибором.

Python пропонує стабільність гнучкість та набір доступних інструментів. В сучасному світі безліч проектів пишуться з використанням машинного навчання на цій мові, Python допомагає розробникам продуктивно працювати та бути впевненими щодо якості програмного забезпечення, яке вони створюють. Переваги, завдяки яким Python найкраще підходить для машинного навчання та проектів із використанням штучного інтелекту, включають до себе простоту і послідовність, вільний доступ до чудових бібліотек і фреймворків для AI та ML, гнучкість, незалежність від платформи та велику спільноту. Python пропонує лаконічний і читабельний код. Python дозволяє розробникам писати надійні системи, хоч за цими бібліотеками стоять складні алгоритми та універсальні робочі процеси. Розробники можуть концентруватися на вирішенні проблеми машинного навчання замість того, аби зосереджувати сили на технічних деталях. Реалізація алгоритмів AI та ML може бути складною і забирає багато часу. В

таких випадках важливо мати добре структуроване та перевірене середовище, щоб розробники могли знаходити найкращі можливі рішення при написанні коду. Для скорочення час розробки, програмісти використовують ряд фреймворків та бібліотек на Python. Python з його багатим стеком технологій має також великий підбір бібліотек які використовуться для роботи із штучним інтелектом та машинним навчанням. Ось кілька з них:

- Keras, TensorFlow і Scikit-learn для машинного навчання
- NumPy для високопродуктивних наукових обчислень та аналізу даних
- SciPy для просунутих обчислень
- Pandas для аналізу даних загального призначення
- Seaborn для візуалізації даних

Scikit-learn включає в собі реалізацію для різних алгоритмів регресії, класифікації та кластеризації, включаючи методи логістичної регресії, випадкового лісу, лінійної регресії, k-середніх та ін., та призначений для роботи з бібліотеками Python NumPy та SciPy. В цій бібліотеці також включені реалізації деяких основні метрик які потрібні для оцінки точності роботи натренованих алгоритмів.

Переваги Python:

● Мова загального призначення — Python вважається найкращим вибором, якщо проект вимагає не тільки статистики. Наприклад — розробка функціонального веб-сайту

● Плавна крива навчання — Python легко вивчити і він легко доступний, що дозволяє швидше знаходити кваліфікованих розробників.

● Велика частина важливих бібліотек — Python використовує незліченну кількість бібліотек для пошуку, збору та контролю інформації. Якщо команді розробників потрібна одна з основних функціональних можливостей R, то існують бібліотеки типу RPy2, які можуть використовувати R.

● Краща інтеграція — як правило, у будь-якому інженерному середовищі Python інтегрується краще, ніж R.

● Підвищує продуктивність — синтаксис Python надзвичайно легко читається.

Недоліки Python:

● Включає мало пакетів статистичних моделей.

● Через наявність глобального блокувальника інтерпретатора (GIL), багатопотокові програми в Python це складна і досить проблематична задача.

2.5.2 Мова програмування R

R був розроблений статистиками і в основному для статистиків, які будь-який розробник може передбачити те саме, дивлячись на його синтаксис. Оскільки мова містить математичні обчислення, пов'язані з машинним навчанням, яке є похідним від статистики, то R стає правильним вибором для тих хто хоче краще зрозуміти основні деталі та будувати інноваційно. Якщо проект багато в чому базується на статистиці то R можна вважати відмінним вибором для звуження ваших проектів, що вимагає одноразового занурення в набір даних

Переваги R:

● Підходить для аналізу — якщо аналіз даних або візуалізація є основою проекту то R можна вважати найкращим вибором, оскільки він дозволяє швидко створювати прототипи і працює з наборами даних для розробки моделей машинного навчання.

● Велика кількість корисних бібліотек та інструментів — подібно до Python, R містить кілька пакетів, які допомагають підвищити продуктивність проектів машинного навчання. Наприклад, Caret розширює можливості машинного навчання R за допомогою спеціального набору функцій, який допомагає ефективно створювати моделі для прогнозування. Розробники R отримують переваги від передових пакетів аналізу даних, які охоплюють етапи до і після моделювання, які спрямовані на конкретні завдання, такі як перевірка моделі або візуалізація даних.

● Підходить для дослідницької роботи — якщо вам потрібна дослідницька робота в статистичних моделях на початкових етапах вашого проекту, тоді R полегшить їх написання, оскільки розробникам потрібно лише додати кілька рядків коду.

Недоліки R:

● Крута крива навчання — важко заперечити, що R є складною мовою, і тому дуже рідко можна знайти експертів в цій мові.

● Непослідовний — оскільки алгоритми R надходять від третіх сторін, буває, що розробник може отримати невідповідності. Кожного разу, коли команда розробників використовує новий алгоритм, усі підключені ресурси повинні вивчати нові способи моделювання даних і прогнозування. Подібно до цього, кожен новий пакет вимагає вивчення також не вистачає детальної документації R, і це веде до негативного впливу на швидкість розробки.

2.5.3 Порівняння R та Python

Коли справа доходить до проектів машинного навчання, і R і Python мають свої переваги. Тим не менш, Python, краще підходить в маніпуляціях з даними та повторюваних завданнях. Отже, це правильний вибір, якщо планується створення продукту на основі машинного навчання, також в нього простіший синтаксис і зрозумілі інтрефейси бібліотек. Якщо ж проводити детальне статистичне дослідження то кращим вибором буде R.

Отже, після проведеного аналізу найбільш поширених мов програмування для задач ML, буде використано мову програмування Python для реалізації та тестування алгоритму виявлення DoS/DDoS атак в мережі IoT.

Висновки до другого розділу

Отже, в даному розділі ми розглянули завдання та принципи роботи системи виявлення вторгнень, формалізували задачу виявлення DoS/DDoS атак в мережі IoT. Було описано та визначено основні завдання та кроки для побудови моделі машинного навчання. Ми також провели дослідження існуючих наборів даних які можна застосувати для побудови системи виявлення DoS/DDoS атак, після проведеного аналізу було вирішено вибрати набір даних BotIoT в ролі основного датасету на якому будуть проводитися експерименти. Також було обрано мову програмування Python як основний інструмент для реалізації програмного

забезпечення яке здатне виявляти атаки направлені на відмову в обслуговуванні в мережах пристроїв Інтернету речей. Були розглянуті методи прийняття рішень в багатокритеріальних задачах, метод ELECTRE-III було обрано для оцінювання алгоритмів для виявлення DoS/DDoS атак в мережах пристроїв Інтернету речей.

3 РОЗРОБКА РІШЕННЯ

Виходячи із інформації отриманої в попередніх двох розділах було прийнято рішення проаналізувати обраний датасет задля використання його для тренування алгоритмів машинного навчання та вибору кращого з них у задачі виявлення DDoS/DoS атак.

3.1 Дані в наборі BoT-IoT

3.1.1 Ознаки у BoT-IoT

Автори датасету надають ознаки які представлено в Таблиці 3.1. Крім того, групи «saddr», «sport», «daddr», «dport», «proto» вважаються ідентифікаторами мережевого потоку, оскільки ця інформація здатна однозначно ідентифікувати потік у будь-який момент часу та допомогти в процесі маркування. Щоб позначити дані для використання в процесах машинного навчання, автори використали запити «alter table» для введення нових стовпців і запити «update» для зміни значень на основі значень saddr і daddr. У наборі даних екземпляри атаки позначаються «1», а звичайні — «0» для навчання та перевірки моделей машинного навчання за допомогою бінарної класифікації. На додаток до цього, було додатково введено атрибути категорії та підкатегорії атаки, які можна використовувати для навчання та перевірки моделей багатокласової класифікації. За допомогою інструменту CICFlowMeter [32] із необроблених даних мережевого трафіку було видобуто 10 найбільш важливих ознак. Він читає файл pcap і створює візуальний представлення вилучених функцій, а також пропонує згенерувати файли csv для набору даних. Цей процес був зроблений, перш за все, для покращення можливостей прогнозування класифікаторів. На таблиці 3.2 можна побачити 14 ознак, які були згенеровані з наведених у таблиці 3.1. Основна мета цього процесу – покращити можливості прогнозування класифікаторів. Необхідно істотно зменшити кількість ознак і використовувати лише необхідні ознаки для навчання та випробування алгоритмів, щоб знайти легке безпечне рішення, придатне для систем Інтернету речей [33].

Таблиця 3.1 – Ознаки в датасеті

Ознака	Опис
pkSeq1D	Ідентифікатор рядка
Stime	Час початку запису
flgs	Потік прапорів стану, які можна побачити в транзакціях
flgs_number	Числове представлення прапорців ознак
Proto	Текстове представлення протоколів транзакцій, присутніх у мережевому потоці
proto_number	Числове представлення ознаки proto
saddr	IP-адреса джерела
sport	Номер порту джерела
daddr	IP-адреса призначення
dport	Номер порту призначення
pkts	Загальна кількість пакетів у транзакції
bytes	Загальна кількість байтів в транзакції
state	Стан транзакції
state number	Числове представлення стану ознак
Itime	кінець запису
seq	Порядковий номер Argus
dur	Загальна тривалість запису
mean	Середня тривалість агрегованих записів
stddev	Стандартне відхилення агрегованих записів
sum	Загальна тривалість агрегованих записів
min	Мінімальна тривалість агрегованих записів
max	Максимальна тривалість агрегованих записів
spkts	Кількість пакетів від джерела до призначення
dpkts	Кількість пакетів від призначення до джерела
sbytes	Кількість байтів від джерела до призначення
dbytes	Кількість байтів від призначення до джерела
rate	Загальна кількість пакетів за секунду в транзакції
srate	Пакети від вихідного призначення в секунду
drate	Пакети від призначення до джерела в секунду
attack	підпис класу: 0 для нормального , 1 для атаки
category	Категорія трафіку
subcategory	Підкатегорія трафіку

Після чого дослідження продовжили і було вибрано 10 кращих ознак із цих за комбінованою оцінкою [24], сюди ввійшли: seq, stddev, N_IN_Conn_P_SrcIP, min, state_number, mean, N_IN_Conn_P_DstIP, drate, srate, max які і будуть використовуватися для оцінки алгоритмів.

Таблиця 3.2 – Згенеровані ознаки

	Ознака	Опис
1	TnBPSrc1P	Загальна кількість байтів на IP джерела
2	TnBPDst1P	Загальна кількість байтів на IP-адресу призначення.
3	TnP-PSrc1P	Загальна кількість пакетів на вихідний IP.
4	TnP-PDst1P	Загальна кількість пакетів на IP-адресу призначення.
5	TnP-PerProto	Загальна кількість пакетів на протокол.
6	TnP-Per-Dport	Загальна кількість пакетів на dport
7	AR-P-Proto-P-Src1P	Середня ставка за протокол на вихідний IP. (розраховується за pkts/dur)
8	AR-P-Proto-P-Dst1P	Середня кільість за протокол на IP-адресу призначення.
9	N-1N-Conn-P-Src1P	Кількість вхідних підключень на IP-адресу джерела.
10	IN-IN-Conn-P-Dst1P	Кількість вхідних підключень на IP-адресу призначення.
11	AR-P-Proto-P-Sport	Середня ставка за протокол на один sport
12	AR-PNr0toNDport	Середня ставка за протокол на один dport
13	Pkts-P-State-P-Protoc01-P-Dest1P	Кількість пакетів, згрупованих за станом потоків і протоколів на IP призначення.
14	Pkts-P-State-P-Protoc01-P-Src1P	Кількість пакетів, згрупованих за станом потоків і протоколів на IP джерела.

3.1.2 Підкатегорії атак BoT-IoT

Нижче на рисунку 3.1 зображена діаграма, побудована в ході дослідження, із кількістю пакетів для кожної підкатегорії даних, ми можемо побачити наскільки більша кількість даних які згенерувалися при проведенні атак DoS/DDoS порівняно з іншими атаками та нормальними даними. З одного боку це свідчить про незбалансованість даних в датасеті, але з іншого це відображує реальну ситуацію, адже при проведенні таких атак генерується значна кількість запитів. Для вирішення проблеми оцінювання алгоритмів за незбалансованості даних існують спеціальні методики, про які буде розказано в наступному підрозділі цієї роботи.

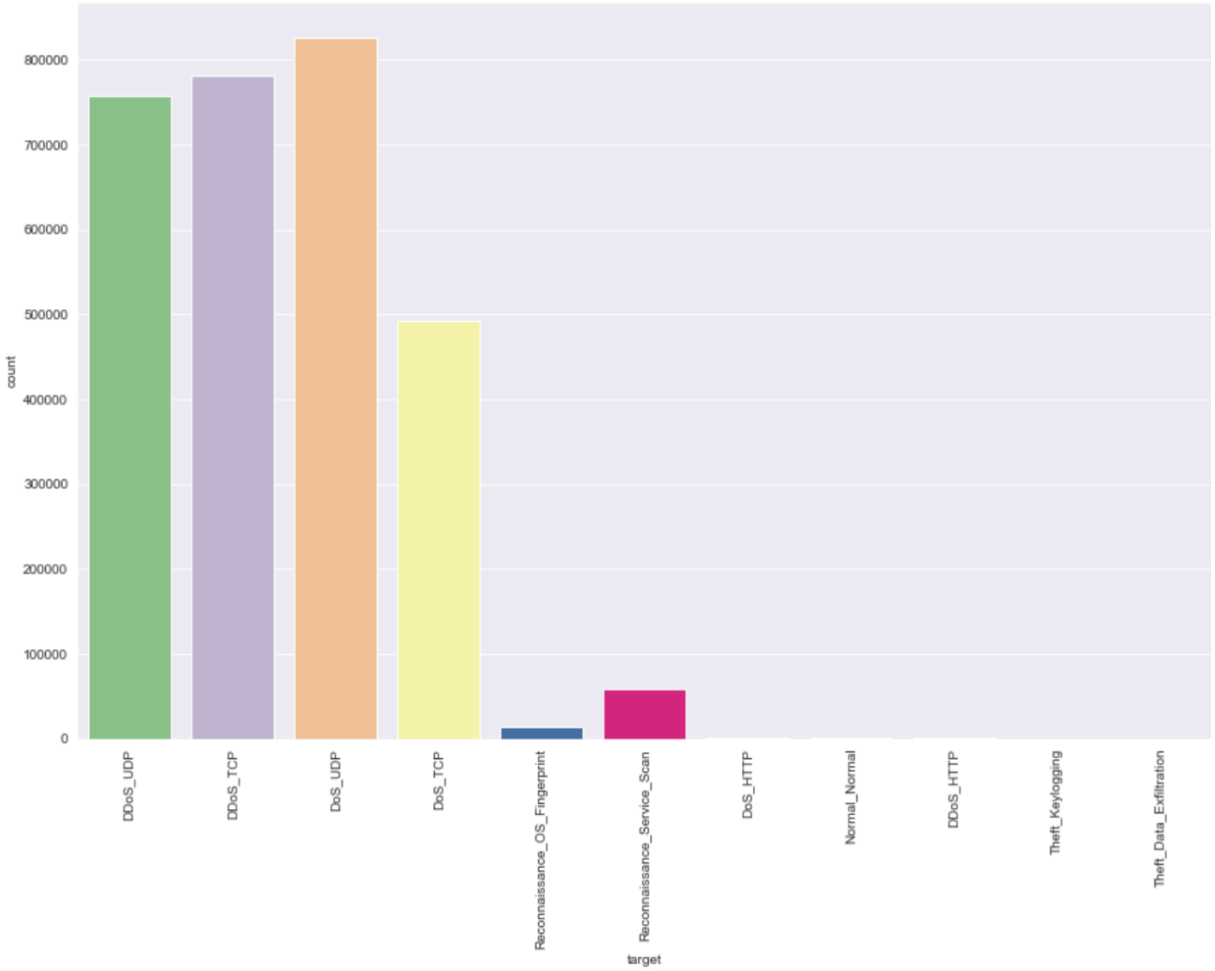

Рисунок 3.1 – Кількість пакетів по категоріям

3.2 Підготовка даних у датасеті

Особливість вибраних наборів даних полягає в тому, що дані вже розмічені фахівцями, які, власне, і генерували ці набори Однак, ми не можемо провести повноцінну дослідницьку роботу без опрацювання даних. На це є кілька причин:

1. Багато алгоритмів машинного навчання здатні працювати з числовими ознаками, тобто їм не можна передавати наші категоріальні та порядкові дані. Навіть деякі числові дані не будуть оброблені, так як вони представлені не в числових типах даних, а текстовому.
2. Існують деякі друкарські помилки у вихідних даних, які можуть сильно позначитися на точності роботи навчених моделей. Таким чином, в цільовій ознаці один і той же клас може бути записаний як з великої, так

і з малої літери. Для будь-якого алгоритму машинного навчання ці літери будуть у різних класах.

3. Дуже велика незбалансованість класів. Є класи з мільйонами записів, а є зовсім незначні, із кількістю записів менше десяти. Найбільшу проблему становлять якраз другі. Їх не можна відразу включати в навчальну вибірку, так як у такому разі будь-яка модель просто перенавчиться на цих класах. Але якщо залишити для перевірки моделі лише одне значення, то ми не зможемо правильно оцінити точність роботи моделі. При збалансованому розбитті маленьких класів на навчальну та тестову вибірку швидше за все моделі погано навчаться, і разом з цим ми не зможемо перевірити яка саме модель краще працює з цими класами через занадто малу тестову вибірку.
4. Деякі алгоритми машинного навчання працюють з усіма ознаками, як із одним вектором, а оскільки в наших даних значення ознак неоднорідні, це може призвести до серйозного зниження якості роботи навчених моделей.

Для правильної оцінки якості роботи класифікаторів та для відсутності проблеми передвиборних даних (проблема перенавчання) дані були поділені на тренувальну та тестову частини.

Моделі навчаються лише з тренувального набору даних. Навіть під час використання кросвалідації визначення оптимальних параметрів моделі, кросвалідація виконується лише з тренувальних даних.

У підсумковому наборі даних повністю відсутні елементи з тренувальної частини. Таким чином, навчена нами модель на тренувальних даних точно не знає точної відповіді на будь-який з елементів підсумкової вибірки і видаватиме передбачення спираючись на правила і закономірності вироблені в ході навчання.

У нашому дослідженні ми розділили вихідний набір даних у співвідношенні 60% на тренувальну вибірку і 40% на тестову вибірку. Це дає досить рівномірний розподіл, тому великий розмір тренувальної вибірки знижує ймовірність проблеми недонавчання, коли моделі не вистачило даних для навчання. Також

великий розмір тестової вибірки підвищує точність отриманих оцінок при масштабуванні як у загальний більший обсяг даних, і більше об'єктів кожного класу. Також було застосовано перемішування даних при розподілі за вибірками параметром random_state рівним десяти для кращого поділу кожного класу на тренувальну та тестову вибірки.

Після проведеного аналізу даних в BoT-IoT були закодовані стопчики: saddr, daddr, proto, target. Пропущені значення було замінено на NaN, ми також залишили лише стопчики які відносяться до DoS/DDoS атак. Підкатегорії атак були переведені в числовий формат:

- DDoS_HTTP – 0,
- DDoS_TCP – 1,
- DDoS_UDP – 2,
- DoS_HTTP – 3,
- DoS_TCP – 4,
- DoS_UDP – 5,
- Normal_Normal – 6

3.3 Методи машинного навчання

Для найбільш ефективного виявлення DoS/DDoS атак в мережі та роботи IDS в цілому, спочатку потрібно вибрати метод який буде задовольняти таким критеріям: – точність навчання; – час навчання. Для вирішення цієї проблеми буде використано бібліотеки машинного навчання написані для мови програмування Python. Усі експерименти були проведені із використанням бібліотек scikit-learn, Mat-plotlib, Pandas і NumPy які мають відкритий програмний код, і користуються найбільшою популярністю серед науковців.

Оптимальними методами виходячи з пов’язаних досліджень [24][33][34][35] та існуючих критеріїв було обрано наступні алгоритми:

- Метод k-найближчих сусідів (KNN);
- Наївний Баєсів класифікатор (NB);

- Випадковий ліс(Random Forest);
- Логістична регресія (Logistic Regression);
- Дерево ухвалення рішень (Decision Tree)

3.4.1 KNN

Алгоритм k-найближчих сусідів (KNN) — це метод класифікації даних для оцінки ймовірності того, що точка даних стане членом тієї чи іншої групи на основі того, до якої групи належать найближчі до неї точки даних. Це представник алгоритмів контрольованого машинного навчання, який може використовуватися для вирішення проблем класифікації та регресії. Однак він, в основному, використовується для задач класифікації. KNN — це непараметричний алгоритм із «ледачим» навчанням.

KNN часто називається алгоритмом лінивого навчання або ледачим учням, тому що він не виконує жодного навчання, коли ви надаєте навчальні дані. Натомість він просто зберігає дані під час навчання і не виконує жодних обчислень. Він не створює модель, поки не буде виконано запит до набору даних. Це робить KNN ідеальним для аналізу даних.

Він вважається непараметричним методом, оскільки він не робить жодних припущень щодо основного розподілу даних. Простіше кажучи, KNN намагається визначити, до якої групи належить точка даних, дивлячись на точки даних навколо неї.

Нехай є дві групи, A і B. Щоб визначити, чи входить точка даних у групу A чи групу B, алгоритм розглядає стан точок даних поблизу неї. Якщо більшість точок даних у групі A, то висока ймовірність, що ця точка даних знаходиться в групі A, і навпаки. Отже, KNN включає в себе класифікацію точки даних, дивлячись на найближчу анотовану точку даних, також відому як найближчий сусід.

Як зазначено вище, алгоритм KNN переважно використовується як класифікатор. На відміну від класифікації за допомогою штучних нейронних мереж, класифікація k-найближчих сусідів проста для розуміння і проста у реалізації. Це ідеально в ситуаціях, коли точки даних чітко визначені або

нелінійні. По суті, KNN виконує механізм голосування, щоб визначити клас невидимого спостереження. Це означає, що клас з більшістю голосів стане класом точки даних, про яку йдеться. Якщо значення K дорівнює одиниці, алгоритм використовує тільки найближчого сусіда для визначення класу точки даних, якщо ж значення K дорівнює 10, ми будемо використовувати десять найближчих сусідів і тд.

3.4.2 Наївний Баєсів класифікатор

Наївний Баєсів класифікатор — це простий алгоритм класифікації даних на основі теореми Байєса. Він вважається керованим алгоритмом машинного навчання, який належить до статистичного сімейства класифікаторів. Слово наївний відноситься до наївного припущення про незалежність ознак; тобто теорія припускає, що значення однієї ознаки не залежать від наявності чи властивостей інших ознак. Наївний класифікатор Байєса використовує наведене далі рівняння (теорема Байеса) для обчислення ймовірності настання конкретної події з урахуванням певного набору значень ознак:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (1)$$

Де A, B – події, P(A) – та P(B) – ймовірності A та B безвідносно одна до одної, P(A | B) – умовна ймовірність A за умови істинності B, та P(B | A) ймовірність спостереження події B, за умови істинності A.

Цей алгоритм є одним із простих та найефективніших алгоритмів класифікації, який допомагає створити швидкі моделі машинного навчання, які можуть робити швидкі прогнози. Це імовірнісний класифікатор, що означає, що він прогнозує на основі ймовірності об’єкта. Деякі популярні приклади застосування його: фільтрація спаму, сентиментальний аналіз і класифікація статей.

Переваги наївного байєсового класифікатора: Naive Bayes є одним із найшвидших і простих алгоритмів ML для прогнозування класу наборів даних. Його можна використовувати і для бінарних, так і для багатокласових

класифікацій. Він добре працює в багатокласових передбаченнях порівняно з іншими алгоритмами. Це також найбільш популярний вибір для задач класифікації тексту.

Недоліки наївного байєсового класифікатора: він припускає, що всі ознаки незалежні або не пов'язані між собою, тому він не може вивчити зв'язок між ознаками, не враховується кореляція ознак.

Існує три типи цього алгоритму, які наведені нижче:

- Гауссова модель: вона передбачає, що ознаки мають нормальний розподіл. Це означає, що якщо предиктори приймають безперервні значення замість дискретних, то модель припускає, що ці значення відбираються з гауссового розподілу.
- Мультиноміальний: мультиноміальний наївний байєсівський класифікатор використовується, коли дані розподілені багаточленами. В основному він використовується для проблем із класифікацією документів, це означає, що певний документ належить до якої категорії, наприклад, спорт, політика, освіта тощо. Класифікатор використовує частоту слів для передбачення.
- Бернуллі: Класифікатор Бернуллі працює аналогічно мультиноміальному, але передбачувачами є незалежні логічні змінні. Наприклад, якщо конкретне слово присутній чи ні в документі. Ця модель також відома завданнями класифікації документів.

3.4.3 Випадковий ліс

Випадковий ліс — простий для використання та гнучкий алгоритм машинного навчання, котрий часто показує гарні результати, без додаткового підбору гіперпараметрів. Цей алгоритм є одним з найпопулярніших через його простоту та різноманітність (цю модель можна як для задач регресії, так і для завдань класифікації). Випадковий ліс — це алгоритм навчання під наглядом. «Ліс», який він будує, являє собою ансамбль дерев рішень, зазвичай навчених методом «мішкування». Загальна ідея методу мішкування полягає в тому, що комбінація

моделей навчання підвищує загальний результат. Однією з великих переваг Random Forest є те, що його можна використати як для задач класифікації, так і для задач регресії, які формують більшість сучасних систем машинного навчання. Алгоритми випадкового лісу мають три основні гіперпараметри, які необхідно встановити перед навчанням. Вони включають розмір вузла, кількість дерев і кількість вибраних об'єктів. Звідти класифікатор випадкового лісу можна використовувати для вирішення завдань регресії або класифікації.

Алгоритм випадкового лісу складається з набору дерев рішень, і кожне дерево в ансамблі складається із вибірки даних, взятої з навчального набору із заміною, званої вибіркою завантаження. З цієї навчальної вибірки одна третина відкладається як тестові дані, відомі як вибірка поза мішком (oob), до якої ми повернемося пізніше. Інший приклад випадковості потім впроваджується за допомогою пакетування функцій, додаючи більше різноманітності до набору даних і зменшуючи кореляцію між деревами рішень. Залежно від типу проблеми, визначення прогнозу буде відрізнятися. Для завдання регресії окремі дерева рішень будуть усереднені, а для завдання класифікації — більшістю голосів, тобто. найчастіша категоріальна змінна — дасть прогнозований клас. Нарешті, вибірка oob потім використовується для перехресної перевірки, завершуючи це передбачення.

Існує ряд ключових переваг і проблем, які дає алгоритм випадкового лісу, коли використовується для задач класифікації або регресії. Деякі з них включають це:

- Знижений ризик переобладнання: дерева рішень мають ризик переобладнання, оскільки вони, як правило, щільно вписуються в усі вибірки в навчальні дані. Однак, коли у випадковому лісі є надійна кількість дерев рішень, класифікатор не буде переповнюватися моделі, оскільки усереднення некорельованих дерев знижує загальну дисперсію та помилку прогнозу.
- Забезпечує гнучкість: оскільки випадковий ліс може справлятися як із завданнями регресії, так і з класифікацією з високим ступенем точності, це популярний метод серед дослідників даних. Розміщення

функцій також робить класифікатор випадкових лісів ефективним інструментом для оцінки відсутніх значень, оскільки він зберігає точність, коли частина даних відсутня.

- Легко визначити важливість ознак: випадковий ліс дозволяє легко оцінити важливість змінної або внесок у модель. Існує кілька способів оцінити важливість функції. Важливість Джіні та середнє зменшення домішки (MDI) зазвичай використовуються для вимірювання того, наскільки зменшується точність моделі, коли дану змінну виключають. Однак важливість перестановки, також відома як точність зменшення середньої величини (MDA), є ще одним показником важливості. MDA визначає середнє зниження точності шляхом випадкової перестановки значень ознак у вибірках oob.

Ключові проблеми:

- Тривалий процес: оскільки алгоритми випадкового лісу можуть обробляти великі набори даних, вони можуть надавати більш точні прогнози, але можуть бути повільними в обробці даних, оскільки вони обчислюють дані для кожного окремого дерева рішень.
- Потребує більше ресурсів: оскільки випадкові ліси обробляють більші набори даних, їм знадобиться більше ресурсів для зберігання цих даних.
- Більш складний: передбачення одного дерева рішень легше інтерпретувати, якщо порівнювати з його лісом.

3.4.4 Логістична регресія

Логістична регресія – це один із підтипів множинної регресії, основне призначеня котрої це аналіз зв'язку між кількома незалежними змінними (регресорами) та залежною зміною. Логістична регресія – це алгоритм класифікації. Він використовується для прогнозування бінарного результату на основі набору незалежних змінних. Бінарний результат — це такий, де є лише два можливі сценарії — або подія відбудеться (1), або не відбудеться (0). Незалежні

змінні - це змінні або фактори, які можуть впливати на результат (або залежну змінну). Логістична регресія — це правильний тип аналізу, який можна використовувати, коли ви працюєте з двійковими даними. Logistic Regression розв'язує цю задачу з допомогою прогнозування безперервної змінної зі значеннями із на [0,1] при будь-яких значеннях всіх незалежних змінних. Такий результат досягається застосуванням наступного регресійного рівняння: $P = \frac{1}{1+e^{-y}}$ де P – ймовірність того, що відбудеться необхідна подія, e – основа натуральних логарифмів, y – стандартне рівняння регресії (y=$w_1x_1 + w_2x_2 + \ldots + w_mx_m$ + b). Залежність, що зв'язує ймовірність події і величину y, показана на рис. 3.2

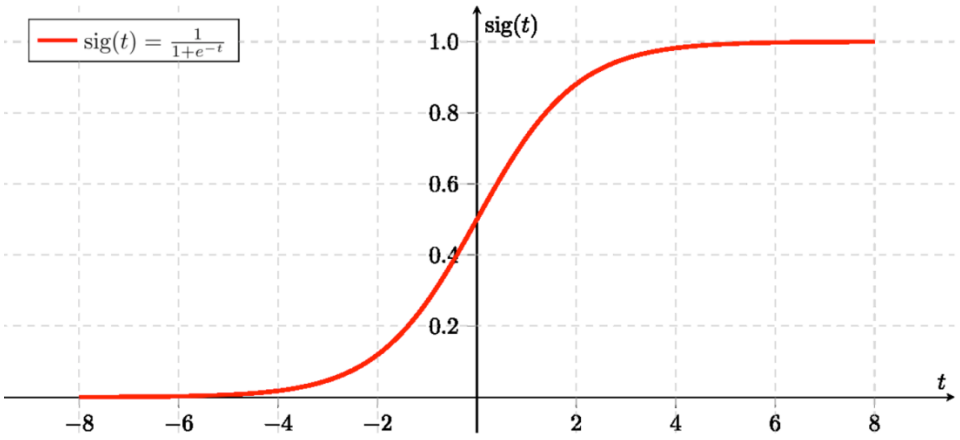

Рисунок 3.2 – Сигмоїд

Проте застосування логістичного перетворення до рівняння логістичної регресії породжує певні проблеми. При розв'язанні задачі лінійної регресії підганяють до значень, які спостерігаються, деяку гіперповерхню - пряму у разі простої регресії, площина - у разі двох незалежних змінних. Також потребується нормальність та некорельованість помилок. При переході до рівняння логістичної регресії поверхня, яка підганяється, вже не матиме такого простого вигляду.

Все це унеможливлює використання методів оцінювання, що застосовуються для лінійних завдань. Наприклад, у разі однієї незалежної змінної для простої регресії застосовувався відомий метод найменших квадратів. У разі простої логістичної регресії такий метод вже не застосовується. Непридатними є й подібні

методи на вирішення завдань із великою кількістю предикторов. Тому для вирішення завдань логістичної регресії використовується лише метод максимальної правдоподібності. Коротко, процес оцінки регресійних коефіцієнтів зводиться до максимізації ймовірності появи конкретної вибірки (при заданих значеннях, що спостерігаються). Це призводить до часто невисокого відсотка коректної класифікації. Логістична регресія також слабо стійка до зайвого припасування.

3.4.5 Дерево ухвалення рішень

Decision Tree є не-параметричним методом навчання з учителем, який використовуєть для класифікації та регресії. Мета полягає в тому, щоб створити модель, яка прогнозує значення цільової змінної, вивчаючи прості правила прийняття рішень, виведені з характеристик даних. Дерево можна розглядати як частково-постійне наближення. Деякі переваги дерев рішень:

- Простий для розуміння та інтерпретації.
- Дерева можна візуалізувати.
- Вимагає невеликої підготовки даних. Інші методи часто вимагають нормалізації даних, створення фіктивних змінних і видалення пустих значень. Однак зауважте, що цей модуль не підтримує пропущені значення.
- Складність використання дерева (тобто прогнозування даних) логарифмічна від кількості точок даних, які використовуються для навчання дерева.
- Здатне обробляти як числові, так і категорійні дані. Однак реалізація scikit-learn наразі не підтримує категоріальні змінні. Інші методи зазвичай спеціалізуються на аналізі наборів даних, які мають лише один тип змінної.
- Здатний вирішувати проблеми з кількома виходами.

- Використовується модель білої коробки. Якщо дана ситуація спостерігається в моделі, пояснення умови легко пояснюється булевою логікою. Навпаки, у моделі чорного ящика(наприклад, у штучній нейронній мережі) результати можуть бути важчими для інтерпретації.
- Можливість перевірити модель за допомогою статистичних тестів. Це дозволяє пояснити надійність моделі.

До недоліків дерев рішень можна віднести:

- Учні з дерева рішень можуть створювати надскладні дерева, які погано узагальнюють дані. Це називається переобладнанням. Щоб уникнути цієї проблеми, необхідні такі механізми, як обрізка, встановлення мінімальної кількості зразків, необхідних у вузлі листя, або встановлення максимальної глибини дерева.
- Дерева рішень можуть бути нестабільними, оскільки невеликі варіації в даних можуть призвести до створення зовсім іншого дерева. Ця проблема пом'якшується використанням дерев рішень в ансамблі.
- Прогнози дерев рішень не є ні гладкими, ні безперервними, а частково-постійними наближеннями, як показано на малюнку вище. Тому вони погано вміють екстраполювати.
- Відомо, що проблема вивчення оптимального дерева рішень є NP-повною за кількох аспектів оптимальності і навіть для простих концепцій. Отже, практичні алгоритми навчання дерева рішень базуються на евристичних алгоритмах, таких як жадібний алгоритм, де локально оптимальні рішення приймаються на кожному вузлі. Такі алгоритми не можуть гарантувати повернення глобально оптимального дерева рішень. Це можна пом'якшити шляхом навчання кількох дерев у учень ансамблю, де функції та зразки випадково відбираються із заміною.

- Існують поняття, які важко засвоїти, оскільки дерева рішень не виражають їх легко, наприклад, XOR, проблеми парності чи мультиплексора.
- Учні дерева рішень створюють упереджені дерева, якщо деякі класи домінують. Тому рекомендується збалансувати набір даних перед поєднанням з деревом рішень.
- Працює добре, навіть якщо його припущення дещо порушуються справжньою моделлю, на основі якої були згенеровані дані.

3.5 Оцінювання алгоритмів

При оцінці ефективності моделей ML, важливо визначити які показники ефективності підходять для вирішення завдання.

Метрика accuracy (точність) не дуже корисна у завданнях з нерівними класами, і це легко показати на прикладі.

Для того, щоб оцінити наші результати, ми використовували найважливіші показники ефективності: час передбачення, влучність (precision), повнота (recall), точність(accuracy) та f-міра(f-measure) як показано в рівняннях нижче [35]:

$$\text{влучність} = \frac{TP}{TP+FP} \quad (1)$$

$$\text{повнота} = \frac{TP}{TP+FN} \quad (2)$$

$$\text{точність} = \frac{TP+TN}{TP+FP+TN+FN} \quad (3)$$

$$f - \text{міра} = \frac{2}{\frac{1}{\text{повнота}} + \frac{1}{\text{влучність}}} \quad (4)$$

де TP – істинно негативне, FN – хибно негативне, TN – істинно негативне, FP – хибно позитивне. Зокрема f-міра є показовою, адже вона зводить до одного числа дві інші основоположні метрики: влучність і повноту[35].

Влучність інтерпретують як частину об'єктів, названих класифікатором правдимими і при цьому вони дійсно є правдимими, а повнота демонструє, яку частку об'єктів правдивого класу з усіх об'єктів знайшов алгоритм. Саме використання precision не дозволяє нам записувати всі об'єкти до одного класу,

тому що в такому випадку ми отримаємо ріст рівня False Positive. Отже, recall показує насікльки точно алгоритм може виявляти конкретний клас, а precision – представляє здатність точно відрізнити даний клас від інших класів.

Precision та recall не залежать від того як співвідносяться класи і тому вони гарно підходять в випадку незбалансованих наборів даних.

3.6 Інтегральні оцінки та метод ELECTRE III

В нашій задачі існувала проблема з тим що ми маємо кілька критеріїв (час та ф-міру) для оцінки алгоритмів, і при цьому ці критерії не рівнозначні, але нам потрібно було зробити вибір врахувавши обидва критерії. Також для більшої наочності результату потрібно відсортувати алгоритми за отриманими оцінками. Для цього було застосовано метод ELECTRE III, як описувалося в розділі 2, він добре підходить для вирішення подібних задач. В ході цінки критеріїв особа(и) яка приймає рішення встановлює наступні пороги:

- Поріг переваги $[p_i]$: різниця, вище якої людина, яка приймає рішення, сильно надає перевагу альтернативному рішенню над іншими для критерію i. Альтернатива b є суворою перевагою перед альтернативою a з точки зору критерію і
  якщо $g_i(\mathrm{b}) > g_i(\mathrm{a}) + p_i(g_i(\mathrm{a}))$.
- Поріг байдужості $[\mathrm{q}_i]$: це різниця, за якої особа, яка приймає рішення, є байдужою між двома альтернативами управління за критерієм і. Альтернатива b має слабку перевагу перед альтернативою a з точки зору критерію i якщо $g_i(\mathrm{b}) > g_i(\mathrm{a}) + \mathrm{q}_i(g_i(\mathrm{a}))$.
- Поріг вето $[v_i]$: блокує визначення переваги між альтернативами для критерію i. Альтернатива a не може випередити альтернативу b, якщо критерій b перевищує показник a на суму, яка перевищує поріг вето, тобто якщо $g_i(\ (\mathrm{b}) \geq g_i(\ (\mathrm{a}) + v_i(g_i(\mathrm{a}))$.

Кроки в методі ELECTRE III зображені на рис. 3.3:

Рисунок 3.3 – Кроки методу ELECTRE III

3.6.1 Обчислення матриці конкордації

Сила гіпотези про те, що альтернатива Ai принаймні така ж добра, як альтернатива Aj, вимірюється індексом узгодженості між парою альтернатив Ai і Aj і розраховується за допомогою рівняння:

$$\mathrm{C(a,b)} = \frac{1}{W}\sum_{j=1}^{n} w_j c_j(a,b) \tag{5}$$

де,

$$\mathrm{W} = \sum_{i=1}^{n} w_i \tag{6}$$

$$C_j(a,b) = \begin{cases} 1, & diff \geq -q_j \\ \dfrac{diff + P_j}{P_j - q_j}, & -q_j > diff > -P_j \\ 0, & diff \leq -P_j \end{cases} \tag{7}$$

3.6.2 Обчислення матриці розбіжностей

Індекс невідповідності вимірює силу доказів проти гіпотези та розраховується за допомогою рівняння:

$$D_j(a,b) = \begin{cases} 1, & diff \geq v_j \\ \dfrac{diff - P_j}{v_j - P_j}, & P_j \leq diff \leq q_j \\ 0, & diff \geq P_j \end{cases} \quad (8)$$

3.6.3 Обчислення матриці достовірності

Матриця достовірності вказує на надійність гіпотези випередження. Якщо індекс конкордантності вищий або дорівнює індексу невідповідності для всіх критеріїв, то ступінь достовірності дорівнює індексу відповідності. Якщо індекс конкордантністі строго нижче індексу розбіжності, то ступінь достовірності дорівнює індексу конкордантності зниженому безпосередньо в залежності від важливості цих розбіжностей. Отже:

$$S_j(a,b) = \begin{cases} C(a,b), & \text{Якщо } D_j(a,b) \leq C(a,b)\ \forall\ j \\ C(a,b) \cdot \pi_{i \in j(a,b)} \dfrac{1 - D_i(a,b)}{1 - C(a,b)}, & \text{інакше} \end{cases} \quad (9)$$

де π(a,b) — набір критеріїв, для яких $D_j(a,b) \geq C_j(a,b)$

3.6.4 Попереднє сортування за зростанням і за спаданням

Перше попереднє сортування отримують за допомогою низхідної перегонки, спочатку вибирають найкращі альтернативи, і закінчують найгіршими. Друге попереднє сортування отримують за допомогою висхідної дистиляції, вибираючи найгірший рейтинг альтернативи спочатку і закінчуючи найкращими. Два попередніх сортування, які встановлюються на основі кваліфікаційного бала для кожної альтернативи наступним чином:

Крок 1: λ_0 встановлюється рівним максимальному значенню S(a,b) у матриці достовірності (A) відповідно до рівняння:

$$\lambda_0 = \max_{a,b \in S} S(a,b) \tag{10}$$

Крок 2: Граничний рівень випередження λ_1 визначається як найбільший бал випередження, який трохи менше, ніж максимальний бал випередження мінус поріг дискримінації.

$$\lambda_1 = \max_{\{S(a,b)<(\lambda_0 - s(\lambda_0))\} \in S} S(a,b) \tag{11}$$

і:

$$S(\lambda_0) = \alpha + \beta\lambda \tag{12}$$

де s(λ_0) – поріг дискримінації на максимальному рівні випередження λ_0.

Значення α і β є зазвичай 0,3 і -0,15

Крок 3: На початковому рівні відсікання a випереджає b, якщо S(a, b) більше, ніж рівень відсікання, а S(a, b) перевищує S(b, a) на більше ніж поріг дискримінації, що задовольняє умову.

Крок 4: Кожен раз, коли a випереджає b, a отримує оцінку +1 (сила), а b отримує – 1 (слабкість). Для кожної альтернативи, сильні та слабкі сторони додаються разом, щоб отримати остаточний кваліфікаційний бал.

Крок 5: У рамках низхідної дистиляції альтернативі з найвищим кваліфікаційним балом присвоюється ранг і вилучається з процедури, і процес повторюється для всіх інших варіантів. У рамках висхідної дистиляції альтернативі з найнижчим кваліфікаційним балом присвоюється ранг і вилучається з процедури та процес повторюється для всіх інших варіантів.

3.6.5 Рейтинг альтернатив

Результати двох процедур низхідного фільтрування та висхідного об’єднуються у повний рейтинг, що відповідає результату обох процедур.

3.7 Практична реалізація

В ході роботи було написано програмний код на мові Python версії 3.9 з використанням бібліотек для машинного навчання. Був використаний набір кроків який буде описано в наступних підрозділах.

3.7.1 Підготовка даних та алгоритмів

Підготовка тестових і навчальних даних була виконана наступним чином з використанням вбудованих функцій в бібліотеку sklearn. Для підготовки алгоритмів та підбору найоптимальніших параметрів було використано реалізацію алгоритмів в бібліотеці sklearn, програмний код знаходиться в додатках.

Також для навчання ми відібрали випадкові проби даних, які є більш збалансованими (Рисунок 3.4)

```
import imblearn
from imblearn.under_sampling import RandomUnderSampler
samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 8:58592, 7:14267, 3:1179, 0:786, 6:332}
#samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 3:1179, 0:786, 6:332}
random_under= RandomUnderSampler(sampling_strategy=samp_strat,random_state=1)
X_rus,y_rus = random_under.fit_resample(x,y)
```

```
pd.Series(y_rus).value_counts()
```

```
1    70000
2    70000
5    70000
4    65000
8    58592
7    14267
3     1179
0      786
6      332
Name: target_enc, dtype: int64
```

```
from imblearn.over_sampling import RandomOverSampler
samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 8:58592, 7:30000, 3:20000, 0:15000, 6:8000}
#samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 3:20000, 0:15000, 6:8000}
random_under= RandomOverSampler(sampling_strategy=samp_strat,random_state=1)
Xres,yres = random_under.fit_resample(X_rus,y_rus)
```

```
plt.figure(figsize=(10,5))
sns.countplot(yres,palette='magma')
```

```
<AxesSubplot:xlabel='target_enc', ylabel='count'>
```

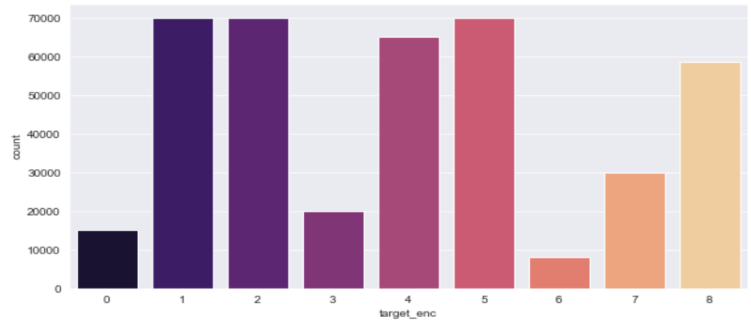

Рисунок 3.4 – Відбір проб даних по категоріях: 0 – DDoS_HTTP, 1 - DDoS_TCP, 2 - DDoS_UDP, 3 - DoS_HTTP, 4 - DoS_TCP, 5 - DoS_UDP, 6 – Normal

3.7.2 Отримання прогнозу від алгоритмів та підрахунок метрик

Було проведено запуск прогнозування для кожної натренованої моделі по 100 разів задля вимірювання витрати часу. Також дана процедура була повторена 10 разів задля усереднення метрик. Програмний код експериментів знаходиться в додатках, на рисунку 3.5 представлений приклад програмного коду та виводу програми для оцінки моделі дерева прийняття рішень.

```
We got maximum accuracy for max_depth= 6 and the maximum accuracy is:  0.9426083855457246
```

```
model_1 = DecisionTreeClassifier(max_depth=6)
model_1.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=6)
```

```
start_time = time.time()
for i in range(100):
    pred_1= model_1.predict(X_test)
print("--- %s seconds ---" % (time.time() - start_time))
score1 = model_1.score(X_test, y_test)
print("Accuracy of base model: ",score1)
print(classification_report(y_test, pred_1, target_names = ['DoS_UDP','DDoS_TCP', 'DDoS_UDP', 'Service_Scan', 'OS_Finge
```

```
--- 1.4833080768585205 seconds ---
Accuracy of base model:  0.9426206828704415
                precision    recall  f1-score   support

       DoS_UDP       0.91      0.98      0.94      6038
      DDoS_TCP       0.91      0.97      0.94     28155
      DDoS_UDP       1.00      0.98      0.99     27879
  Service_Scan       0.99      0.99      0.99      7938
OS_Fingerprint       0.99      0.99      0.99     26186
       DoS_TCP       0.98      1.00      0.99     28044
      DoS_HTTP       0.97      0.95      0.96      3147
     DDoS_HTTP       0.99      0.57      0.73     11984
 Normal_Normal       0.81      0.90      0.85     23266

      accuracy                           0.94    162637
     macro avg       0.95      0.93      0.93    162637
  weighted avg       0.95      0.94      0.94    162637
```

Рисунок 3.5 – Програмний код та виводи для оцінки моделі дерева прийняття рішень.

Всі зібрані дані для тестування алгоритмів на наборі даних BoT-IoT представлено на таблиці 3.3. Як можна побачити з таблиці, усі алгоритми показали дуже високі результати в виявленні прикладів DoS атаки через HTTP протокол, відхилення в результатах не значні, тому для виявлення цієї атаки в реальних задачах можна буде використати найшвидший варіант, проте в цій роботі ми мали на меті отримати більш універсальне рішення для різноманітних типів атак DoS та DDoS. Також виходячи з отриманих даних у випадку використання логістичної регресії, наївного Баєсового класифікатору та дерева прийняття рішень помітно нижчий відсоток виявлення нормального трафіку, що

означає підвищений ризик отримання хибних спрацьовувань якщо використовувати ці алгоритми. Також проаналізувавши час витрачений на отримання прогнозу можна виявити явного «аутсайдера» - метод KNN, який в десятки разів повільніший за випадковий ліс, який в свою чергу повільніший за решту. Для Логістичної регресії, та дерева прийняття рішень та наївного Баєсового класифікатору затрати часу на 100 прогнозів співставні та знаходяться в межах 1.5 секунд. В даному випадку без інтегральної оцінки критеріїв важко визначити який з алгоритмів краще справився із задачею, і яке співвідношення часу і точності можна вважати оптимальним.

Таблиця 3.3 – Результати алгоритмів

Критерій		KNN	Random Forest	Logistic Regression	NB	Decision Tree
час на 100 прогнозів		73,1	2,87	1,5	1,7	1,45
DoS UDP	f-міра	0,99	**1**	0,04	0,5	0,94
	повнота	0,99	1	0,66	0,96	0,98
	влучність	0,98	1	0,02	0,34	0,9
DDoS UDP	f-міра	**0,986**	0,983	0,99	0,817	0,967
	повнота	0,99	0,988	1	0,77	0,975
	влучність	0,982	0,979	0,99	0,87	0,96
DDoS TCP	f-міра	0,99	**1**	0,96	0,65	0,94
	повнота	0,98	1	0,93	0,53	0,9
	влучність	0,99	1	0,99	0,85	0,91
DoS TCP	f-міра	0,97	0,983	**0,99**	0,92	0,99
	повнота	0,98	0,988	0,99	0,84	0,99
	влучність	0,972	0,979	1	0,99	0,98
DoS HTTP	f-міра	0,98	0,994	0,985	**0,999**	0,962
	повнота	0,98	0,991	0,99	0,999	0,953
	влучність	0,98	0,998	0,98	0,999	0,971
DDoS HTTP	f-міра	0,964	**1**	0,545	0,364	0,726
	повнота	0,99	1	0,67	0,27	0,57
	влучність	0,94	1	0,46	0,56	0,998
Normal	f-міра	0,979	**0,989**	0,758	0,618	0,853
	повнота	0,96	0,99	0,702	0,46	0,901
	влучність	0,998	0,999	0,823	0,95	0,81

3.7.3 Підрахунок оцінок для алгоритмів

В даній дипломній роботі для оцінок роботи алгоритмів було використано f-міру та час який потрібен на здійснення оцінки. Для підрахунку результатів було використано ПЗ XLSSTAT [36] для MS Excel, яке надає інтерфейс користувача (рис. 3.6) для застосування методів ELECTRE II та ELECTRE III. В даному дослідженні було використано ELECTRE III , який був описаний детально в підрозділах раніше. Для цього потрібно було підготувати дані, та критерії з порогами. З таблиці 3.3 було взято без перетворень значення f-міри для виявлення кожної підкатегорії даних. Час було перетворено в безрозмірну величину на проміжку (0; 1], де за 1 було взято найкраще значення з-поміж доступних (найшвидший результат з-поміж п'яти обраних алгоритмів. Підготовані та перетворені дані оцінок для алгоритмів було зібрано в файлі xls, дані на таблиці 3.4,

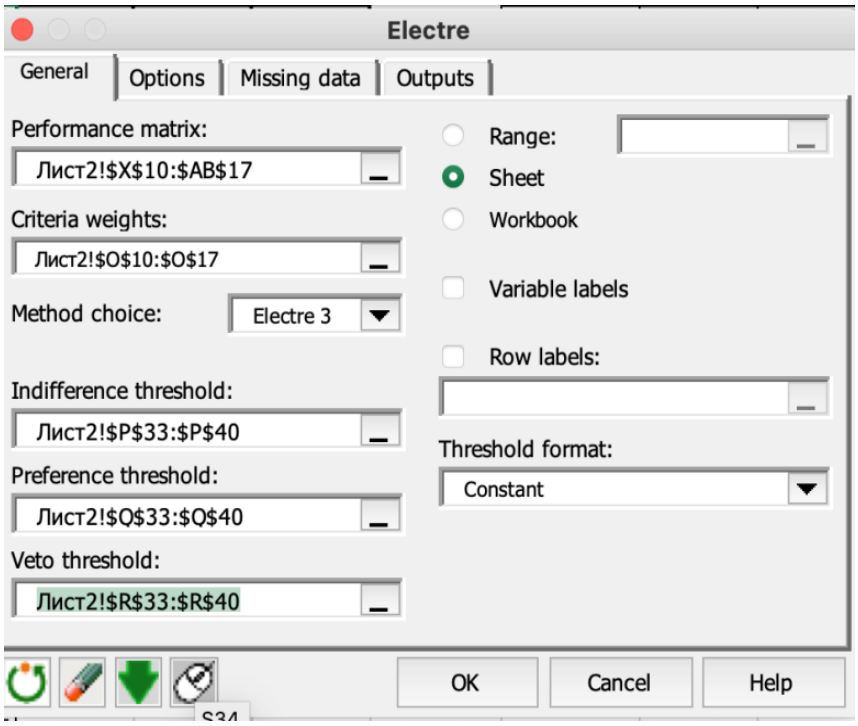

Рисунок 3.6 – Інтерфейс користувача для надбудови XLSSTAT [34]

Таблиця 3.4 – Підготовані результати для оцінки.

Критерій	KNN	Random Forest	LogisticRegression	NB	Decision Tree
Час	0,0198	0,395	0,967	0,85	1,0000
DOS UDP	0,990	**1,000**	0,040	0,500	0,94
DDoS UDP	**0,986**	0,983	0,990	0,817	0,97
DDoS TCP	0,990	**1,000**	0,960	0,650	0,94
DoS TCP	0,980	0,994	0,985	**0,999**	0,99
DoS HTTP	0,980	0,994	0,985	**0,999**	0,96
DDoS HTTP	0,964	**1,000**	0,545	0,364	0,73
Normal	0,979	**0,989**	0,758	0,618	0,85

Далі за допомогою оцінок від двох експертів було отримано інформацію про порівняльну важливість критеріїв. Обидва експерти визначили що виявлення всіх підкатегорій DoS/DDoS атак однаково важливе, а правильна робота при виявленні нормального трафіку більш важлива за окремо взяті категорії атак. Перший визначив наступні коефіцієнти: час – 0.7, виявлення нормального трафіку – 2, для DoS/DDoS - по 0,9 на кожну підкатегорію. Інший вказав наступні: час – 0.9, для виявлення нормального трафіку – 2, DoS/DDoS – по 1,1 на кожну. Усереднивши оцінки отримуємо наступні вагові коефіцієнти: час – 0.8, виявлення нормального трафіку – 2, виявлення атак на відмову в обслуговуванні – 6 (по 1 на кожен критерій).

Потім було проведено опитування задля визначення трьох порогів, переваги(p), байдужості(q) та вето(v), результати зібрані в таблиці 3.5. В результаті ПЗ згенерувало звіт в вигляді таблиць 3.6 –для матриці відповідності(Concordance), 3.7 – матриці достовірності(Credibility) та 3.8 – для матриці випередження (Outranking). У матриці випередження використано наступні позначення:

- aPb означає, що дія a є кращою перед дією b

- a NP b означає, що дія a не є кращою перед дією b
- a I b означає, що дія a байдужа до дії b.

Таблиця 3.5 – Пороги визначені експертами.

q	p	v
0,10	0,25	0,60
0,01	0,02	0,20
0,01	0,02	0,20
0,01	0,02	0,20
0,01	0,02	0,20
0,01	0,02	0,20
0,03	0,04	0,20
0,01	0,02	0,20

Таблиця 3.6 – Матриця відповідності

a/b	KNN	Random Forest	Logistic Regression	NB	Decision Tree
KNN	1,000	0,750	0,909	0,705	0,909
Random Forest	1,000	1,000	0,909	0,909	0,909
Logistic Regression	0,432	0,432	1,000	0,795	0,545
NB	0,318	0,318	0,422	1,000	0,288
Decision Tree	0,239	0,250	0,659	0,886	1,000

Таблиця 3.7 – Матриця достовірності

a/b	KNN	Random Forest	Logistic Regression	NB	Decision Tree
KNN	1,000	1,000	0,000	0,000	0,000
Random Forest	0,750	1,000	0,000	0,000	0,000
LogisticRegression	0,000	0,800	1,000	0,000	0,659
NB	0,000	0,909	0,000	1,000	0,886
Decision Tree	0,000	0,000	0,000	0,000	1,000

Таблиця 3.8 – Матриця випередження

a/b	Random Forest	Decision Tree	Logistic Regression	KNN	NB
Random Forest	I	P	P	P	P
Decision Tree	NP	I	P	P	P
Logistic Regression	NP	NP	I	P	P
KNN	NP	NP	NP	I	P
NB	NP	NP	NP	NP	I

В результаті отримано наступні місця в рейтингу для алгоритмів:

1. Random Forest – кращий вибір за 4 інші алгоритми.
2. Decision Tree – випереджає 3 інші альтернативи
3. Logistic Regression – кращий за два алгоритми.
4. KNN – кращий за один алгоритм.
5. NB – останній в рейтингу.

3.7.4 Аналіз отриманих результатів і порівняння з іншими рішеннями

Після використання методу ELECTRE III для отримання рейтингу алгоритмів та після визначення порогових значень і коефіцієнтів для нього було визначено що алгоритм випадковий ліс найкращим чином підходить для виявлення DDoS/DoS атак в мережі IoT. Алгоритм дерева прийняття рішень зайняв 2 місце, через меншу точність порівняно з першим, особливо в виявленні нормальних запитів, але швидшим виявленням. Логістична регресія показала 2-гий по швидкості результат, але надто неточно виявляла деякі підкатегорії атак, KNN займає передостаннє місце переважно через те що він в 45 разів повільнішиий ніж в конкуренти, хоча має високі показники точності. Останнє місце в наївного Баєсового класифікатора через низьку точність виявлення атак і не досить швидкі прогнози, хоч для підкатегорій DoS TCP та DoS HTTP він був найточнішим.

Порівнюючи отриманий результат для алгоритму випадковий лс з схожою роботою [35], в якій також використовували датасет BoT-IoT, маємо кращі або такі ж результати в усіх підкатегоріях крім DoS TCP (таблиця 3.9), проте варто

відмітити що у цій роботі оцінювали час роботи алгоритму іншим способом та на іншій машині та не описали результати при виявленні нормального трафіку.

Таблиця 3.9 – Порівняння результатів з [35]

Підкатегорія атак	Інша робота [35], f-міра	Ця робота, f-міра
DOS UDP	0.97	1
DDoS UDP	0.98	0,98
DDoS TCP	0,99	1
DoS TCP	1	0,99
DoS HTTP	0.95	0,99
DDoS HTTP	0.96	1
Normal	Дані відсутні	0,98

Висновки до третього розділу

Отже, в даному розділі були описані кроки реалізації алгоритму виявлення DoS/DDoS атак в мережах IoT. Було описано основні ознаки в наборі даних BoT-IoT які найкраще підходять для навчання алгоритмів машинного навчання. Було детально розглянуто 5 методів машинного навчання, які були використані в дослідженні. Для обраних методів було зібрано дані про точність виявлення окремих підкатегорій атак та час на прогнозування, з чого було отримано інтегральну оцінку для алгоритмів за допомогою методу прийняття рішень для багатокритеріальних задач – ELECTRE III. Як результат було визначено що алгоритм випадкового лісу найкраще справився з задачею і може бути використаний для побудови системи виявлення вторгнень. Результати моделі було порівняно з іншою роботою і зроблено висновок що ефективність виявлення підвищилася, а вибір алгоритму має більшу обгрунтованість.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї стартап-проекту

На початку розробки проекту відбувається аналіз і подання змісту ідеї стартапу, можливі напрямки та вигоди, якими може бути зацікавлений потенційний користувач. Головна ідея, напрямки застосування, вигода для користувача зібрані у Таблиці 4.1. Відмінності від основних існуючих аналогів та замінників у Таблиці 4.2

Таблиця 4.1 – Опис ідеї, напряму та вигоди стартап-проекту

Зміст ідеї	Напрям застосування	Вигоди для користувачів
Застосування, тренованого на спеціальних наборах даних, методу машинного навчання для виявлення DoS/DDoS атак в мережах пристроїв Інтернету речей, основою цього проекту повинен стати проект розроблений в межах минулого розділу	Компанії та підприємства які використовують IoT	Захист мережі IoT від кібератак на відмову в обслуговуванні
	Фізичні особи які використовують мережі IoT	Швидкість та точність механізму виявлення
		Пом’ягшення наслідків проведених DoS/DDoS атак
		Легкість в розгортанні системи

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W (слабка сторона)	N(нейтральна сторона)	S(сильна сторона)
		Мій проект	Конкурент			
1.	Покриття питань захисту пристроїв IoT	Виявлення DDoS/DoS атак	Не спеціалізується на мережах IoT.	Не покриття інших сценаріїв атаки.	Виявлення різних типів DDoS/DoS атак.	Вибір найсучасніших даних в процесі тренування.
2.	Швидкодія пропонованого методу	Використання алгоритму найкращого за характеристикою швидкодія/точність	Використання внутрішніх методів вибору алгоритму виявлення атак.	Існують швидші методи для різних підкатегоій атак	Немає	Розробка з врахуванням балансу швидкості та якості прогнозів
3.	Точність пропонованого методу	Оцінка здійснена з врахуванням незбалансованості даних в навчальному наборі. >98%	Оцінка здійснена з врахуванням незбалансованості даних в навчальному наборі >98%	Не оцінювалась точність на прикладі не DoS атак	Немає	Враховано точність виявлення нормальних даних для запобігання хибних сповіщень

4.2 Технологічний аудит ідеї проекту

У цьому підрозділі було здійснено аудит технологій для реалізації ідеї, у таблиці 4.3 зібрані результати аналізу можливості здійснення ідеї проекту.

Таблиця 4.3 – Технологічні можливості здійснення ідеї

Ідея	Наявність технологій	Технології та реалізації	Доступність технологій
Зчитування мережевого трафіку	В системі можливо переадресовувати трафік на IDS	Wireshark, HTTP/gRPC запити	Програмні засоби та протоколи відкриті.
Аналіз трафіку IoT	В системи є можливість аналізу за допомогою моделі ML	Бібліотеки scikit-learn, pandas, numpy, tensor-flow	Бібліотеки надають інтерфейс до алгоритмамів. Ліцензія BSD/Apache-2.0
Обробка трафіку	Необхідна розробка	Необхідно обробляти дані перед аналізом	Необхідна розробка

4.3 Аналіз можливостей запуску стартап-проекту на ринок

В даному підрозділі було проведено аналіз попиту: його наявність, динаміка розвитку ринку, обсяг попиту. Визначено ринкові можливості, які можна використати під час впровадження проекту, а також ринкових загроз, які можуть перешкодити реалізації проекту. Результати зібрано в таблиці 4.4

Таблиця 4.4 – Характеристика потенційного ринку для проекту

№ п/п	Показники стану ринку	Характеристика
1 .	Кількість основних гравців	Не знайдено
2.	Загальний обсяг продаж грн/ум.од	15млн ум.од
3.	Динаміка ринку	Зростає
4.	Наявність обмежень для виходу	Немає
5.	Специфічні умови для сертифікацій чи дотримання стандартів	Немає
6.	Середня норма рентабельності на ринку, %	>= 120%

З результатів дослідження випливає що ринок відкритий та зараз зростає, не було знайдено продуктизованих конкурентів, через що ринок можна вважати інвестиційно привабливим.

Інформацію про потенційних клієнтів та групи клієнтів було зібрано у Таблиці 4.5. Після визначення груп потенційних клієнтів проведено аналіз ринкового середовища: було складено таблиці факторів, які сприяють ринковому впровадженню проекту, а також тих факторів, які йому перешкоджають (таблиця 4.6-4.7).

Таблиця 4.5 – Потенційні клієнти продукту

№ п/п	Потреба, яка формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Відсутність прикладних методів аналізу трафіку для виявлення атак в мережах IoT	1. Підприємства які займаються захистом від кібератак. 2. Підприємства які зазнали DoS/DDoS атак на мережу пристроїв IoT. 3. Фізичні особи які використовують мережі пристроїв IoT.	Перша група займається поширенням послуг кібербезпеки і хочуть отримати грошову вигоду. Друга хоче уникнути можливих втрат від повторення атак. Третя група це люди які хочуть вберегти себе від можливих атак на мережу IoT.	Своєчасне та точне виявлення атак DoS/DDoS

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція
1.	Високий рівень хибних прогнозів	Високий рівень не правильних прогнозів через певні відмінності в окремих мережах	Збір додаткових даних для тренування і покращення моделі в конкретних мережах
2.	Відсутність попиту	Використання звичних інструментів IDS, замість спеціалізованих на IoT, або недооцінка можливих наслідків від атак на пристрої IoT	Рекламна компанія про важливість захисту пристроїв Інтеренету речей, а також про важливість спеціалізації на конкретній проблемі.

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція
1.	Відсутність альтернатив	Існуючі системи не задовольняють набором своїх можливостей для конкретних проблем в IoT	Розширення спектру можливостей

Таблиця 4.8 – Ступеневий аналіз ринкової конкуренції

Особливість конкурентного середовища	У чому саме проявляється особливість	Можливі дії компанї щоб залишатися конкурентноспроможними
1. Вказати тип конкуренції - олігополія	Немає повноцінних конкурентів з спеціалізацією на мережах IoT	Можливість завоювати частку ринку за рахунок вузької спеціалізації
2. За рівнем конкурентної боротьби -національна	Надавання послуг на території України і за її межами	Вдосконалення підходів до аналізу, розширення набору послуг
3. За галуззю - міжгалузева	Можливість особистого використання та в компаніях різного напрямку діяльності	Пошук нових клієнтів серед населення та промислових підприємств
4. За видами товарів - товарно-видова	Ця послуга задовольняє потреби споживачів, є унікальною і відмінною з конкурентами.	Надання послуг у сфері кібербезпеки
5. За характером конкурентних переваг - цінова	Головним фактором для клієнта є забезпечення кібербезпеки своєї мережі або зменшення ризиків та наслідків атак	Налаштування системи під конкретного клієнта, розширення набору послуг що надаються
6. За інтенсивностю -марочна	Привабливість самої послуги	Покращення методів виявлення

Таблиця 4.9 - Аналіз галузевої конкуренції за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти в галузі	Постачальники	Клієнти	Товари замінники
	Немає	Патенти на програмні продукти	IT компанії та дослідники які займаються кібербезпекою	Компанії та особи які використовують мережі IoT	Відсутні
Висновки	Інтенсивність боротьби за вихід на ринок не висока	Можливість виходу на ринок висока	Не диктують умови роботи ринку	Зацікавлені в ефективності роботи IDS	Обмежень на ринку через товари-замінники немає

З огляду на ситуацію на ринку цей проект має можливості виходу та розвитку на ринку. Сильними сторонами є унікальність методу аналізу та вузька спеціалізація на IoT, а також відсутність потенційних конкурентів. Стрімке зростання та розповсюдження IoT мереж грає значну роль у якості сприяння розширення множини потенційних клієнтів та подальшого розвитку проекту. Так як прямих конкурентів не було виявлено, то порівняння було проведено із подібними послугами в цій галузі. В Таблиці 4.10 обґрунтовано фактори конкурентоспроможності продукту, а в Таблиці 4.11 – проведено порівняльний аналіз сильних і слабких сторін.

Таблиця 4.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1.	Якість продукту	Основний алгоритм був протестований і показує відмінні результати по виявленню атак
2.	Наявність прямих конкуретів	Аналогічних продуктів з спеціалізацією на IoT немає в вільному продажі.
3.	Масштабованість продукту	Продукт потребує масштабованості

Таблиця 4.11 – Порівняльний аналіз сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим методом						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Відсутність прямих конкурентів на ринку надання послуг	15							+
2	Гнучкість методу	18					+		
3	Застосування для компаній із різних галузей	20							+
4	Ціна послуги	15					+		

Заключним етапом ринкового аналізу можливостей впровадження стартап-проекту став SWOT – аналіз, на основі виявленних ринкових загроз та можливостей, а також слабких та сильних сторін. SWOT аналіз складається з виділення сильних і слабких сторін, а також можливостей і загроз. Результати зібрано у таблиці 4.12.

Таблиця 4.12 – Порівняльний аналіз сторін проекту

Сильні сторони: унікальність моделі аналізу трафіку, ціна послуги, відсутність прямих конкурентів на ринку, застосування у компаніях із різних галузей та в особистому використанні.	Слабкі сторони: залежність від маркетингу, надто вузька спеціалізація
Можливості: можливості виходу на міжнародний ринок, розвиток суміжних послуг, розширення кількості клієнтів за рахунок фізичних осіб, створення окремого незалежного продукту для аналізу мережевого трафіку в мережах пристроїв IoT	Загрози: вихід на ринок прямих конкурентів, інфляція, поява нових способів проведення DDoS/DoS атак

На основі проведеного SWOT-аналізу розроблено альтернативи ринкової поведінки (перелік можливих заходів) для виведення проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації, дані зібрано у Таблиці 4.13

Таблиця 4.13 – Альтернативи для виходу на ринок

№ п/п	Альтернатива – орієнтовний перелік заходів поведінки	Ймовірність отримання ресурсів	Строки реалізації
	Укладання угоди з IT компанією	Висока	до 12 місяців
	Впровадження рекламних засобів	Середня	до 4 місяців
	Участь в IT конференціях, публікування в журналах	Середня	до 2 місяців
	Оформлення патенту	Висока	до 3 місяців
	Додання нового функціоналу	Мала	до 8 місяців

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії для стартапу, перш за все, передбачає окреслення стратегії охоплення ринку: опис цільових груп потенційних споживачів, це було зроблено у Таблиці 4.14.

Таблиця 4.14 – опис цільових груп клієнтів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах груп	Інтенсив-ність конкуренції в секторі	Простота входу
1.	IT компанії які займаються кібербезпекою	Клієнти потребують такого продукту	Середній	Низька /немає	Немає, проте ринок не сформований
2.	Фізичні особи та компанії які використовують мережі IoT	Клієнти потребують такого продукту	Середній	Низька /немає	Немає складності

Таблиця 4.15 - Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Розвиток завдяки висвітленню особливостей проекту через інтенсивну рекламну кампанію	Рекламна кампанія	Великі перспективи та визначні якості наявного рішення	Стратегія спеціалізацій

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

Чи є проект «першопроходьцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Шукати нових серед населення, та забирати існуючих	Частково, виявлення атак	Стратегія лідера

Таблиця 4.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Точність та швидкодія виявлення атак DoS/DDoS	Стратегія спеціалізації	Виявлення атак за допомогою моделей ML	Запобігання наслідків ботнет атак. Виявлення атак DoS/DDoS в IoT. Швидкість

4.5 Розроблення маркетингової програми проекту

В цьому підрозділі за допомогою даних у таблицях була описано розробка а також формування концепції кінцевого продукту (товару), яка була сформована на основі проаналізованих аспектів конкурентоспроможності цього товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№	Потреби	Вигоди які пропонує товар	Ключові переваги над конкуурентами
1.	Виявлення DDoS/DoS атаки в мережі пристроїв IoT	Точне виявлення	Баланс точності та швидкості прогнозу через використання навченого на IoT трафіку RandomForest
2.	Сповіщення користувачів про можливу атаку	Негайна реакція на знайдену атаку	Швидкодія і точність в IoT мережах

Таблиця 4.19 - Опис 3 рівнів моделі товару

Рівні	Сутність та складові		
Товар за задумом	Система з простими інтерфейсами для аналізу трафіку і виявлення атак в мережі.		
Товар у реальному виконанні	Властивості	М/Нм	Вр/Тх /Тл/Е/Ор
	Отримання трафіку з мережі	М	Тх, Тл, Е
	Підготовка трафіку за допомогою виділення ознак	М	Тх
	Виявлення аномалій за допомогою алгоритмів машинного навчання	М	Тх, Е
	Якість: поділ вибірки на тренувальну та тестову, апробація результатів за допомогою експертних оцінок та інтегральної оцінки методом ELECTRE III. Марка: “Компанія Б.”, консультаційні послуги		
Товар із підкріпленням	До продажу: початок рекламної кампанії		
	Після продажу: продовження рекламної компанії		
За рахунок чого потенційний товар буде захищено від копіювання: патент			

Таблиця 4.20 - Визначення меж встановлення ціни

Рівень цін на товари - замінники	Рівень цін на товари - аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі
12000 у. о.	14500 у. о.	Високий	9000-16000 у.о.

Таблиця 4.21 - Формування системи збуту

Специфіка закупівельної поведінки цільових груп	Функції збуту, які має виконувати постачальник	Глибина каналів збуту	Оптимальнна система збуту
Власна система збуту	Усні та письмові консультації	Продаж користувачем, нульовий рівень	Прямий збут без посередників

Таблиця 4.22 - Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комінукацій якими користуються клієнти	Ключові пропозиції для позиціонування	Завдання рекламного повідомлення
Консервативна поведінка	Корпоративна пошта та телефонні дзвінки	Гнучке налаштування, спеціалізація на IoT, післяпродажна підтримка	Висвітлення унікальних характеристик послуги

Висновки до розділу 4

В четвертому розділі дипломної роботи було здійснено аналіз перспектив для запуску стартап-проекту, який ставить собі за ціль надання послуг в сфері виявлення атак на відмову в обслуговуванні в мережах пристрої Інтернету речей. Після отримання результатів аналізу було показано, що запуск такого проєкту є можливим, хоч і матиме певні труднощами. Для даного проекту спостерігається попит який зростає через активний розвиток IoT. Не висока конкуренція дає шанси виходу на ринок.

ВИСНОВКИ

Дана дипломна робота була спрямована на дослідження можливості використання методів машинного навчання для побудови системи виявлення DoS/DDoS атак в мережах пристроїв Інтернету речей. Для цього були проаналізовані існуючі роботи на схожу тематику. Було виявлено недостатню увагу в літературі до проблем кібербезпеки в мережах IoT, а також не досконалий підхід для оцінки успішності алгоритмів виявлення аномалій. Результати аналізу існуючих досліджень були представлені в межах першого розділу.

В межах другого розділу було проведено пошук найкращого набору даних який підходить для навчання і тестування алгоритмів, ним став BotIoT. Також було вирішено використовувати метод прийняття рішень для багатокритеріальних задач – ELECTRE III для оцінювання успішності виявлення атак алгоритмами.

В результаті в ході роботи над третім розділом був імплементований проект в якому для навчання і тестування використовувався набір даних - BotIoT. На ньому було проведено навчання п’яти різних алгоритмів машинного навчання, і більшість з них досягла високої точності виявлення атак, що довело їхню ефективність. Для обраних алгоритмів було проведено збір інформації про час прогнозування та точність прогнозів в умовах незбалансованих даних. Для цих даних було зібрано експертні критерії для оцінки та пороги для прийняття рішень, вибір найкращих алгоритмів був здійснений з використанням, який вирішує задачу сортування вибраних алгоритми по отриманій інтегральній оцінці, це дозволяє мати більш універсальний та точний підхід для визначення найкращого рішення аніж в існуючих дослідженнях. Після застосування методу було показано що для вирішення задачі виявлення DoS/DDoS атак в мережах IoT найкраще з-поміж решти підійшов алгоритм Random Forest.

Також результати даного дослідження можуть служити основою для створення більш надійного алгоритму виявлення який буде об’єднувати різні алгоритми ML як багатошарову модель для покращення ефективності виявлення.

В четвертому розділі було здійснено дослідження можливості запуску стартап проекту із використанням запропонованої ідеї. Було проведено аналіз за різними показниками. На основі аналізу було зроблено висновок про доцільність подальшої розробки цього проекту щоб створити продукт у вигляді системи виявлення вторгнень яка спеціалізується на мережах пристроїв Інтернету речей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. N. Moustafa, B. Turnbull, K.-K.R. Choo, Towards automation of vulnerability and exploitation identification in iiot networks, in: 2018 IEEE International Conference on Industrial Internet.
2. N. Moustafa, B. Turnbull, K.-K.R. Choo, An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of IoT, IEEE Internet Things J. (2018).
3. M. Tayyab, B. Belaton, and M. Anbar, ‘‘ICMPv6-based DoS and DDoS attacks detection using machine learning techniques, open challenges, and blockchain applicability: A review,’’ IEEE Access, vol. 8, pp. 170529–170547, 2020.
4. Ahmed, Sheikh. (2021). A Study of ML Algorithms for DDoS Detection. International Journal for Research in Applied Science and Engineering Technology.
5. Das, Saikat & Mahfouz, Ahmed & Venugopal, Deepak & Shiva, Sajjan. (2019). DDoS Intrusion Detection through ML Ensemble.
6. K. Ashton, "That 'Internet of Things'," 22 червня 2009. [Online]. URL: http://www.rfidjournal.com/articles/view?4986
7. L. Dignan, “Iot devices to generate 79.4zb of data in 2025, says idc”, ZDnet, 2019, URL: https://www.zdnet.com/article/iot-devices-to-generate-79-4zb-of-data-in-2025-says-idc.
8. Kasinathan P. Denial of service detection in 6LoWPAN based internet of things [Текст]. / P. Kasinathan, C. Pastrone, M.A. Spirito and M. Vinkovits // 9-та міжнародна конференція безпровідного та мобільного обчислення – 2013. – pp. 600-607.
9. M. A. Simplício, M. V. M. Silva, R. C. A. Alves, and T. K. C. Shibata, “Lightweight and escrow-less authenticated key agreement for the internet of things”, Comput. Commun., vol. 98, pp. 43–51, 2017.
10. J. Czyz, M. J. Luckie, M. Allman, and M. Bailey, “Don’t forget to lock the back door! a characterization of ipv6 network security policy”, in NDSS, 2016.

11. K. Angrishi, “Turning internet of things(iot) into internet of vulnerabilities (iov) : Iot botnets”, CoRR, vol. abs/1702.03681, 2017. arXiv: 1702.03681. Доступ: http://arxiv.org/abs/1702.03681.
12. A. Chapman, Hacking into internet connected light bulbs, URL: https://www.contextis.com/en/blog/hacking-into-internet-connected-light-bulbs, 2014.
13. B. Rodrigues, Arris cable modem has a backdoor in the backdoor, URL: https://w00tsec.blogspot.com/2015/11/arris-cable-modem-has-backdoor-in.html, 2015.
14. Jmaxxz, Backdooring the frontdoor, DEF CON, https://doi.org/10.5446/36251 Lastaccessed : 10 липня 2020, 2016
15. E. Fernandes, J. Jung, and A. Prakash, “Security analysis of emerging smart home applications”, 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 636– 654
16. A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, “A large-scale analysis of the security of embedded firmwares”, 2014, ст. 95–110.
17. X. Lei, G. Tu, A. X. Liu, C. Li, and T. Xie, “The insecurity of home digital voice assistants - vulnerabilities, attacks and countermeasures”, 2018 IEEE Conference on Communications and Network Security (CNS), 2018, ст. 1–9. DOI: 10.1109/CNS. 2018.8433167
18. R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, “Systematic classification of side-channel attacks: A case study for mobile devices”, IEEE Communications Surveys Tutorials, vol. 20, no. 1, pp. 465–488, 2018.
19. A. A. Pammu, K. Chong, W. Ho, and B. Gwee, “Interceptive side channel attack on aes-128 wireless communications for iot applications”, в 2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2016, pp. 650–653.
20. D. Genkin, A. Shamir, and E. Tromer, “Rsa key extraction via low-bandwidth acoustic cryptanalysis”, in Advances in Cryptology – CRYPTO 2014, J. A. Garay and R. Gennaro, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 444–461, ISBN: 978-3-662-44371-2.

21. What Is an ACK Flood DDoS Attack? (2020). What Is an ACK Flood DDoS Attack? | Types of DDoS Attacks. Cloudflare. URL: https://www.cloudflare.com/ learning/ddos/whatisanackflood/

22. Vlajic, N. and Zhou, D. (July 2018). “IoT as a Land of Opportunity for DDoS Hackers”. In: Computer 51.7, pp. 26–34. ISSN: 15580814. DOI: 10.1109/MC.2018. 3011046.

23. Antonakakis, M. et al. (2017). “Understanding the Mirai Botnet”. In: p. 18

24. K. Nickolaos, N. Moustafa, E. Sitnikova, and B. Turnbull. "Towards the development of realistic botnet dataset in the IoT for network forensic analytics: Bot-iot dataset." Future Generation Comp. Systems 100 (2019).

25. Opricovic, S. and Tzeng, G.-H. (2006). Extended vikor method in comparison with outranking methods. *Eu- ropean Journal of Operational Research*, 178(2):514– 529.

26. Roy, B. (1968), Classement et choix en présence de critères multiples (la méthode ELECTRE), RIRO, 8, 57-75.

27. Hokkanen, J. and P. Salminen (1997), ELECTRE III and IV decision aids in an environmental problem, Journal of Multi-criteria Decision Analysis, 6, 215-226.

28. Simpson, L. (1996), Do decision makers know what they prefer?: MAVT and ELECTRE II, Journal of the Operational Research Society, 47, 919-929.

29. Al-Garadi, M.A.; Mohamed, A.; Al-Ali, A.; Du, X.; Ali, I.; Guizani, M. A Survey of machine and deep learning methods for internet of things (IoT) security. arXiv 2018, arXiv:1807.11023. [CrossRef]

30. Sebastian Garcia, Agustin Parmisano, & Maria Jose Erquiaga. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Набір даних]. Doi: http://doi.org/10.5281/zenodo.4743746”

31. S.T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against ddos flooding attacks, IEEE commun. Surv. Tutor. 15 (4) (2013)

32. Mehryar Mohri, Afshin Rostamizadeh, & Ameet Talwalkar. (2018). Foundations of Machine Learning (2nd. ed.). The MIT Press.

33. Alsemiri, Jadel & Alsubhi, Khalid. (2019). Internet of Things Cyber Attacks Detection using ML. International Journal of Advanced Comp. Science and Applications.

34. Соловей Б., Гальчинський Л. (2021). ВИЯВЛЕННЯ DOS/DDOS АТАК В ІОТ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ. *Матеріали конференцій Молодіжної наукової ліги*. вилучено із https://ojs.ukrlogos.in.ua/index.php/liga/article/view/15495

35. Jadel Alsamiri and Khalid Alsubhi, "Internet of Things Cyber Attacks Detection using Machine Learning" International Journal of Advanced Computer Science and Applications(IJACSA), 10(12), 2019. DOI: http://dx.doi.org/10.14569/IJACSA.2019.0101280

36. XLSTAT (2007) Програмне забезпечення для статистики в MS Excel. URL: https://www.xlstat.com

ДОДАТКИ

ДОДАТОК А

Програмний код

```
train = pd.read_csv('training.csv')
test = pd.read_csv('testing.csv')
train.dtypes[train.dtypes=='object']
train['category'].value_counts()
train.groupby(['category','subcategory']).count()
train['target'] = train['category'] + "_" + train['subcategory']
train.head()
plt.figure(figsize=(15,10))
plt.xticks(rotation=90)
sns.countplot(train['target'],palette='Accent')
index_names = train[train['category']=='Theft'].index
print(index_names)
train.drop(index_names , inplace=True)
index_names = test[test['category']=='Theft'].index
print(index_names)
test.drop(index_names , inplace=True)
test['target'] = test['category'] + "_" + test['subcategory']
test.head()
train.drop(["pkSeqID","seq"], axis=1, inplace=True)
test.drop(["pkSeqID","seq"], axis=1, inplace=True)
train[train['category']=='Normal']
check='0x'
s_res = set([i for i in train['sport'] if i.startswith(check)])

train['sport']=train['sport'].replace(['0x0303'],'771')
train['sport']=train['sport'].replace(['0x0011'],'17')
train['sport']=train['sport'].replace(['0x000d'],'13')
train['sport']=train['sport'].replace(['0x0008'],'8')

test['sport']=test['sport'].replace(['0x0303'],'771')
```

```
test['sport']=test['sport'].replace(['0x0011'],'17')
test['sport']=test['sport'].replace(['0x000d'],'13')
test['sport']=test['sport'].replace(['0x0008'],'8')
train["sport"] = train["sport"].astype(str).astype(int)
test["sport"] = test["sport"].astype(str).astype(int)
check='0x'
d_res = set([i for i in train['dport'] if i.startswith(check)])
print(len(d_res))
train["dport"] = train["dport"].apply(lambda x: int(x,16) if len(x)>1 and x[1]=="x" else int(x))
test["dport"] = test["dport"].apply(lambda x: int(x,16) if len(x)>1 and x[1]=="x" else int(x))

#Since dport can't be negative, we are dropping it
index_names = train[train['dport'] == -1].index
train.drop(index_names, inplace=True)
index_names = test[test['dport']==-1].index
test.drop(index_names , inplace=True)
train.drop(["category","subcategory"], axis=1, inplace=True)
test.drop(["category","subcategory"], axis=1, inplace=True)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train["saddr_enc"]= le.fit_transform(train.saddr)
train["daddr_enc"]= le.fit_transform(train.daddr)
train["proto_enc"]= le.fit_transform(train.proto)
train["target_enc"]= le.fit_transform(train.target)
train.head()
train.drop(['saddr','daddr','proto','target'], axis=1, inplace=True)
test["saddr_enc"]= le.fit_transform(test.saddr)
test["daddr_enc"]= le.fit_transform(test.daddr)
test["proto_enc"]= le.fit_transform(test.proto)
test["target_enc"]= le.fit_transform(test.target)
test.drop(['saddr','daddr','proto','target'], axis=1, inplace=True)
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
features = train.iloc[:,:-1]
cols=features.columns
scaled_features= scaler.fit_transform(features)
train= pd.DataFrame(scaled_features,columns=cols)
ytest = test['target_enc']
features = test.iloc[:,:-1]
cols=features.columns
scaled_features= scaler.fit_transform(features)
test= pd.DataFrame(scaled_features,columns=cols)
x = train
xtest = test
import imblearn
from imblearn.under_sampling import RandomUnderSampler
samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 8:58592, 7:14267, 3:1179, 0:786, 6:332}
#samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 3:1179, 0:786, 6:332}
random_under= RandomUnderSampler(sampling_strategy=samp_strat,random_state=1)
X_rus,y_rus = random_under.fit_resample(x,y)
from imblearn.over_sampling import RandomOverSampler
samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 8:58592, 7:30000, 3:20000, 0:15000, 6:8000}
#samp_strat= {5:70000, 1:70000, 2:70000, 4:65000, 3:20000, 0:15000, 6:8000}
random_under= RandomOverSampler(sampling_strategy=samp_strat,random_state=1)
Xres,yres = random_under.fit_resample(X_rus,y_rus)
plt.figure(figsize=(10,5))
sns.countplot(yres,palette='magma')
from sklearn import model_selection
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, classification_report, f1_score
X_train, X_test, y_train, y_test = model_selection.train_test_split(Xres,yres, test_size=0.40)
scores_gini = []
print('Accuracy score for values of max_depth of a decision tree are :')
print('Max Depth     Accuracy Score')
```

```
for i in range(1,7):
  dtree_test_gini = DecisionTreeClassifier(criterion='gini',max_depth=i)
  dtree_test_gini.fit(X_train,y_train)
  scores_gini.append(dtree_test_gini.score(X_test,y_test))
  print(i,'         ', scores_gini[i-1])
model_1 = DecisionTreeClassifier(max_depth=6)
model_1.fit(X_train, y_train)

start_time = time.time()
for i in range(100):
    pred_1= model_1.predict(X_test)
print("--- %s seconds ---" % (time.time() - start_time))
score1 = model_1.score(X_test, y_test)
print("Accuracy of base model: ",score1)
print(classification_report(y_test, pred_1, target_names = ['DoS_UDP','DDoS_TCP', 'DDoS_UDP',
'Service_Scan', 'OS_Fingerprint','DoS_TCP', 'DoS_HTTP', 'DDoS_HTTP', 'Normal_Normal']))
```