МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія розпізнавання ботнет-атак у кіберфізичних системах»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Москаленко В.В.**

Студента групи ІК.мз-91с — **Долгополова С.В.**

СУМИ 2020

ЗМІСТ

ВСТУП

Останнім часом кіберфізічні системи (CPS, cyber-physical system) набувають все більшої популярності в різних галузях промисловості та приватного сектору. Як і було спрогнозовано раніше, зараз ці системи починають глибоко інтегруватися у повсякденне життя людини для забезпечення різних побутових потреб і дозвілля. Також CPS набули неабиякої популярності в системах виробництва, що дозволило значно скоротити витрати та підвищити якість продукції.

Під CPS розуміють інтеграцію різного виду фізичних об’єктів та обчислювальних ресурсів. Ця синергія охоплює всі необхідні ресурси для управління інформацією, а саме: комп'ютери, програмне забезпечення та мережі, що дозволяє створювати, здійснювати моніторинг, передавати і зберігати інформацію.

Обчислення в кіберфізичних системах розподілене по всій фізичній системі і пов'язане з її складовими елементами, такими як інформаційні системи. Обладнання і датчики можуть виходити за фізичні межі певної локації, регіону чи країни, що не заважає їм взаємодіяти один з одним за допомогою стандартних протоколів для обміну даними, самоналаштування, адаптування та прогнозування.

Інтернет речей (IoT, Internet of things) – наступний великий еволюційний крок у світі інтернету. Ця революційна технологія, що вбудовує Інетрнет в фізичні пристрої, відкриває можливість з'єднання фізичної речі з будь-якої точки світу. IoT неймовірно величезна мережа підключених пристроїв, за допомогою якої можна легко обмінюватися даними і породжувати концепцію Глобального обчислювального шару. Ця концепція об'єднує функції Інтернету, комунікацій і цифрових технологій в єдину технологію, що дозволяє техніці стати по-справжньому незалежною.

Однак більшість пристроїв не мають достатнього захисту на рівні програмного забезпечення і мають як відомі, так і приховані вразливості. Через це можуть бути використані кіберзлочинцями для масштабних кібератак за допомогою побудованих ботнетів Інтернету речей. Такі атаки можуть призводити до пошкодження інфраструктури, перебоїв в обслуговуванні, викликати величезні фінансові збитки та руйнувати іміжд і репутацію великих компаній.

Один із підходів захисту зводиться до проактивного блокування таких ботнетів інтернету речей через автоматичне сканування, викриття вразливих пристроїв Інтернету речей та ізолювання їх від інтернету перш ніж вони будуть скомпрометовані, а також стануть частиною IoT ботнету. Також можна виділити заходи, що направлені на виявлення та ізоляцію IoT мереж через маршрутизатори доступу. Цей підхід допомагає запобігти компроментації пристроїв та ізоляції ботнету IoT.

Також існує інший підхід в пасивному виявленні та розпізнанні ботнет атак за допомогою аналізу всієї мережевої та внутрішної активності IoT. Через використання величезних масивів даних за одиницю часу людино-ресурси обмежені, тому досить дорого використовувати їх для аналізу і прийняття рішення. Більш практично використовувати Штучний інтелект і Машинне навчання.

Штучний інтелект (AI, Artificial intelligence) – це здатність механічного пристрою імітувати людські інтелектуальні процеси, а машинне навчання (ML, Machine Learning) – це технологічна група AI, яка виконує пошук даних, аналізує та классифікує їх. Потім на підставі цього аналізу прогнозує закономірність і будує алгоритм, що дозволяє автоматично програмувати, налаштовувати свої підсистеми і навіть стати незалежним. Все це дозволяє швидко та з мінімальною похибкою приймати рішення про розпізнавання кібератак.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

За останні роки кількість розумних пристроїв IoT різко зросла. Через невиликі витрати на обладнання та програмне забезпечення з відкритим кодом, багато компаній виробляють пристрої IoT. Звіт, опублікований компанією HP у рамках проекту захисту веб-додатків (OWASP), доводить що виробники ігнорують аспекти безпеки під час розробки цих пристроїв [1]. Тому пристрої IoT стали потенційно вразливими об'єктами для кіберзлочинців. Крім того, спеціалістам із безпеки стало важко обслуговувати величезну кількість даних, що знаходяться на пристроях та передаються в мережах IoT.

Складність, обумовлена кількістю пристроїв та мереж IoT, надає хакерам можливість перетворити прості пристрої, такі як телевізори, камери, DVD-диски та концентратори, у шкідливі бот-мережі для запуску небезпечних кібератак [2].

Для застосування основного заходу безпеки, такого як криптографія в пристрої IoT, є дві основні проблеми:

- нестабільна архітектура, інфраструктура та стандарти;

- непідтримувані та недостатні ресурси.

Застосування відповідного захисного механізму (послаблення) необхідне для блокування супротивників, щоб зменшити вплив на пристрої та/або кінцевих споживачів. Незважаючи на атаки, що постійно зростають, важко повністю пом'якшити моніторинг мережі в режимі реального часу за допомогою системи виявлення та запобігання проникненню. Але прийняття потужного механізму контролю доступу та автентифікації може запобігти атакам.

1.1 Сучасний стан та тенденції розвитку систем кібер-захисту мереж інтернету речей

IoT – це мільярд взаємозв’язаних через Інтернет різнорідних об’єктів. Кількість підключених інтелектуальних пристроїв IoT перевершила людську популяцію. У 2015 р Ericsson прогнозувала, що в 2020 році до Інтернету буде підключено 20 мільярдів пристроїв, але вже на сьогодні число IoT-гаджетів досягло 26,6 мільярдів.

На рисунку 1.1 показані різні сфери застосування пристроїв IoT, що включає Smart Grid, Smart Retail, Smart Supply Chain, Smart Agriculture, Smart Industry, Smart Transport, Smart Health, Smart Wearables, Smart Housing & Building and Smart City. Зі згаданих вище статистичних даних та сфер застосування стає зрозуміло, що IoT є майже в кожному секторі, і тому необхідно розуміти, як працює пристрій IoT.

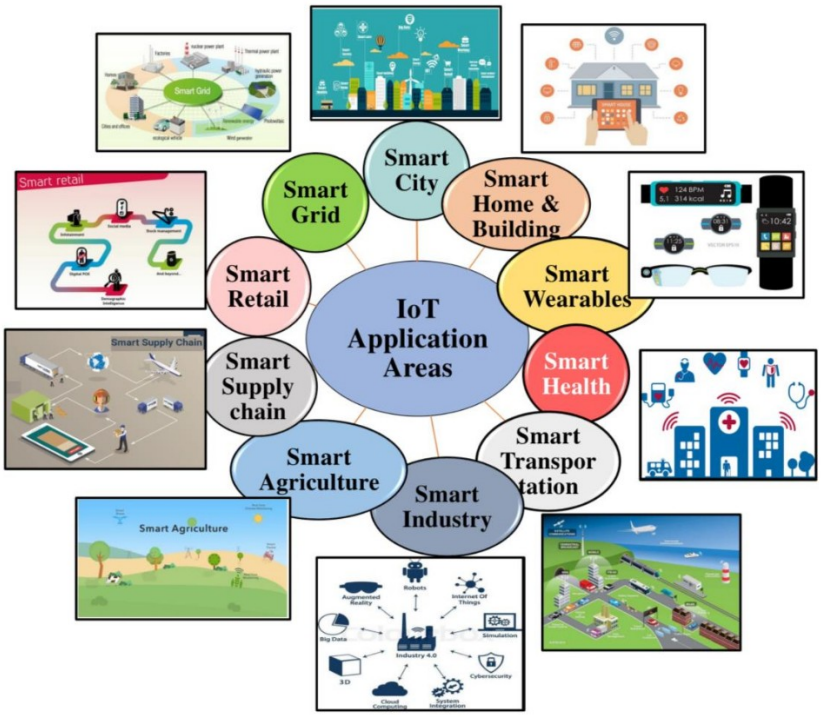

Рисунок 1.1 – Області застосування IoT [3]

Будь-який пристрій IoT працює у 3 фази:

- фаза збору;
- фаза передачі;

- фаза обробки, управління та використання.

Фаза збору – це початковий крок для збору даних із фізичного середовища за допомогою пристроїв та технологій зондування комунікацій короткого діапазону. Пристрої для цієї фази мають менший заряд акумулятора, обробну потужність та обмежену пам’ять. Призначення протоколів зв'язку складається таким чином, що він споживає менше енергії, працює на обмеженій швидкості передачі даних, малій пам'яті та обробній потужності на короткі відстані. Через вищезазначені причини ці мережі називають мережами з низьким енергоспоживанням та втратою енергії (LLN). Отже, механізми безпеки повинні адаптуватися до обмеженості ресурсів цих пристроїв.

Фаза передачі передає зібрані протягом фази збору дані користувачам та програмам. При цьому використовуються такі технології передачі, як Ethernet, Wi-Fi, Bluetooth, Bluetooth з низьким енергоспоживанням (BLE), коаксіальна гібридна волоконна (HFC) та цифрова абонентська лінія (DSL). Більшість із цих технологій вразливі до атак. Шлюзи, що використовуються на фазі збору, інтегрують протоколи LLN з Інтернет-протоколами фази передачі.

Фаза обробки, управління та використання за допомогою програм обробляє зібрані дані, щоб отримати інформацію про навколишнє середовище. Іноді додатки повинні самі приймати рішення на основі зібраної інформації [7]. Вони також мають проміжний пакет для інтеграції зв'язку із фізичними об'єктами та багатофункціональними додатками.

Вищезазначені фази експлуатації потребують захисту щоб забезпечити належне надання послуг. Далі ми будемо досліджувати періодичні атаки безпеки на архітектуру IoT [3].

1.1.1 Атаки на рівні IoT

Хоч і не існує стандартизованої моделі архітектури IoT, а останні досягнення додали до них більш абстрактні рівні, основними типами архітектур, які широко використовуються, є 3-рівневі, 4- рівневі та 5- рівневі

архітектури [4]. Можна виділити атаки на трьох рівнях: сприйняття, транспорт та мережа. На рисунку 1.2 показані три основні рівні IoT, оскільки саме на них переважно націлені кібер-атаки.

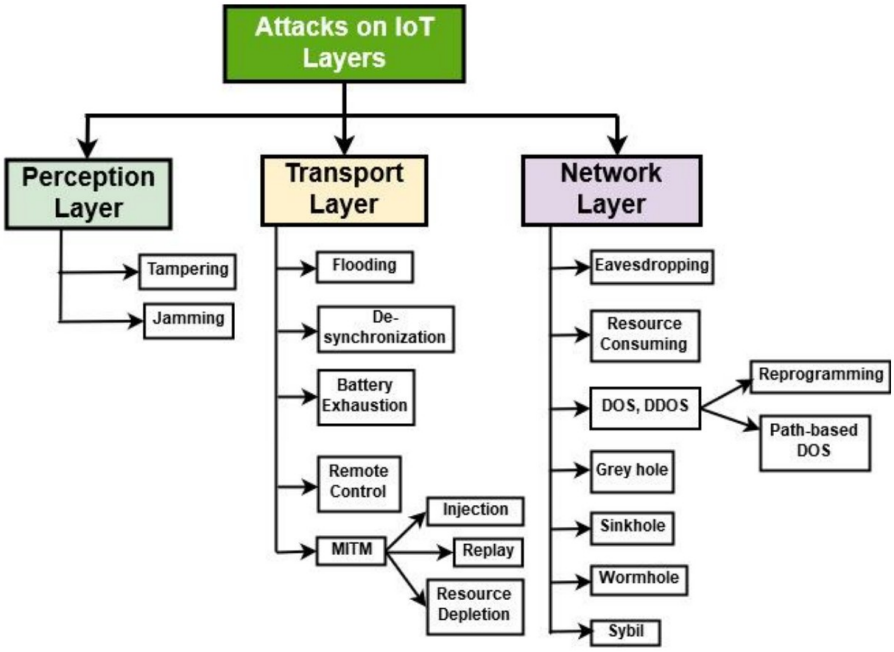

Рисунок 1.2 – Атаки на рівні IoT [3]

Типи атак, згадані на рисунку 1.2, є найбільш актуальними, але існують і інші атаки, які не враховувались через їх непопулярність або через те що вони виходять за межі механізмів безпеки, необхідних для таких атак.

Рівень сприйняття (Perception Layer).

Це перший рівень, який складається з фізичних датчиків та виконуючих пристроїв IoT для сприйняття навколишнього середовища та збору інформації.

Поширена атака на цей рівешь – глушіння (jamming) та темперинг (tampering). Під час атаки глушіння атакуючий порушує роботу мережі способом стискання або перешкоджаючи за допомогою високочастотних сигналів [5]. Противник може атакувати будь-який вузол датчика щоб заблокувати всю мережу, що призведе до атаки відмови в обслуговуванні (DoS) або розподіленої відмови в обслуговуванні (DDoS). Як підсумок нижче наведена узагальнююча таблиця [3].

Таблиця 1.1

Атака	Техніки та наслідки	Механізм захисту
Бездротова атака глушіння	Перешкоди зв'язку з використанням високочастотних сигналів.	Розподіл використання за спектром та постійний моніторинг когнітивного спектру.
		Доступ до сили отриманого сигналу за допомогою архітектури MAPE.
	Випадкові атаки глушіння на основі зондування за допомогою Deep Learning.	Система глибокого навчання для перенаправлення та пошкодження рішень глушителя.
	Повне заклинення сигналів Wi-Fi та погіршення продуктивності мережі	Підсилене навчання для послаблення перешкод на основі алгоритму Q-навчання.

Транспортний рівень (Transport Layer).

Цей рівень контролює наскрізні посилання і він, в основному, стикається з двома типами атак – флудінгом та асинхронізацією. При флудінг атаці (TCP Synchronization / TCP-SYN) ресурси пам'яті пристроїв вичерпуються шляхом повторного розповсюдження керуючого сигналу. Тоді як під час несинхронізованої атаки зловмисник перериває повністю встановлений зв'язк між двома справжніми кінцевими вузлами шляхом повторної синхронізації (нескінченний цикл) їх передачі. Це порушує зв'язок і вичерпує ресурси мережі. Такий тип атак призводить до виснаження мережі.

У таблиці 1.2 показано кібератаки на цьому рівні,а також можливі заходи безпеки [3].

Таблиця 1.2

Атака	Техніки та наслідки	Механізм захисту
Флудінг атака, ICMP / TCP / UDP / HTTP / DNS	Сигнал управління, що повторюється, вичерпує пам’ять та висніжує акумулятор. Може призвести до DDoS-атаки або атаки перешкод.	Механізм обмеження швидкості в Contiki OS.
		IDS на основі SDN для моніторингової діяльності.
		Модуль динамічного виявлення аномалій шляхом вивчення поведінки атак.
Атака на виснаження батареї	Шкідливі коди для розширення завдань які споживають більше енергії, іноді роблять пристрій неефективним.	IDS контролює використану пристроєм енергію. Якщо цей показник перевищує порогове значення – спрацьовує попередження.
Атака дистанційного керування	Перехоплює зв’язок за допомогою ботнетів або атак MITM щоб отримати повний контроль над пристроєм і зруйнувати його ресурси Руйнівний в розумному домі, ICS, інтелектуальній мережі, системах управління енергопостачанням.	Модель безпеки TLS і DTLS для LLN на основі CoAP.
		Система моніторингу ідентичності для ICS з використанням криптографії, обробки зображень, автентифікації та авторизації.
		Багатоканальні опціональні шлюзи IoT. Приховані за допомогою служб Tor вузли IoT, роблять їх доступними лише для авторизованих користувачів.
Атака людина посередині (MITM)	Нападник атакує дані для маніпулювання або видалення інформації. Це може призвести до DoS, відтворення, вичерпання ресурсів та атаки введення.	Контрольований IDS для класифікації атак.
		Клієнт-серверна модель, автентифікація ключа сервера із значенням даних датчика.

Мережевий рівень (Network Layer).

Цей рівень використовує для зв'язку з пристроями різні технології, такі як ідентифікація радіочастот (RFID), IFR, 3G, GSM, BLE, Універсальна система мобільного зв'язку (UMTS), WiFi, ZigBee тощо. Зв'язок в пристроях IoT відбувається за допомогою маршрутизації, і вона вразлива до різних атак [6].

Маршрутизуючі атаки включають спуфінг, вибіркову переадресацію, зміну шляхів маршрутизації або повторне відтворення пакетів, пробоїн, теплих отворів тощо. Ці атаки можуть призвести до загроз DoS.

У таблиці 1.3 наведено деякі атаки на мережу або рівень каналу передачі даних та їх захисні заходи. Хоча DoS або DDoS також можна запускати на транспортному рівні [3].

Таблиця 1.3

Атака	**Техніки та наслідки**	**Механізм захисту**
Прослуховування	Атака краде дані під час передачі так, що супротивник може прослуховувати або змінювати дані.	Інноваційний метод зв’язку видимого світла (VLC), заснований на кореляції каналів та оцінці помилок.
Атаки на споживання ресурсів	Атаки Unfairness зіткнення та виснаження (Collision and Exhaustion). Неможливість надання послуг.	Симетричне шифрування та багаторівневий механізм захисту за допомогою TLS.
Атаки модифікаційного типу (атаки маршрутизації)	Атаки Сірої діри (Grey hole), Sinkhole, чорної діри (Black hole) та червоточини (Wormhole). Вбудовані заходи безпеки, такі як автентифікація та контроль доступу, не можуть пом'якшити або виявити такі атаки.	IDS для поглинальних (sinkhole) і вибіркових атак.
		IDS для виявлення атак червоточини (Wormhole) в середовищі IoT.
		Підхід, заснований на специфікації протоколу RPL, контролює втручання в мережі та зловмисну поведінку.
Атака Сивілли	Вузол із помилковою ідентифікацією, DoS або загрозою відтворення.	IDS на основі хоста з використанням SDN блокує пристрій жертви. Модель SAAS.
Відмова в обслуговуванні	Атаки DoS, DDoS , Заборона сну (Denial of Sleep), SYN Flood, DNS Flood, Ping Flood, UDP Flood та ICMP Broadcast.	Архітектура SDN для ідентифікації DdoS та сканування портів. IDS в поєднанні з ними забезпечують кращий захист.
		Атаки ухилення проти ML IDS можна пом'якшити, використовуючи Gradient-based підхід.

Всі зазначені атаки важко пом’якшити, використовуючи традиційні заходи безпеки. Всі вони потребують оновлення та модернизації. Тому можна зробити висновок, що звичайні контрзаходи, що включають основні механізми, неефективні.

У більшості випадків сильна система автентифікації, контролю доступу та моніторингу є ефективною для виявлення, послаблення та припинення кібератак. Крім того, IDS здатний виявляти більшість типів атак на рівні сприйняття, транспорту та мережі.

1.1.2 Заходи безпеки

Далі більш детально розглянемо заходи безпеки, що можуть бути використані для протистояння подібним типам атак.

Велика кількість підключених IoT надає багато різних децентралізованих способів проникнення зловмисників або шкідливих програм. Заходи високого рівня безпеки створюють вузьке місце для адаптивності та роблять пристрій складним, а це в свою чергу викликає нові проблеми безпеки [7].

IoT вимагає різних налаштування для різних цілей. Вбудований захист повинен забезпечувати пристосованість пристрою та повинен бути масштабованим щоб мати можливість додавати до мережі більшу кількість пристроїв. Для забезпечення кращого захисту та основних особливостей безпеки, а саме автентифікації, контролю доступу, конфіденційності, цілісності та відмови на пристроях IoT, необхідне вдосконалення наступних практик безпеки.

Надійний механізм автентифікації.

Пристрої IoT мають функцію автентифікації за допомогою пароля для доступу до своїх послуг. Велике занепокоєння щодо автентифікації викликають слабкі паролі або паролі за замовчуванням, ботнети, троянці, що викрадають паролі, словникові та грубі атаки [8]. Сьогодні фахівці з безпеки рекомендують інтегрувати два методи для посилення автентифікації.

1) Біометрична аутентифікація. Заміна процесу автентифікації на основі пароля на біометричну автентифікацію гарантує більш високий рівень безпеки, оскільки вона є надійною проти звичайних атак злому паролів.

2) Багатофакторна автентифікація (MFA). Вона передбачає багатоступеневий процес автентифікації. Він може бути 2-кроковий або 3-кроковий, що включає комбінацію на основі знань (паролі), власності (картка), біологічної (відбиток пальця) особливості. Для одноразової автентифікації потрібні дві або три функції (облікові дані) користувача, такі як PIN-код та OTP для підтвердження банківської операції.

Надійний механізм контролю доступу.

Контроль доступу та захист даних на пристроях IoT із низькою потужністю став необхідністю захисту від розширення кібератак. Згідно із дослідженнями, біометричний контроль доступу є найбільш бажаним у IoT. Він приймає біологічні атрибути особи для перевірки та ідентифікації. У цьому процесі він порівнює діяльність індивіда із збереженими шаблонами в системі.

Цей механізм життєво важливий для уникання атак на основі хосту мережі [9]. При цьому модель перетворює зображення райдужної оболонки в ірисовий код (Iris Image). Перевірка коду райдужної оболонки здійснюється за допомогою дистанції Хеммінга. Він зберігає головний ключ у системі, використовує менше обчислень і має дуже невеликі накладні витрати.

Програмно-конфігурована мережа.

Програмно-конфігурована мережа (SDN, Software-defined Networking) – це управління мережевою безпекою в різних сферах застосування, таких як бізнес, розумні будинки та системи електронної охорони здоров’я. Будь-яка комп'ютерна мережа складається з комутаторів і маршрутизаторів як основних компонентів. Важливими функціями комутаторів/маршрутизаторів є площина управління та площина даних.

Контрольна площина відповідає за те, куди направити трафік, тоді як площина даних перенаправляє трафік до певного пункту призначення. У звичайних мережах площина даних і площина управління пов'язані між собою.

В архітектурі SDN площина управління відокремлена від площини даних. Суб’єкт на основі програмного забезпечення, званий контролером, віддалено керує завданнями площини управління. Площина даних виконується в апаратному забезпеченні та площині управління в програмному забезпеченні і знаходиться в логічно централізованому вигляді. SDN здатний контролювати мережевий трафік та виявляти шкідливі дії. Він визначає та ізолює пошкоджені вузли від решти мережі [10].

Система виявлення вторгнень.

Системи виявлення вторгнень (IDS, Intrusion detection system) – це програма або алгоритм, який намагається розпізнати шкідливу діяльність у мережі. Він також намагається виявити, коли комп'ютер атаковано або зловмисник намагається його скомпроментувати. Крім того, він визначає, чи законний користувач намагається збільшити привілеї або отримати доступ до несанкціонованих даних та послуг.

IDS став важливим елементом захисту ICT-інфраструктури. В даний час кожна мережа має IDS або IPS для виявлення та послаблення кібератак. IDS класифікується як мережевий, хостовий або прикладний, відповідно до моделі аналізу даних. У деяких контекстах системний та прикладний розглядаються як два випадки IDS на основі хостів. Більше того, на основі техніки/методу, що використовуються, IDS класифікується як Signature-based, Anomaly-based та Specification-based [3].

Система IDS повинна точно, швидко та ефективно розрізняти атаки з меншою кількістю помилок. Будь-який IDS, який точно ідентифікує атаки, але витрачає тривалий час для виявлення, не підходить для поточних мереж IoT.

Отже, стало необхідним дослідити метод, який здатний виявляти нові атаки з меншою кількістю помилкових тривог та який може обробляти

величезну кількість даних і швидко приймати рішення для виявлення атак в режимі реального часу.

Система, що базується на підписах, виявляє атаки на основі підписів та відомих моделей атак, але важко розкрити відомі відхилення або невідомі атаки. У літературі більшість реалізацій IDS базуються на правилах, що неефективні для виявлення нових атак [11]. Однак, якщо база даних підписів атак оновлюється шляхом додавання кожного разу нових підписів атак, тоді цей метод ефективний. Подібним чином, на основі специфікації передбачено визначення правил адміністратором. Система виявлення аномалій знаходить відхилення від заздалегідь визначеної нормальної поведінки, але створює багато помилкових тривог для законної поведінки коли профіль користувача складний і невідомий.

Складно оновлювати базу даних IDS через неоднорідну мережу та мінливі середовища, такі як топологія мережі, протоколи зв'язку, відкриті порти, сервери та кілька підключених пристроїв. Щоб подолати цю проблему, дослідники зосереджуються на адаптованих методах, таких як штучний інтелект, техніка машинного навчання та глибоке навчання [12].

З наведеного вище, можна вивести декілька кількісних показників у таблиці 1.4 для системи IDS для визначення її ефективності.

Кількісні показники IDS

Таблиця 1.4

Метрика	Опис
Покриття	Кількість і типи атак, які IDS може виявити у реальному середовищі.
Обробка пропускнії здатності трафіка	Властивість IDS обробляти трафік із великою пропускною здатністю, блокувати або протистояти пропускній здатності, що перевищує таку в каналі.
Протидія атакам на IDS	Нечисленні зловмисники націлюються на IDS так, що коли IDS скомпрометована, стає легше атакувати мережу та пристрої. Отже, IDS повинна бути здатна протистояти атакам, спрямованим на неї.
Ймовірність виявлення	Визначає наскільки точно система виявляє вторгнення та має точність до 70-90%. Тим не менше, мережа в режимі реального часу вимагає вищої точності виявлення різних атак.
Ймовірність помилкових тривог	Іноді діяльність, що не стосується атаки, класифікується як атака, і навпаки. Такі типи рішень IDS можуть спричинити фатальні наслідки. Останні алгоритми машинного навчання та глибокого навчання використовують матрицю рішення для правильної класифікації події.
Можливість виявлення невідомих атак	Занепокоєння щодо атак Zero-day зростає день за днем. Система, яка не здатна розпізнати незнайомі атаки, неефективна.
Можливість ідентифікувати напад	Важливо щоб система правильно визначала атаку і адміністратор зміг прийняти подільші відповідні рішення. Якщо атака класифікується неправильно, наприклад, атака «Червоточини» (Wormhole) класифікується як Сіра діра (Grey hole), це може ввести адміністратора в оману рівнем ризику атаки.
Можливість визначити успіх атаки	Система також повинна мати можливість визначати статус атаки.Наприклад, її успіх чи невдачу, наскільки вона успішна та якою мірою вона пошкодила ресурси.
Інші	Інші вимірювання включають простоту використання, розгортання та обслуговування. IDS повинен відповідати вимогам до ресурсів, продуктивності та якості обслуговування.

Легке шифрування.

Для захисту даних у стані спокою, передачі даних, збереження конфіденційності та цілісності, необхідне шифрування, яке інкапсулює дані у нечитабельному форматі та розшифровується лише одержувачем.

Найпоширеніші типи алгоритмів шифрування – симетричні та асиметричні. Кожен із цих підходів здатний захистити дані від деяких атак, що при іншому підході важко. Однак асиметричні алгоритми передбачають більшу кількість обробки, яка не підходить для пристроїв IoT [13].

В даний час Physical Unclonable Function (PUF) використовується для додавання додаткового апаратного рівня захисту для протистояння атакам фізичного рівня та для захисту кінцевих точок. Цей метод споживає менше ресурсів порівняно з іншими алгоритмами шифрування.

1.1.3 Ботнет з Інтернетом речей

Ризики для пристроїв IoT є важливою проблемою, оскільки їх важко відремонтувати. На них даже легко можуть вплинути зловмисники. Наявність ботнетів IoT було визнано після 2008 р. Проте, масштаби ризику, які вони представляють, не були визнані аж до 2016 р [14].

Як правило, заражена мережа кінцевих хостів (так званих ботів) визначається бот-мережами, і ці мережі фактично контролюються людиною, яку називають Botmaster. Вразливі та скомпрометовані пристрої вербуються ботнетами з використанням стратегій, які застосовуються іншими класами шкідливого програмного забезпечення (наприклад, соціальна інженерія, вразливості програм, що експлуатуються віддалено тощо) [15]. Такі пристрої встановлюють інфраструктуру управління та керування (C&C), щоб можна було здійснити шкідливу діяльність. Тому Botmaster отримує такі послуги від ботів [4]:

- botmaster забезпечує легке відновлення та моніторинг;
- кожен бот має обмежений вплив на ботнет;
- чітке розповсюдження трафіку, а також шифрування;
- надійне підключення до мережі.

Компоненти ботнета.

Сервер управління та керування (C&C) – це централізована система, яка може отримувати інформацію та надсилати інструкції ботам, що локалізуються у цій конкретній мережі. Ця інфраструктура складається з багатьох технічних частин, а також безлічі серверів. Багато ботнетів використовують структуру клієнтського сервера, але кілька ботнетів використовують однорангову (P2P) архітектуру, що складається з ботнетів, які мають функціональність C&C.

Peer-to-Peer (P2P) ботнет – для забезпечення додаткового захисту від видалення використовується децентралізована бот-мережа, яка називається ботнетом P2P. Хоча бот-мережі P2P можуть мати C&C-сервер, вони можуть додатково працювати і без цього, а також бути довільно організованими, щоб додатково закрити бот-мережу.

Botmaster, також відомий як бот-пастух або контролер ботнетів, виконує функцію оператора ботнетів. Бот-мережа регулюється командами віддаленого бот-майстра певним ботнетам, а також C&C-серверу в системі. Щоб запобігти переслідуванню правоохоронних органів та встановленню особи ботмайстера, місцезнаходження та ім’я ботмайстера залишаються незрозумілими.

Бот визначається як пристрій, який підключений до Інтернету в мережі ботнетів. В основному в якості бота використовується комп'ютерна система, але з появою технології IoT-пристрій, смартфон також може використовуватися як частина бот-мережі. Інструкція з експлуатації надсилається ботнетам від ботів тієї самої мережі, від ботмайстра безпосередньо або від сервера C&C. Зомбі – це не що інше, як синонім бота. Використовується як зовнішня особа або обчислювальний пристрій для управління ботом, тому бот називається "зомбі", а ботнет - "армія зомбі".

Приклад Ботнет атаки

- Ботмайстер отримує бот-мережу, поширюючи шкідливе програмне забезпечення для зараження ПК та додаткових систем. Він також може орендувати поточний ботнет у іншого злочинця.

• C&C центру повідомляється про нещодавно зібраний бот або "зомбі".

• Тепер ці боти контролюються C&C, а списки адрес потенційних жертв, шаблони електронної пошти та виконувані файли шкідливих програм розподіляються ботами відповідно до інструкцій C&C.

• Багато потенційних жертв отримують шкідливе програмне забезпечення, що містить повідомлення електронної пошти від заражених ботів на замовлення botmaster.

Методи виявлення ботнетів IoT середовища.

Зазвичай виявлення ботнетів, а також моніторинг є важливою темою розслідування внаслідок посилення шкідливої діяльності. На рисунку 1.3 показано таксономію методів виявлення ботнетів. Існує дві основні категорії методів виявлення ботнетів, Системи виявлення вторгнень (IDS, Intrusion Detection Systems) та Мереж приманок (HoneyNets) [16].

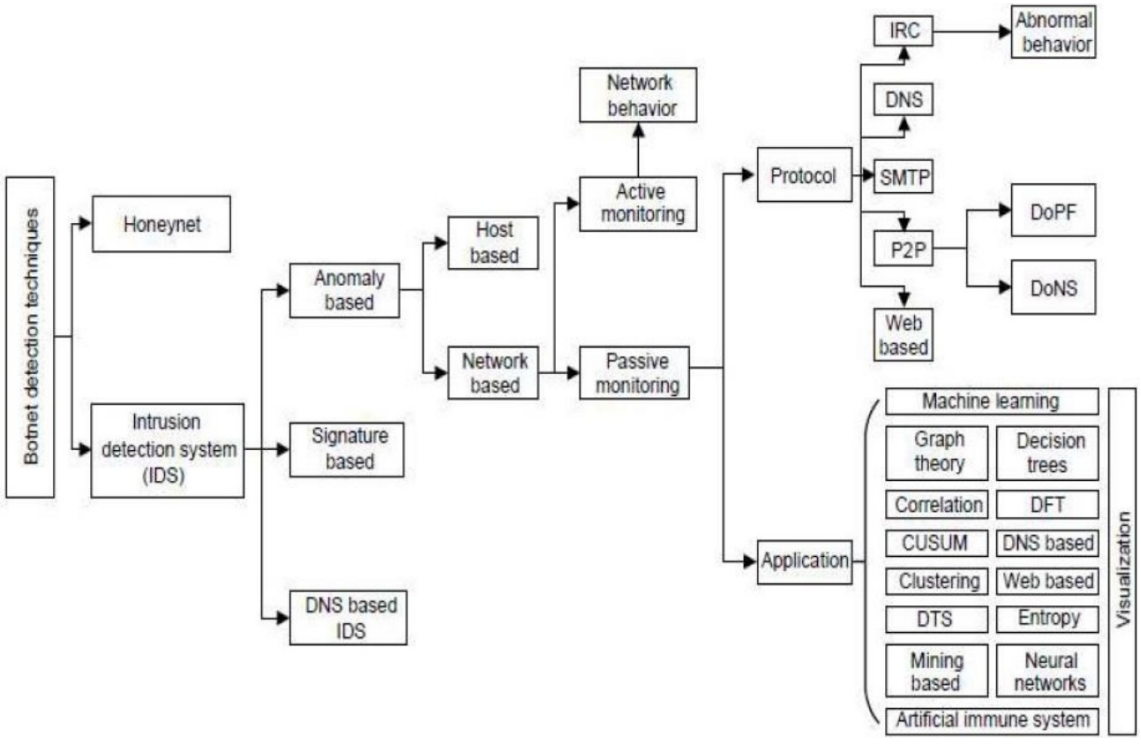

Рисунок 1.3 – Таксономія техніки виявлення ботнетів [17]

Honeynet: зазвичай Honeypot (“горшик з медом”, “приманка”) та Honeywall (“медовий бар'єр”) сумісно створюють метод, заснований на Honeynet [11]. З точки зору безпеки, Honeypot розглядається як комп'ютерна

система, яка допомагає заманити зловмисників, щоб зловмисник атакував певну комп'ютерну систему. Цей Honeypot повинен бути кінцевим хостом. Він може бути скомпрометований за короткий час і набагато більш сприйнятливий до зловмисних атак, тоді як honeywall складається з програмного забезпечення, за допомогою якого трафік через медовий пункт контролюється, збирається, а також модифікується [18].

Комп’ютерні системи, що використовуються як Honeypot, не мають жодної виробничої цінності. Виходячи з цього конкретного принципу, більшість взаємодій між іншими системами та Honeypot є недовірливими або підозрілими і повинні бути обстежені. Наприклад, у вашій мережі ви можете встановити веб-сервер Honeypot у певному місці.

Щоб записати детальні кроки ботнетів, Раджаб та ін. [19] створили розподілену та багатогранну інфраструктуру вимірювань, об’єднавши Honeynets із зміненою версією платформи .

Системи виявлення вторгнень: Системи виявлення вторгнень (IDS) використовуються для повідомлення про керуючу веб-сторінку. Коли системна політика порушується, системні служби відстежуються на предмет зловмисних дій або будь-які інші порушення. Ці IDS можуть бути як апаратною машиною, так і програмним додатком.

Перевага функції виявлення IDS полягає в тому, що вона має сигнатурні добірки розпізнаних ботнетів. Тим не менше, головними недоліками таких методів є:

- по-перше, збільшення аномалії через малу швидкість оновлення сигнатур IDS;

- по-друге, для виявлення щойно активованих ботнетів необхідно часто оновлювати сховище бази знань сигнатур [16].

Ще одна класифікація методів виявлення на основі сигнатур, аномалій та DNS [22].

Виявлення сигнатур. У цьому конкретному методі достатнім для захоплення ботів є підпис поточного бота відповідно до доступної інформації.

Для ідентифікації ботнетів збирається бібліотека певних імен функцій ботнетів, а також інструкції, які можуть бути включені та узагальнені в IDS. Після того, як IDS знайшов відповідні пошукові фрази під час вивчення вмісту корисного навантаження, він може видати попередження та вжити додаткових заходів. Хоча слід відзначити що цей конкретний метод обмежений для ідентифікації лише розпізнаних ботнетів.

Виявлення аномалій. Цей метод реалізується шляхом пошуку нерегулярних або ненормальних дій системи. Тут "ненормальна" дія зазвичай означає виявлення ботів як відхилення від "звичайної" дії, яка вже визначена відповідним чином у деяких керівних принципах.

Сайг і Бінклі [20] запропонували хороший метод виявлення аномалій на основі TCP з токенізацією IRC, статистику повідомлень IRC для ідентифікації клієнтів Botnet, а також викриття серверів Botnet.

Спочатку компонент розбору IRC, реалізований у цьому конкретному методі заснованому на аномалії, збирає інформацію про пакет TCP, а також визначає канал IRC.

Далі, дії сканування виконуються на великому наборі вибіркових даних, який корелює з трафіком каналів IRC [14].

Більше того, нарешті, маршрути IRC з гарною корелюючою ознакою будуть позначені як доступні станції ботнету. Передбачалося, що всі боти будуть демонструвати схожу синхронізацію, відповідь та взаємозв'язок, що належить одному і тому ж ботнету.

Гу та інші запропонували BotHunter – систему виявлення ботнетів [21]. Ця система виявляє ботів за допомогою визначеної користувачем моделі життєвого циклу зараження ботами. Потім запускає алгоритм кореляції, який допомагає розпізнати фазу зараження ботами.

Визначення на основі DNS: Цей метод виявлення є гібридом методів добування даних та методів поведінки, що використовуються для трафіку DNS. Взагалі, Botmaster може легко приховати та підтримувати своїх ботів, тому треба подумати про фактори, які застосовуються до запитів DNS у декілька фаз ботнету. До таких факторів відносяться оновлення сервера C&C, ініціація шкідливих атак та процес збору після зараження, що є перевагою стратегії.

Ботнет атаки

Атака відбувається, коли пристрій, заражений ботом та підключений до інтренету. Отже, бот-мережа також є частиною зараженої мережі пристроїв, за якою стежить один зловмисник або група. Часто ботнетів називають комп’ютерними черв’яками чи арміями зомбі, а господарі ботів або пастухи ботів є їх власниками.

Командування та управління (C&C). C&C ботнетів унікальні і, ймовірно, не змінюватимуться між ботнетами. На додаток до підтримки діючого, ефективного ботнета, важливим є ботнет C&C. Також вважається, що C&C є найслабшою ланкою в операційному вимірі ботнетів. Якщо нам вдається порушити активний C&C або просто спричинити порушення зв'язку, ботмайстри не можуть підключити занадто багато ботів або ініціювати великі, скоординовані атаки. Тому для боротьби з ботнетами важливо розуміти функцію C&C в ботнетах.

Принципи управління сервером C&C досить простий. Зазвичай IRC-сервер створюється Botmaster. Якщо хост заражений бот-вірусом, він повернеться до серверу в C&C, щоб зачекати команду від ботмастера. Бот приєднується до певного IRC-каналу в типовій бот-мережі IRC для читання повідомлень від свого хозяїна.

Механізми об'єднання. Механізми згуртування ботнетів – ще одна вивчена особливість. Механізми об'єднання ботнетів для пошуку та збору нових ботів під їх ботмайстрами мають вирішальне значення.

Розглянемо основні механізми збору.

• Жорстко закодована IP-адреса. Це найпоширеніша процедура, яка використовується для збору нових ботів. Бот має IP-адреси серверу C&C, які жорстко закодовані у своїй базі даних. Якщо бот спочатку заражає комп'ютер, IP-адреса жорстко закодованого сервера, що міститься в двійковому коді, може бути використана для зворотного з'єднання з сервером C&C.

• Динамічне доменне ім’я DNS. Сьогодні боти часто надають жорстко закодовані доменні імена динамічними провайдерами DNS. Перевага динамічного DNS полягає у тому, щоб дозволити ботмайстрам легко відновити управління, створивши новий сервер C&C та оновивши IP-адресу у відповідному динамічному вході DNS. Боти робитимуть запити DNS і будуть відправлені назад на новий сервер C&C, коли з’єднання зі старим сервером C&C не вдасться.

• Розподілена служба DNS. Багато найновіших підвидів ботнетів працюють з власним розподіленим сервісом DNS, недоступним правоохоронним органам. Боти містять адреси DNS-сервера і спілкуються з ними для пошуку IP-адрес сервера C&C. Щоб уникнути виявлення пристроями безпеки на шлюзах, ці послуги часто використовуються за умови великих номерів портів.

Протоколи зв'язку. Протоколи зв'язку, що використовуються в ботнетах, є одними з основних характеристик ботнетів. У цьому відношенні, як і у випадку з багатьма іншими програмними засобами, боти спілкуються в певних чітко визначених мережевих протоколах між собою та своїми ботмайстрами. Ботнети, як правило, не створюють нових мережевих протоколів зв'язку. Вони використовують існуючі протоколи зв'язку, які реалізовані програмними засобами і є загальнодоступними [16].

1.2 Аналіз сучасних моделей та методів машинного навчання

В цьому розділі представлений огляд різних методів машиного навчання (Machine Learning, ML) на основі середовища IoT. У таблиці 1.4 наведено короткий огляд методів ML, їх переваги та обмеження. Рисунок 1.4 ілюструє найпоширеніші методи ML, що використовуються для проектування в мережах IoT.

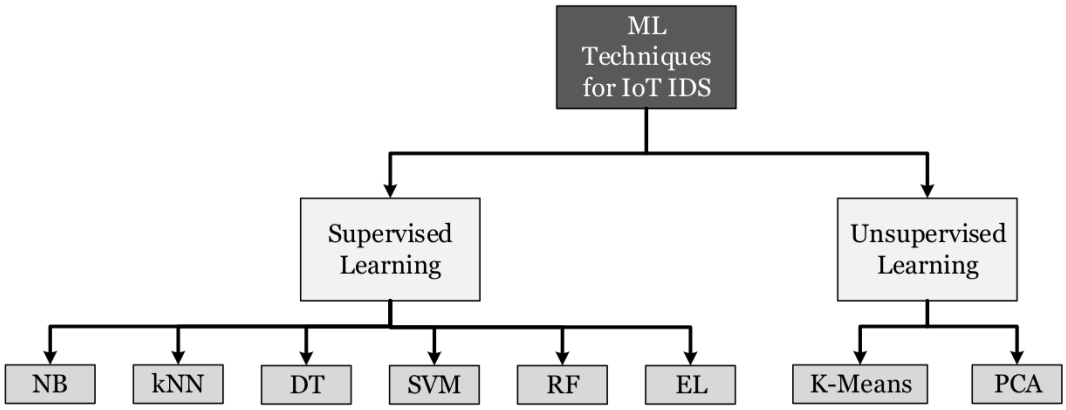

Рисунок 1.4 – Таксономія методів ML на основі IoT [22]

Класифікатор Naive Bayes (NB).

Цей алгоритм використовує теорему Байєса (Bayes) для прогнозування ймовірності настання події на основі попередніх спостережень за подібними подіями. У сценаріях ML це може бути використано для класифікації нормальної та ненормальної поведінки на основі попередніх спостережень у режимі контрольованого навчання.

Класифікатор NB – це простий загальновживаний контрольований класифікатор. NB обчислює ймовірність і на основі цього приймається рішення про маркування, щоб класифікувати немаркований трафік як звичайний або аномальний. Незалежний набір ознак відстежуваного трафіку (як приклад, прапорці стану, протокол, затримка) використовуються для прогнозування ймовірності нормального або іншого трафіку. Легке впровадження алгоритму, дозволяє застосовувати класифікатор NB для виявлення аномального трафіку. Для цього потрібно досить мало зразків для навчання і классифікації. Однак

цей класифікатор не враховує взаємозалежності між ознаками для цілей класифікації, що впливає на його точність [23].

K-Найближчий сусід (K-Nearest Neighbor, KNN).

KNN не вимагає жодних параметрів для своєї роботи. Евклідова відстань використовується для вимірювання відстані між сусідами [24]. На рисунку 1.5 показано основний принцип, що лежить в основі алгоритму класифікації KNN. Цей принцип використовується для класифікації нового екземпляру даних на основі його відносної відстані до будь-якого з класів.

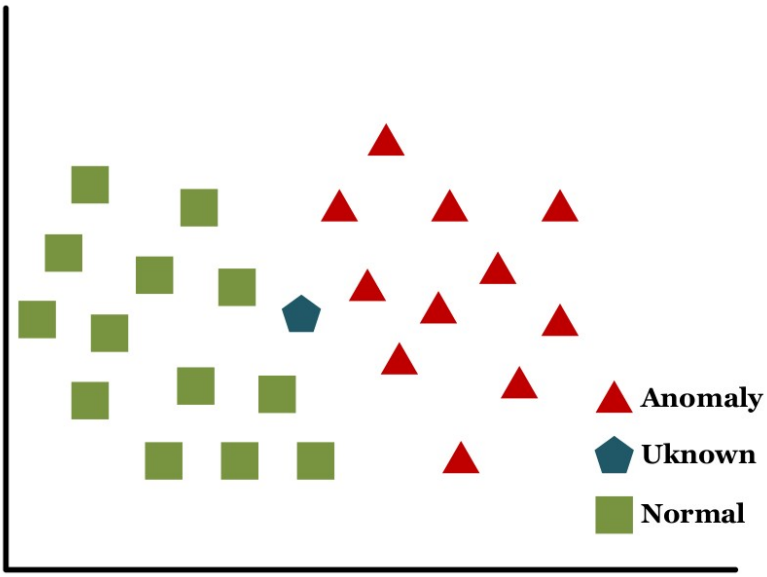

Рисунок 1.5– Принцип класифікації K-Найближчий сусід (KNN) [24]

Зелені квадрати відображають клас нормальної поведінки, а червоні трикутники – клас ненормальної поведінки. Будь-який знайдений невідомий екземпляр (синій шестикутник) тепер може бути класифікований на основі кількості максимально близьких сусідів будь-якого з класів. Відповідно, цей новий екземпляр класифікується як відомий клас. k – кількість найближчих сусідів, використаних для класифікації.

Класифікація зміниться зі значенням k. Для k = 1 червоний шестикутник буде класифікований як ненормальний клас, але для k = 2 і k = 3 він буде класифікований як нормальний клас. Отже, отримання оптимального значення k шляхом тестування є життєво важливим для точності цього алгоритму.

В деяких дослідженнях з достатньою точністю використовували KNN класифікацію для виявлення аномалій та вторгнень вцілому та вторгнень в мережі на основі IoT. Такий підхід був використаний при виявленні атак користувача до root (U2R) та атак віддаленого до локального (R2L). Незважаючи на те, що KNN простий у використанні, визначення оптимального значення k та виявлення відсутніх вузлів трудозатратні та дорогі з точки зору точності [22].

Дерева прийняття рішень (Decision Tree, DT).

Дерева рішень (DT) працюють шляхом вилучення об’єктів зразків у наборі даних та упорядковують в дерева на основі значення об’єкта. Кожна ознака представлена вузлом дерева, а відповідні значення представлені гілками, що відходять з цього вузла. Будь-який функціональний вузол, який оптимально ділить дерево навпіл, вважається початковим вузлом дерева. Для ідентифікації початкового вузла використовуються різні метрики, які оптимально ділять набори навчальних даних, такі як індекс Джині та Інформаційний приріст [25].

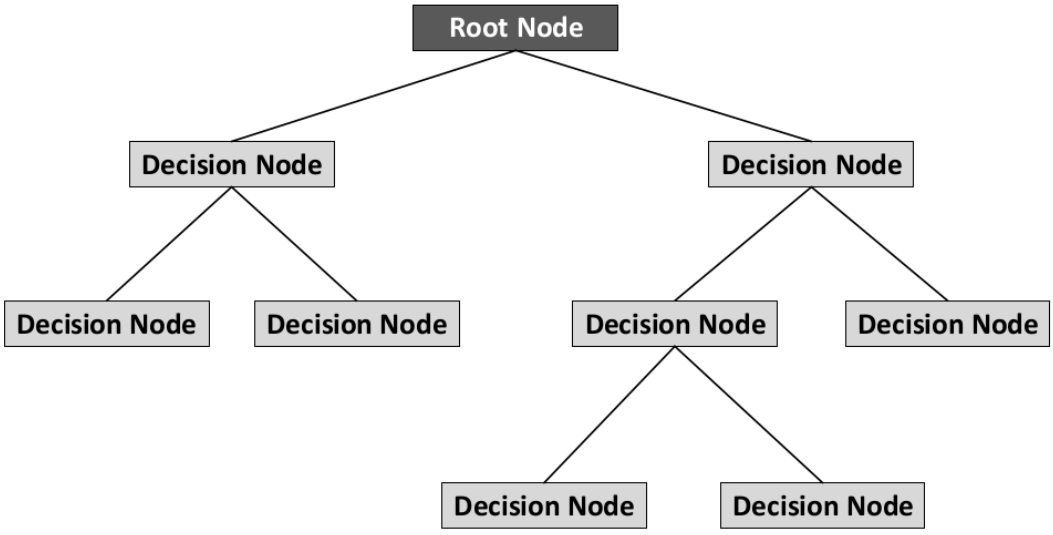

Рисунок 1.6 – Зображення структури дерева рішень [22]

Рисунок 1.6 ілюструє вузли дерева рішень. Алгоритми DT включають два процеси, а саме індукцію та умови, що спрямовані на побудову моделі, а потім проведення класифікації [26]. Під час індукційного процесу побудова DT починається з додавання вузлів і гілок. Спочатку ці вузли не зайняті, а потім за

допомогою процесу вибору ознак за допомогою отримання інформації та інших заходів вибирається функція, яка вважається розділеною на вибірки навчальних наборів даних. Потім ця функція призначається як початкова вершина DT.

Процес продовжує обирати кореневі вузли функцій, щоб мінімізувати накладення між різними класами, що знаходяться в наборі даних навчальних програм. Як результат, точність класифікатора збільшується при ідентифікації окремих екземплярів класу. Зрештою, листя кожного суб-дерева ідентифікуються та класифікуються відповідно до їх відповідних класів. Після побудови DT розпочинається процес виведення, де будь-які невідомі екземпляри класів з ознаками можуть бути класифіковані шляхом ітераційного порівняння з побудованим DT.

Після приєднання відповідного листового вузла процес класифікації нового зразка завершується. В контексті виявлення вторгнень DT мають потенціал для використання як класифікатор. Однак слід враховувати аспекти більших вимог до сховища та складності обчислень [22].

Метод опорних векторів (Support Vector Machine, SVM).

SVM – це ще один тип класифікатора, який працює шляхом створення гіперплощини в наборі функцій двох або більше класів. Гіперплощину розщеплення знаходимо через максимальну відстань до найближчої точки даних кожного порівнюваного класу [27], як показано на рисунку 1.7.

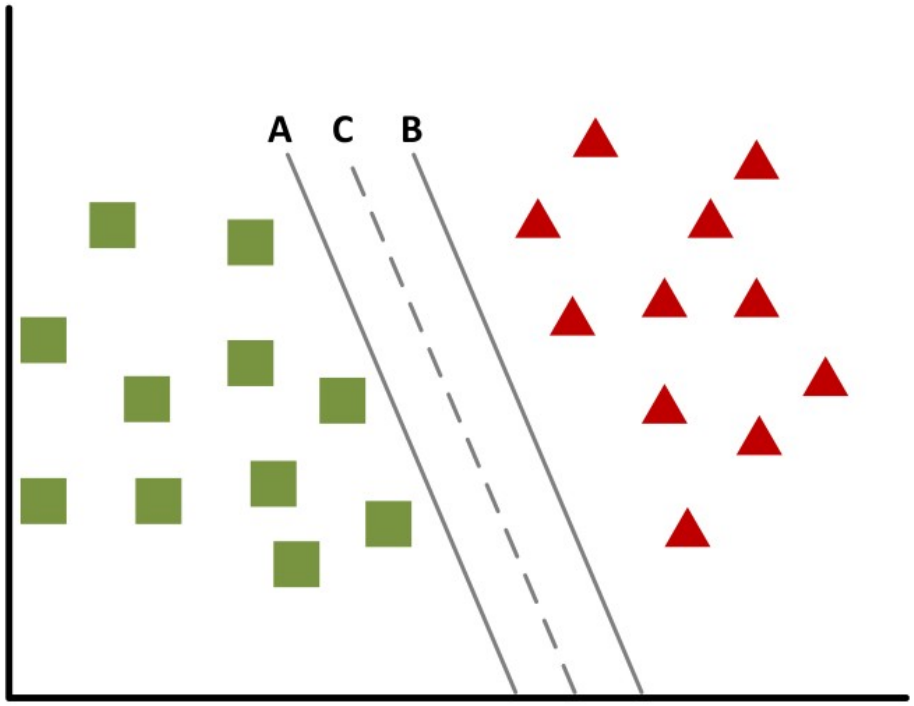

Рисунок 1.7 – Зображення гіперплощини розбиття
методом опорних векторів (SVM) [22]

SVM найбільше підходять для випадку, коли класи, що містять великі набори ознак, повинні класифікуватися на основі меншої кількість вибірок даних. На основі статистичного навчання, SVM ідеально підходить для виявлення аномалій, де необхідна класифікація між нормальними та аномальними класами. SVM є дуже масштабованим завдяки простоті і здатні виконувати такі завдання, як виявлення вторгнень на основі аномалій в режимі реального часу, включаючи навчання в режимі онлайн [28–30].

Ще однією перевагою використання SVM є використання меншого обсягу пам'яті. Використання SVM у системі IoT оцінювалось у різних наукових дослідженнях [22], де SVM показав більш точні результати, ніж інші алгоритми ML, включаючи DT, NB та Random Forest. Однак використання оптимальної функції ядра в SVM, яка використовується для розділення даних, коли вони не лінійно відокремлювані, залишається проблемою для досягнення бажаної швидкості класифікації.

Ансамблеве навчання (Ensemble Learning, EL).

EL працює, спираючись на сильні сторони різних класифікаторів, комбінуючи їх результати, а потім генеруючи більшість голосів за класифікацію, як показано на рисунку 1.8.

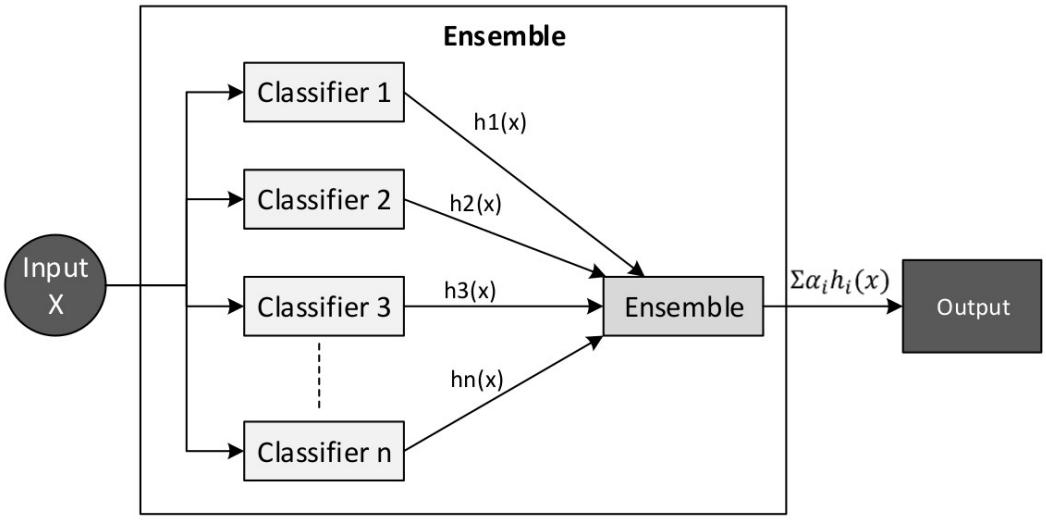

Рисунок 1.8 – Робота класифікатора ансамблю [22]

Це покращує точність класифікації завдяки поєднанню результатів різних однорідних/неоднорідних класифікаторів. EL базується на дослідженні, де було встановлено, що кожен алгоритм класифікації ML залежить від точності застосування та супутніх даних. Отже, жоден алгоритм ML не можна охарактеризувати як «єдиний розмір, що підходить для всіх рішень». Для узагальнених застосувань комбінації, подібні до EL методи, можуть бути найбільш підходящими для максимізації точності за рахунок зменшення дисперсії та уникнення перенавчання [31].

Ефективність EL для виявлення вторгнень вивчали в різних дослідженнях. Доступність EL в умовах обмежених ресурсів, таких як IoT, вивчалася із запропонованою узагальненою полегшеною структурою EL для онлайн-виявлення аномалій у мережах IoT. Це дослідження показало, що такий алгоритм EL давав кращі та точніші результати, ніж кожен класифікатор-член окремо [32].

Випадковий ліс (Random Forest, RF).

RF можна класифікувати як керований алгоритм ML. RF будується з використанням декількох дерев рішень (DT) для прогнозування більш точних та стійких до помилок результатів класифікації. Випадково побудовані дерева рішень навчаються виводити результати класифікації на основі більшості голосів.

Хоча DT можна розглядати як компоненти RF, існує два різних алгоритми класифікації, оскільки, на відміну від DT, які будують набір правил під час навчання для подальшої класифікації нових зразків, RF створює підмножину правил, використовуючи всі DT-члени. Це призводить до більш надійного та точного виходу, який стійкий до перенавчання і вимагає значно меншої кількості входів і не вимагає процесу вибору особливостей. Як пропонують деякі дослідження [33], RF підходить для виявлення аномалій та вторгнень в мережі IoT. Більше того, інше дослідження [34] показало, що RF кращий, ніж KNN, штучна нейронна мережа (ANN) та SVM при виявленні DDoS в мережах IoT, оскільки вимагає меншої кількості вхідних функцій і може обійти важкі обчислення, необхідні для вибору функцій в реальному часі [22].

Кластеризація методом k-середніх (k-Means Clustering).

Це некерований алгоритм, який базується на виявленні k кластерів у зразках даних. Кожен екземпляр зразків даних присвоюється певному кластеру на основі його особливостей. Зразки розподіляються по k кластерам відповідно до їх особливостей, використовуючи оцінку центроїдів відповідно до квадрату Евклідової відстані. Потім проводиться перерахунок центроїдів кожного кластера, беручи середнє значення точок даних, виділених для цього кластера, як показано на рисунку 1.9.

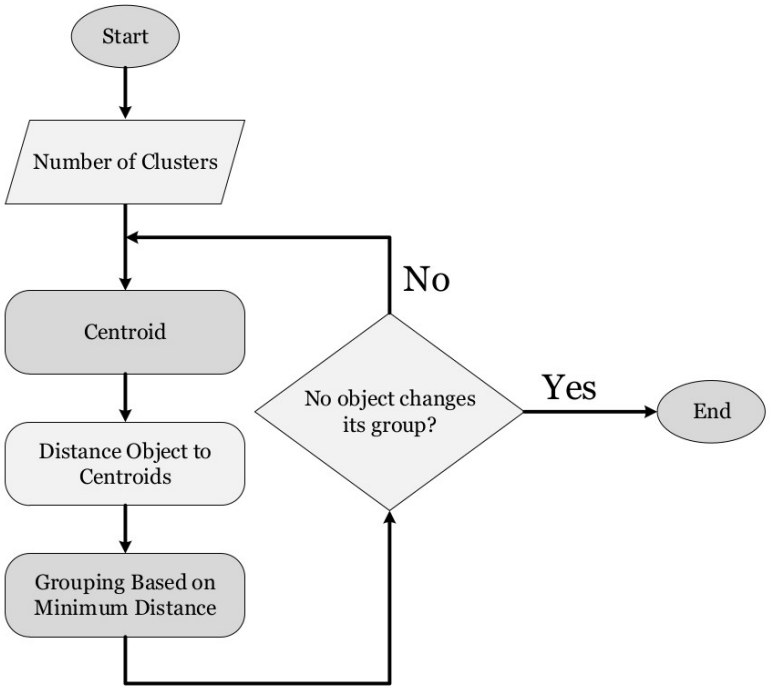

Рисунок 1.9 – Ілюстрація k-середньої кластеризації [22]

Процес триває ітеративно, доки буде не можливо вносити зміни в кластери. Вибір відповідного значення k та припущення, що вибірковий набір даних буде рівномірно розподілений по k кластерам, виступають обмеженнями для алгоритму кластеризації k-ознак. Попередні дослідження, представлені в [35], припускають придатність кластеризації k-середнє для виявлення аномалії шляхом обчислення подібності ознак. Автори [36] запропонували поєднувати DT з кластеризацією k-середнє для виявлення аномалій у мережах IoT для підвищення продуктивності.

Метод головних компонент (Principle Component Analysis, PCA).

PCA не є методом виявлення аномалій, але він зазвичай використовується як вибір об'єкта або метод зменшення ознак із великого набору даних. Потім вибрані набори функцій можна використовувати разом з деякими іншими класифікаторами ML для виявлення аномалій у мережі IoT. Техніка PCA перетворює великий набір змінних у зменшений набір функцій, не втрачаючи значної частини інформації. У різних дослідницьких роботах [22]

використовувалася комбінація PCA з різними класифікаторами для виявлення аномалій у мережах IoT.

Таксономія методів, заснованих на ML, для безпеки систем IoT

Таблиця 1.5

Метод ML	**Плюси**	**Мінуси**
KB	- потрібно дуже мало зразків - може класифікуватися як у двійковій, так і в багатозначній класифікації -показує стійкість до нерелевантних особливостей.	- не враховує взаємозалежності між ознаками для цілей класифікації, що впливає на його точність
KNN	- простий у використанні	- визначення оптимального значення K та виявлення відсутніх вузлів є складним завданням
DT	-легкий і простий у використанні метод	- вимагає більшого зберігання - обчислювально складний
SVM	- масштабовані завдяки простоті і здатні виконувати завдання виявлення вторгнень на основі аномалій - вважаються придатними для даних, що містять велику кількість атрибутів ознак - використовують менше пам’яті	- використання оптимальної функції ядра у SVM - важко зрозуміти та інтерпретувати моделі, засновані на SVM
EL	- надійний до перенавчання - виконує більше, ніж одиничний класифікатор - зменшує дисперсію	- збільшена ресурсоємкість за рахунок використання кількох класифікаторів паралельно
RF	- дає більш надійний і точний результат, який стійкий до перенавчання - вимагає значно меншої кількості вхідних даних і не вимагає процесу вибору особливостей	- оскільки RF створює кілька дерев рішень, його використання може бути недоцільним у програмах реального часу, що вимагають великих наборів даних
K-Means	- кластеризація не вимагає мічених даних	- менш ефективний у порівнянні з технікою навчання з вчителем, зокрема виявляючи відомі атаки
PCA	- підходить там, де набір даних включає великий набір змінних, оскільки PCA перетворює його на зменшений набір функцій, не втрачаючи багато інформації - може зменшити складність даних	- метод не виявляє аномалій - він повинен використовуватися з деякими іншими методами ML для розробки моделі

1.3 Формалізована постановка задачі

Головне завдання цієї роботи це побудувати модель і оптимізувати її параметри для максимізації за тестовою вибіркою критерія валідації.

Для вирішення даного завдання у роботі пропонується вирішити ряд задач:

- проаналізувати сучасні моделі і методи маштинного навчання для захисту та розпізнования атак на інтернет речей (IoT);
- сформувати ознаковий опис початкової вибірки з використанням препроцесінгу даних;
- запропонувати моделі класифікаційного аналізу даних;
- релізувати програмне рішення для обробки набору данних, нормалізації, оптимізації, навчання та тестування;
- виконати обробку вхідного набору данних, нормалізацію, оптимізацію, навчання моделі та тестування;
- зробити висновки.

Для цього нам потрібно виконати такі етапи:

- підібрати достовірний набір данних;
- побудувати модель данних;
- проаналізувати ти оптимізувати (нормалізувати) вхідні дані;
- визначити критеріі валідаціі;
- валідувати модель на тестовому наборі даних.

В нашому пипадку корисні дані ми можемо отримати аналізуючи пакети данних та потоки обміну даних для мереж IoT. Спочатку потрібно отримати такі дані у вихідному форматі захоплення (pcap файли), потім ці дані потрібно екстрактувати у більш зручний формат для аналізу та обробки (csv файли).

Для роботи нам потрібні справжні дані, що містять як різноманітний сучасний звичайний мережевий трафік, так і різноманітні сценарії вторгнень із поглибленою структурованою інформацією про трафік мережі. Тому для набору данних було використано достовірну збірку з потрібним нам складом показників.

Набір даних IoT Botnet можна отримати з [37]. Критерії валідації визначають мережевий трафік як звичайний потік або аномальний потік, тоді як критерії типу категорії класифікують мережевий трафік як Звичайний, DDoS, DoS, розвідка або крадіжка. Критерій підкатегорії класифікує та визначає мережевий трафік як Звичайний, DDoS-HTTP, DDoS-TCP, DDoS-UDP, DoS-HTTP, DoS-TCP, DoS-TCP, OS-Fingerprint, Service-Scan, Keylogging та вилучення даних (Data-Exfiltration).

В роботі буде використано Метод головних компонент (Principle Component Analysis) для зменшення великого набору ознак. Оскільки PCA перетворює його на зменшений набір, це також дозволить швидше його опрацьовувати.

Максимізація швидкості та точності виявлення шляхом мінімізації показників середньоквадратичної похибки та перехресної перевірки є основною метою пропонованої нами моделі системи виявлення вторгнень. Тому для цього для порівняння будемо використовувати 2 моделі.

Виходячи з літературного огдяду, в роботі будуть розглянуті найбільш перспективні моделі для цієї задачі: RandomForestClassifier та GradientBoostingClassifier (як представник Ансамблевого навчання), для яких використаємо оптимизацію гіперпараметрів із використанням пакету RandomSearchCV.

РОЗДІЛ 2
ОПИС ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ЕКСТРАКЦІЇ МЕТАДАНИХ

2.1 Моделі та методи екстракції ознакового опису мережевих атак на мережі інтернету-речей

Кібератаки зростають із кожним днем, і їх ефект стає все більш руйнівним. Система виявлення вторгнень покращує кібербезпеку, контролюючи мережевий трафік на наявність ненормальних шаблонів. В результаті Система виявлення вторгнень (IDS) стала критичним аспектом захисту мереж IoT.

IDS – це апаратний пристрій або програмне забезпечення, яке контролює систему чи мережу на предмет зловмисних дій або порушень політики. IDS може бути корисним лише в тому випадку, якщо він генерує своєчасні, точні попередження та надає корисну та ефективну інформацію. Засновані на аномалії та зловживані – це, як правило, цілеспрямовані та мотивовані методи знаходження в зоні виявлення вторгнень.

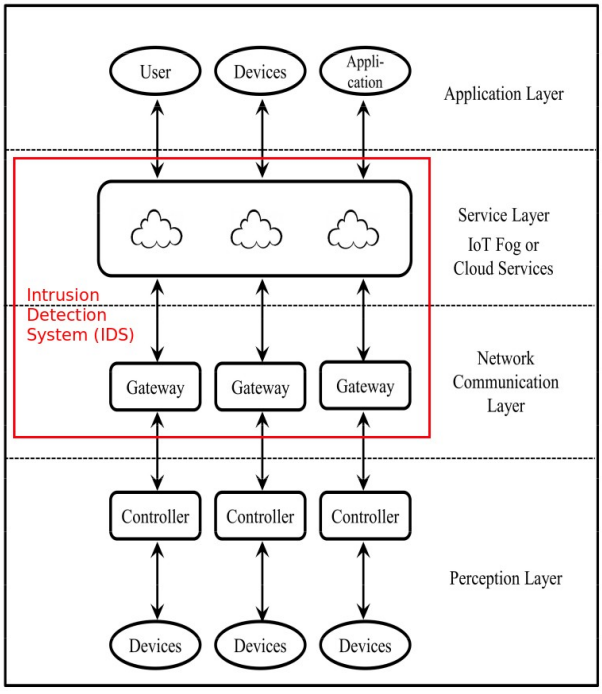

Рисунок 2.1 – Схематичне розміщення IDS [3]

В якості програми перехоплення трафіку був обраний Wireshark. Це, мабуть, один з найкращих інтрументів для аналізу мережі. Програма має відкритий код, доступна для користування широкому колу людей та дозволяє вивчати дані у режимі реального часу.

Wireshark генерує двонаправлені потоки (Biflow), де перший пакет визначає прямий (джерело до місця призначення) і зворотній (місце призначення до джерела) напрямки. Звідси виходять 84 статистичні характеристики, такі як тривалість, кількість пакетів, кількість байтів, довжина пакетів, і т.ін., що розраховуються окремо в прямому і зворотному напрямках. Вихідними даними додатка є формат файлу CSV з шістьма стовпчиками, поміченими для кожного потоку, а саме FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort і Protocol з більш ніж 80 ознаками мережевого трафіку.

Крім того Wireshark можна використовувати для аналізу пакетів, якщо вам доводиться адмініструвати комп’ютерні мережі. Він має доступ до всієї мережі, що підключено до роутера.

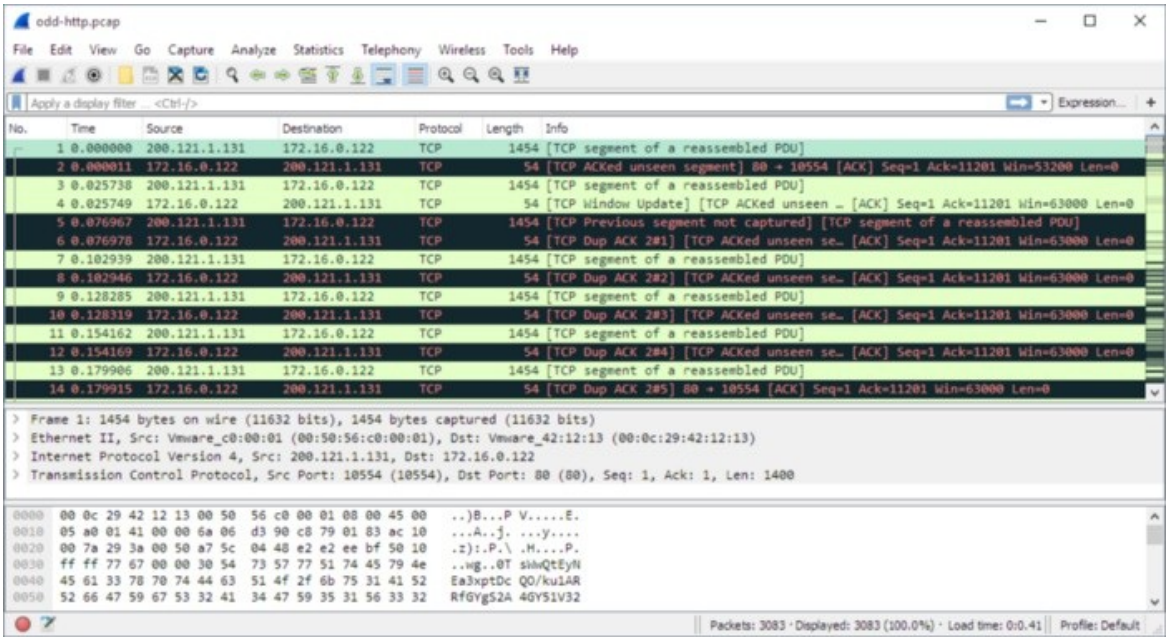

Рисунок 2.2 – Інтерфейс програми Wireshark

Якщо з’єднання підтримують https протокол, то можливі спроби розшифрувати дані не будуть результативними. Але все одно це можливість

захвату пакетів даних в режимі реального часу та подальший їх аналіз на пошук вразливостей.

Щоб проаналізувати мережевий потік необхідно мати достатню кількість ознак. Рішенням є обчислення нових ознак на основі головних ознак потоку. Інтервал передачі потоку серйозно порушує роботу системи виявлення вторгнень, і механізм виявлення не зможе виділити короткочасні атаки, якщо інтервал передачі потоку триває довше.

Екстракційні набори метаданих (DataSet).

Отримання даних та ML спрямовані на методи виявлення зловживань порівняно з методами, що засновані на аномаліях. Саме тому аномалії дуже розповсюджені як методи класифікації та кластеризації в мережевій безпеці.

Основною проблемою для таких підходів вважається незріла технологія в бізнес-інструментах. Система виявлення вторгнень (IDS) – це здатність виявляти нові атаки на основі раніше відміченого майнінгу та машинного навчання – головних методів класифікації та кластеризації в мережі.

Для навчання моделі нейронної мережі для системи виявлення атак на мережеві ресурси буде використовуватися підхід навчання з вчителем, що буде проводитися на наборі даних IoTID20 [54].

Набір даних IoTID20 містить як нормальні, так самі сучасні поширені атаки, максимально реальні дані. Він також включає в себе результати аналізу мережевого трафіку з використанням Wireshark із маркованими потоками на основі тимчасової мітки, IP-адреси джерела і призначення, портів джерела і призначення, протоколів і атак (CSV-файли).

Набір даних ботнету IoT має більш комплексні ознаки на основі мережі та потоків. Ознаку на основі потоків можна використовувати для аналізу та оцінки системи виявлення вторгнень на основі потоків. Цей набір даних стане відправною точкою для виявлення аномальної активності в мережах IoT.

Вибірка даних містить дані, що були зібрані на основі попредніх атак на IoT пристрої та дані по штатній роботі за допомогою моделювання. Це

дозволить побачити не лише факт атаки на пристрій, але і її тип, що допоможе вибрати вірну стретегію для боротьби або пом’якшення.

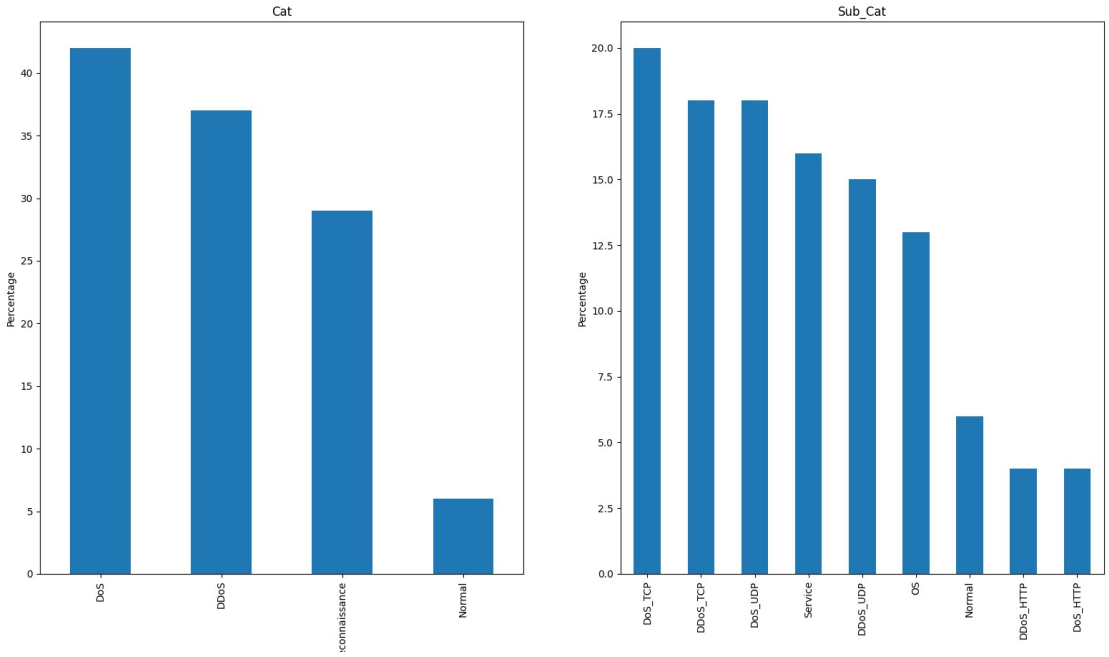

Рисунок 2.3 – Співвідношення кількості зразків за категоріями

Дані містять більше 80 параметрів, які задають вхідні цільові ознаки, що будуть використані у роботі. Опис параметрів, що містяться в датасетах приведено у Додатку А.

Вибір ознак.

Кожна ознака може бути охарактеризована певною величиною: число, строка, булева перемінна – яка виступає в ролі параметру і визначає цільовий маркер. Наприклад, за зростом людини можливо спрогнозувати розмір його ноги, а за країною проживання – расу та колір шкіри.

З одного боку, чим більше ознак, тим точніше ми можемо спрогнозувати значення маркеру, виходячи з існуючих або вихідних даних. Але з іншого боку із збільшенням розмірності ми отримуємо експонційний ріст кількості даних, що тягне за собою збільшення необхідних обчислювальних ресурсів для обробки. Ця закономірність має назву «прокляття розмірності».

Для компенсування та оптимізації ресурсів потрібно використовувати вибір ознак для підвищення продуктивності класифікаторів. На цьому етапі слід визначити найбільш визначальні ознаки, використання яких дозволить значно покращити продуктивність класифікаторів.

Спочатку потрібно буде перевірити та виключити з вибірки такі дані, як синтетичі ідентифікатори кожного потоку та мітки часу пакетів. В результаті цієї роботи нам не потрібно ідентифікувати тип та різновид самої анормальної активності, а важливо її деактивувати, тому такі характристики підлягають видаленню. Список видалених категорій: Flow_ID, Timestamp, Cat, Sub_Cat.

Екстраційний набір даних містить велику кількість ознак і ми зі стовідсотковою точністю не можемо передбачити вплив кожної на цільовий маркер. Існує багато методів для визначення та підбору методів, але для роботи, в якості методів, було обрано алгоритм машинного навчання RandomForestClassifier та показник feature_importances для початкової оцінки впливу на результат навчання.

Вилучення ознак.

Вилучення ознак, на відміну від вибору ознак, необхідне для перетворення невалідних даних для машинного навчання у валідний критерій. В робоі ми взяли за основу тип даних і в залежності від цього були виявлені категорії, що відрізняються від числових значень. Це Src IP, Dst IP, що знаходяться в строковому вигляді.

IP-адреси є важливою ознакою для деактивування атак, бо всі вузли інтренету характеризуються цим параметром і явно вказують на «товариша» або «ворога» у тестовій виборці. IP-адреси представлені у вигляді октету із чисел від 0 до 255 і тому легко можуть бути переведені в числову форму, наприклад, двійкову, десяткову та шістнадцяткову системи обчислення. Ми будемо використовувати десяткову.

Дуже часто можна зустріти зв’язок певних IP адрес з географічними даними. Наприклад, можна отримати такі дані як місцезнаходження (країна,

місто тощо), дані провайдера і т.ін. Ці дані в значній мірі покращать якість моделі та зменшать кількість помилкових спрацювань. Проте є великий мінус в тому, що нам потрбіно звертатится до зовнішніх служб щоб отримати розширену інформацію. Більше того, користування цими сервісами часто платне та лімітоване у кількості запросів, а це може негативно вплинути на швидкість підготовки даних для навчання, а також на великих об’ємах даних.

Набір ознак.

Таким чином ми визначили корисні відібрані ознаки (Таблиця 2.1)

Відібрані ознаки з IoT Botnet dataset

Таблиця 2.1.

Назва ознаки	**Назва ознаки**
Src_IP	Subflow_Fwd_Byts
Dst_IP	TotLen_Bwd_Pkts
Src_Port	Bwd_IAT_Std
ACK_Flag_Cnt	Fwd_Act_Data_Pkts
Tot_Bwd_Pkts	RST_Flag_Cnt
Fwd_Pkt_Len_Mean	Pkt_Len_Mean
Down/Up_Ratio	Dst_Port
Subflow_Bwd_Pkts	Bwd_IAT_Mean.1
Bwd_Header_Len	Tot_Fwd_Pkts
Bwd_IAT_Max	Subflow_Fwd_Pkts

2.2 Алгоритм машинного навчання для розпізнавання кібер-атак на мережі інтернету речей

Випадковий ліс (Random Forest, RF).

Випадковий ліс – це керований алгоритм машинного навчання класифікації, який використовує ансамблевий метод. Простіше кажучи, випадковий ліс складається з численних дерев рішень і допомагає вирішити проблему перенавчання дерев рішень. Ці дерева рішень будуються випадковим чином шляхом вибору випадкових ознак із заданого набору даних.

Випадковий ліс приймає рішення або прогноз на основі максимальної кількості голосів, отриманих від дерев рішень. Результат, до якого приходять протягом максимальної кількості разів через численні дерева рішень, випадковим лісом вважається кінцевим результатом.

Випадкові ліси засновані на техніках ансамблевого навчання. Ансамбль просто означає групу або колекцію, яка в даному випадку являє собою сукупність дерев рішень, які разом називаються випадковим лісом. Точність ансамблевих моделей краща, ніж точність окремих моделей, завдяки тому, що вона збирає результати з окремих моделей і забезпечує кінцевий результат.

Ознаки вибираються випадково з використанням методу агрегування або завантаження. З набору ознак, доступних у наборі даних, створюється ряд навчальних підмножин шляхом вибору випадкових ознак із заміною. Це означає, що одна ознака може повторюватися одночасно в різних підгрупах тренувань.

Потім кожне дерево дає свій голос, тобто окремий результат. Вихідні дані, які отримують максимальну кількість голосів, будуть вибрані випадковим лісом як остаточний результат. У регресійному випадку буде використовуватися середнє значення всіх голосів.

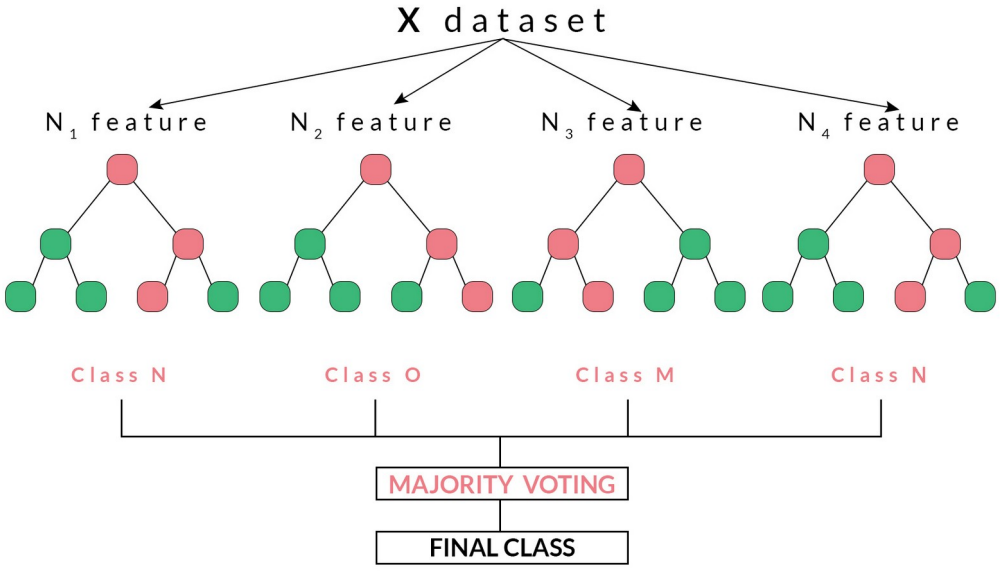

Рисунок 2.4 – Спрощена схема побудови Випадкового лісу [3]

Переваги:

- уникає перенавчання;
- може використовуватися як для класифікації, так і для регресії;
- може обробляти відсутні значення.

Недоліки:

- мелика кількість дерев може займати простір динамічної пам'ятті.

Градієнтний бустінг (Gradient boosting).

Бустінг є різновидом жадібних алгоритмів побудови композиції. Маючи безліч відносно слабких алгоритмів навчання, можно лекго побудувати їх хорошу лінійну комбінацію. Навчання базових алгоритмів відбувається ітеративно і кожен наступний алгоритм компенсує недоліки всіх попередніх.

Градієнтний бустінг – це техніка машинного навчання для вирішення завдань регресії та класифікації, яка зазвичай створює модель прогнозування у вигляді ансамблю слабких моделей прогнозування, а саме – дерева рішень. Він будує модель поетапно, дозволяючи оптимізувати довільну диференційовану функцію втрат.

На прикладі бустінга стає зрозуміло, що гарною якістю можуть володіти як завгодно складні композиції класифікаторів. Це розвіяло існуюче довгий час уявлення про те, що для підвищення узагальнюючої здатності необхідно обмежувати фскладність алгоритмів.

Дійсно, посилення градієнта для відбору використовує градієнтний спуск: на кожній ітерації ми збільшуємо ваги для слабкого учня до протилежного градієнту. По-перше, можна написати теоретичний процес градієнтного спуску над моделлю ансамблю:

$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.)$$

де c_l - коефіцієнти;

w_l - слабкі учні.

Знайти оптимальну модель за цієї формулою з першого разу занадто складно, і тому необхідний ітераційний підхід. Головна відмінність полягає у визначені послідовного процесу оптимізації. На кожній ітерації ми пристосовуємо слабкого учня до протилежного градієнту поточної помилки підгонки. Можна написати теоретичний процес градієнтного спуску над моделлю ансамблю

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

де E (.) - похибка підгонки даної моделі,

c_l - коефіцієнт, що відповідає розміру кроку та

$$-\nabla_{s_{l-1}} E(s_{l-1})(.)$$

є протилежною градієнту помилки щодо моделі ансамблю на кроці l-1. Ця (досить абстрактна) протилежність градієнту – це функція, яка на практиці може бути оцінена лише для спостережень у навчальному наборі даних (для

яких ми знаємо вхідні та вихідні дані). Ці оцінки називаються псевдо-залишками, доданими до кожного спостереження.

Отже, припустимо, що ми хочемо використовувати техніку посилення градієнта з даною сім'єю слабких моделей. На самому початку алгоритму (перша модель послідовності) псевдо-залишки встановлюються рівними значенням спостереження. Потім повторюємо L разів (для L моделей послідовності) наступні кроки:

- підібрати найкращого з можливих слабких учнів до псевдо-залишків;
- обчислити значення оптимального розміру кроку, що визначає, наскільки ми оновлюємо модель ансамблю в напрямку нового слабкого учня;
- оновити модель ансамблю, додавши нового слабкого учня, помноженого на розмір кроку (зробіть крок градієнтного спуску);
- обчислити нові псевдо-залишки, які вказують для кожного спостереження, в якому напрямку ми хотіли б оновити наступні передбачення моделі ансамблю.

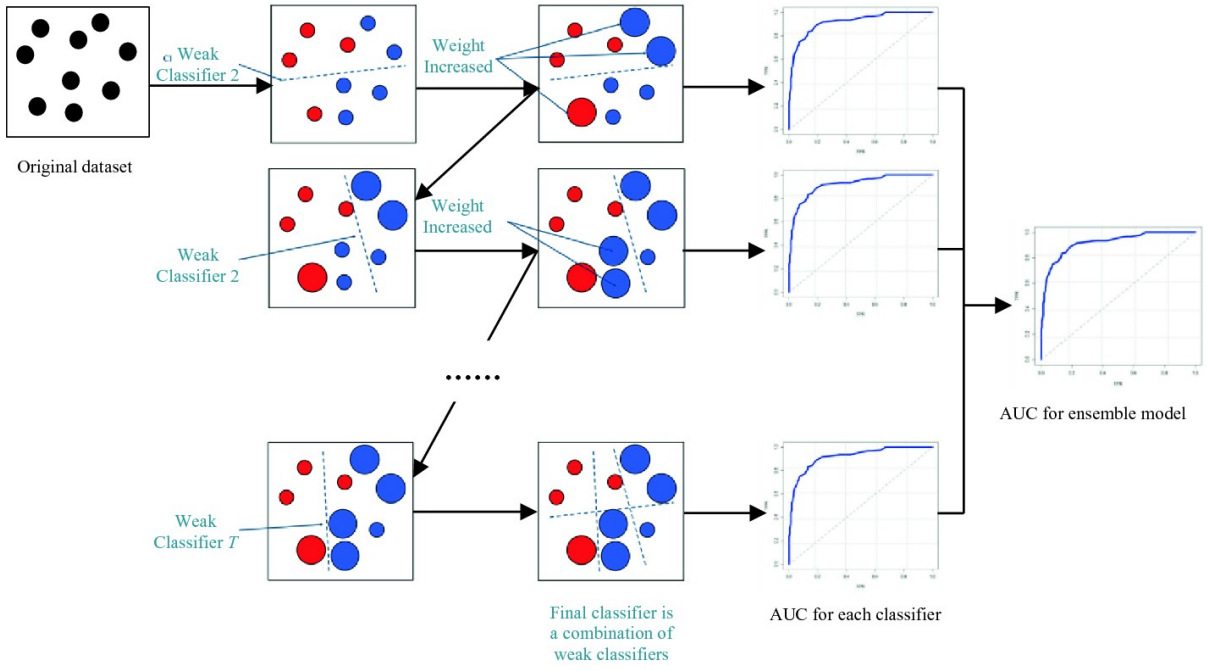

Рисунок 2.5 – Спрощена схема навчання Градієнтного бустінгу

Переваги:

- легкий для читання та інтерпретації передбачень;
- ефективне прогнозування;
- добре справляється з перенавчанням.

Недоліки:

- досить чутливий до шуму та «забруднень»;
- результат не можна масштабувати, бо коректність засновується на предикторах.

Метод головних компонент (Principal component analysis, PCA).

Цей метод намагається пояснити кореляційну структуру набору змінних-предикторів, використовуючи менший набір лінійних комбінацій цих змінних, які називаються компонентами. Зменшення лінійної розмірності виконується з використанням декомпозиції одиничного значення даних для проектування їх на нижчий розмірний простір. Важливо розуміти, що вхідні дані центруються, але не масштабуються для кожної ознаки.

Враховуючи набір даних з m змінними, для його представлення можна використовувати набір k лінійних комбінацій (це означає, що k містить майже стільки інформації, скільки m змінних), також $k << m$.

Переваги:

- зменшення обсягу даних;
- дозволяє оцінювати ймовірності у високовимірних даних;
- він відображає набір компонентів, які не пов'язані між собою.

Недоліки:

- він потребує високого обчислювального ресурсу, тому його не можна застосовувати до дуже великих наборів даних;
- незалежні змінні стають менш зрозумілими, втрачаються оригінальні показники ознак.

2.3 Критерій валідації алгоритмів машинного навчання

Після навчання моделі нам потрібно оцініти, наскільки точно класифікатор може передбачити цільовій результат: нормальна чи анормальна мережева активність, базуючись на відібраних ознаках. Також ми використовували різні алгоритми машинного навчання, з разноманітною підготовкою даних та гіперпараметрів. Саме завдяки критеріям валідації ми зможемо порівняти разні алгоритми навчання та моделі даних.

Як основа використовується Позитивно/Негативна техніка валідації:

- Positive (P) - Позитивні кортежі
- Negative (N) - Негативні кортежі
- True positive (TP) - Справжні позитиви: це позитивні кортежі, які правильно було позначено класифікатором
- True nagative (TN) - Справжні негативи: це негативні кортежі, які були правильно позначені класифікатором
- False positive (FP) - Хибні позитиви: це негативні кортежі, які неправильно було позначено позитивними
- False negative (FN) - Хибні негативи: це позитивні кортежі, які були неправильно позначені як негативні

Ці значення підсумовуються у матриці плутанини (табл. 2.2). Вона підсумовує, наскільки класифікатор може розпізнати класи даних.

Матриця плутанини

Таблиця 2.2

	Прогнозування «так» (Predicted Positives)	**Прогнозування «ні» (Predicted Negatives)**
Так (Positives)	TP	FN
Ні (Negatives)	FP	TN

У таблиці 2.3 описані різні допоміжні показники оцінки класифікатора при прогнозуванні.

Міри оцінки

Таблиця 2.3

Міра	**Формула**	**Визначення**
Точність (Accuracy)	$\frac{TP+TN}{P+N}$	Міра розпізнавання до загальної кількості.
Коефіцієнт помилок	$\frac{FP+FN}{P+N}$	Похибка просто 1 – точність.
Повнота (чутливість, Recall)	$\frac{TP}{P}$	Це позначається як справжня позитивна ставка. Це міра повноти (тобто, який відсоток позитивних кортежів позначений як такий).
Специфічність	$\frac{TN}{N}$	Це позначається як справжня негативна ставка.
Точність (Precision)	$\frac{TP}{TP+FP}$	Це розглядається як міра точності (тобто, який відсоток кортежів, позначених позитивом, насправді такий)
F-міра, гармонійне середнє значення точності повноти	$\frac{2\times точність\times повнота}{точність+повнота}$	Це альтернативний спосіб використання точності та повноти, поєднуючи їх разом.
F , де β – негативне дійсне число	$\frac{(1+\beta^2)\times точність\times повнота}{\beta^2\times точність+повнота}$	Це зважений показник точності та повноти

Середньоквадратична похибка (Mean squared error, MSE).

Середньоквадратична похибка регресії є числом, обчисленим як сума квадратів обчислених залишків, а не невидимих похибок. Якщо цю суму квадратів поділити на n (кількість спостережень), то результатом буде середнє

значення квадратичних залишків. MSE – це функція ризику, що оголошує випадковість.

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

де $\hat{y}_i$ - передбачуване значення і-го зразка,

y_i - відповідне справжнє значення, тоді середня квадратична помилка (MSE), оцінена за n_{samples}.

Перехресна перевірка (Cross-validation, CV).

Вивчення параметрів функції прогнозування і тестування її на одних і тих же даних є методологічною помилкою. Модель, яка буде просто повторювати мітки зразків, які вона щойно побачила, матиме ідеальну оцінку, але поки не зможе передбачити невидимі дані. Така ситуація називається перенавчанням. Щоб цього уникнути, при проведенні експерименту з машинним навчанням зазвичай використовується частина наявних даних у вигляді набору тестів.

Але існує вірогідність, що якась частина даних буде використовуватися тільки як тренувальній набір і ніколи не буде протестований. Щоб вирішити цю проблему, ще одна частина набору даних може бути представлена як так званий «набір перевірки»: навчання триває на навчальному наборі, після чого виконується оцінка на перевірочному наборі, і коли експеримент здається успішним, остаточна оцінка може бути зроблена на тестовому наборі.

Вирішенням цієї проблеми є процедура перехресної перевірки (скорочено CV). Набір тестів все ще повинен бути наданий для остаточної оцінки, але набір для перевірки більше не потрібен при виконанні CV. У базовому підході, званому k- кратним CV, навчальний набір розбивається на k менших наборів (інші підходи описані нижче, але зазвичай відповідають тим же принципам).

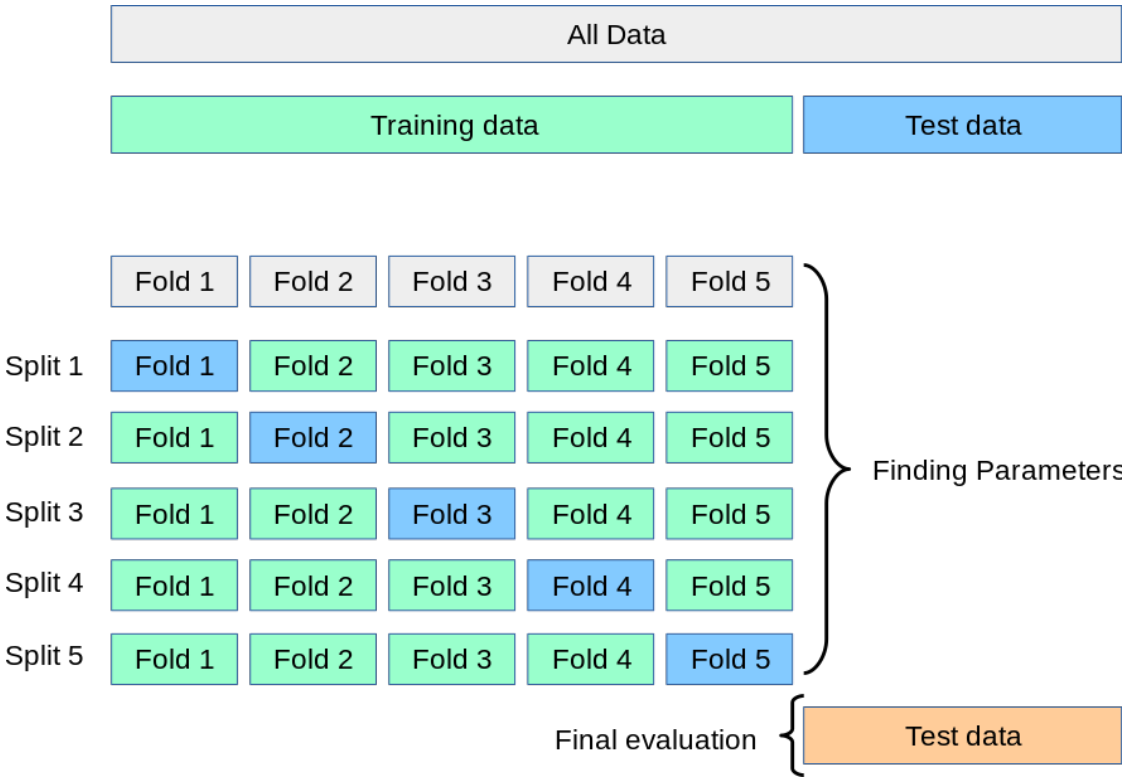

Рисунок 2.6 – Схема виконання перехресної перевірки

РОЗДІЛ 3
РЕАЛІЗАІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАХИСТУ ВІД КІБЕР-АТАК НА МЕРЕЖУ ІНТЕРНЕТУ РЕЧЕЙ

3.1 Формування навчальних та тестових вибірок

Навчання моделі буде проводитися на наборі даних IoTID20 [54], який представлений у виглялі csv файлу з даними (кома виступає в ролі делімітера), де кожна нова строка – це новий набір данних. Файл можна відкрити для перегляду простим текстовим редактором.

Файл містить 1940389 записів та 86 колонок із різним типами данних, що ускладнює спроби обробки масиву даних вручну чи програмами обробки таблиць.

Графіки на рисунку 3.1 продемонструють кількість данних за категоріями, які знаходяться в наборі. У таблиці 3.1 знаходяться кількісні характеристики кожної ознаки

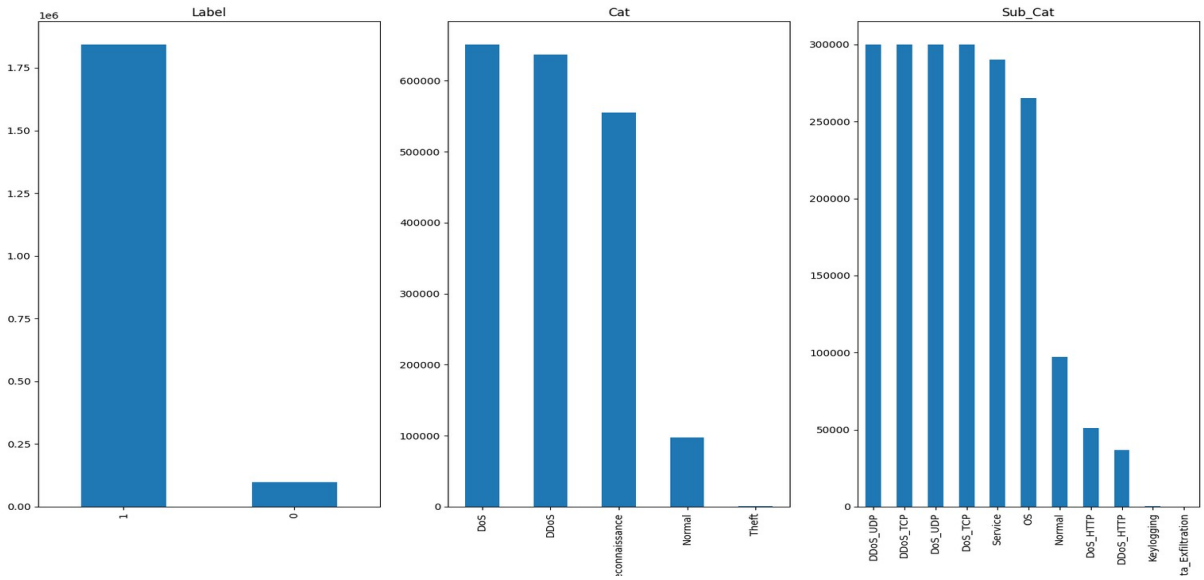

Рисунок 3.1 – Співвідношення кількості зразків за категоріями

Таблиця 3.1

Label	**Кількість**	**Category**	**Кількість**	**Subcategory**	**Кількість**
1 (Anormal)	1843192	DoS	651122	DoS_UDP	300000
				DoS_TCP	300000
				DoS_HTTP	51122
		DDoS	636539	DDoS_UDP	300000
				DDoS_TCP	300000
				DDoS_HTTP	36539
		Reconnaissance	555011	Service	289990
				OS	265021
		Theft	520	Keylogging	384
				Data_Exfiltration	136
2 (Normal)	97197	Normal	97197	Normal	97197

У додатку Б можна побачити приклад запису з полями та типом данних pandas датафрейму.

Вибір ознак

Згідно з основним завданням цієї роботи, нам потрібно классифікувати лише за Label ознакою, тобто визначити тільки ознаку ненормальної поведінки (Label=1) та нормальної (Label=0). Тому ми повинні позбудися явних ознак “Cat” та “Sub_Cat”.

На додачу ми маємо першу колонку із синтетичним ідентифікатором потоку, що в собі містить конкатеновані параметри з інших ознак, і тому, “Flow_ID” також повинен буди видалений.

Мітка часу фісксує час виконання потіку, та не носить системний характер. Вилучаємо і “Timestamp” ознаку.

Таким тином ми вибрали 81 ознаку, що залишились, але і це забагато. Потрібно буде затратити значні ресурси для обробки цих даних. Однак ми поки

залишаємося з такою їх кількістю, бо подальше виключення їх, без додаткового аналізу, може значно зашкодити разрахунку вірного результату.

Вилучення ознак

На відміну від вибору ознак, аналізуємо типи данних, які потрібні буде привести к числовому формату для подальшої обробки та масштабування.

IP адрес має об’єктиний тип данних, трансформуємо його в десяткове число, використовуючи бібліотеку ipaddress зі зміною типу даних.

Таблиця 3.2

Ознака	**Попередній запис (приклад)**	**Попередній тип даних**	**Змінений запис (приклад)**	**Змінений тип даних**
Src_IP	192.168.100.147	object	3232261267	int64
Dst_IP	192.168.100.3	object	3232261123	int64

Результуючий набір ознак та типи даних

Результуючий набір ознак: Src IP, Src Port, Dst IP, Dst Port, Protocol, Flow Duration, Tot Fwd Pkts, Tot Bwd Pkts, TotLen Fwd Pkts, TotLen Bwd Pkts, Fwd Pkt Len Max, Fwd Pkt Len Min, Fwd Pkt Len Mean, Fwd Pkt Len Std, Bwd Pkt Len Max, Bwd Pkt Len Min, Bwd Pkt Len Mean, Bwd Pkt Len Std, Flow Byts/s, Flow Pkts/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min, Fwd IAT Tot, Fwd IAT Mean, Bwd IAT Mean, Fwd IAT Max, Fwd IAT Min, Bwd IAT Tot, Bwd IAT Mean.1, Bwd IAT Std, Bwd IAT Max, Bwd IAT Min, Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Header Len, Bwd Header Len, Fwd Pkts/s, Bwd Pkts/s, Pkt Len Min, Pkt Len Max, Pkt Len Mean, Pkt Len Std, Pkt Len Var, FIN Flag Cnt, SYN Flag Cnt, RST Flag Cnt, PSH Flag Cnt, ACK Flag Cnt, URG Flag Cnt, CWE Flag Count, ECE Flag Cnt, Down/Up Ratio, Pkt Size Avg, Fwd Seg Size Avg, Bwd Seg Size Avg, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, Bwd Blk Rate Avg, Subflow Fwd Pkts, Subflow Fwd Byts, Subflow Bwd Pkts, Subflow Bwd Byts, Init Fwd Win Byts, Init

Bwd Win Byts, Fwd Act Data Pkts, Fwd Seg Size Min, Active Mean, Active Std, Active Max, Active Min, Idle Mean, Idle Std, Idle Max, Idle Min, Label

Результуючі типи даних: dtypes: float64(45), int64(37)

3.2 Короткий опис програмного забезпечення

Основною мовою програмування для всих створюваних додатків в дипломному проекті обраний Python. Це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Вибір Python, як мови програмування, не випадковий. Ця мова має низький поріг входження, можливість, без особливих проблем, запуститися на будь-який операційній системі (ОС) і, найголовніше, на цій мові написано безліч бібліотек для побудови різних моделей.

NumPy – один із найпопулярніших програмних пакетів Python. Це ефективний та зручний інструмент для роботи із векторними та матричними операціями. Завдяки присутності корисним вбудованим функціям, дозволяє уникнути перевантаження коду. Багато інших бібліотек можуть напряму взаємодіяти з NumPy.

Pandas – не менш відомий пакет для роботи зі структурованими даними, що мають відкритий код та ліцензування BSD. Ця бібліотека – це високопродуктивні, прості у використання структури даних та інструменті аналізу даних.

Pyplot Matplotlib та Seaborn – чудовий набір для візуалізації даних. Методи цих двох бібліотек дозволяють легко сконфігурувати будь-які типи графіків, гістограм і т.ін. Це, мабуть, найкращий спосіб графічного представлення даних. Seaborn є надбудовою над базовою бібліотекою візуалізації Matplotlib.

Scikit-learn – найкращий набір функціоналу для машинного навчання. Бібліотека має великий вибір компонент для навчання із вчителем та без нього.

Не дивлячись на об’ємну та складну предметну частину, він забезпечує зручний, зрозумілий та добре описаний інтерфейс для роботи з бібліотеками та функціями. Ще однією перевагою є наявність великої кількості навчальних посібників та довідкової інформації, в яких використовуються алгоритми класичного машинного мислення. Саме завдяки документації та великій кількості прикладів, ці бібліотеки виокристовуються як професіоналами, так і початківцями, які щойно знайомляться з машинним навчанням.

3.3 Результати машинного навчання системи кібер-захисту мережі інтернету речей

Існує безліч способів для створення класифікаційних моделей, як популярних так і не зовсім. Один з найвідоміших та надійних методів навчання полягає у використанні «випадкового лісу». Також існує відомий набір методів під загальною назвою Бустінг. В рамках цієї работи розглядається його підвид – Градієнтний бустинг.

Для того, щоб спробувати поліпшити показники моделей, використовується оптимізація гіперпараметрів моделі. Крім того, це поширений підхід, відповідно до якого, дані перед передачею до моделі обробляють за допомогою Методу головних компонент (PCA , Principal Component Analysis).

Тому спробуємо порівняти різні підходи та зробити висновки.

Підготовка тестового набору даних

Спочатку розділемо набір тестових даних з використанням Scikit-learn train_test_split функції. Будемо використовувати співвідношення 90% тренувальні та 10% тестові дані. Загалом, таке співвідношення ми можемо забеспечити через великий набір даних, що містить приблизно 2 мільйони строк.

До того як почати, нам потрібно зробити «центрування» та «стандартизацію» даних шляхом їх масштабування, бо значення знаходяться в різних одиницях виміру. Скористаємося бібліотекою StandardScaler з набору sklearn.preprocessing.

Приклад даних після масштабування:

```
[ 0.01870706 -0.91918155  0.03803938 -0.4453907   1.4759318  -0.16810052
 -0.02161741 -0.03118988 -0.02094447 -0.00900395 -0.26596342 -0.33608529
 -0.28783919 -0.2345614  -0.28293491 -0.289167   -0.27244251 -0.31024063
 -0.02567371 -0.12391848 -0.31184741 -0.41532871 -0.40912126 -0.20644786
  0.08598708 -0.11340054 -0.34378906 -0.21462028  0.10604433 -0.30312204
 -0.30134956 -0.20894545 -0.32499942 -0.2366933   0.         -0.04912933
  0.         -0.01675303 -0.03180197 -0.03402693 -0.03148228 -0.13153317
 -0.36218219 -0.27163362 -0.3628692  -0.24399098 -0.00481293 -0.0729089
 -1.34027067 -0.08411431 -0.04912933 -0.26753866 -0.01675303 -0.00945183
 -0.00951223 -0.5869948  -0.42484455 -0.28783919 -0.27244251  0.
  0.          0.          0.          0.          0.         -0.02161741
 -0.02094447 -0.03118988 -0.00900395  0.         -0.42371306 -0.04701692
  0.         -0.20445752 -0.08733658 -0.21027013 -0.19313727 -0.42423479
 -0.33322529 -0.44800919 -0.39637378]
```

Використання Random Foresf моделі

Використаємо базову модель RandomForestClassifier з бібліотеки sklearn.ensemble з мінімально сконфігурованими гіперпараметрами (max_depth=2) та визначимо валідаційні показники.

Показник 1. Середньоквадратична похибка (Mean squared error, MSE) передбачення для тренувального набору даних. Позначимо цей параметр як MSE_TRAIN.

Показник 2. Перехресна перевірка (Cross-validation, CV) тренувального набору (cv=3, scoring='neg_mean_squared_error'). Позначимо цей параметр як CV_SCORE.

Показник 3. Середньоквадратична похибка (Mean squared error, MSE) передбачення для тестового набору даних. Позначимо цей параметр як MSE_TEST.

Результат базової RF моделі:

- MSE_TRAIN: 0.009
- CV_SCORE: [-0.00937956 -0.00927992 -0.00962695]
- MSE_TEST: 0.009

Визначимо головні ознаки:

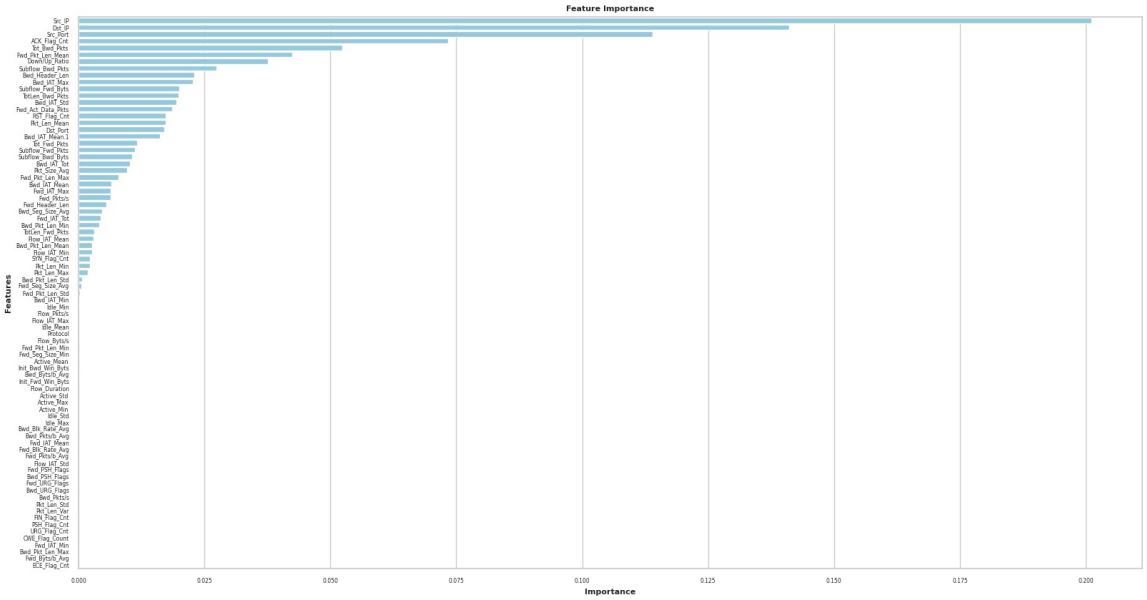

Рисунок 3.2 – Головні ознаки базової RF моделі

Візуалізація «важливості» ознак

Таблиця 3.3

Features	**Gini-Importance**	**Features**	**Gini-Importance**
Src_IP	0.2	Subflow_Fwd_Byts	0.02
Dst_IP	0.14	TotLen_Bwd_Pkts	0.02
Src_Port	0.11	Bwd_IAT_Std	0.02
ACK_Flag_Cnt	0.07	Fwd_Act_Data_Pkts	0.02
Tot_Bwd_Pkts	0.05	RST_Flag_Cnt	0.02
Fwd_Pkt_Len_Mean	0.04	Pkt_Len_Mean	0.02
Down/Up_Ratio	0.04	Dst_Port	0.02
Subflow_Bwd_Pkts	0.03	Bwd_IAT_Mean.1	0.02
Bwd_Header_Len	0.02	Tot_Fwd_Pkts	0.01
Bwd_IAT_Max	0.02	Subflow_Fwd_Pkts	0.01

Спробуєм покращити базову модель за допомогою метода головних компонент.

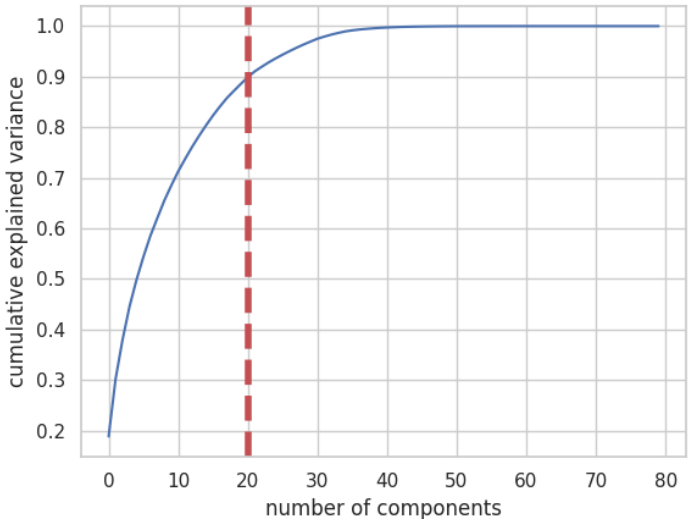

Рисунок 3.3 – Дисперсія впливу ознак для базової RF моделі

Після того, як число компонент, що використовуються, перевищує 20, зростання їх кількості не надто сильно підвищує дисперсію. Якщо поглянути на вищенаведений графік, то виявиться, що використання PCA для переходу від 80 змінних до 20 компонентів дозволяє пояснити 90% дисперсії даних. Інші 60 компонент пояснюють менше 10% дисперсії, а це значить, що від них ми можемо відмовитися. За такою логікою, скористаємося PCA для зменшення числа компонент з 80 до 20 для X_train і X_test та розрахуємо валідаційні показники.

Результат RF+PCA моделі:

- MSE_TRAIN: 0.010
- CV_SCORE: [-0.00979013 -0.01030377 -0.01054601]
- MSE_TEST: 0.010

Спробуємо оптимізувати гіперпараметри з використанням RandomizedSearchCV. Задаємо змінними такі параметри: n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, bootstrap.

Визначення параметрів виконувалося близько 10 годин. Результат гіперпараметрів: {'n_estimators': 400, 'min_samples_split': 23, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 14, 'bootstrap': False}

Результат RF+PCA+RandomizedSearchCV моделі:

- MSE_TRAIN: 0.012
- CV_SCORE: [-0.01030013 -0.01138970 -0.01145561]
- MSE_TEST: 0.012

Використання Gradient Boosting (GB) моделі

Використаємо базову модель GradientBoostingClassifier з бібліотеки sklearn.ensemble з мінімально сконфігурованими гіперпараметрами (n_estimators=10, tol=0.01) та визначемо валідаційні показники.

Результат базової GB моделі:

- MSE_TRAIN: 0.000
- CV_SCORE: [-0.00000000e+00 -1.71786771e-06 -0.00000000e+00]
- MSE_TEST: 0.000

Результат базової GB+PCA моделі:

- MSE_TRAIN: 0.006
- CV_SCORE: [-0.00553669 -0.0056432 -0.00586996]
- MSE_TEST: 0.006

Результат базової GB+PCA+RandomizedSearchCV моделі:

- Час виконання більше 12 годин

Загальний висновок ми можемо зробити проаналізувавши валідаційні характеристики для моделей (Середньоквадратична похибка навчального та

тестового набору; перехресна перевірка тренувального набору даних) в таблиці 3.4.

Збірна таблиця валідаційних характеристик

Таблиця 3.4

Модель	**Критерій валідації**	**Випадковий ліс (Random Forest)**	**Градієнтний бустінг (Gradient Boosting)**
Базова модель	MSE_TRAIN	0.009	**0.000**
	CV_SCORE (min)	-0.00962695	**-1.71786771e-06**
	MSE_TEST	0.009	**0.000**
Базова модель з методом головних компонентів (PCA)	MSE_TRAIN	0.010	0.006
	CV_SCORE (min)	-0.01054601	-0.00586996
	MSE_TEST	0.010	0.006
Базова модель з методом головних компонентів (PCA) та Randomized Search CV	MSE_TRAIN	0.012	*
	CV_SCORE (min)	-0.01145561	*
	MSE_TEST	0.012	*

Базова модель, в якій використовується метод головних компонент і широкомасштабна оптимізація гіперпараметрів, може працювати не так добре, як звичайна модель зі стандартними налаштуваннями. Також базова модель Градієнтного бустінгу показала кращі результати, ніж Випадковий ліс.

ВИСНОВОК

Інтернет речей (IoT) – це взаємозв'язок різнорідних розумних пристроїв через Інтернет з різноманітними сфери застосування. Звичайних засобів безпеки контролю недостатньо для запобігання численим атакам на ці багаті інформацією пристрої. Разом із вдосконаленням існуючих підходів, периферійна оборона, Система виявлення вторгнень (IDS), виявилася ефективною у більшості сценаріїв. Аналізуючи мережевий трафік, ми можемо розпізнати атакуючі дії зловмисників та вчасно вжити заходи для зменшення шкідливого впливу.

В результаті цієї роботи були розглянуті сучасні та ефективні технології розпізнавання різних видів анормальной активності. Зібрані ознаки таких активностей лягли в основу великого набору даних, які допоможуть навчити штучний інтелект і в подальшому використовувати його.

Було проаналізовано вхідний набір даних, на базі якого був сформований рузультуючий набір ознак, що повністю подготовлений для навчання та тестування моделей. Для классифікації були обрані базові моделі: рандомний ліс та градієнтний бустінг – що добре показали в процесі навчання з такими типами ознак.

Було розроблено програмну реалізацію алгоритму підготовки ознак, а також навчання моделей. Завдяки оптимізації та комбінації методів, валідації та оцінки прогнозів ми змогли сформувати найкращі схеми процессингу ознак та гіперпараметрів класифікатора.

Таким чином, використовуючи найкращу за критеріями модель даних, навчену на наданому наборі даних, ми зможемо вміло та швидко визначити категорію мережевої активності.

Як результат виконання роботи було вирішено ряд завдань, а саме:

- проаналізовано сучасні моделі та методи машинного навчання для захисту та розпізнавання атак на інтернет речей (IoT);
- підібрано найбільш відповідний та повний за ознаками набір даних;
- сформований ознаковий опис начальної вибірки з використанням препроцесінгу даних;
- запропоновано моделі класифікаційного аналізу даних;
- релізовано програмне рішення для обробки набору даних, нормалізації, оптимізації, навчання та тестування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kolias, Constantinos, Angelos Stavrou, and Jeffrey Voas. Securely Making Things Right // Computer 48, no. 9 – 2015.

2. Hilton, Scott, Dyn analysis summary of Friday, October 21 attack 2016 [Електронний ресурс] - https://dyn.com/blog/Dyn-analysis-summary-of-friday-october-21-attack

3. Aliya Tabassum, Wadha Lebda. Security Framerork for IoT Devices against cyber-attacks. Department of Computer Science and Engineering – 2019 - Zurich, Switzerland

4. Tabassum, Aliya, Aiman Erbad, and Mohsen Guizani. A Survey on Recent Approaches in Intrusion Detection System in IoTs. – 15th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE – 2019.

5. Alaba FA, Othman M, Hashem IA, Alotaibi F. Internet of Things security // Journal of Network and Computer Applications – 2017.

6. Madakam S, Ramaswamy R, Tripathi S. Internet of Things (IoT) // Journal of Computer and Communications – 2015.

7. Suo, Hui, Jiafu Wan, Caifeng Zou, and Jianqi Liu. Security in the internet of things: a review. - In 2012 international conference on computer science and electronics engineering - vol. 3 – 2012.

8. Hossain, M. Shamim, Ghulam Muhammad, Sk Md Mizanur Rahman, Wadood Abdul, Abdulhameed Alelaiwi, and Atif Alamri. Toward end-to-end biomet rics-based security for IoT infrastructure. // IEEE Wireless Communications 23 – no. 5 – 2016.

9. Ali, Bako, and Ali Awad. Cyber and physical security vulnerability assessment for IoT-based smart homes // Computer Networks – 2019.

10. Giotis, Kostas, Christos Argyropoulos, Georgios Androulidakis, Dimitrios Kalogeras, and Vasilis Maglaris. Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. // Computer Networks 62 – 2014

11. Le A, Loo J, Lasebae A, Aiash M, Luo Y. A study on QoS security threats and countermeasures using intrusion detection system approach. // International Journal of Communication Systems – 2012 Sep.

12. Fadlullah, Zubair Md, et al. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. // IEEE Communications Surveys & Tutorials 19.4 – 2017.

13. Stergiou, Christos, et al. Secure integration of IoT and cloud computing. // Future Generation Computer Systems 78 – 2018.

14. Engrish, Kishore. Turning internet of things (not) into the internet of vulnerabilities (Nov): It botnets. // arXiv preprint – 2017.

15. Chao Li, Wei Jiang, Xin Zou. Botnet: Survey and Case Study – 4th International Conference on Innovative Computing, Information and Control – 2009

16. Kamal Alieyan, Ammar Almomani, Rosni Abdullah, Badr Almutairi, Mohammad Alauthman. Botnet and Internet of Things (IoT): A Definition, Taxonomy, Challenges, and Future Directions – DOI: 10.4018/978-1-5225-9742-1.ch013 – 2020.

17 Karim, Ahmad, et al. Botnet detection techniques: review, future trends, and issues. // Zhejiang University SCIENCE – 2014.

18. G. Schaffer. Worms and Viruses and Botnets, Oh My: Rational Responses to Emerging Internet Threats - IEEE Security & Privacy – 2006.

19. Rajab, M., Zarfoss, J., Monrose, F., &Terzis. A multifaceted approach to understanding the botnet phenomenon – Retrieved October 31, 2009 [Електронний ресурс] – http://www.imconf.net/imc- 2006/papers/p4-rajab.pdf

20. J. Binkley and S. Singh. An algorithm for anomaly-based botnet detection // In Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI) – 2006.

21. G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting malware infection through ids-driven dialog correlation // In Proceedings of the 16th USENIX Security Symposium – 2007.

22. Javed Asharf, Nour Moustafa, Hasnat Khurshid, Essam Debie , Waqas Haider and Abdul Wahab. Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions – 2020

23. Ng, A.Y.; Jordan, M.I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Advances in Neural Information Processing Systems [Електронний ресурс] – https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf

24. Soucy, P.; Mineau, G.W. A simple KNN algorithm for text categorization. // IEEE International Conference on Data Mining, San Jose, CA, USA – 2001.

25. Kotsiantis, S.B.; Zaharakis, I.; Pintelas, P. Supervised machine learning: A review of classification techniques. Emerg. Artif. Intell. Appl. Comput. Eng – 2007.

26. Kotsiantis, S.B. Decision trees: A recent overview. Artif. Intell. Rev. – 2013.

27. Tong, S.; Koller, D. Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. – 2001.

28. Liu, Y.; Pi, D. A novel kernel svm algorithm with game theory for network intrusion detection. KSII Trans. Internet Inf. Syst – 2017.

29. Hu, W.; Liao, Y.; Vemuri, V.R. Robust Support Vector Machines for Anomaly Detection in Computer Security. ICMLA – 2003.

30. Wagner, C.; François, J.; Engel, T. Machine learning approach for ip-flow record anomaly detection // In International Conference on Research in Networking; Springer: Berlin, Germany – 2011

31. Zhang, H.; Liu, D.; Luo, Y.; Wang, D. Adaptive Dynamic Programming for Control: Algorithms And Stability; Springer Science & Business Media: Berlin, Germany // Electronics – 2020.

32. Bosman, H.H.; Iacca, G.; Tejada, A.; Wörtche, H.J.; Liotta, A. Ensembles of incremental learners to detect anomalies in ad hoc sensor networks. // Ad Hoc Netw – 2015.

33. Zhang, J.; Zulkernine, M. A hybrid network intrusion detection technique using random forests // In Proceedings of the First International Conference on Availability, Reliability and Security (ARES’06), Vienna, Austria – 2006.

34. Doshi, R.; Apthorpe, N.; Feamster, N. Machine learning ddos detection for consumer internet of things devices // In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA – 2018.

35. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network anomaly detection: Methods, systems and tools // IEEE Commun. Surv. Tutor – 2013

36. Muniyandi A.P.; Rajeswari R.; Rajaram R. Network anomaly detection by cascading k-Means clustering and C4. 5 decision tree algorithm // Procedia Eng.- 2012

37. Ullah, I.; Mahmoud, Q.H. IoT-Botnet Dataset 2020 [Електронний ресурс] – https://sites.google.com/view/iotbotnetdatset

ДОДАТКИ

Додаток А. Список ознак з набіру данних

Таблиця 4.1.

Назва параметра	**Значення**
Destination Port	Порт призначення
Flow Duration	Тривалість потоку
Total Fwd Packets	Всього пакетів в прямому напрямку
Total Backward Packets	Всього пакетів в зворотному напрямку
Total Length of Fwd Packets	Загальний розмір пакетів в прямому напрямку
Total Length of Bwd Packets	Загальний розмір пакетів в зворотному напрямку
Fwd Packet Length Max	Максимальний розмір пакета в прямому напрямку
Fwd Packet Length Min	Мінімальний розмір пакета в прямому напрямку
Fwd Packet Length Mean	Середній розмір пакета в прямому напрямку
Fwd Packet Length Std	Розмір стандартного відхилення пакету в прямому напрямку
Bwd Packet Length Max	Максимальний розмір пакета в зворотному напрямку
Bwd Packet Length Min	Мінімальний розмір пакета в зворотному напрямку
Bwd Packet Length Mean	Середній розмір пакета в зворотному напрямку
Bwd Packet Length Std	Розмір стандартного відхилення пакету в зворотному напрямку
Flow Bytes/s	Швидкість потоку в байтах, тобто кількість пакетів, переданих в секунду
Flow Packets/s	Швидкість потоку пакетів, тобто кількість пакетів, переданих в секунду
Flow IAT Mean	Середній час між двома потоками
Flow IAT Std	Стандартне відхилення часу двох потоків
Flow IAT Max	Максимальний час між двома потоками
Flow IAT Min	Мінімальний час між двома потоками
Fwd IAT Total	Загальний час між двома пакетами, відправленими в прямому

	напрямку
Fwd IAT Mean	Середній час між двома пакетами, відправленими в прямому напрямку
Fwd IAT Std	Час стандартного відхилення між двома пакетами, відправленими в прямому напрямку
Fwd IAT Max	Максимальний час між двома пакетами, відправленими в прямому напрямку
Fwd IAT Min	Мінімальний час між двома пакетами, відправленими в прямому напрямку
Bwd IAT Total	Загальний час між двома пакетами, відправленими в зворотному напрямку
Bwd IAT Mean	Середній час між двома пакетами, відправленими в зворотному напрямку
Bwd IAT Std	Час стандартного відхилення між двома пакетами, відправленими в зворотному напрямку
Bwd IAT Max	Максимальний час між двома пакетами, відправленими в зворотному напрямку
Bwd IAT Min	Мінімальний час між двома пакетами, відправленими в зворотному напрямку
Fwd PSH Flags	Кількість разів, коли прапор PSH був встановлений в пакетах, що рухаються в прямому напрямку (0 для UDP)
Bwd PSH Flags	Кількість разів, коли прапор PSH був встановлений в пакетах, що рухаються у зворотному напрямку (0 для UDP)
Fwd URG Flags	Кількість разів, коли прапор URG був встановлений в пакетах, що проходять в прямому напрямку (0 для UDP)
Bwd URG Flags	Скільки разів був встановлений прапор URG в пакетах, що рухаються у зворотному напрямку (0 для UDP)
Fwd Header Length	Всього байт, які використовуються для заголовків в прямому напрямку
Bwd Header Length	Всього байт, які використовуються для заголовків в прямому

	напрямку
Fwd Packets/s	Кількість прямих пакетів в секунду
Bwd Packets/s	Кількість зворотних пакетів в секунду
Min Packet Length	Мінімальна довжина пакета
Max Packet Length	Максимальна довжина пакета
Packet Length Mean	Середня довжина пакета
Packet Length Std	Стандартне відхилення довжини пакета
Packet Length Variance	Мінімальний час прибуття пакета
FIN Flag Count	Кількість пакетів з FIN
SYN Flag Count	Кількість пакетів з SYN
RST Flag Count	Кількість пакетів з RST
PSH Flag Count	Кількість пакетів з PUSH
ACK Flag Count	Кількість пакетів з ACK
URG Flag Count	Кількість пакетів з URG
CWE Flag Count	Кількість пакетів з CWE
ECE Flag Count	Кількість пакетів з ЄЕК
Down/Up Ratio	Коефіцієнт завантаження і вивантаження
Average Packet Size	Середній розмір пакета
Avg Fwd Segment Size	Середній розмір сегмента в прямому напрямку
Avg Bwd Segment Size	Середній розмір сегмента в зворотному напрямку
Fwd Header Length	Розмір заголовка в прямому напрямку
Fwd Avg Bytes/Bulk	Середній обсяг байтів в прямому напрямку
Fwd Avg Packets/Bulk	Середня кількість пакетів в прямому напрямку
Fwd Avg Bulk Rate	Середня кількість насипного курсу в прямому напрямку
Bwd Avg Bytes/Bulk	Середній обсяг байтів в зворотному напрямку
Bwd Avg Packets/Bulk	Середня кількість пакетів в зворотному напрямку

Bwd Avg Bulk Rate	Середня кількість пакетів в зворотному напрямку
Subflow Fwd Packets	Середня кількість пакетів в підпотоків в прямому напрямку
Subflow Fwd Bytes	Середня кількість байтів в підпотоків в прямому напрямку
Subflow Bwd Packets	Середня кількість пакетів в підпотоків в зворотному напрямку
Subflow Bwd Bytes	Середня кількість байтів в підпотоків в зворотному напрямку
Init_Win_bytes_forward	Кількість байтів, відправлених в початковому вікні в прямому напрямку
Init_Win_bytes_backward	Кількість байтів, відправлених в початковому вікні в зворотному напрямку
act_data_pkt_fwd	пакетів з не менше 1 байта корисного навантаження даних TCP в прямому напрямку
min_seg_size_forward	Мінімальний розмір сегмента спостерігається в прямому напрямку
Active Mean	Середній час, коли потік був активний, перш ніж став вільним
Active Std	Стандартне відхилення часу, протягом якого потік був активний до простою
Active Max	Максимальний час, протягом якого потік був активний до простою
Active Min	Мінімальний час активності потоку до простою
Idle Mean	Середній час, коли потік простоював, перш ніж стати активним
Idle Std	Стандартне відхилення часу, протягом якого потік простоював до того, як став активним
Idle Max	Максимальний час простою потоку до його активації
Idle Min	Мінімальний час, протягом якого потік простоював, перш ніж стати активним
Label	Маркування пакета
Cat	Категорія маркеру
Sub_Cat	Субкатегорія маркеру

Додаток Б. Приклад запису з полями та типом даних

Таблиця 4.2.

Ознака	Варіант значення	Тип данних	Ознака	Варіант значення	Тип данних
Flow_ID	192.168.100.147-192.168.100.3-27516-80-6	object	Bwd_Pkts/s	0.08	float64
Src_IP	192.168.100.147	object	Pkt_Len_Min	0	float64
Src_Port	27516	int64	Pkt_Len_Max	100	float64
Dst_IP	192.168.100.3	object	Pkt_Len_Mean	75	float64
Dst_Port	80	int64	Pkt_Len_Std	46.29	float64
Protocol	6	int64	Pkt_Len_Var	2142.86	float64
Timestamp	04/06/18 03:07 AM	object	FIN_Flag_Cnt	0	int64
Flow_Duration	24418968	int64	SYN_Flag_Cnt	1	int64
Tot_Fwd_Pkts	5	int64	RST_Flag_Cnt	0	int64
Tot_Bwd_Pkts	2	int64	PSH_Flag_Cnt	0	int64
TotLen_Fwd_Pkts	400	float64	ACK_Flag_Cnt	0	int64
TotLen_Bwd_Pkts	100	float64	URG_Flag_Cnt	0	int64
Fwd_Pkt_Len_Max	100	float64	CWE_Flag_Count	0	int64
Fwd_Pkt_Len_Min	0	float64	ECE_Flag_Cnt	0	int64
Fwd_Pkt_Len_Mean	80	float64	Down/Up_Ratio	0	float64
Fwd_Pkt_Len_Std	44.72	float64	Pkt_Size_Avg	85.71	float64
Bwd_Pkt_Len_Max	100	float64	Fwd_Seg_Size_Avg	80	float64
Bwd_Pkt_Len_Min	0	float64	Bwd_Seg_Size_Avg	50	float64
Bwd_Pkt_Len_Mean	50	float64	Fwd_Byts/b_Avg	0	int64
Bwd_Pkt_Len_Std	70.71	float64	Fwd_Pkts/b_Avg	0	int64

Flow_Byts/s	20.48	float64	Fwd_Blk_Rate_Avg	0	int64
Flow_Pkts/s	0.29	float64	Bwd_Byts/b_Avg	0	int64
Flow_IAT_Mean	4.07E+06	float64	Bwd_Pkts/b_Avg	0	int64
Flow_IAT_Std	3.06E+06	float64	Bwd_Blk_Rate_Avg	0	int64
Flow_IAT_Max	6.39E+06	float64	Subflow_Fwd_Pkts	5	int64
Flow_IAT_Min	64767	float64	Subflow_Fwd_Byts	400	int64
Fwd_IAT_Tot	1.80E+07	float64	Subflow_Bwd_Pkts	2	int64
Fwd_IAT_Mean	4.51E+06	float64	Subflow_Bwd_Byts	100	int64
Bwd_IAT_Mean	2.84E+06	float64	Init_Fwd_Win_Byts	-1	int64
Fwd_IAT_Max	6.22E+06	float64	Init_Bwd_Win_Byts	29200	int64
Fwd_IAT_Min	254491	float64	Fwd_Act_Data_Pkts	4	int64
Bwd_IAT_Tot	1.85E+07	float64	Fwd_Seg_Size_Min	0	int64
Bwd_IAT_Mean.1	1.85E+07	float64	Active_Mean	254491	float64
Bwd_IAT_Std	0	float64	Active_Std	0	float64
Bwd_IAT_Max	1.85E+07	float64	Active_Max	254491	float64
Bwd_IAT_Min	1.85E+07	float64	Active_Min	254491	float64
Fwd_PSH_Flags	0	int64	Idle_Mean	6.04E+06	float64
Bwd_PSH_Flags	0	int64	Idle_Std	320655	float64
Fwd_URG_Flags	0	int64	Idle_Max	6.39E+06	float64
Bwd_URG_Flags	0	int64	Idle_Min	5.69E+06	float64
Fwd_Header_Len	100	int64	Label	1	int64
Bwd_Header_Len	44	int64	Cat	DDoS	object
Fwd_Pkts/s	0.2	float64	Sub_Cat	DDoS_TCP	object

Додаток В.

Програмна реалізація машинного методу навчання Рандомній ліс (Random Forest, RF).

```
import ipaddress as ipc
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import time
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from IPython.display import display
from sklearn.metrics import recall_score
from sklearn.metrics.regression import mean_squared_error
from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle

display('Reading Data...')
data = pd.read_csv('IoT-BoT.csv')

# Statistics
print(data.tail(1).T)
display(data.info())
print(data.shape)
print(data.describe(include='all').T)
print(data['Label'].value_counts())
fig = plt.figure(figsize=(100, 30))
for i, column in enumerate(['Label','Cat', 'Sub_Cat']):
    ax = fig.add_subplot(1, 3, i + 1)
    ax.set_title(column)
    data[column].value_counts().plot(kind="bar", axes=ax)
plt.subplots_adjust(hspace=0.7, wspace=0.2)
plt.show()

# Preparation
data['Src_IP'] = data['Src_IP'].apply(lambda x: int(ipc.ip_address(x))).astype(int)
data['Dst_IP'] = data['Dst_IP'].apply(lambda x: int(ipc.ip_address(x))).astype(int)
data = data.drop(['Flow_ID', 'Timestamp', 'Cat', 'Sub_Cat'], axis=1)
display(data.info())

# Data test split
X = data.drop('Label', axis=1)
y = data['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size=0.1,
                                    random_state = 2020)
display(X_train.tail(1))
columns = X_train.columns
```

```
# Scaling
display('Scaling ...')
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
y_train = np.array(y_train)
display(X_train_scaled[-1])

# Learning
rfc = RandomForestClassifier(n_jobs=-1,
                             max_depth=2,
                             verbose=1,
                             random_state = 2020)
display('Forest Fit ...')
start = time.time()
rfc.fit(X_train_scaled, y_train)
display('>timing:', time.time() - start)
display('Forest Score ...')
start = time.time()
display(rfc.score(X_train_scaled, y_train))
display('>timing:', time.time() - start)
print("Mean squared error (train): %.3f" %
    mean_squared_error(rfc.predict(X_train_scaled), y_train))
print("Mean squared error (cv): ",
    cross_val_score(estimator=rfc,
                    X=X_train_scaled,
                    y=y_train,
                    cv = 3,
                    scoring='neg_mean_squared_error'))
print("Mean squared error (test): %.3f" %
    mean_squared_error(rfc.predict(X_test_scaled), y_test))

# feature_importances
display('Feature Importances ...')
feats = {}
for feature, importance in zip(columns, rfc.feature_importances_):
    feats[feature] = importance
importances = pd.DataFrame.from_dict(feats,
    orient='index').rename(columns={0: 'Gini-Importance'})
importances = importances.sort_values(by='Gini-Importance',
    ascending=False)
importances = importances.reset_index()
importances = importances.rename(columns={'index': 'Features'})
sns.set(font_scale=0.5,
        style="whitegrid",
        color_codes=True)
fig, ax = plt.subplots()
fig.set_size_inches(30,15)
sns.barplot(x=importances['Gini-Importance'],
            y=importances['Features'],
            data=importances,
            color='skyblue')
plt.xlabel('Importance', fontsize=8, weight = 'bold')
```

```
plt.ylabel('Features', fontsize=8, weight = 'bold')
plt.title('Feature Importance', fontsize=8, weight = 'bold')
plt.show()
display(importances.head(20))

# PCA
display('PCA ...')
start = time.time()
pca_test = PCA(n_components=80)
pca_test.fit(X_train_scaled)
sns.set(style='whitegrid')
plt.plot(np.cumsum(pca_test.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.axvline(linewidth=4, color='r', linestyle = '--', x=20, ymin=0, ymax=1)
plt.show()
evr = pca_test.explained_variance_ratio_
cvr = np.cumsum(pca_test.explained_variance_ratio_)
pca_df = pd.DataFrame()
pca_df['Cumulative Variance Ratio'] = cvr
pca_df['Explained Variance Ratio'] = evr
display(pca_df.head(20))
display('>timing:', time.time() - start)

# Reduce from 80 to 20 components
display('PCA fit...')
start = time.time()
pca = PCA(n_components=20)
pca.fit(X_train_scaled)
X_train_scaled_pca = pca.transform(X_train_scaled)
X_test_scaled_pca = pca.transform(X_test_scaled)
display('>timing:', time.time() - start)

display('Forest Fit ...')
start = time.time()
rfc.fit(X_train_scaled_pca, y_train)
display('>timing:', time.time() - start)

# display the mean accuracy on the given test data and labels.
display('Forest Score ...')
start = time.time()
display(rfc.score(X_train_scaled_pca, y_train))
display('>timing:', time.time() - start)
print("Mean squared error (train): %.3f" %
    mean_squared_error(rfc.predict(X_train_scaled_pca), y_train))
print("Mean squared error (cv): ",
    cross_val_score(estimator=rfc,
                    X=X_train_scaled_pca,
                    y=y_train,
                    cv = 3,
                    scoring='neg_mean_squared_error'))
print("Mean squared error (test): %.3f" %
    mean_squared_error(rfc.predict(X_test_scaled_pca), y_test))
```

```
# RandomizedSearchCV
display('RandomizedSearchCV ...')
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, num = 10)]
max_features = ['log2', 'sqrt']
max_depth = [int(x) for x in np.linspace(start = 1, stop = 15, num = 15)]
min_samples_split = [int(x) for x in np.linspace(start = 2, stop = 50, num = 10)]
min_samples_leaf = [int(x) for x in np.linspace(start = 2, stop = 50, num = 10)]
bootstrap = [True, False]
param_dist = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}
rs = RandomizedSearchCV(rfc,
                        param_dist,
                        n_iter = 10,
                        cv = 3,
                        verbose = 1,
                        n_jobs=-1,
                        random_state=0)
rs.fit(X_train_scaled_pca, y_train)
display(rs.best_params_)

# Learning RF + PCA + SearchCV set
rfc = RandomForestClassifier(n_estimators=400,
    min_samples_split=23,
    min_samples_leaf=2,
    max_features='sqrt',
    max_depth=14,
    bootstrap=False,
    n_jobs=-1,
    verbose=1)
display('Forest Fit + PCA + SearchCV set ...')
rfc.fit(X_train_scaled_pca, y_train)
display('Train Score ...')
display(rfc.score(X_train_scaled_pca, y_train))
print("Mean squared error (train): %.3f" %
    mean_squared_error(rfc.predict(X_train_scaled_pca), y_train))
print("Mean squared error (cv): ",
    cross_val_score(estimator=rfc,
                    X=X_train_scaled_pca,
                    y=y_train,
                    cv = 3,
                    scoring='neg_mean_squared_error'))
print("Mean squared error (test): %.3f" %
    mean_squared_error(rfc.predict(X_test_scaled_pca), y_test))
```

Додаток Г.

Програмна реалізація машинного методу навчання

Градієнтний бустінг (Gradient boosting)

```
import ipaddress as ipc
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import time
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.decomposition import PCA
from IPython.display import display
from sklearn.metrics import recall_score
from sklearn.metrics.regression import mean_squared_error
from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle

display('Reading Data...')
data = pd.read_csv('IoT-BoT.csv')

# Statistics
print(data.tail(3).T)
print(data.shape)
print(data.describe(include='all').T)
print(data['Label'].value_counts())
fig = plt.figure(figsize=(100, 100))
for i, column in enumerate(['Cat', 'Sub_Cat']):
    ax = fig.add_subplot(1, 2, i + 1)
    ax.set_title(column)
    ax.set(ylabel='Percentage')
    data[column].value_counts().plot(kind="bar", axes=ax)
plt.subplots_adjust(hspace=0.7, wspace=0.2)
plt.show()

# Preparation
data['Src_IP'] = data['Src_IP'].apply(lambda x: int(ipc.ip_address(x))).astype(int)
data['Dst_IP'] = data['Dst_IP'].apply(lambda x: int(ipc.ip_address(x))).astype(int)
data = data.drop(['Flow_ID', 'Timestamp', 'Cat', 'Sub_Cat'], axis=1)
display(data.info())

# Data test split
X = data.drop('Label', axis=1)
y = data['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size=0.1,
                                    random_state = 2020)
display(X_train.tail(1))
columns = X_train.columns
```

```
# Scaling
display('Scaling ...')
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
y_train = np.array(y_train)
display(X_train_scaled[-1])

# Learning
gbc = GradientBoostingClassifier(verbose=1,
                                 n_estimators=10,
                                 tol=0.01,
                                 random_state = 2020)
display('Boosting Fit ...')
start = time.time()
gbc.fit(X_train_scaled, y_train)
display('>timing:', time.time() - start)
display('Boosting Score ...')
start = time.time()
display(gbc.score(X_train_scaled, y_train))
display('>timing:', time.time() - start)
print("Mean squared error (train): %.3f" %
    mean_squared_error(gbc.predict(X_train_scaled), y_train))
print("Mean squared error (cv): ",
    cross_val_score(estimator=gbc,
                    X=X_train_scaled,
                    y=y_train,
                    cv = 3,
                    scoring='neg_mean_squared_error'))
print("Mean squared error (test): %.3f" %
    mean_squared_error(gbc.predict(X_test_scaled), y_test))

# feature_importances
display('Feature Importances ...')
feats = {}
for feature, importance in zip(columns, gbc.feature_importances_):
    feats[feature] = importance
importances = pd.DataFrame.from_dict(feats,
    orient='index').rename(columns={0: 'Gini-Importance'})
importances = importances.sort_values(by='Gini-Importance',
    ascending=False)
importances = importances.reset_index()
importances = importances.rename(columns={'index': 'Features'})
sns.set(font_scale=0.5,
        style="whitegrid",
        color_codes=True)
fig, ax = plt.subplots()
fig.set_size_inches(30,15)
sns.barplot(x=importances['Gini-Importance'],
            y=importances['Features'],
            data=importances,
            color='skyblue')
plt.xlabel('Importance', fontsize=8, weight = 'bold')
```

```
plt.ylabel('Features', fontsize=8, weight = 'bold')
plt.title('Feature Importance', fontsize=8, weight = 'bold')
plt.show()
display(importances.head(20))

# PCA
display('PCA ...')
start = time.time()
pca_test = PCA(n_components=80)
pca_test.fit(X_train_scaled)
sns.set(style='whitegrid')
plt.plot(np.cumsum(pca_test.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.axvline(linewidth=4, color='r', linestyle = '--', x=20, ymin=0, ymax=1)
plt.show()
evr = pca_test.explained_variance_ratio_
cvr = np.cumsum(pca_test.explained_variance_ratio_)
pca_df = pd.DataFrame()
pca_df['Cumulative Variance Ratio'] = cvr
pca_df['Explained Variance Ratio'] = evr
display(pca_df.head(20))
display('>timing:', time.time() - start)

# Reduce from 80 to 20 components
display('PCA fit...')
start = time.time()
pca = PCA(n_components=20)
pca.fit(X_train_scaled)
X_train_scaled_pca = pca.transform(X_train_scaled)
X_test_scaled_pca = pca.transform(X_test_scaled)
display('>timing:', time.time() - start)

display('Boosting Fit ...')
start = time.time()
gbc.fit(X_train_scaled_pca, y_train)
display('>timing:', time.time() - start)
# display the mean accuracy on the given test data and labels.
display('Boosting Score ...')
start = time.time()
display(gbc.score(X_train_scaled_pca, y_train))
display('>timing:', time.time() - start)
print("Mean squared error (train): %.3f" %
    mean_squared_error(gbc.predict(X_train_scaled_pca), y_train))
print("Mean squared error (cv): ",
    cross_val_score(estimator=gbc,
                    X=X_train_scaled_pca,
                    y=y_train,
                    cv = 3,
                    scoring='neg_mean_squared_error'))
print("Mean squared error (test): %.3f" %
    mean_squared_error(gbc.predict(X_test_scaled_pca), y_test))
```