ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Засоби виявлення аномалій мережевого трафіку за допомогою машинного навчання»

зі спеціальності *125 Кібербезпека*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Іван ВЕРЕСТЮК
(підпис)

Виконав: здобувач вищої освіти групи БСД-41
ВЕРЕСТЮК Іван
(прізвище, ім'я)

Керівник асистент кафедри, ШУЛІМОВА Дар'я
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2025

ВСТУП

Актуальність дослідження. Комп'ютерні мережі забезпечують роботу важливих інформаційних систем у державному, корпоративному та інфраструктурному секторах. Звичайні методи захисту, як-от міжмережеві екрани та сигнатурні системи виявлення атак, поступово втрачають ефективність через ускладнення мереж та появу нових загроз. Вони покладаються на заздалегідь визначені правила та шаблони, що потребують постійного оновлення, ручної класифікації, та не здатні виявляти нові, раніше невідомі типи атак. Водночас методи машинного навчання відкривають нові можливості для виявлення аномалій у мережевому трафіку. Завдяки здатності аналізувати великі обсяги даних, знаходити складні та неочевидні патерни та адаптуватися до змін у поведінці мережі, ML-моделі можуть істотно підвищити рівень кіберзахисту. Особливе значення мають системи виявлення аномалій (ANIDS), які орієнтовані на виявлення відхилень від нормальної поведінки замість пошуку сигнатур.

Застосування таких підходів дозволяє не лише класифікувати поточну активність у мережі, але й оперативно реагувати на загрози шляхом реєстрації, блокування або обмеження шкідливого трафіку. Саме тому розробка ефективних ML-рішень для виявлення мережевих аномалій є актуальним завданням в умовах сучасної кібербезпеки.

Об'єкт дослідження – мережевий трафік у комп'ютерних системах.

Предмет дослідження – засоби виявлення аномалій у мережевому трафіку за допомогою машинного навчання.

Мета роботи – дослідити та реалізувати підхід до виявлення аномального мережевого трафіку з використанням алгоритмів машинного навчання та розробити прототип системи виявлення загроз (NIDS)

Наукові завдання:

- дослідити класифікацію мережевих аномалій та принципи роботи систем IDS/ANIDS;

- проаналізувати сучасні підходи до виявлення атак за допомогою алгоритмів машинного навчання;

- провести обробку та підготовку даних на прикладі CICIDS2017;

- побудувати, навчити та порівняти кілька моделей на основі датасету;

- реалізувати прототип NIDS-системи з використанням найкращої моделі;

- протестувати її в умовах симульованих атак;

- сформулювати рекомендації щодо розширення функціоналу, зменшення кількості хибнопозитивних спрацювань і можливості інтеграції у корпоративне середовище.

Методи дослідження – аналіз літературних джерел, робота з відкритими датасетами мережевого трафіку (CICIDS2017), побудова моделей машинного навчання, оцінка їх ефективності, практичне тестування в симульованому середовищі.

Практичне значення отриманих результатів полягає в створенні NIDS-прототипу, здатного виявляти аномалії у трафіку за допомогою ML-моделі. Щоб система показала можливість виявлення та класифікування атак типу Port Scanning, Brute Force, DoS, DDoS, Web Attacks та Bots, а також потенціал до інтеграції із засобами реагування (IPS, NDR). Результати можуть бути використані як основа для подальшої розробки адаптивних систем виявлення аномалій у реальному середовищі.

Галузь використання – кібербезпека, автоматичний моніторинг та аналіз мережевого трафіку в організаціях.

МЕРЕЖЕВИЙ ТРАФІК, ВИЯВЛЕННЯ АНОМАЛІЙ, СИСТЕМИ IDS, МАШИННЕ НАВЧАННЯ, NIDS

1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АНОМАЛІЙ

1.1 Класифікація аномалій у мережевому трафіку

При аналізі великих наборів даних, зокрема в контексті мереж, шаблони, що відхиляються від очікуваної норми, називають аномаліями. У різних сферах застосування їх також називають відхиленнями, винятками, аномальностями, сюрпризами, особливостями або дисонансами. Серед цих термінів найпоширенішими в контексті виявлення мережевих аномалій є “аномалії” та “відхилення”. Виявлення аномалій має критичне значення в багатьох галузях — від виявлення шахрайства з кредитними картками й моніторингу здоров’я до кібербезпеки та військової розвідки. У комп’ютерних мережах аномальний трафік часто свідчить про те, що скомпрометована машина передає конфіденційні дані несанкціонованим одержувачам.

Комп’ютерна мережа за своєю природою об’єднує безліч компонентів для забезпечення складних комунікаційних послуг. Будь-який збій або зловмисне втручання в ці компоненти може призвести до виникнення аномалій. Загалом аномалії в мережі поділяють на два основних типи: *аномалії продуктивності* та *аномалії безпеки*[1].

Аномалії Продуктивності

Аномалії продуктивності виникають переважно через несправності в інфраструктурі мережі. Вони можуть проявлятися через проблеми з навколишнім середовищем, як-от втрата контролю за температурою, що призводить до перегріву чи пошкодження процесорів, або через фізичні вразливості, коли недостатній захист обладнання дозволяє здійснювати крадіжки чи саботаж.

Важливим аспектом є конфігурація мережевих периметрів. Відсутність чітко визначеної межі безпеки або неправильне налаштування брандмауерів може призвести до витоку даних або несанкціонованого доступу. Неналежне ведення журналів та відсутність ефективного моніторингу ще більше знижують здатність

швидко реагувати на інциденти. Регулярний та системний моніторинг мережевої активності є необхідною умовою для запобігання пошкодженню чи порушенню роботи мережі.

Вразливості комунікаційних каналів також становлять ризик. Відсутність ідентифікації критичних шляхів передачі даних створює приховані “бекдори”. Використання стандартних протоколів, як-от FTP, Telnet або NFS без належних заходів безпеки, відкриває мережу для експлуатації через інструменти аналізу протоколів. Слабкість аутентифікації користувачів, даних чи пристроїв збільшує ризик спуфінгу та несанкціонованого доступу, а брак контролю цілісності даних дозволяє атакуючим непомітно змінювати комунікації.

Уразливості бездротових мереж ще більше посилюють ризики. Слабка автентифікація між клієнтами та точками доступу дозволяє зловмисникам створювати фальшиві точки доступу. Відсутність шифрування переданих даних полегшує перехоплення конфіденційної інформації. В Таблиці 1.1 наведено приклади аномалій продуктивності

Таблиця 1.1

Типи аномалій продуктивності

Тип аномалії	Опис
Недостатній контроль середовища	Відсутність моніторингу температури чи вологості може призвести до перегріву процесорів і аварійного вимкнення систем.
Недостатні механізми фізичного доступу	Відсутність обмежень на фізичний доступ до мережевого обладнання створює ризики несанкціонованих змін або крадіжки даних.
Відсутність резервного копіювання	Неналежне резервування критичних даних може призвести до повної втрати інформації у разі збою.
Недостатній захист периметра мережі	Відсутність чітких зон контролю та помилки у налаштуванні міжмережевих екранів сприяють поширенню шкідливого ПЗ.
Неналежний моніторинг трафіку та ведення логів	Нестача ретельного аудиту подій і некоректна фільтрація трафіку призводить до прихованого накопичення загроз у мережі.

Аномалії Безпеки

На відміну від аномалій продуктивності, аномалії безпеки виникають унаслідок навмисних зловмисних дій, спрямованих на порушення функціонування мережі. Вони включають аномалії мережевої діяльності, сплескові аномалії навантаження та аномалії затоплення трафіку, однак головною загрозою залишаються навмисні атаки.

Аномалії безпеки поділяються на три основні типи:

Точкові аномалії — це одиничні випадки даних, що істотно відрізняються від загальної маси. Наприклад, нехарактерний великий обсяг вихідного трафіку вночі.

Контекстні аномалії — це дії, що нормальні в одному контексті, але є аномальними в іншому. Наприклад авторизація адміністратора з незвичної геолокації або в нетиповий час доби.

Колективні аномалії — це послідовність дій, кожна з яких окремо не є аномальною, але разом вони формують підозрілий шаблон. Наприклад, серія успішних входів у систему з різних IP-адрес за короткий проміжок часу, що завершується завантаженням великої кількості даних з файлового сервера..

Після розгляду типів аномалій безпеки, важливо також знати приклади конкретних атак, які можуть викликати такі аномалії. Типологія атак наведена в таблиці 1.2 та виглядає наступним чином:

Таблиця 1.2

Типи аномалій безпеки

Основна категорія атаки	Опис	Приклади атак
Інфікування	Встановлення шкідливого ПЗ для отримання контролю або пошкодження системи	Віруси, черв’яки, троянські програми
Перевантаження	Переповнення системи помилками або надлишковими даними	Атаки типу Buffer Overflow, Ping of Death, TCP SYN Flood, Teardrop
Розвідка (Probe)	Збір інформації про мережу чи систему для планування подальших атак	IP Sweep, Port Sweep, сканування портів через Nmap

Продовження таблиці 1.2.

Типи аномалій безпеки

Обман (Cheat)	Використання підроблених даних або протоколів для обходу захисту	IP-спуфінг, MAC-спуфінг, DNS-спуфінг, захоплення сесії, XSS-атаки
Проникнення (Traverse)	Отримання доступу шляхом підбору паролів чи ключів	Брутфорс-атаки, словникові атаки, Doorknob-атаки
Паралельні атаки (Concurrency)	Створення перевантаження через масові запити	DDoS (розподілена відмова в обслуговуванні), Flooding
Інші атаки	Використання відомих вразливостей без складної підготовки	Експлуатація недоліків у ПЗ

Таким чином, було розглянуто основні типи аномалій у мережах — як продуктивностні, так і безпекові, а також типові приклади атак, які можуть їх спричиняти. Однак для реального захисту мережевої інфраструктури недостатньо лише розуміти природу аномалій — важливо також вміти своєчасно їх виявляти та реагувати на них.

1.2 Системи виявлення вторгнень (IDS)

Виявлення проникнення в інформаційні системи є одним із ключових напрямків кібербезпеки, який активно досліджується вже понад два десятиліття. Основна концепція IDS (Intrusion Detection Systems) полягає у фіксації відхилень у поведінці користувачів чи мережевого трафіку, що може свідчити про спробу атаки або іншу зловмисну активність. Системи виявлення вторгнень призначені для своєчасного виявлення несанкціонованого доступу, порушень політик безпеки або спроб експлуатації вразливостей у комп’ютерних мережах і системах. Залежно від місця їхнього встановлення та природи даних, які вони аналізують, IDS поділяються на дві основні категорії: системи на основі хоста (HIDS) та мережеві системи виявлення вторгнень (NIDS)[1][5].

HIDS орієнтовані на моніторинг внутрішньої активності окремих пристроїв, таких як сервери або робочі станції. Вони аналізують журнали подій, зміни у файлових системах, активні процеси та спроби доступу до важливих ресурсів.

NIDS, у свою чергу, контролюють трафік на рівні мережі, зчитуючи вхідні та вихідні пакети у реальному часі. Їхнє основне завдання — виявлення підозрілих зразків у мережевій активності, що можуть свідчити про спроби сканування портів, перебір паролів, спроби експлуатації вразливостей або витік даних. В таблиці 1.3 наведено порівняння цих типів

Таблиця 1.3

Порівняння характеристик HIDS та NIDS

Критерій	HIDS	NIDS
Місце встановлення	На окремому хості	На мережевих вузлах або розгалуженнях
Тип даних для аналізу	Файлові системи, журнали, процеси	Пакетний та потоковий мережевий трафік
Сфокусованість на загрозах	Виявлення внутрішніх порушень	Виявлення зовнішніх атак
Основні переваги	Глибокий аналіз стану системи	Охоплення великої кількості пристроїв
Основні недоліки	Потрібна інсталяція на кожному пристрої	Складність виявлення атак при шифруванні трафіку

Як видно з таблиці, вибір типу IDS залежить від цілей моніторингу: чи то контроль внутрішньої поведінки систем, чи аналіз периметру мережі.

Методи виявлення загроз у IDS також можуть відрізнятися залежно від використовуваного підходу. Вони поділяються на:

Сигнатурний підхід, який базується на порівнянні трафіку або подій із базою відомих шаблонів атак. Прикладом систем, що використовують цей метод, є Snort, Suricata та Zeek. Основною перевагою такого підходу є висока точність у виявленні

відомих загроз. Однак він має суттєвий недолік — неспроможність виявляти нові типи атак, для яких ще не створено відповідних сигнатур.

Аномальний підхід, який передбачає побудову профілю “нормальної” поведінки системи та виявлення істотних відхилень від нього. Такий метод дозволяє виявляти нові або змінені атаки, але часто супроводжується підвищеною кількістю хибнопозитивних спрацювань і потребує високих обчислювальних ресурсів.

Гібридний підхід поєднує можливості сигнатурного та аномального виявлення, намагаючись мінімізувати їхні слабкі сторони та підвищити загальну ефективність системи.

На практиці для розгортання IDS використовуються різноманітні програмні рішення. Найбільш популярними є:

- *Snort* — гнучка система, яка функціонує як IDS і може працювати у режимі IPS (системи запобігання вторгнень).
- *Suricata* — альтернатива Snort, що підтримує мультипотокову обробку та здатна аналізувати великі обсяги трафіку.
- *Zeek* — орієнтований на глибокий аналіз протоколів і збирання детальних логів для подальшого вивчення інцидентів.
- *OSSEC* — популярне рішення HIDS-класу для централізованого моніторингу стану систем та журналів подій.

Таким чином, правильний вибір інструменту IDS має базуватись на конкретних потребах організації, архітектурі мережі, ресурсах та рівні загроз.

Дані системи дозволяють своєчасно виявляти спроби атак, аналізувати аномальну активність та підтримувати контроль над станом інформаційних ресурсів. Проте зі зростанням обсягів і складності мережевого трафіку, а також появою нових типів загроз, класичні IDS стають недостатньо ефективними. Саме тому наступний крок у розвитку захисних механізмів — це використання методів машинного навчання[15] для автоматизації процесу виявлення загроз та покращення адаптивності систем безпеки.

1.3 Застосування машинного навчання для аналізу мережевого трафіку

Класичні методи виявлення аномалій, засновані на ручному аналізі журналів або використанні попередньо визначених сигнатур, стають дедалі менш ефективними. Якщо раніше було можливим виявляти підозрілі дії шляхом зіставлення логів і трафіку з відомими шаблонами аномальної поведінки, то сьогодні масштаби даних настільки великі й різнорідні, що виявлення порушень традиційними методами стало складнішим завданням. Багато вторгнень замасковані під нормальну активність і можуть залишатися непоміченими навіть досвідченими фахівцями.

Аномальні активності можуть бути виявлені шляхом аналізу логів і мережевого трафіку з використанням шаблонів відомої аномальної поведінки. Проте величезний обсяг даних, їх постійне зростання та різнорідність ускладнюють ідентифікацію порушень. Патерни атак можуть бути нечіткими або складно відокремлюваними від нормальної поведінки. У таких умовах ідеї машинного навчання набувають особливого значення, пропонуючи нові підходи до виявлення потенційних загроз та своєчасного інформування персоналу, відповідального за безпеку мережі.

Машинне навчання стало інструментом для автоматизації аналізу великих обсягів даних та виявлення нових, раніше невідомих закономірностей, що можуть свідчити про порушення безпеки. *Основною метою машинного навчання є автоматичне виявлення нових, достовірних і корисних закономірностей у великих обсягах даних[12].*. Алгоритми машинного навчання здатні розпізнавати складні патерни у наявних наборах даних і здійснювати інтелектуальні рішення або прогнози стосовно нових, раніше не бачених прикладів.

Методи машинного навчання умовно поділяють на кілька основних типів:

Навчання з учителем (Supervised Learning) передбачає використання мічених даних, де кожен приклад має визначену категорію або клас, наприклад: «нормальний трафік» або «атакуючий трафік». Це дозволяє створювати точні моделі для класифікації нових прикладів.

Навчання без учителя (Unsupervised Learning) не потребує попереднього маркування даних і орієнтується на виявлення внутрішніх закономірностей, групуючи схожі об’єкти в кластери або виявляючи аномальні відхилення від загальної картини.

Напівкероване навчання (Semi-supervised Learning) поєднує обидва підходи, використовуючи невелику кількість мічених даних поряд із великою кількістю немічених для побудови більш гнучких моделей.

Таким чином, машинне навчання відкриває широкі можливості для створення сучасних та адаптивних систем виявлення загроз. Правильне поєднання різних підходів дозволяє не тільки виявляти вже відомі атаки з високою точністю, але й значно підвищує шанси на виявлення нових, раніше невідомих загроз, що є критично важливим у динамічному середовищі сучасних кіберпросторів.

Одним із практичних напрямів застосування машинного навчання у сфері кібербезпеки є побудова систем виявлення мережевих аномалій на основі аналізу трафіку — *ANIDS (Anomaly-Based Network Intrusion Detection Systems)[3]*. Розглянемо детальніше архітектуру даної технології, основні компоненти та принципи функціонування.

Виявлення мережевого вторгнення, заснованого на аномаліях, є одним із головних напрямів досліджень і розробок в області виявлення загроз. Саме машинне навчання виступає ключовою технологією, що дозволяє автоматизувати процес виявлення аномальної активності у складних мережах.

У зв’язку з цим з’являються різні системи виявлення мережевого вторгнення на основі аномалій (ANIDS), які активно використовують методи машинного навчання для аналізу мережевого трафіку та виявлення відхилень від нормальної поведінки. Наразі вивчаються та розробляються багато нових підходів до побудови таких систем, Однак ця тема ще не до кінця опрацьована, і ще належить вирішити ключові питання, перш ніж стане можливим широкомасштабне розгортання ANIDS

Типова архітектура ANIDS складається з кількох ключових компонентів (Рис. 1.1)[1].

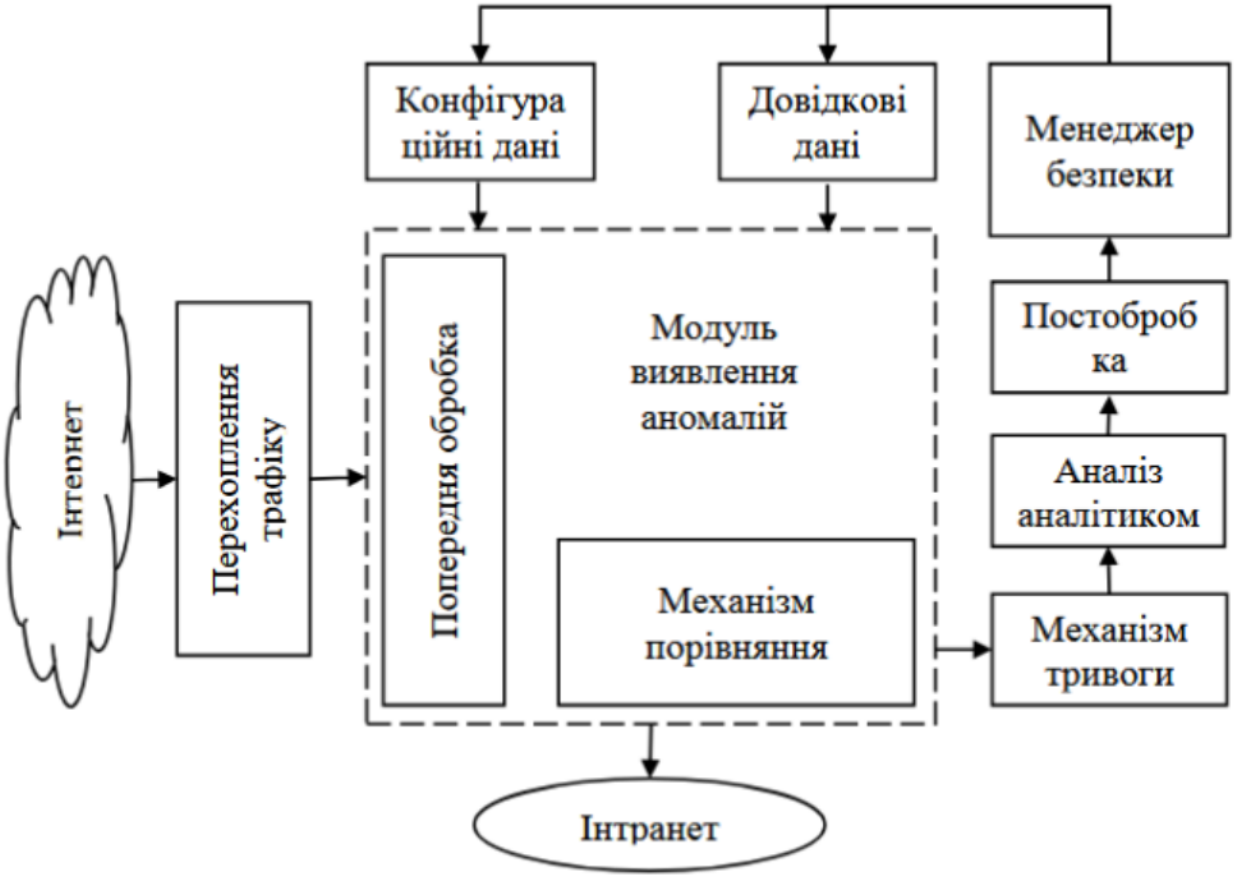

Рис. 1.1. Архітектура ANIDS

- *Модуль виявлення аномалій* є основою будь-якої системи виявлення вторгнень. Його головне завдання полягає у виявленні будь-яких аномальних дій як в режимі онлайн, так і в автономному режимі. Перед передачею мережевого трафіку до модуля виявлення аномалій дані обов'язково проходять етап попередньої обробки для усунення шумів і приведення їх до уніфікованого вигляду. У випадку відомих атак їх виявлення може здійснюватися за допомогою сигнатурних методів. Проте для виявлення нових, раніше невідомих атак критичну роль відіграють методи виявлення аномалій, які часто базуються на алгоритмах машинного навчання. Ці алгоритми дозволяють будувати адаптивні моделі нормальної поведінки мережевого трафіку та автоматично виявляти відхилення від них. Виявлення здійснюється за допомогою *механізму порівняння*, який відповідає за пошук відповідного шаблону або профілю в потоці даних. Профілі можуть бути побудовані шляхом постійного моніторингу мережевого трафіку з урахуванням як

звичайної активності, так і відомих типів атак та вразливостей. Для розробки ефективного механізму порівняння необхідно дотримуватись кількох вимог: механізм має точно визначати, чи належить новий екземпляр до певного класу, заданого профілем високої розмірності; процес порівняння має бути швидким і масштабованим; ефективна організація профілів дозволяє прискорити пошук відповідностей у великих масивах даних[1].

- *Довідкові дані* – компонент, який зберігає інформацію про сигнатури відомих атак, вразливостей, а також профілі нормальної поведінки мережевого трафіку. Для ефективного функціонування системи довідкові дані мають бути організовані таким чином, щоб забезпечувати швидкий доступ і оновлення. Типовими видами довідкових даних у загальній архітектурі ANIDS є профілі, сигнатури та правила. Профілі постійно вдосконалюються шляхом накопичення нових знань про поведінку мережевих об’єктів, причому оновлення відбувається регулярно і в пакетному режимі.

- *Конфігураційні дані* – модуль, відповідальний за зберігання тимчасової або неповної інформації, зокрема частково сформованих сигнатур вторгнень. Через динамічний характер змін у мережевому середовищі обсяг таких даних може бути досить великим. Механізм оновлення конфігураційних даних представлений на рисунку 1.2[1].

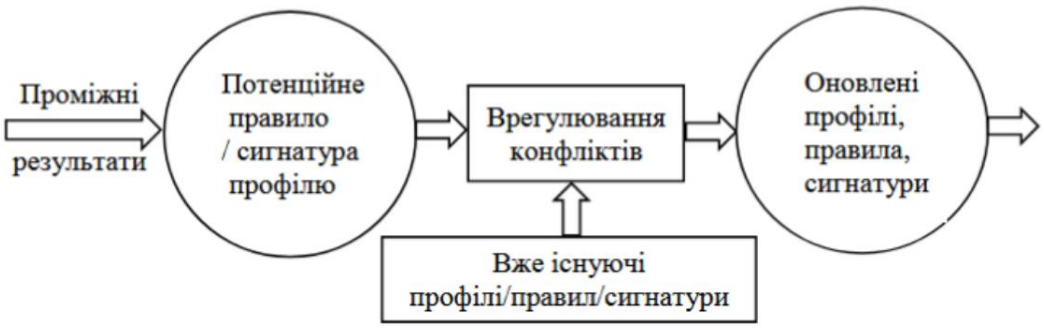

Рис. 1.2 – Оновлення конфігураційних даних в загальній архітектурі ANIDS

- *Механізм тривоги* – відповідає за генерацію сигналів тривоги на основі показників, отриманих від модуля виявлення аномалій.

- Аналіз аналітиком – відповідає за аналіз, інтерпретацію та прийняття необхідних заходів на підставі сигнальної інформації, наданої модулем виявлення аномалій. Аналітик також вживає необхідних заходів для діагностики інформації про сигнали тривоги в якості постобробки для підтримки еталонного або профільного оновлення за допомогою менеджера безпеки.

- Постобробка – модуль, який забезпечує верифікацію та поглиблений аналіз тривожних сповіщень для відокремлення реальних атак від хибних спрацювань..

Менеджер безпеки – відповідає за оновлення сигнатур вторнень, по мірі того як з'являються нові види атак.

Виявлення мережевих аномалій розглядається як задача виявлення виняткових закономірностей у трафіку, які відхиляються від очікуваної нормальної поведінки. Такі закономірності визначаються як аномалії або відхилення.

Системи виявлення мережевих аномалій (ANIDS) виконують моніторинг мережевого трафіку на рівні пакетів і застосовують різні методи аналізу для класифікації вхідних даних на нормальні та аномальні. В залежності від доступності навчальних даних та обраної стратегії, системи можуть працювати в одному з трьох режимів:

- *Під наглядом (supervised mode)* — використовується навчальна вибірка, яка містить як нормальні, так і аномальні приклади.
- *Напівкерований режим (semi-supervised mode***)** — система навчається лише на даних нормальної поведінки, виявляючи відхилення без явних прикладів атак.
- *Без нагляду (unsupervised mode)* — немає потреби у попередньо розмічених даних; кластеризація або інші алгоритми самостійно виділяють аномальні шаблони.

Ключовим етапом побудови ефективної системи ANIDS є формалізація поняття *нормальності*. Нормальність визначається через модель, яка описує взаємозв'язки між основними змінними мережевого середовища. У разі значного відхилення нових даних від цієї моделі спостереження класифікуються як аномальні.

2 АНАЛІЗ, ПОБУДОВА ТА ОЦІНЮВАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

2.1 Типи машинного навчання та їх порівняння

Виявлення мережевих аномалій є багатокроковим процесом, що спрямований на виявлення відхилень від нормальної поведінки мережі для своєчасного виявлення атак, збоїв або атипових подій. Зазвичай аномалії виникають унаслідок шкідливих дій, збоїв обладнання чи програмного забезпечення або навіть законних, але рідкісних подій, таких як великі передавання файлів чи миттєві сплески трафіку.

З точки зору машинного навчання, завдання виявлення аномалій розглядається як задача класифікації, де потрібно розрізнити нормальні та аномальні приклади поведінки на основі аналізу мережевого трафіку. Загалом підходи до виявлення аномалій поділяються на кілька основних типів, залежно від наявності попередніх знань, методики аналізу та архітектури системи.

На основі аналізу наукової літератури існуючі методи виявлення аномалій у мережах класифікуються на шість основних категорій (Рис. 2.1)[1][2]:

- *Навчання з учителем (Supervised Learning)*: передбачає використання попередньо мічених даних для побудови моделей, здатних класифікувати нові приклади.
- *Навчання без учителя (Unsupervised Learning)*: працює без мічених даних, спрямоване на виявлення закономірностей та відхилень у даних.
- *Ймовірнісне навчання (Probabilistic Learning):* базується на побудові статистичних моделей для оцінки ймовірності аномальної поведінки.
- *М'які обчислення (Soft Computing):* включає нечіткі системи, еволюційні алгоритми та інші підходи для роботи з невизначеністю та неповнотою даних.
- *Методи на основі знань (Knowledge-Based Methods):* використовують правила та експертні системи для виявлення порушень.

- *Комбіновані методи (Combination Learners):* поєднують кілька підходів для підвищення точності та стійкості систем виявлення аномалій.

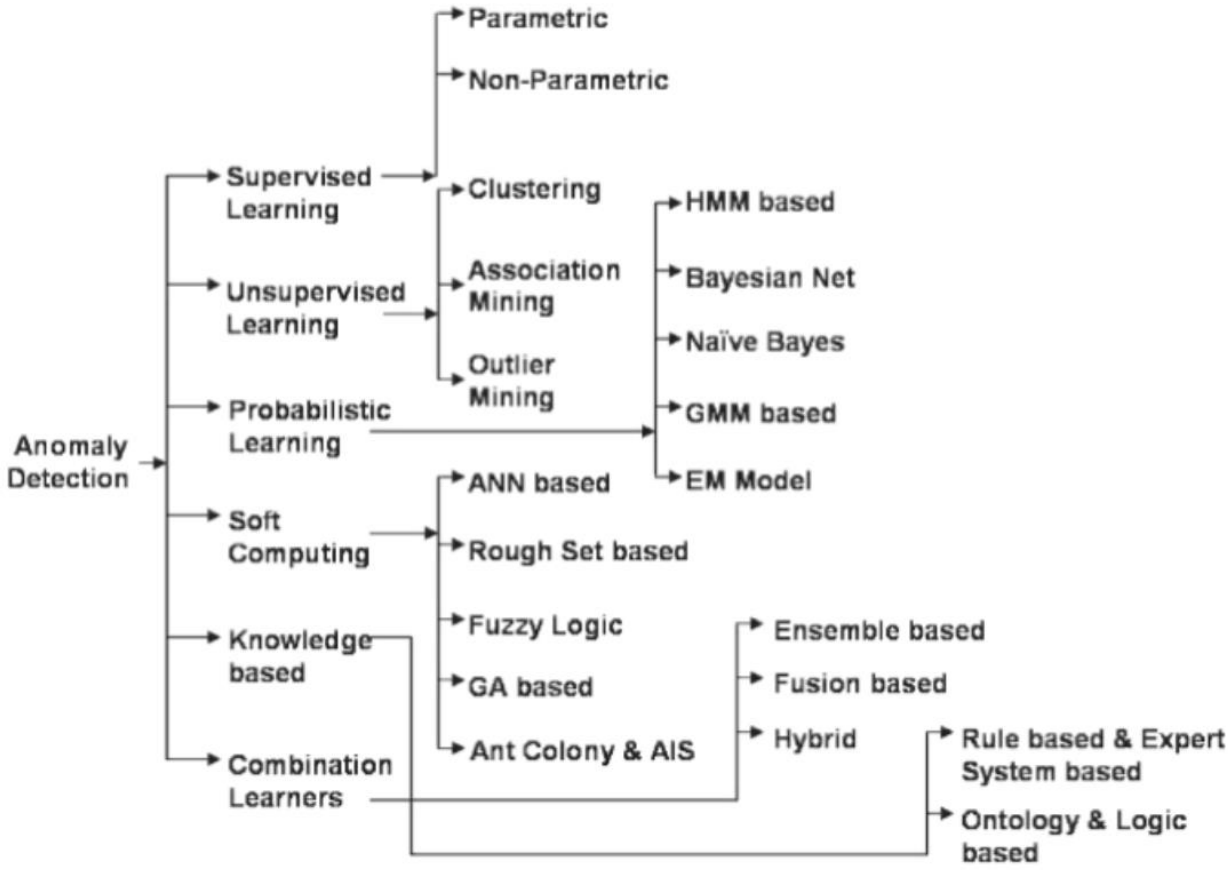

Рис 2.1 – Класифікація методів виявлення мережевих аномалій

Рис. 2.1 ілюструє загальну класифікацію методів машинного навчання, що застосовуються для аналізу аномальної активності в мережах. Кожна категорія включає набір специфічних підходів і алгоритмів, які використовуються залежно від поставленого завдання, наявності даних і бажаних характеристик системи.

У наступних розділах буде детально розглянуто основні типи навчання:

- Спочатку буде проаналізовано *навчання з учителем*, його математичні основи, типи, застосування і обмеження у виявленні аномалій.
- Далі буде висвітлено *навчання без учителя*, з описом основних алгоритмів та формулювань задачі.
- Також коротко буде охарактеризовано інші специфічні напрями, такі як ймовірнісні моделі, м’які обчислення, знання-орієнтовані та комбіновані системи.

У підходах на основі *навчання з учителем* (supervised learning) передбачається наявність навчального набору даних, де кожен екземпляр має відповідну мітку: «нормальний трафік» або один із відомих класів аномалій. Типовим завданням є побудова предиктивної моделі, яка дозволяє розрізняти нормальну та аномальну поведінку мережевого трафіку. Нові дані перевіряються через цю модель для визначення їх належності до відповідного класу.

Однак у використанні цього підходу виникають дві основні проблеми:

- Кількість нормальних екземплярів зазвичай значно переважає кількість аномальних у навчальному наборі даних, що створює проблему незбалансованості класів.
- Отримання точних та репрезентативних міток, особливо для класів аномалій, є надзвичайно складним завданням.

У практичних умовах сформувати повністю чисті набори нормального трафіку майже неможливо, оскільки завжди існує ризик прихованих аномалій у даних. Для подолання цієї проблеми часто застосовуються техніки, які дозволяють штучно генерувати аномальні приклади та змішувати їх із нормальними даними для створення навчальних вибірок.

Таким чином, виходячи з наявності відповідного навчального набору даних, *побудова достовірної предиктивної моделі* є основним завданням у супервізованому виявленні аномалій. Серед методів виявлення мережевих аномалій за допомогою навчання з учителем розрізняють два основних типи[1]:

- *Параметричні методи (Parametric methods)* – припускають заздалегідь певний розподіл даних (наприклад, нормальний розподіл) і будують моделі на основі цих припущень.
- *Непараметричні методи (Non-Parametric methods)* – не роблять жорстких припущень щодо форми розподілу даних, що робить їх більш гнучкими в реальних умовах.

Параметричні Методи

Визначення та Підхід

Параметричні методи припускають, що нормальні дані генеруються з параметричного розподілу з певними параметрами (θ) та функцією щільності ймовірності (pdf) f(p, θ), де p представляє спостережений екземпляр. Ці методи визначають статистичну модель з фіксованим набором параметрів, отриманих з навчальних даних.

Оцінка аномальності тестового екземпляра p зазвичай розраховується як обернена величина до pdf, f(p, θ). Чим нижча ймовірність p за вивченим розподілом, тим більша ймовірність, що це аномалія.

Процес Впровадження

1. Визначення параметричної моделі розподілу (часто гаусів/нормальний розподіл)
2. Оцінка параметрів (θ) з навчальних даних
3. Для кожного нового спостереження розрахунок його ймовірності за розподілом
4. Позначення екземплярів з ймовірністю нижче порогу як аномалій

Параметричні Методи на Практиці

З наданої літератури[], кілька параметричних підходів для виявлення мережевих аномалій включають(Таблиця 2.1):

Таблиця 2.1

Параметричні методи

Метод	Опис
Модель суміші	Модель суміші для виявлення аномалій без навчання на нормальних даних. Оцінює розподіл ймовірностей у даних і застосовує статистичні тести для виявлення аномалій. Застосовано до трасувань системних викликів UNIX для оцінки.

Продовження таблиці 2.1

Параметричні методи

Статистична модель	Автоматичний, адаптивний, проактивний та ієрархічний багаторівневий багатовіконний метод виявлення аномалій. Застосовується як до дротових, так і до бездротових адхок-мереж. Продуктивність оцінено за допомогою наборів даних DARPA98 та в реальному часі.
Модель ARMA	Використовує аналіз та фільтрацію трафіку в “частотній області”. Трафік розділяється на базову складову, що включає більшість низькочастотного трафіку з низькою буферизацією, та короткостроковий трафік, що включає найбільш динамічну частину. Продуктивність оцінено за допомогою симулятора OPNET.
Розподілена модель прихованого Маркова	Використовує стратегію прихованої марковської моделі (HMM) з теорії управління для виявлення вторгнень з використанням розподілених спостережень на декількох вузлах. Включає розподілений двигун HMM, який виконується на випадково вибраному вузлі моніторингу та функціонує як частина механізму зворотного зв’язку.

Непараметричні методи

Визначення та Підхід

Непараметричні методи не припускають жодної базової моделі розподілу для даних. Замість підгонки заздалегідь визначеної статистичної моделі, ці методи адаптують свій механізм виявлення до характеристик самих даних. Вони зазвичай роблять менше припущень про дані порівняно з параметричними методами.

Процес Впровадження

1. Побудова представлень даних (гістограми, графи тощо) на основі навчальних даних
2. Для нових спостережень обчислення їх зв’язку з вивченою структурою даних

3. Розрахунок оцінок аномальності за допомогою непараметричних метрик

4. Позначення екземплярів з оцінками за межами порогу як аномалій

Непараметричні Методи на Практиці

З наданої літератури, кілька непараметричних підходів для виявлення мережевих аномалій включають:

Таблиця 2.2

Непараметричні методи

Метод	Опис
Система IntruShield	Адаптивна система статистичного виявлення аномалій. Застосовує статистичне профілювання використання для постійної модифікації базової лінії, за якою вимірюється вся активність для виявлення аномальної поведінки. Підтримує два набори даних використання: (a) довгостроковий профіль використання та (b) короткострокове спостережуване використання. Порівнює короткострокове використання з довгостроковим профілем під час виявлення аномалій і повідомляє про значні відхилення як потенційні атаки.
Мультиграфік CUSUM	Алгоритм мультиграфіка CUSUM (Кумулятивна Сума) для виявлення DoS-атак. Використовує мінімальну кількість доступної інформації про моделі трафіку до та після зміни.

Параметричні методи мають перевагу в точному виявленні аномалій за умови правильно налаштованих порогів і використовуються для аналізу очікуваної поведінки системи, однак вимагають чіткого визначення статистичних моделей і більше часу для навчання. Непараметричні методи, навпаки, більш гнучкі, не потребують припущень про розподіли даних, краще працюють з високовимірною інформацією та складною структурою, але часто є обчислювально затратними й

чутливими до шуму. Вибір між ними залежить від ресурсоємності, складності даних і цілей моніторингу.

Виявлення аномалій за допомогою навчання без вчителя (Unsupervised Learning)

Підходи навчання без вчителя до виявлення аномалій пропонують значні переваги у ситуаціях, коли важко отримати марковані дані. На відміну від методів з учителем, методи без вчителя не потребують навчальних даних з явними мітками, що вказують на нормальну або аномальну поведінку. Це робить їх особливо цінними в контексті мережевої безпеки, де[3]:

- Отримання точно маркованих даних є надмірно дорогим
- Ручне маркування експертами займає багато часу
- Нормальну поведінку легше охарактеризувати, ніж усі можливі аномалії
- Динамічний характер аномальної поведінки означає, що постійно з’являються нові типи

Методи без вчителя можуть виявляти потенційні вторгнення, приховані в даних, без попереднього навчання, і ці виявлені випадки можуть згодом використовуватися для навчання більш цілеспрямованих методів з учителем. Розглянемо 3 основні методи.

1) Методи виявлення аномалій на основі кластеризації

Визначення та підхід

Методи кластеризації(Рис. 2.1) групують точки даних у кластери на основі мір подібності або обчислень відстані. Процес зазвичай включає вибір репрезентативних точок для кожного кластера та призначення точок даних до найближчого кластера. Цей підхід може вивчати та виявляти аномалії без необхідності отримання явних пояснень типів аномалій від системних адміністраторів.

Процес впровадження

1. Вибір репрезентативних точок для початкових кластерів

2. Групування точок даних на основі близькості до цих репрезентативних точок

3. Повторення процесу для уточнення кластерів

4. Ідентифікація менших кластерів або ізольованих точок як потенційних аномалій

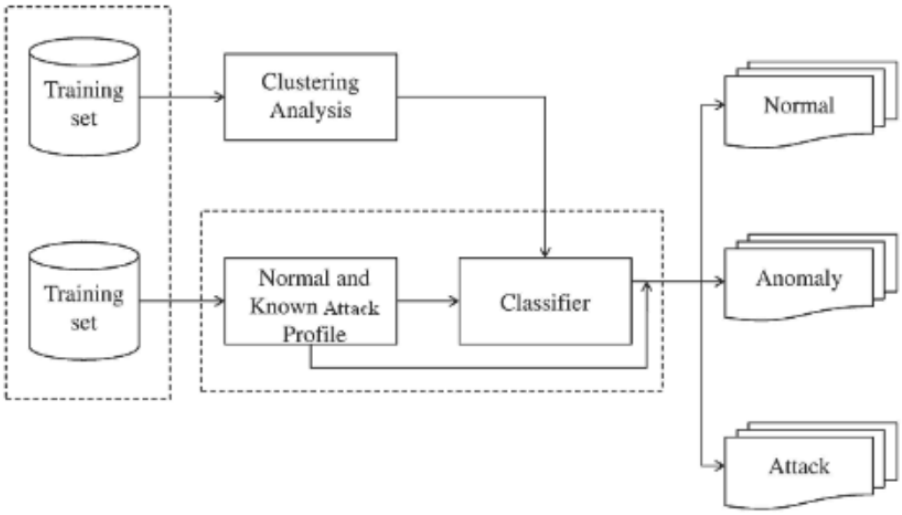

Рис. 2.1 - Виявлення аномалій на основі кластеризації

Приклад: Алгоритм виявлення мережевих аномалій без вчителя (UNADA)

UNADA — це незалежний від знань підхід, який використовує кластеризацію щільності у підпросторі для виявлення викидів або невідповідних патернів(Рис. 2.2). Метод:

1. Захоплює трафік у послідовних часових інтервалах фіксованої довжини
2. Агрегує трафік у IP-потоки, використовуючи різні ключі агрегації
3. Будує часові ряди з показниками трафіку
4. Використовує виявлення змін для ідентифікації значних відхилень
5. Застосовує кластеризацію для ранжування потоків за ступенем аномальності
6. Позначає викиди з найвищим рейтингом як аномалії

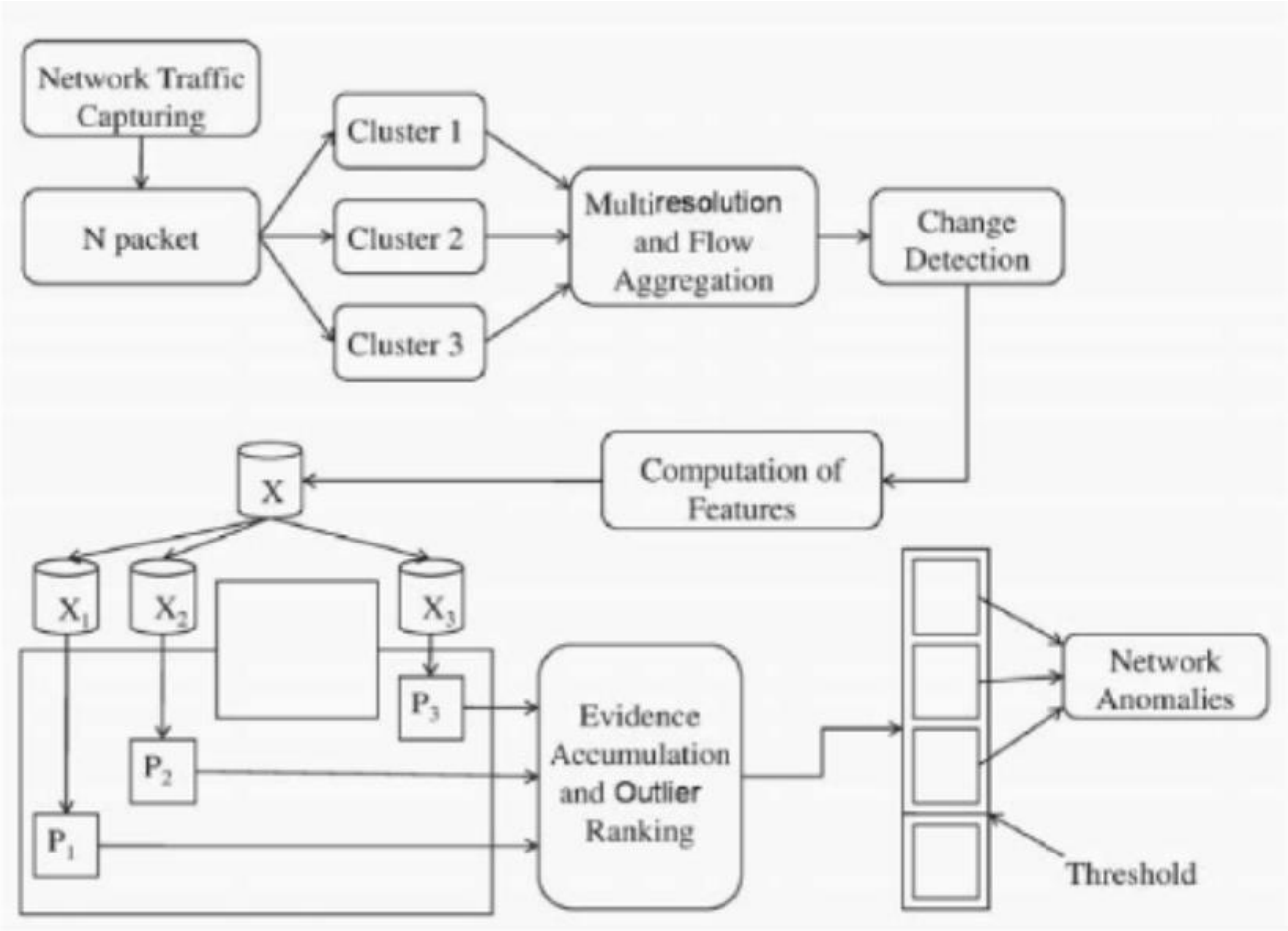

Рис. 2.2 - Концептуальна структура UNADA[1]

UNADA може виявляти різні типи аномалій:

- Аномалії 1-до-N (трафік від одного джерела до багатьох призначень)
- Аномалії N-до-1 (трафік від багатьох джерел до одного призначення)
- Розподілені аномалії завдяки комбінованому використанню ключів IP-джерела та призначення

2) Виявлення аномалій за допомогою пошуку викидів

Пошук викидів шукає об’єкти, які не відповідають правилам і очікуванням, яким слідує більшість даних. З точки зору кластеризації, викиди — це об’єкти, які випадають за межі встановлених кластерів і можуть вважатися потенційними атаками.

Типи методів виявлення викидів

1. *На основі відстані:* Обчислює відстані між об’єктами за допомогою геометричних мір
2. *На основі щільності:* Оцінює щільність сусідніх об’єктів для кожного екземпляра даних

– Об’єкти в районах з низькою щільністю вважаються викидами

– Об’єкти в районах з високою щільністю вважаються нормальними

3. *Підходи м’яких обчислень:* Використовують нечітку логіку, нейронні мережі або інші методи м’яких обчислень

Приклад: Система виявлення вторгнень Міннесоти (MINDS)

MINDS(Архітектура представлена на рисунку 2.3) використовує методи пошуку викидів для виявлення невідомих мережевих аномалій через:

1. Вилучення ознак та узагальнення на основі часових вікон
2. Початкову фільтрацію відомих атак
3. Обчислення Локального Фактору Викиду (LOF) для кожної точки даних

– LOF враховує щільність сусідніх об’єктів навколо кожної точки

– Вищі значення LOF вказують на більшу невідповідність (викиди)

4. Людський аналіз найбільш аномальних з’єднань
5. Зворотний зв’язок для покращення майбутнього виявлення

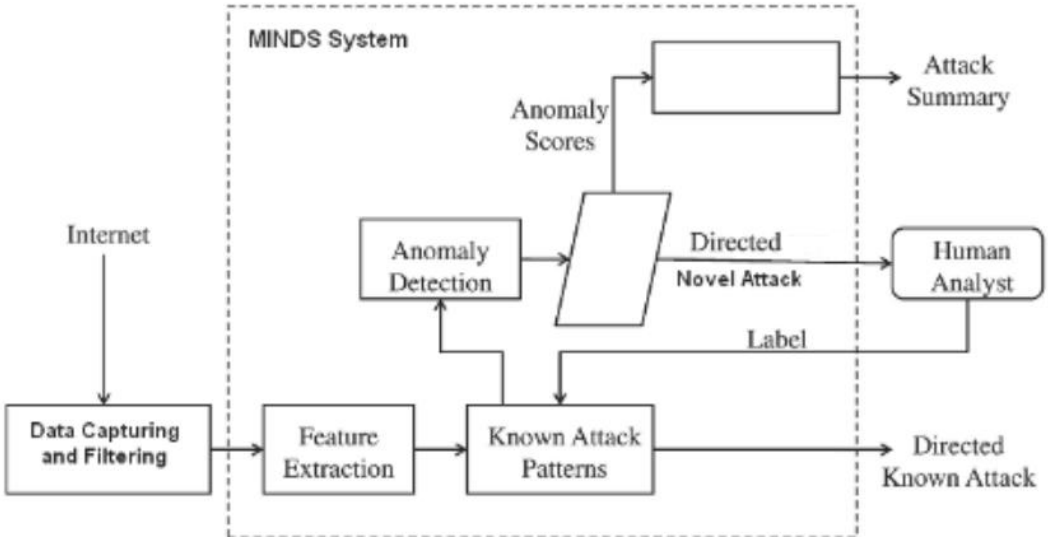

Рис. 2.3 - Архітектура системи MINDS[1]

3) Виявлення аномалій за допомогою асоціативного пошуку

Пошук асоціативних правил (ARM) ідентифікує події, які часто виникають разом. Асоціативне правило — це імплікація виду A → B, де A і B — набори значень атрибутів, з конкретними показниками підтримки та впевненості:

- Підтримка (m%): Відсоток транзакцій, що містять як A, так і B
- Впевненість (t%): Відсоток транзакцій, що містять A, які також містять B

Приклад: Аналіз та видобуток аудиторських даних (ADAM)

ADAM поєднує пошук асоціативних правил і класифікацію для виявлення атак у даних мережевого аудиту:

1. Будує репозиторій “нормальних” частих наборів елементів під час періодів без атак
2. Використовує алгоритм ковзного вікна для пошуку частих наборів елементів із поточних з’єднань
3. Порівнює їх із збереженими нормальними наборами елементів для виявлення підозрілих з’єднань
4. Застосовує навчений класифікатор для категоризації підозрілих з’єднань як відомі атаки, невідомі типи або помилкові тривоги

ADAM складається з трьох модулів(Рис. 2.4, 2.5):

- Препроцесор: Вилучає інформацію заголовків з трафіку TCP/IP
- Модуль видобутку: Видобуває асоціативні правила в режимі навчання або виявлення
- Модуль класифікації: Класифікує неочікувані правила як нормальні або аномальні події

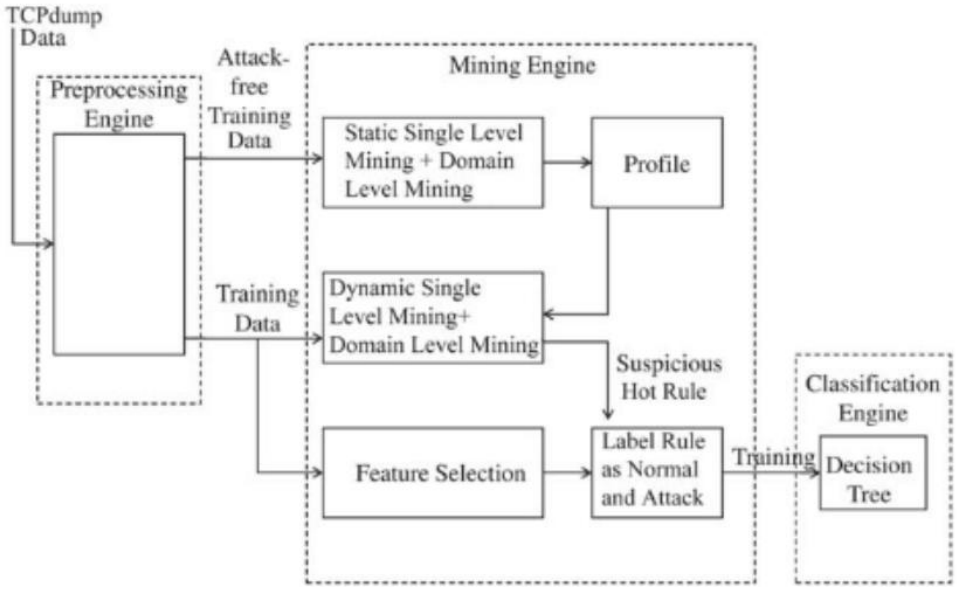

Рис. 2.4 - Фаза навчання ADAM

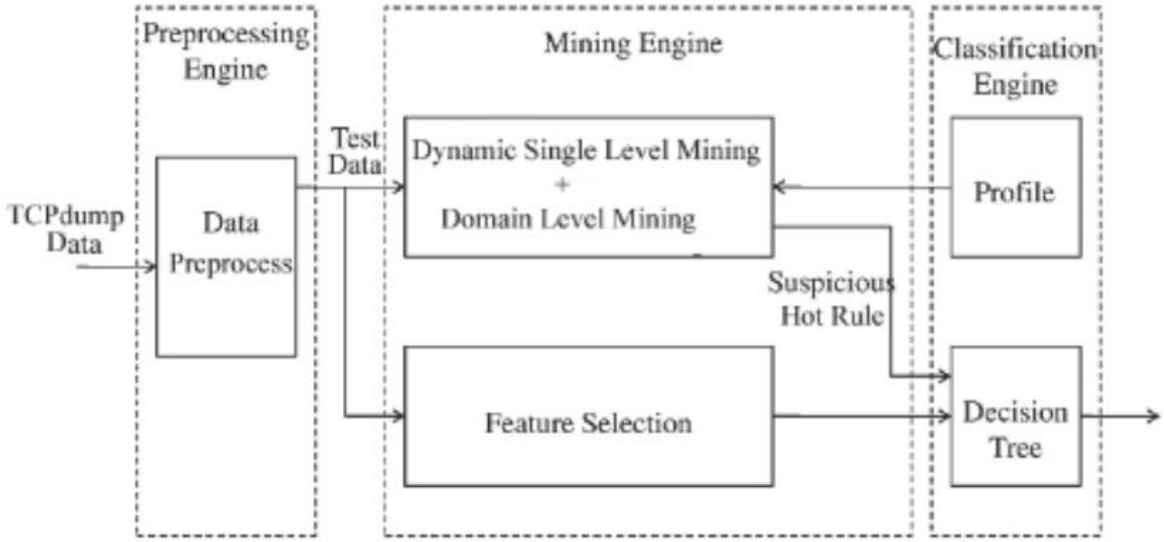

Рис. 2.5 - Режим виявлення за допомогою ADAM

Наведемо деякі способи використання методів навчання без вчителя на практиці у таблиці 2.3

Таблиця 2.3

Практичні методи

Метод	Опис
Кластеризація часових послідовностей	Метод виявлення аномалій, залежний від профілю користувача. Використовує технологію кластеризації часових послідовностей для забезпечення виявлення вторгнень у реальному часі з збором та обробкою даних на основі хоста. У фазі навчання створює профілі користувачів, а у фазі тестування перевіряє дані потоку команд користувача відносно відповідних профілів.
Виявлення рідкісних подій на основі правил	Вивчає правила для пошуку рідкісних подій у номінальних часових рядах даних з довгими залежностями. Знаходить аномалії в мережевих пакетах та нові сеанси вторгнення TCP.
Детектор аномалій на основі корисного навантаження	Представляє детектор аномалій на основі корисного навантаження. Моделює нормальне корисне навантаження додатків мережевого трафіку повністю автоматично, без вчителя та дуже ефективно. Обчислює розподіл частоти байтів профілю та стандартне відхилення корисного навантаження додатку, що надходить до одного хоста та порту під час фази навчання.
k-means та Naive Bayes (KMNB)	Представляє модель кластеризації k-means та класифікатора Naive Bayes. Використовує кластеризацію для групування зразків на зловмисні та незловмисні класи. Класифікатор NB використовується для класифікації даних на кінцевому етапі в точні класи.

Перевагами методів виявлення аномалій без учителя є те, що вони не потребують попередньо маркованих даних, тобто можуть функціонувати без

заздалегідь класифікованих прикладів атак. Коли кількість кластерів (k) визначено точно, кластеризація на основі розбиття стає ефективною. Підходи, що використовують кластеризацію за щільністю або пошук викидів, інтуїтивно відповідають задачам виявлення вторгнень. Крім того, групування великих обсягів даних у кластери суттєво знижує обчислювальне навантаження при аналізі. Після формування профілів і відповідних асоціативних правил виявлення відомих атак стає простим і швидким.

Разом з тим, такі підходи мають *низку недоліків*. Більшість методів найкраще працюють із неперервними атрибутами, що обмежує їх універсальність. Часто вони припускають, що великі кластери є нормальними, а малі — потенційно шкідливими, що не завжди є вірним. Якість виявлення може суттєво залежати від вибраної метрики близькості: невідповідна метрика знижує ефективність класифікації. Крім того, динамічне оновлення поведінкових профілів вимагає значних ресурсів і часу. І нарешті, під час аналізу рідкісних подій традиційні фреймворки асоціативного видобування, засновані на підтримці та впевненості, можуть давати хибні результати через упередженість до домінуючих класів.

Загалом, вибір підходу без учителя для виявлення мережевих аномалій має ґрунтуватися на особливостях середовища, обсягах доступних обчислювальних ресурсів та типах атак, які необхідно виявляти. Кожен метод має свої переваги, які можуть бути ефективними залежно від конкретного сценарію.

Крім класичних алгоритмів, у виявленні аномалій використовують ймовірнісні моделі (наприклад, HMM, GMM, Naive Bayes), м’які обчислення (нейронні мережі, генетичні алгоритми, нечітка логіка), знаннєві системи (Snort, онтології), а також комбіновані підходи (ансамблі, гібриди SVM і дерев рішень). Ці методи підвищують гнучкість і точність, але потребують якісних даних і складнішої розробки. Загалом кожен із методів має свої переваги й обмеження, а вибір залежить від типу трафіку, доступності даних і вимог до системи. На практиці важливими залишаються точність, швидкодія та здатність працювати в реальному часі.

2.2. Підготовка, очищення та формування ознак на основі датасету CICIDS2017

Виділення ознак (англ. *feature extraction*) є одним із ключових етапів у будь-якому проєкті з машинного навчання[8]. На цьому етапі сирі, часто неструктуровані або надмірно багатовимірні дані перетворюються у компактні, репрезентативні та інформативні характеристики, які краще відображають приховану структуру. Метою є не просто скоротити обсяг інформації, а зберегти її суть, тобто залишити лише те, що дійсно має вплив на прогнозування. Цей процес охоплює цілу низку дій: від вибору найбільш релевантних характеристик і зменшення розмірності (dimensionality reduction), до побудови нових похідних ознак (feature construction) та попередньої обробки — такої як нормалізація, логарифмування, масштабування або перетворення змінних у більш зручний для алгоритму формат. Якщо пояснити прикладом: у реальному житті ми не аналізуємо об’єкт за всіма можливими властивостями, а концентруємось на ключових ознаках. Так і модель — вона не повинна “читати все”, вона має “бачити головне”.

Виділення ознак є критично важливим із кількох причин. По-перше, це підвищує точність моделі. Алгоритми машинного навчання мають обмежену здатність абстрагуватись від шуму та другорядних залежностей. Якщо модель навчається на сирих даних, вона або взагалі не побудує корисних закономірностей, або побачить помилкові — тобто знайде патерни там, де їх немає. Якісно відібрані ознаки дозволяють алгоритму тренуватись швидше, з меншою похибкою і надійніше узагальнювати правила з навчальної вибірки.

Другою перевагою є зменшення ризику перенавчання (overfitting). Якщо модель отримує надто багато несуттєвих або корельованих ознак, вона починає “запам’ятовувати” дані, а не вивчати їх. Це означає, що вона працює добре на тренувальному наборі, але погано — на нових, реальних даних. Видалення зайвих параметрів дозволяє досягти кращої генералізації і збільшити стійкість до шумів.

Третій аспект — це обчислювальна ефективність. Кожна зайва ознака вимагає додаткових ресурсів для збереження, обробки та аналізу. Це особливо критично при

роботі з високочастотними даними або потоками великого обсягу — наприклад, у сфері обробки мережевого трафіку, де кожна секунда містить тисячі записів. Виділення ключових параметрів дозволяє істотно скоротити час тренування моделей та ресурсоспоживання в режимі реального часу.

Окремо варто згадати про візуалізацію та інтерпретованість. Дані, що зведені до 2–3 репрезентативних вимірів, можна не лише подати графічно (наприклад, через PCA або t-SNE), але й пояснити — як керівництву, так і користувачу. Це важливо як з погляду довіри до системи, так і в контексті юридичної відповідальності у сферах, пов’язаних із безпекою.

Отже, виділення ознак має такі наслідки: покращується точність класифікації або кластеризації; скорочується час тренування; підвищується загальна продуктивність моделей; зменшується ризик перенавчання; знижується навантаження на обчислювальні ресурси; а також поліпшується можливість пояснення результатів і їх практичного використання.

Важливу роль цей етап відіграє також у задачах виявлення аномалій у мережевому трафіку — як в даній роботі, де аналізується датасет CICIDS2017. Тут кожен запис — це окремий мережевий потік, описаний десятками параметрів: кількість байтів, прапори TCP, час між пакетами, кількість з’єднань з тим самим хостом, середня довжина пакета тощо. Часто дані містять дублікати, постійні значення або взаємно скорельовані колонки, що не тільки заважають моделі, а й створюють хибне уявлення про структуру трафіку. Без якісного виділення ознак, модель або не зможе виявити аномалії, або видасть занадто багато хибнопозитивних результатів. Далі буде розглянуто, як ці принципи реалізовані на практиці при підготовці даних CICIDS2017[7] до задачі виявлення аномального трафіку.

CICIDS2017 — це один із найпопулярніших датасетів для дослідження систем виявлення вторгнень. Його було створено в Canadian Institute for Cybersecurity з метою подолання обмежень застарілих наборів даних, які страждали від низької різноманітності трафіку, відсутності сучасних атак або неповних міток. Дані були зібрані протягом 5 днів активного мережевого моніторингу (3–7 липня

2017 року) з моделюванням реалістичного трафіку від 25 користувачів, які працювали з такими протоколами, як HTTP, HTTPS, FTP, SSH та email. Крім цього, були виконані актуальні атаки: Brute Force, DoS, DDoS, Web-атаки (SQLi, XSS), Botnet, Infiltration, Port Scan, Heartbleed тощо. Уся мережа була побудована з урахуванням повної топології, включаючи firewall, NAT, сервери DNS, комутатори, а також різні операційні системи на стороні клієнтів (Windows, Ubuntu, MacOS). Трафік повністю зафіксовано через mirror-порт і представлено у форматі PCAP-файлів та CSV-таблиць, згенерованих інструментом CICFlowMeter, який формує понад 80 мережевих ознак для кожного потоку. Особливу увагу приділено маркуванню даних, що дозволяє точно ідентифікувати моменти атак та їх типи. Таким чином, CICIDS2017 забезпечує повноту, різноманітність і якість, необхідну для побудови, тренування та тестування моделей виявлення аномалій у мережах. Цей датасет повністю задовольняє усі 11 критерії якості IDS-датасетів (Gharib et al., 2016), що робить його еталоном для сучасних досліджень у сфері кібербезпеки та машинного навчання.

Попри свою високу якість, сирий датасет містить ряд проблем, таких як відсутні значення, дублікатні записи, неконсистентність ознак, наявність зайвих або слабкоінформативних стовпців, а також сильний дисбаланс класів[13]. Тому для підготовки до моделювання було виконано низку етапів з використанням Python, бібліотеки pandas, а також методів статистичного аналізу та візуалізації даних[6]:

1) Об'єднання CSV-файлів. Оригінальний датасет складався з 8 окремих CSV-файлів, які були об'єднані в єдиний набір для комплексного аналізу. Після злиття було виявлено 2,830,743 зразки та 79 ознак. Первинний огляд показав надмірну кількість ознак, суміш числових і категоріальних типів, а також незначну кількість пропущених значень (<0.1%) у колонках Flow Bytes/s і Flow Packets/s.

2) Видалення дублікатів За допомогою методу drop_duplicates() було видалено 308,381 дублікатів рядків, а також усунено дублікати колонок, залишивши 67 унікальних ознак. Це підвищило цілісність даних і уникнуло зайвої ваги окремих трафікових шаблонів у моделі.

3) Обробка нескінченних значень. Нескінченні значення, що виникли внаслідок операцій ділення (наприклад, у Flow Bytes/s), були замінені на NaN.

4) Обробка пропущених значень. Усі рядки з пропущеними значеннями (менше 1% датасету) були видалені. Це дозволило уникнути спотворення моделі через імпутацію. Фінальний розмір датасету склав 2,520,751 зразок.

5) Очищення назв колонок. . Було видалено пробіли на початку та в кінці назв ознак для забезпечення уніфікації і зручності при обробці.

6) Видалення ознак з єдиним значенням. Було вилучено всі ознаки, які містили лише одне унікальне значення у всіх зразках, адже вони не несуть жодного значення для класифікації.

7)Аналіз кореляцій Проведено матричний аналіз кореляцій, виявлено пари з майже ідеальним зв'язком ($r \geq 0.99$), з яких одна ознака з кожної пари була видалена. Наприклад:

- Total Length of Fwd Packets ↔ Subflow Fwd Bytes
- Total Length of Bwd Packets ↔ Subflow Bwd Bytes. Це дозволило уникнути мультиколінеарності та підвищити інтерпретованість моделей.

8) Оцінка важливості ознак (H-статистика та Random Forest). Застосовано:

- Крускала-Валліса H-тест — для визначення статистично значущих ознак щодо типів атак.
- Random Forest — для машинного відбору ознак на основі дерева рішень.

У результаті було збережено 53 найінформативніші ознаки, а інші, як-от Fwd URG Flags або Idle Std, — відфільтровано.

*9) Групування міток атак*Колонка Label містила 15 типів атак, з яких деякі мали менше 50 зразків. З метою покращення узагальнення моделі:

- подібні атаки були згруповані (наприклад, DoS Hulk і DoS GoldenEye → DoS);
- рідкісні типи (Infiltration, Heartbleed) видалено. Фінальні класи: Normal Traffic, DoS, DDoS, Port Scanning, Brute Force, Web Attacks, Bots.

10)Попередній аналіз дисбалансу класів

Клас Normal Traffic становить 83.1% усіх зразків, що може призвести до упередженості моделі. Тому балансування (SMOTE або ваги класів) буде реалізовано на етапі тренування, а не попередньо, щоб уникнути витоку інформації в модель.

В підсумку для навчання моделей з датасету було виділено 53 ознаки, які охоплюють характеристики трафіку як у прямому, так і в зворотному напрямку. Серед них — порти, тривалість потоку, кількість і розміри пакетів, швидкість передачі, інтервали між пакетами (IAT), параметри заголовків, використання TCP-флагів (FIN, PSH, ACK), розміри вікна, активність, час простою, а також зведені статистичні характеристики (середні значення, максимум, мінімум, дисперсія). Цільовою змінною виступав тип трафіку — нормальний або атакуючий.(розділений по класам)

2.3 Побудова та порівняння моделей виявлення аномалій

Перш ніж тренувати моделі машинного навчання, необхідно правильно підготувати дані. Це дозволяє уникнути помилок, пов’язаних із нерівномірним масштабом ознак, дисбалансом класів або некоректним оцінюванням. Дані зазвичай розділяються на два основні набори: *тренувальний і тестовий*. Тренувальний набір використовується безпосередньо для навчання моделі, тоді як тестовий — не бере участі в процесі навчання і застосовується лише для остаточної перевірки того, як модель працює з новими, раніше невідомими даними. У межах тренувального набору додатково застосовується k-кратна крос-валідація(k-fold cross-validation) — це техніка, яка підвищує надійність оцінки моделі. Вона полягає в тому, що дані кілька разів розбиваються на частини, і модель проходить етапи тренування та тестування на різних комбінаціях, що дозволяє виявити стабільність її роботи.

Ще одним важливим кроком є масштабування ознак (feature scaling)[13] — тобто приведення числових параметрів до одного масштабу. Це особливо важливо

для алгоритмів, що базуються на відстанях, таких як KNN, де одна “вимірювана” ознака не повинна переважати інші лише через свої великі значення. Залежно від характеру даних обираються різні типи масштабування: StandardScaler — для нормально розподілених даних, MinMaxScaler — для ознак із чіткими межами, та RobustScaler — для даних з великою кількістю викидів. Викиди — це поодинокі значення, що сильно відрізняються від інших і можуть спотворювати результати. У нашому випадку, оскільки в багатьох ознаках присутні значні викиди, обрано RobustScaler, що забезпечує стійкість моделі до таких аномалій.

Останнім важливим етапом підготовки є балансування класів. У вихідному датасеті спостерігається значний дисбаланс: нормального трафіку у десятки разів більше, ніж атак, що може призвести до “ігнорування” рідкісних, але важливих класів атак. Щоб компенсувати цей ефект, використовується поєднання двох технік. По-перше, ми зменшуємо кількість прикладів класу «Normal Traffic» до 500 000 записів (undersampling). По-друге, за допомогою методу SMOTE (Synthetic Minority Over-sampling Technique) штучно генеруємо нові приклади менш представлених класів. Це дозволяє досягти кращого балансу та навчити модель ефективно виявляти навіть рідкісні типи атак.

Техніки оцінки[8]:

Оцінювання моделей здійснюється поетапно, починаючи з перевірки на тренувальному наборі за допомогою k-кратної крос-валідації. Як згадувалося раніше, ця техніка полягає у тому, що дані діляться на кілька частин, і модель тренується на одних частинах, а тестується на інших. Це дозволяє обчислити середню точність класифікації на різних підмножинах тренувальних даних і зрозуміти, наскільки стабільно модель поводиться при зміні розподілу даних. Результатом є середнє значення точності та стандартне відхилення, які показують узагальнену ефективність моделі на навчальних даних.

Після цього виконується фінальна оцінка на окремому тестовому наборі, який не використовувався в процесі навчання. Тут розраховується показник accuracy, який демонструє, яка частка усіх передбачень моделі є правильними. Формально:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

Цей показник дозволяє оцінити загальну якість класифікації на нових, раніше невідомих даних.

Наступним кроком є *перевірка продуктивності моделі* — вимірюються витрати ресурсів під час навчання, зокрема споживання оперативної пам'яті, середнє та пікове навантаження на процесор і загальний час тренування. Ці метрики є критичними для практичного застосування моделей у середовищах з обмеженими ресурсами або в режимі реального часу.

Детальнішу картину дає *класифікаційний звіт* (classification report), який формується на основі тестового набору і містить ключові метрики для кожного класу.

Precision показує, скільки з передбачених як «атака» випадків дійсно були атаками

$$precision = (TP / (TP + FP)).$$

Recall демонструє, яку частку реальних атак модель змогла правильно виявити

$$Recall = (TP / (TP + FN)).$$

Для збалансованої оцінки використовується F1-score, що є гармонічним середнім між precision і recall:

$$F1 = 2 \times (precision \times recall) / (precision + recall).$$

Окрім цього, support вказує кількість прикладів кожного класу у тестовому наборі — чим більше значення, тим більш надійною є відповідна метрика.

Порівняння алгоритмів для створення моделей

1) *Random Forest* — ансамблевий метод машинного навчання, що базується на поєднанні кількох дерев рішень. Його основною перевагою є стійкість до варіацій у даних, а також здатність працювати з немасштабованими ознаками без

суттєвого впливу на точність[2]. Це робить Random Forest надійним вибором для роботи з великими, “грубими” мережевими даними, де не завжди можливо точно нормалізувати вхідні характеристики.

Для досягнення максимальної ефективності моделі було проведено підбір гіперпараметрів за допомогою RandomizedSearchCV[13]. Цей підхід дозволяє випадковим чином перебирати комбінації параметрів і знаходити ті, які демонструють кращу продуктивність, ніж стандартні значення. На відміну від GridSearchCV, що перебирає всі можливі варіанти, RandomizedSearchCV є менш ресурсоємним і підходить для систем із обмеженою обчислювальною потужністю, як у нашому випадку.

Підібрані гіперпараметри[6]:

- n_estimators = 200 — кількість дерев у моделі. Більше дерев підвищує стабільність, але збільшує час навчання.
- min_samples_split = 5 — мінімальна кількість зразків для розбиття вузла. Впливає на глибину дерева і запобігає перенавчанню.
- min_samples_leaf = 2 — мінімальна кількість зразків у листі дерева. Дозволяє уникнути надто дрібних (нестійких) рішень.
- max_features = ‘sqrt’ — кількість ознак, які використовуються для кожного дерева. Значення ‘sqrt’ пришвидшує навчання та зменшує кореляцію між деревами.
- max_depth = None — відсутність обмеження на глибину дерева, що дозволяє моделі адаптуватися до складних структур даних.

Після навчання моделі виконано її оцінку:

Спершу було проаналізовано середнє значення точності на крос-валідації, що дало змогу оцінити стабільність моделі під час тренування на різних підмножинах даних.

Результат: середня точність = 0.9987, стандартне відхилення = 0.0001.

Це свідчить про надзвичайну узгодженість результатів та відсутність серйозних коливань між підмножинами даних.

Далі оцінено accuracy на тестовому наборі — *модель досягла 0.9988*, що підтверджує її здатність узагальнювати нові, невідомі дані з високою точністю.

Оцінка продуктивності моделі показала наступне:

- *Споживання пам'яті: 3063.80 МБ*
- *Час навчання: 280.25 секунд*
- *Пікове завантаження CPU: 100%*
- *Середнє завантаження CPU: 99.38%*

Для глибшого аналізу якості класифікації було сформовано класифікаційний звіт, який містить precision, recall та F1-score для кожного класу. Це дозволяє оцінити, наскільки добре модель розпізнає конкретні типи атак(Рис. 2.6)

	precision	recall	f1-score	support
Bots	0.71	0.83	0.77	584
Brute Force	1.00	1.00	1.00	2745
DDoS	1.00	1.00	1.00	38404
DoS	1.00	1.00	1.00	58124
Normal Traffic	1.00	1.00	1.00	628518
Port Scanning	0.99	1.00	0.99	27208
Web Attacks	0.98	0.97	0.98	643
accuracy			1.00	756226
macro avg	0.95	0.97	0.96	756226
weighted avg	1.00	1.00	1.00	756226

Рис. 2.6 - звіт класифікації для Random Forest

На завершення можна зробити висновок, що при середньому значенні крос-валідації 0.9987 та точності на тесті 0.9988 модель Random Forest демонструє відмінну ефективність без ознак перенавчання чи недонавчання. Особливо важливою є покращена recall для класу 'Bots' — зросла з 0.74 до 0.83 (в порівнянні з моделю тренованую без оптимізації гіперпараметрів), що критично важливо для

систем виявлення загроз. Хоча precision для цього класу трохи зменшився (з 0.86 до 0.71), цей компроміс вважається прийнятним, оскільки в системах типу NIDS набагато небезпечніші пропущені атаки, ніж хибні спрацьовування. Отже, оптимізація гіперпараметрів дала позитивний ефект і зробила модель більш чутливою до важливих, але рідкісних загроз.

2) XGBoost

Ще однією моделлю, що була розглянута в рамках цього дослідження, є XGBoost (Extreme Gradient Boosting)[2] — високоефективна реалізація градієнтного бустінгу. Вона добре зарекомендувала себе в задачах із дисбалансованими наборами даних, де клас атаки значно менш представлений, ніж нормальний трафік. На відміну від Random Forest, який створює всі дерева паралельно, XGBoost формує дерева послідовно, де кожне нове дерево навчається на помилках попереднього. Такий підхід дозволяє моделі краще адаптуватися до складної структури даних і часто досягати вищої точності.

Для вибору найкращих параметрів було застосовано RandomizedSearchCV, що дозволяє перебирати випадкові комбінації параметрів і знаходити ті, що забезпечують кращі результати. Це дозволяє зменшити навантаження на систему порівняно з GridSearchCV, який повністю перебирає всі варіанти.

Найкращі підібрані гіперпараметри для XGBoost:

- subsample = 1.0 — визначає, яку частину даних використовувати для побудови кожного дерева. Значення 1.0 означає використання всього обсягу доступних даних.
- n_estimators = 150 — кількість дерев, що входять до ансамблю. Впливає на загальну потужність моделі.
- min_child_weight = 1 — мінімальна вага (кількість) зразків у листовому вузлі. Менші значення дозволяють моделі навчатися на рідкісних шаблонах.
- max_depth = 3 — максимальна глибина дерева. Обмеження глибини допомагає уникати перенавчання.

- learning_rate = 0.3 — темп, з яким модель навчається. Чим менше значення, тим повільніше, але стабільніше навчання.
- colsample_bytree = 0.7 — частка ознак, які використовуються для кожного дерева. Це допомагає уникати перенавчання шляхом випадкової вибірки ознак.

Після тренування моделі було виконано оцінку її ефективності. Середня точність на крос-валідації склала 0.9991 при стандартному відхиленні 0.0001, що свідчить про дуже стабільні результати на різних підмножинах даних. На окремому тестовому наборі точність (accuracy) *склала 0.9990*, що підтверджує високу узагальнюючу здатність моделі.

Показники продуктивності моделі:

- *Споживання пам'яті: 3356.53 МБ*
- *Час навчання: 72.70 секунд*
- *Пікове завантаження CPU: 100%*
- *Середнє завантаження CPU: 99.01%*

	precision	recall	f1-score	support
Bots	0.70	0.95	0.80	584
Brute Force	1.00	1.00	1.00	2745
DDoS	1.00	1.00	1.00	38404
DoS	1.00	1.00	1.00	58124
Normal Traffic	1.00	1.00	1.00	628518
Port Scanning	0.99	1.00	0.99	27208
Web Attacks	0.99	0.99	0.99	643
accuracy			1.00	756226
macro avg	0.95	0.99	0.97	756226
weighted avg	1.00	1.00	1.00	756226

Рис. 2.7 - звіт класифікації для XGBoost

Детальний класифікаційний звіт(рис. 2.7) показав, що модель XGBoost особливо добре справляється з виявленням класу ‘Bots’ — метрика recall суттєво покращилась у порівнянні з Random Forest. Це є критично важливим для задач виявлення вторгнень, де непомічені атаки (false negatives) є значно більш небезпечними, ніж хибні спрацювання. У той же час відмінності в інших метриках, таких як precision, F1-score та загальна accuracy, є незначними, але із незначною перевагою на користь XGBoost. Таким чином, модель демонструє високий рівень точності й водночас ефективно виявляє критично важливі типи атак[8].

3) K-Nearest Neighbors (KNN)

Останньою з розглянутих моделей є *K-Nearest Neighbors (KNN)* — простий та інтуїтивно зрозумілий алгоритм класифікації, який визначає клас об’єкта на основі більшості серед його *k* найближчих сусідів у просторі ознак. Його основною особливістю є те, що він не будує внутрішню модель під час навчання, а зберігає дані і проводить класифікацію безпосередньо під час передбачення. Саме тому KNN називають “лінивим” алгоритмом навчання[13][2]. Однак модель є дуже чутливою до масштабу ознак, оскільки використовує метрики відстані. Тому перед тренуванням було обов’язково виконано масштабування вхідних даних.

Для налаштування моделі було застосовано метод RandomizedSearchCV, що дозволив перебрати кілька варіантів ключових гіперпараметрів:

- n_neighbors = 3 — кількість найближчих сусідів, які беруть участь у голосуванні. Менше значення забезпечує локальніші рішення, але підвищує ризик шуму.
- weights = ‘distance’ — вага сусідів залежить від відстані: чим ближчий сусід, тим більше впливу він має. Це дозволяє моделі краще адаптуватися до локальної структури даних і зменшити вплив віддалених точок.

Середня точність моделі на крос-валідації *становила 0.9881* при стандартному відхиленні 0.0007, що вказує на стабільність її роботи при зміні

тренувальних підмножин. Точність на тестовому наборі склала *0.9890*, демонструючи хорошу здатність узагальнювати нові дані.

Оцінка використання ресурсів показала дуже помірне навантаження:

- *Споживання пам'яті: 3018.07 МБ*
- *Час тренування: лише 1.84 секунди*
- *Пікове завантаження CPU: 28.6%*
- *Середнє завантаження CPU: 22.98%*

Це робить KNN привабливим з точки зору швидкої підготовки та розгортання моделі, особливо у системах, де необхідна низька затримка у відповіді на запит.

	precision	recall	f1-score	support
Bots	0.45	0.75	0.56	584
Brute Force	0.98	0.99	0.98	2745
DDoS	0.94	0.98	0.96	38404
DoS	0.95	0.99	0.97	58124
Normal Traffic	1.00	0.99	0.99	628518
Port Scanning	0.96	0.99	0.97	27208
Web Attacks	0.96	0.97	0.96	643
accuracy			0.99	756226
macro avg	0.89	0.95	0.91	756226
weighted avg	0.99	0.99	0.99	756226

Рис 2.8 - звіт класифікації для KNN

У класифікаційному звіті модель KNN також показала високі значення precision, recall та F1-score, проте трохи поступалася попереднім моделям — Random Forest та XGBoost[8], особливо у виявленні класу 'Bots'. Усі метрики для цього класу залишаються нижчими, що вказує на складність його виявлення незалежно від обраного алгоритму. Проте загальна ефективність KNN залишається

високою, що дозволяє розглядати цю модель як легкий і надійний варіант для задач виявлення аномалій.

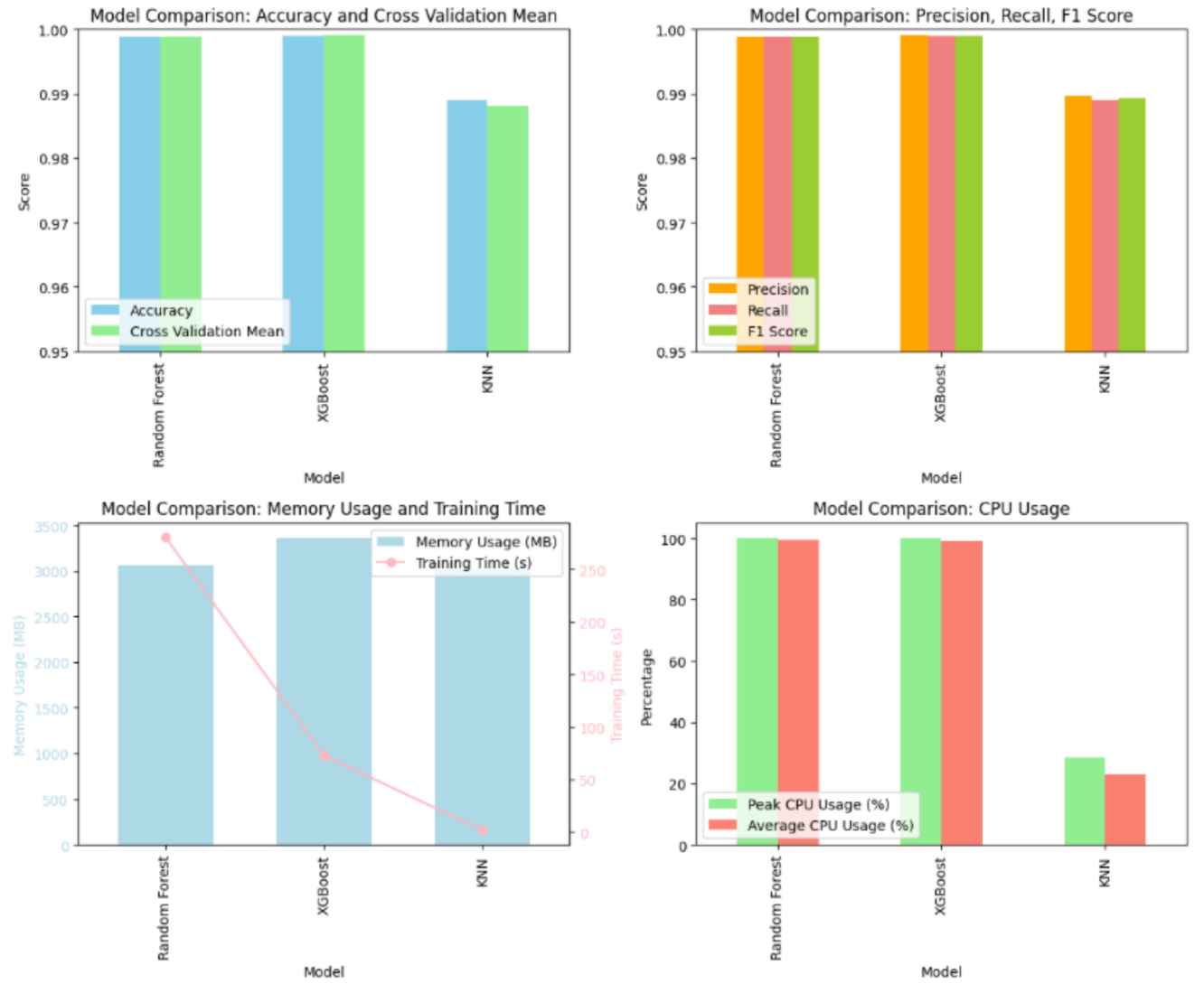

Рис 2.9 - Порівняння результатів всіх моделей

Графіки(Рис. 2.9) наочно ілюструють ефективність кожної моделі як на етапі тренування, так і при тестуванні. Моделі Random Forest (RF) та XGBoost (XGB) досягли найвищих значень точності та якості класифікації, однак це супроводжувалося значно вищими витратами ресурсів. З іншого боку, слід пам’ятати, що KNN є “лінивим” алгоритмом, тобто реальне навантаження на модель виникає не під час навчання, а під час передбачення нових даних. Тому справжню продуктивність KNN можна буде оцінити вже після розгортання прототипу та його тестування в режимі реального часу.

Unsupervised algorytms.

Для моделей без вчителя (unsupervised), навчання відбувається без використання міток — моделі самостійно аналізують структуру даних, щоб виявити відхилення, які можуть свідчити про аномалії[3]. Проте для оцінки ефективності такі мітки (y) все ж зберігаються та використовуються лише під час тестування, де класи спрощуються до бінарного вигляду: нормальний трафік (0) та атаки/аномалії (1).

На етапі підготовки:

- Ознаки (X) масштабуються за допомогою RobustScaler, що забезпечує стійкість до викидів, особливо важливу для K-Means, який чутливий до відстаней.
- Для Isolation Forest масштабування необов’язкове, оскільки цей алгоритм базується на деревах.

У цьому підході не застосовується крос-валідація, а навчання проводиться на повному тренувальному наборі. Моделі оцінюються на тестовому наборі з використанням ключових метрик:

- *Precision, Recall ma F1-score* — залишаються основними метриками, як і в попередньому розділі.
- *ROC-AUC* використовується лише для моделі Isolation Forest. У цьому випадку розраховується крива ROC, що показує, як змінюється співвідношення хибнопозитивних (FPR) і істиннопозитивних (TPR) відповідей при різних порогах.

1) Isolation Forest

Isolation Forest — алгоритм, що ізолює спостереження шляхом випадкового вибору ознаки та випадкового вибору значення розділення в межах діапазону цієї ознаки. Завдяки рекурсивному процесу ізоляції, аномальні об’єкти, що мають унікальні або рідкісні значення, ізолюються швидше, ніж нормальні[11].

У контексті виявлення мережевих атак Isolation Forest особливо корисний, оскільки:

- орієнтований саме на виявлення аномалій, а не нормальної поведінки;

- ефективний на багатовимірних даних;
- добре працює при великій перевазі одного класу над іншим, що характерно для мережевого трафіку, де атаки — рідкісні явища.

Хоча навчання відбувається без міток, для налаштування гіперпараметрів було використано власну функцію оцінки, яка враховувала F1-метрику на бінарних мітках (0 — нормальний трафік, 1 — атака). Враховуючи реальний відсоток атак у тренувальному наборі (приблизно 16.89%), було використано відповідне значення параметра contamination.

Найкращі підібрані гіперпараметри для Isolation Forest:

- n_estimators = 200 — кількість дерев в ансамблі. Більше дерев покращує здатність до ізоляції рідкісних зразків.
- max_samples = 512 — максимальна кількість зразків, що використовуються для побудови кожного дерева. Обмеження дозволяє зменшити перенавчання та пришвидшити тренування.
- contamination = 0.169 — очікувана частка аномалій у даних. Дозволяє моделі визначити, скільки прикладів потрібно вважати аномальними.
- max_features = 1.0 — частка ознак, що використовуються для кожного дерева. Значення 1.0 означає, що всі ознаки доступні для розділення.

Після тренування модель була протестована на відкладеному тестовому наборі. Аномальні значення були визначені на основі ROC-кривої, яка показує співвідношення між TPR (істинно позитивними) та FPR (хибнопозитивними) при різних порогах. Для обрання оптимального порогу було використано статистику Юдена (Youden’s J), що максимізує різницю між TPR і FPR, забезпечуючи збалансовану якість виявлення.

Підсумкові метрики бінарної класифікації:

- *Accuracy: 0.7781*
- *Precision: 0.3972*
- *Recall: 0.6064*
- *F1-score: 0.4800*

Хоча результати поступаються моделям з учителем, вони ілюструють типову ситуацію для алгоритмів без вчителя, які не мають доступу до міток під час навчання. Це підтверджується також іншими дослідженнями, наприклад, Sirisha et al. (2021), де було виявлено подібні труднощі з досягненням високих показників на наборі CICIDS2017[7] через нечітке розділення ознак між нормальним трафіком та атаками.

Оцінка використання ресурсів:

- *Споживання пам'яті: 3538.82 МБ*
- *Час навчання: 142.19 секунд*
- *Пікове завантаження CPU: 90%*
- *Середнє завантаження CPU: 27.87%*

	precision	recall	f1-score	support
Normal	0.91	0.81	0.86	628518
Attack	0.40	0.61	0.48	127708
accuracy			0.78	756226
macro avg	0.65	0.71	0.67	756226
weighted avg	0.82	0.78	0.79	756226

Рис 2.10 - звіт для Isolation forest

Результати(Рис. 2.10) показують, що модель досягає precision = 0.91 та recall = 0.81 для класу Normal, що свідчить про високу точність при виявленні нормального трафіку, хоча деякі нормальні зразки все ж залишаються неідентифікованими (recall = 0.81). Для класу Attack точність нижча — 0.40, тобто значна частина передбачених атак є хибнопозитивними (нормальні приклади

помилково класифікуються як атаки). Водночас, recall для Attack = 0.61, що означає, що модель правильно виявляє 61% реальних атак, хоча є простір для покращення щодо зменшення кількості пропущених атак (false negatives).

Узагальнюючи, модель має F1-score = 0.86 для класу Normal і 0.48 для Attack, що демонструє компроміс між precision та recall. Загальна точність (accuracy) становить 0.78, що свідчить про помірну ефективність, але також вказує на труднощі з одночасним якісним виявленням обох класів, особливо менш представленого класу Attack. Макро-середній F1-score = 0.67, а зважений F1-score = 0.79, що вказує на те, що попри дисбаланс класів модель демонструє загалом хорошу продуктивність. Проте для покращення здатності моделі до виявлення атак без значного зниження точності доцільно розглянути подальше налаштування порогу класифікації або гіперпараметрів.

2) K means

Останньою моделлю, яку було протестовано в межах цього дослідження, стала K-means — алгоритм кластеризації без вчителя, що ділить дані на k кластерів на основі метрик відстані. На відміну від моделей з учителем, K-means не потребує міток для тренування, що робить його корисним у ситуаціях, коли марковані дані відсутні або обмежені. У системах виявлення вторгнень (NIDS) цей алгоритм може бути ефективним у виявленні нових або невідомих атак, кластеризуючи нормальний трафік в один кластер, а решту — як потенційно аномальні.

Оскільки K-means вимагає заздалегідь вказати кількість кластерів k[4], було проведено експериментальне тестування з використанням трьох метрик(рис 2.11):

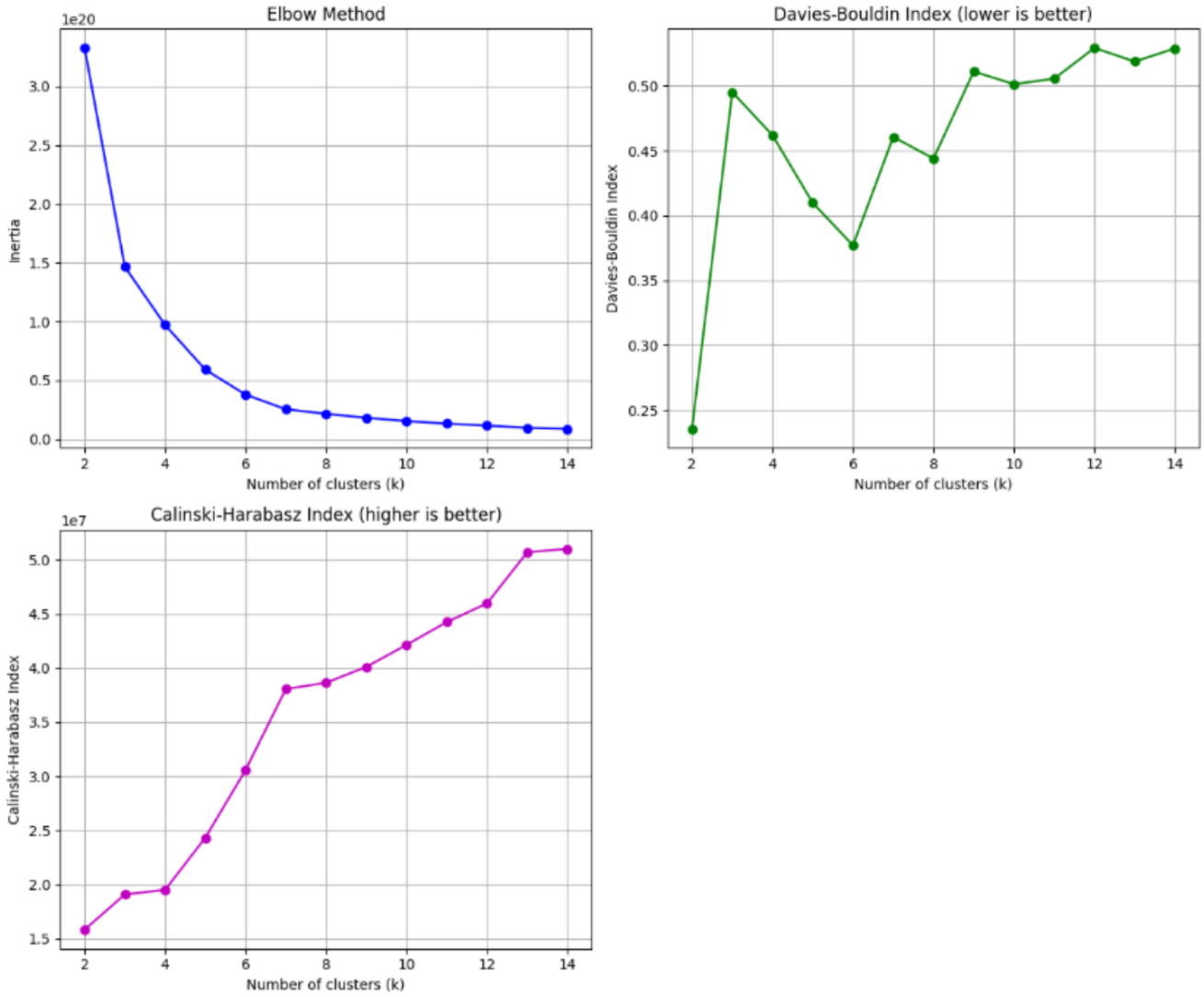

Рис. 2.11— графіки для визначення оптимальних кластерів

На рисунку 2.11 зображено результати трьох основних метрик, які допомагають визначити оптимальну кількість кластерів для алгоритму K-means:

- Elbow Method (Інерція) — інерція відображає суму квадратів відстаней між кожною точкою та центроїдом її кластера. Ідея методу полягає у знаходженні “зламу” (elbow) на графіку: точка, після якої додавання нових кластерів не дає суттєвого зменшення інерції. На графіку помітно різке падіння до k = 3, після чого зменшення уповільнюється. Це вказує, що 3 кластери можуть бути оптимальним значенням, бо подальше збільшення k не дає суттєвого приросту якості кластеризації.
- Davies-Bouldin Index — ця метрика вимірює, наскільки добре кластери відокремлені один від одного. Чим нижче значення, тим краще. Графік демонструє

найнижче значення при k = 2, що теоретично є найкращим, але також видно локальний мінімум при k = 6, що може свідчити про структурні зміни в розподілі даних.

- Calinski-Harabasz Index — оцінює якість кластеризації на основі міжкластерної дисперсії та внутрішньокластерної щільності. Чим вище значення, тим краще. Графік показує стабільне зростання з піковим значенням на k = 12-14, однак зростання уповільнюється після k = 7, що може свідчити про насичення структури.

Результати трьох метрик дають неоднозначну картину, що типово для реальних даних. Elbow Method показує k = 3 як оптимальне значення, яке забезпечує суттєве зменшення інерції без надмірної складності. Davies-Bouldin Index натякає на простішу структуру при k = 2, але також демонструє цікаву поведінку при k = 6. Calinski-Harabasz Index підтримує поступове ускладнення моделі, але з повільним зростанням після k = 7.

Беручи до уваги ці результати, а також локальні експерименти з тренування моделі, було прийнято рішення зупинитися на k = 3, як оптимальному для даного завдання. Це дозволяє зберегти баланс між точністю кластеризації та інтерпретованістю результатів, що критично важливо для застосування в системах виявлення аномалій у мережах (NIDS)[12].

Щоб адаптувати кластеризацію під задачу виявлення аномалій, найбільший кластер був умовно прийнятий як нормальний трафік, а решта — як потенційні аномалії. Це дозволило трансформувати результати кластеризації у бінарну форму для оцінки.

Оцінка використання ресурсів:

- *Споживання пам'яті: 3952.88 МБ*
- *Час навчання: 3.02 секунд*
- *Пікове завантаження CPU: 100%*
- *Середнє завантаження CPU: 62.01%*

	precision	recall	f1-score	support
Normal	0.89	0.95	0.92	628518
Attack	0.63	0.40	0.49	127708
accuracy			0.86	756226
macro avg	0.76	0.68	0.70	756226
weighted avg	0.84	0.86	0.85	756226

Рис. 2.12 - звіт для K-Means

На основі результатів кластеризації(рис 2.12) методом K-Means спостерігається висока загальна точність — 86%, однак із суттєвою різницею в ефективності між класами нормального трафіку та атак. Модель добре виявляє нормальний трафік, демонструючи високі показники precision (0.89), recall (0.95) та F1-міри (0.92). Це свідчить про те, що кластер нормального трафіку чітко відокремлений у простору ознак.

Водночас ефективність для класу атак значно нижча: precision — 0.63, recall — 0.40, F1-міра — 0.49. Це вказує на те, що модель має труднощі з точним виявленням і розмежуванням аномального трафіку, можливо, через відсутність чітких меж між типовою поведінкою і атаками. Особливо низьке значення recall означає, що модель пропускає значну кількість атак, що веде до високого рівня хибнонегативних спрацювань.

Макроусереднені значення (0.76 для precision, 0.68 для recall, 0.70 для F1) відображають дисбаланс між класами, тоді як зважені середні значення (0.84, 0.86 і 0.85 відповідно) є вищими завдяки сильній роботі моделі саме на нормальному трафіку.

У підсумку, хоча K-Means показує прийнятну загальну точність, він має суттєві обмеження у виявленні атак, що свідчить про доцільність подальшого тюнінгу або використання іншого алгоритму кластеризації, здатного краще впоратися зі складністю та дисбалансом набору CICIDS2017.

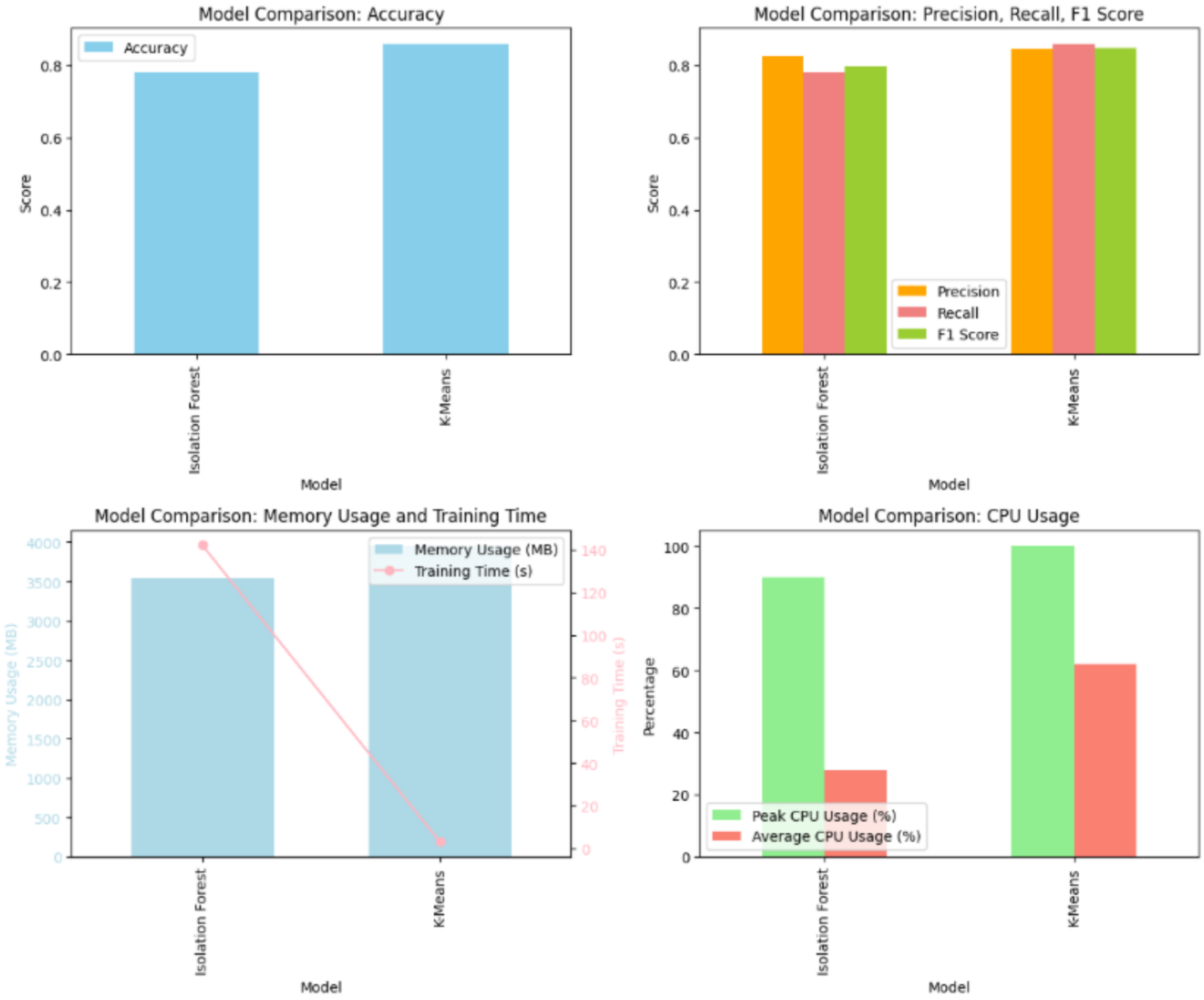

Рис. 2.13 - порівняння моделей

Аналіз ефективності виявлення: Алгоритми навчання без вчителя — Isolation Forest (IF) та K-Means — продемонстрували помітно нижчу продуктивність, порівняно з раніше навченими моделями з учителем. Такий результат відповідає висновкам з наукової літератури, особливо в контексті датасету CICIDS2017, де моделі без вчителя часто мають труднощі з точним виявленням. Хоча в інших наборах даних, наприклад UNSW-NB15, результати виявлялися кращими, CICIDS2017 створює унікальні виклики — зокрема, перекриття класів атак і відсутність чітких меж між нормальним та аномальним трафіком.

Поглиблена обробка даних могла б покращити результати[13]. Це може включати ретельніший відбір та інженерію ознак, з акцентом на ті, що найбільше впливають на виявлення аномалій[11]. Також варто розглянути зменшення розмірності, фільтрацію викидів або перетворення ознак, які можуть покращити здатність моделей відрізняти нормальний трафік від атак.

Обчислювальні ресурси:

З точки зору ефективності використання ресурсів, K-Means тренується швидше, ніж Isolation Forest, але водночас вимагає більше оперативної пам’яті та навантажує CPU, що робить його більш ресурсоємким варіантом, особливо для великих наборів даних. У свою чергу, Isolation Forest забезпечує більш збалансоване співвідношення між споживанням ресурсів і точністю виявлення. Примітно, що саме в плані продуктивності моделі без вчителя перевершили моделі з учителем, якщо не враховувати K-NN, який через свою архітектуру «лінивого навчання» відкладає основні обчислення на етап тестування.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ АНОМАЛІЙ У МЕРЕЖЕВОМУ ТРАФІКУ

3.1 Розробка NIDS на основі XGBoost

Методи і засоби: Для розробки [10][6] системи виявлення мережевих аномалій (NIDS) було використано мову програмування Python, яка відзначається гнучкістю, простим синтаксисом та широким набором бібліотек для аналізу даних, машинного навчання та мережевої взаємодії. Розробка та навчання моделі відбувалася в середовищах Kaggle та Google Colab, що дозволило скористатися обчислювальними ресурсами хмари для тренування моделей (XGBoost — як модель яка показала найкращі результати взято за основу в даному проєкті) на великому обсязі даних (датасет CICIDS2017).

Для реалізації NIDS були використані такі бібліотеки:

- *Scapy* — бібліотека для роботи з мережевими пакетами[16]. Вона дозволила організувати захоплення трафіку в реальному часі, обробку TCP/UDP/IPv4 пакетів, витягнення необхідних атрибутів (портів, прапорців, розмірів тощо), а також визначення потоків для подальшого аналізу.
- *XGBoost* — потужна бібліотека градієнтного бустингу, яка використовувалася як основна модель класифікації трафіку. Саме ця модель показала найвищу точність на тестовому наборі даних серед усіх протестованих.
- *Scikit-learn* — допоміжна бібліотека для обробки даних та обчислення ймовірностей передбачень моделі, збереження/завантаження моделі (разом із joblib), та інших супутніх завдань машинного навчання.
- *Pandas та NumPy* — використовувалися для побудови таблиць ознак (DataFrame), обчислення статистичних показників (середні значення, дисперсії, стандартні відхилення) та роботи з масивами даних.

- *Netifaces* — для автоматичного виявлення активного мережевого інтерфейсу на хості запуску, що дозволяє уникнути ручного налаштування мережевих параметрів.
- *Tkinter* — бібліотека для створення графічного інтерфейсу користувача. Була використана для реалізації моніторингу роботи NIDS в режимі реального часу, з відображенням кількості оброблених пакетів, списку виявлених атак та логів подій.
- *CSV, datetime, re, threading ma os* — стандартні бібліотеки Python, що забезпечують логування, регулярні вирази для фільтрації потоків, управління потоками та файлову систему.

Розробка системи виявлення мережевих аномалій відбувалась у декілька ключових етапів[6] *(у нашому випадку аномалії — це всі типи шкідливого трафіку, що класифікуються як ненормальні)*:

- На першому етапі було здійснено попередню обробку набору даних CICIDS2017, що включала очищення, нормалізацію та виділення релевантних ознак для навчання (детальніше у розділі 2.2).
- На другому етапі було протестовано декілька моделей машинного навчання. Найкращі результати продемонструвала модель XGBoost, яку й було обрано для подальшого впровадження (розділ 2.3).
- Третім етапом стала безпосередня розробка прототипу NIDS, що включає захоплення мережевого трафіку в реальному часі, обчислення ознак потоку, класифікацію трафіку та відображення результатів у графічному інтерфейсі.

Прототип був реалізований мовою Python і складається з двох основних компонентів:

- *NetworkAnomalyDetector* — клас, що відповідає за обробку мережевого трафіку, побудову ознак (features), виявлення аномалій та генерацію сповіщень.
- *NetworkAnomalyGUI* — графічний інтерфейс користувача, що дозволяє запускати/зупиняти захоплення трафіку, налаштовувати параметри аналізу, переглядати попередження у реальному часі та вести журнал виявлених атак.

Детальний опис програми:

Принцип роботи системи виявлення аномалій базується на поетапній обробці мережевого трафіку в режимі реального часу, подальшому аналізі характеристик мережевих потоків (flows) та класифікації[14] за допомогою заздалегідь натренованої моделі XGBoost. Нижче описано повний цикл роботи системи:

На початковому етапі програма автоматично визначає активний мережевий інтерфейс, з якого буде відбуватись захоплення трафіку. Це реалізовано за допомогою бібліотеки netifaces, яка надає інформацію про шлюзи та інтерфейси у системі:

```
INTERFACE = default_gateway[netifaces.AF_INET][1]
```

Далі, у класі NetworkAnomalyDetector, відбувається завантаження попередньо натренованої моделі XGBoost, що зберігається у форматі .joblib. Шлях до неї визначається динамічно відносно директорії запуску:

```
nids_script_dir = os.path.dirname(os.path.abspath(__file__))
MODEL_PATH = os.path.join(nids_script_dir,'../ml_models/supervised/xgboost.joblib')
with open(model_path, 'rb') as f:
    self.model = joblib.load(f)
```

Після цього запускається фоновий процес, що слухає інтерфейс у реальному часі. Для захоплення трафіку використовується бібліотека scapy, яка дозволяє перехоплювати пакети безпосередньо з мережевого інтерфейсу:

```
sniff(
 iface=interface,
 prn=self.process_packet,
 store=0,
 timeout=1,
 stop_filter=lambda x: not self.capture_running)
```

Кожен перехоплений пакет передається у функцію process_packet, де здійснюється його розбір. Зокрема, з пакету вилучаються IP-адреси джерела та призначення, протокол (TCP/UDP), порти та розмір. Пакети групуються у логічні потоки (flows), що ідентифікуються комбінацією IP-адрес, портів та протоколу.

Для кожного потоку ведеться статистика: кількість пакетів у кожному напрямку, кількість байтів, прапори TCP (SYN, ACK, FIN тощо), довжини заголовків, проміжки часу між пакетами (IAT), активні та неактивні періоди тощо. Усі ці характеристики накопичуються у структурі self.flow_stats.

Коли час життя потоку перевищує встановлений таймер (наприклад, 60 секунд), або коли зібрано достатню кількість пакетів (наприклад, більше 3), ініціюється процес виявлення аномалії. Спочатку з потоку витягуються статистичні ознаки через метод extract_features. Вони відповідають ознакам, що використовувались під час тренування моделі (наприклад: середня довжина пакету, кількість пакетів в секунду, середній IAT тощо).
Отримані ознаки конвертуються у таблицю DataFrame:

df = pd.DataFrame([features])

Ця таблиця передається моделі XGBoost для класифікації:

predicted_class_idx = self.model.predict(df)[0]
predicted_class = class_names[predicted_class_idx]

Кожен потік може бути класифікований як один з таких класів: нормальний трафік, DoS, DDoS, Port Scanning, Brute Force, Web Attacks або Bots. Якщо виявлено атаку і рівень впевненості моделі перевищує встановлений поріг (наприклад, 70%), генерується сповіщення, яке додається до журнала подій та може бути передане у графічний інтерфейс.

Інтерфейс користувача(Рис. 3.1), реалізований через Tkinter, дозволяє запускати/зупиняти процес захоплення трафіку, встановлювати поріг виявлення аномалій, переглядати кількість оброблених пакетів та переглядати історію сповіщень. Кожна аномалія фіксується з часовою міткою, IP-адресами, типом атаки та рівнем впевненості. Лог також зберігається у CSV-файл у папці nids_alerts, яка створюється автоматично, що дозволяє переглядати результати навіть після завершення роботи GUI.

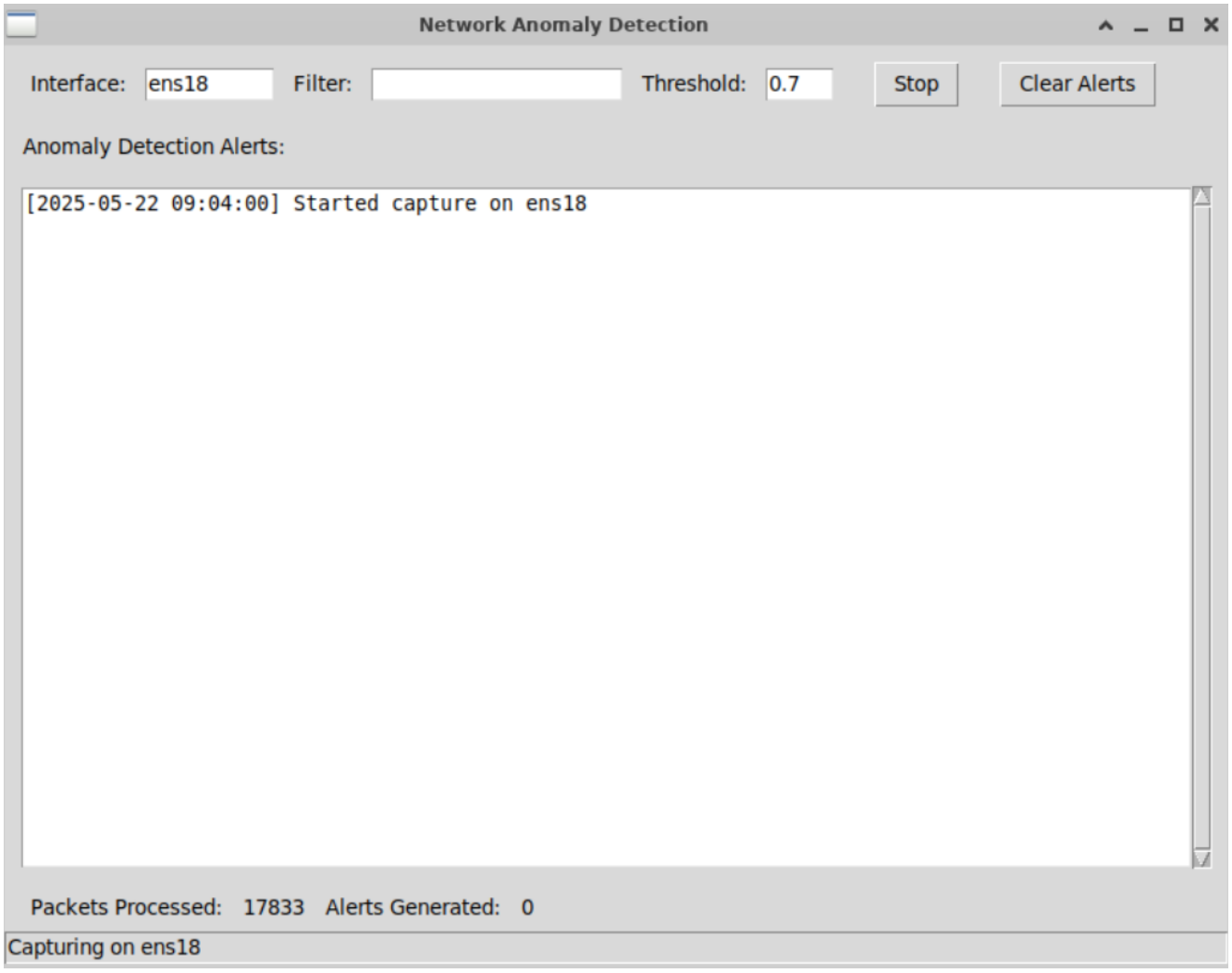

Рис. 3.1 — Інтерфейс користувача

Користувацький інтерфейс містить наступні елементи:

- Поле *Interface* — вибір інтерфейсу для захоплення трафіку (наприклад, ens18).
- Поле *Filter* — можливість задати фільтр для пакетів, як у Wireshark (напр. tcp або port 80).

- Поле *Threshold* — встановлення мінімального рівня впевненості для визначення атаки (наприклад, 0.7).
- Кнопка *Start/Stop* — запускає або зупиняє перехоплення трафіку.
- Кнопка *Clear Alerts* — очищає список попереджень на екрані.
- Область журналу — виводить усі аномалії, виявлені в реальному часі.
- Статусний рядок — показує загальну кількість оброблених пакетів та кількість згенерованих попереджень.

Варто зазначити, що система нативно підтримується лише у середовищі Linux, оскільки бібліотека Scapy[10] для захоплення трафіку потребує доступу до низькорівневих мережевих інтерфейсів через libpcap, що повноцінно реалізовано тільки в Unix-подібних операційних системах. Також на Linux необхідний графічний інтерфейс для коректної роботи Tkinter. У середовищі Windows запуск можливий, але потребує встановлення додаткових компонентів, таких як Npcap (альтернатива WinPcap), що дозволяє Scapy взаємодіяти з мережевими адаптерами.

Завдяки потоковому підходу та ефективному використанню бібліотек Scapy, joblib, pandas, numpy та tkinter, система забезпечує виявлення мережевих аномалій у реальному часі без потреби у складному розгортанні або зовнішніх залежностях.

Інструкція з розгортання та запуску системи

Для запуску програми необхідно виконати наступні кроки:

1. Операційна система.:

Рекомендовано використовувати систему на базі Linux з графічним середовищем (наприклад, GNOME або XFCE).

Альтернативно можливий запуск у середовищі Windows, однак для коректної роботи бібліотеки Scapy необхідно додатково встановити пакет Npcap.

2. Встановлення Python та Git

У Linux-системах у терміналі слід виконати такі команди:

sudo apt update

sudo apt install python3 python3-pip python3-venv git -y

3. Клонування репозиторію з кодом

git clone https://github.com/anacletu/ml-intrusion-detection-cicids2017.git

cd ml-intrusion-detection-cicids2017

4. Завантаження натренованої моделі(.joblib файл)

Після цього перемістити модель у каталог:

ml_models/supervised/

5. Створення віртуального середовища для Python у поточному каталозі

python3 -m venv venv

source venv/bin/activate

(У середовищі Windows: venv\Scripts\activate)

6. Встановлення необхідних залежностей всередині віртуального середовища

*pip install pandas>=2.2.2 numpy>=1.26.4 psutil>=5.9.0 joblib==1.4.2 *

*scikit-learn==1.5.1 xgboost==2.1.3 scipy>=1.13.1 logging>=0.5.1 *

scapy>=2.6.1 netifaces>=0.11.0

7. Запуск системи

Знайти файл nids_prototype_xgboost.py у кореневій директорії проєкту та виконати його:

python nids_prototype_xgboost.py

Після запуску відкриється вікно графічного інтерфейсу користувача (рис. 3.1), і система почне автоматичне захоплення та аналіз мережевого трафіку на обраному інтерфейсі. Результати будуть виводитися у реальному часі

3.2 Перевірка роботи системи на прикладах атак

Після запуску моделі, згідно з попередніми інструкціями (див. попередній розділ), наступним етапом є тестування системи шляхом симуляції різних типів атак. Для цього у репозиторії ml-intrusion-detection-cicids2017[6] у папці tests передбачено скрипт manual_test.py, що реалізує генерацію атак по всіх класах, передбачених навченою моделлю: Port Scanning, Brute Force, Web Attacks, DoS, DDoS, Bots.

Цей скрипт дозволяє виконувати тестування як по одному обраному типу, так і одразу всі типи по черзі, за допомогою відповідних параметрів командного рядка. Перед запуском важливо виконати наступні кроки:

- Переконатися, що на машині встановлені потрібні утиліти:

  sudo apt install nmap apache2-utils curl sshpass -y

  Ці інструменти використовуються для генерації відповідних атак: nmap — сканування портів, sshpass — SSH brute force, ab — DoS/DDoS, curl — атаки на веб-додатки.

- Запустити на машині де працює NIDS тестовий веб-сервер на порту 80, наприклад:

  python3 -m http.server 80 або *apt install nginx -y*

  Це необхідно, оскільки деякі атаки (SQL Injection, XSS, Directory Traversal) надсилаються на порт 80.

- За потреби відключити фільтрацію у файлі nids_prototype_xgboost.py: якщо машина з якою йде тестова атака знаходиться в локальній мережі, необхідно перевірити, щоб IP-адреса джерела не потрапляла у WHITELIST_PATTERNS, інакше трафік не буде класифікований як потенційна атака.

- Формат запуску скрипта:

  python3 manual_test.py --target <IP-АДРЕСА> --category <назва_категорії>

  Де <назва_категорії> може бути однією з: port_scanning, brute_force, web_attacks, dos, ddos, bots, або all.

Наприклад, щоб запустити повний набір тестів:
python3 manual_test.py --target 192.168.0.99 --category all

Скрипт по черзі генерує атаки відповідного типу, із затримкою між кожною (типово 30 секунд), після чого результати зберігаються в CSV-файл у директорії nids_test_results/.
Кожна категорія реалізована як окремий метод класу SimpleAttackTester:

- port_scanning_tests() — виконує базове та стелс-сканування портів, а також визначення версій сервісів.
- brute_force_tests() — здійснює спроби входу в SSH з випадковими логінами та паролями.
- web_attacks_tests() — імітує SQL-ін’єкцію, XSS та directory traversal.
- dos_tests() — запускає легке HTTP-навантаження з ab.
- ddos_tests() — здійснює інтенсивне flood-навантаження з більшим числом запитів.
- botnet_simulation_tests() — імітує C&C-зв’язок та ексфільтрацію даних.

Запуск скрипта дозволяє перевірити, як система реагує на різні сценарії атак та чи з’являються відповідні сповіщення у GUI.

Запустимо тест(Рис. 3.2) , бачимо що атаки йдуть один за одною у відповідний час, вони виконуються.

```
root@DESKTOP-HLUUP8P:~# python3 ml_test.py --target 192.168.0.99 --category all
2025-05-22 12:13:36,924 - INFO - Starting Simple NIDS Testing Framework
2025-05-22 12:13:36,925 - INFO - Configuration: target=192.168.0.99, category=all, delay=30s, output=nids_test_results
2025-05-22 12:13:36,926 - INFO - Starting port_scanning tests
2025-05-22 12:13:36,926 - INFO - Running Port Scanning attack: Basic port scan
2025-05-22 12:13:38,856 - INFO - Attack completed: Port Scanning (Success: True)
2025-05-22 12:14:08,857 - INFO - Running Port Scanning attack: Stealth scan
2025-05-22 12:14:11,895 - INFO - Attack completed: Port Scanning (Success: True)
2025-05-22 12:14:41,896 - INFO - Running Port Scanning attack: Service version detection
2025-05-22 12:14:49,719 - INFO - Attack completed: Port Scanning (Success: True)
2025-05-22 12:14:49,719 - INFO - Waiting 60 seconds before next category...
2025-05-22 12:15:49,720 - INFO - Starting brute_force tests
2025-05-22 12:15:49,721 - INFO - Running Brute Force attack: SSH login attempt as admin
2025-05-22 12:15:51,396 - INFO - Attack completed: Brute Force (Success: False)
2025-05-22 12:15:53,398 - INFO - Running Brute Force attack: SSH login attempt as root
2025-05-22 12:15:54,049 - INFO - Attack completed: Brute Force (Success: True)
2025-05-22 12:15:56,050 - INFO - Running Brute Force attack: SSH login attempt as root
2025-05-22 12:15:56,398 - INFO - Attack completed: Brute Force (Success: True)
2025-05-22 12:15:58,399 - INFO - Waiting 60 seconds before next category...
2025-05-22 12:16:58,400 - INFO - Starting web_attacks tests
2025-05-22 12:16:58,401 - INFO - Running Web Attacks attack: Directory traversal attempt
2025-05-22 12:16:58,474 - INFO - Attack completed: Web Attacks (Success: True)
2025-05-22 12:17:28,475 - INFO - Running Web Attacks attack: SQL injection attempt
2025-05-22 12:17:28,511 - INFO - Attack completed: Web Attacks (Success: True)
2025-05-22 12:17:58,511 - INFO - Running Web Attacks attack: XSS attempt
2025-05-22 12:17:58,529 - INFO - Attack completed: Web Attacks (Success: True)
2025-05-22 12:17:58,529 - INFO - Waiting 60 seconds before next category...
2025-05-22 12:18:58,530 - INFO - Starting dos tests
2025-05-22 12:18:58,530 - INFO - Running DoS attack: Light HTTP flood
2025-05-22 12:19:01,353 - INFO - Attack completed: DoS (Success: True)
2025-05-22 12:19:31,353 - INFO - Waiting 60 seconds before next category...
2025-05-22 12:20:31,353 - INFO - Starting ddos tests
2025-05-22 12:20:31,354 - INFO - Running DDoS attack: HTTP flood simulation using AB
2025-05-22 12:20:57,852 - INFO - Attack completed: DDoS (Success: True)
2025-05-22 12:20:57,853 - INFO - Waiting 60 seconds before next category...
2025-05-22 12:21:57,854 - INFO - Starting bots tests
2025-05-22 12:21:57,855 - INFO - Running Bots attack: C&C server communication
2025-05-22 12:21:58,176 - INFO - Attack completed: Bots (Success: True)
2025-05-22 12:22:28,178 - INFO - Running Bots attack: Data exfiltration
2025-05-22 12:22:28,490 - INFO - Attack completed: Bots (Success: True)
2025-05-22 12:22:28,491 - INFO - All tests completed. Results saved to nids_test_results/manual_test_results.csv
2025-05-22 12:22:28,491 - INFO - NIDS Testing Framework completed
```

Рис. 3.2 - запуск атаки та вивід скрипту

подивимось на реакцію моделі(Рис. 3.3):

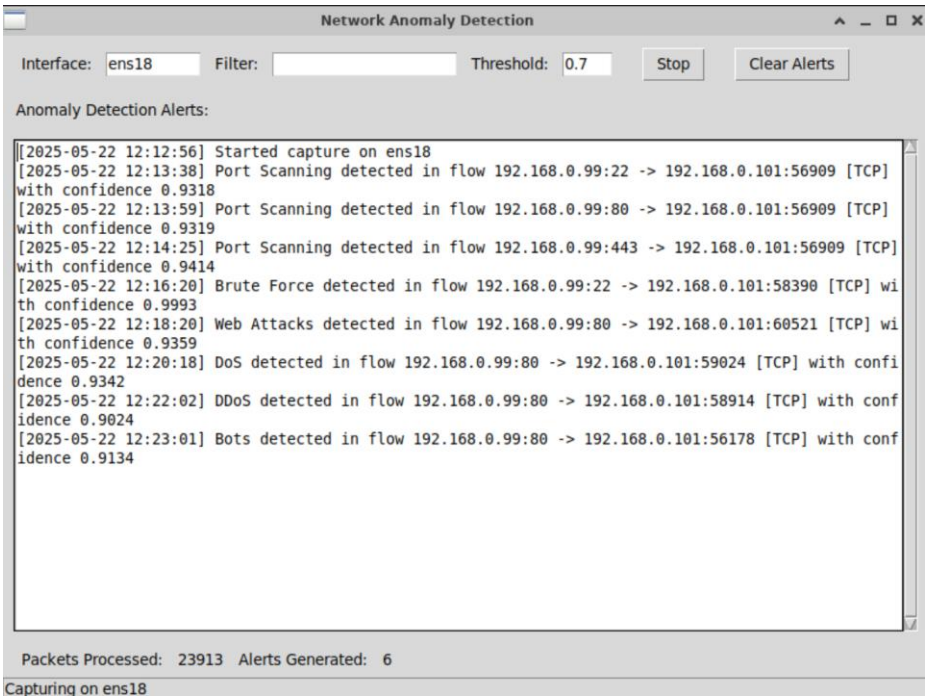

Рис. 3.3 — Спрацювання моделі на атаки

бачимо сповіщення про атаки , модель класифікувала атаку та видала час коли це було. Алерти зберігаються для подальшого аналізу в інших файлах також

У результаті тестування роботи моделі виявлення аномалій можна зробити висновок про її здатність класифікувати мережеві атаки. Кожне спрацювання містить детальну інформацію про цільовий порт нашого вузла (який був атакований), IP-адресу та порт машини-джерела трафіку, а також використаний протокол (TCP). Крім цього, для кожного спрацювання модель надала значення впевненості у правильності класифікації, яке в усіх випадках було високим (понад 90%). Під час симуляції сканування портів було згенеровано три окремі спрацювання, що пов’язано з тим, що зловмисник ініціював підключення до кількох портів (22, 80 та 443), які система розпізнала як окремі потоки. Таким чином, модель коректно виявила факт сканування з різними ознаками. Інші типи атак — Brute Force, Web Attacks, DoS, DDoS та Bots — були класифіковані по одному разу кожна. Це пояснюється тим, що кожен із цих типів трафіку був згенерований як єдиний потік, у якому на цільову IP-адресу йшло декілька пакетів до одного порту,

що й викликало спрацювання на завершенні відповідного тайм-вікна. Таким чином, модель виявила кожну атаку один раз, що відповідає логіці обробки потоків і підтверджує її ефективність. Усі виявлені події були записані до лог-файлу, що дозволяє провести подальший аналіз або автоматизувати відповідні дії на стороні захисту. *Повний код можна знайти за джерелом[6]*

3.3 Розробка рекомендацій щодо виявлення мережевих аномалій за допомогою машинного навчання

У рамках практичної реалізації було розглянуто базову систему виявлення мережевих аномалій (NIDS), яка використовує модель машинного навчання XGBoost для класифікації типів атак на основі характеристик мережевого трафіку. На відміну від класичних міжмережевих екранів, що працюють на основі фіксованих сигнатур і потребують постійного ручного оновлення, модель ML здатна адаптивно виявляти як відомі, так і нові патерни атак, одразу класифікуючи тип загрози. Це значно пришвидшує реагування на інциденти та спрощує подальший аналіз.

Система генерує алерти в реальному часі та зберігає їх у файл, що відкриває можливість автоматичної побудови політик блокування — фільтрації, обмеження, або сповіщення на основі зафіксованих подій. Таким чином, у разі інтеграції з механізмами реагування, система здатна виконувати не лише функції IDS, а й частково реалізовувати *IPS (Intrusion Prevention System)* або стати базою для створення повноцінного *NDR (Network Detection and Response)*. Власне, NDR можна розглядати як розширення IDS з доданою можливістю реагування, аналізу та автоматизованої обробки інцидентів.

Система успішно виявляла атаки типу Port Scanning, Brute Force, DoS, DDoS, Web Attacks та Bots, відображаючи результати в графічному інтерфейсі та логуючи всі події. Проте ця реалізація має обмеження, характерні для прототипу, і потребує вдосконалення для повноцінного впровадження в умовах розгалуженої мережевої інфраструктури.

Одним із ключових напрямів покращення є використання реального трафіку організації для побудови моделей та створення і обробка власних датасетів. Дані, зібрані з Zeek, NetFlow, sFlow, а також логів з вебсерверів (Nginx, Apache), маршрутизаторів і свічів, дозволяють сформувати точніші моделі, адаптовані до специфіки конкретної мережі. Це усуває ризик перенавчання на зовнішніх датасетах і підвищує достовірність класифікації.

Також варто поєднувати різні типи алгоритмів: supervised-моделі (як-от XGBoost) забезпечують високу точність при виявленні відомих загроз, тоді як unsupervised-підходи (Isolation Forest, автоенкодери)[3] ефективні для виявлення нових, нетипових відхилень. Проте моделі без учителя чутливі до хибнопозитивних спрацювань через відсутність контексту. Тому важливо впроваджувати механізми зменшення FP(False Positive) — адаптивні пороги, поведенкові профілі, а також можливість верифікації тривог вручну аналітиком або за допомогою додаткових джерел.

Вбудована сигнатурна база може значно підсилити якість класифікації, слугуючи додатковим підтвердженням у випадках, де ML-модель не має високої впевненості. У комбінації з поведінковим аналізом та механізмами профілювання та реагування, це дозволяє створювати гібридні системи, що виходять за межі звичайного IDS.

Ще одним важливим компонентом є адаптивна побудова базових ліній нормальної поведінки, як це реалізовано в Kentik Protect[9]. Такі системи враховують добові, тижневі чи сезонні коливання, навчаючись тому, як виглядає типовий трафік компанії. Відхилення від цієї динамічної норми інтерпретуються як потенційна загроза, що суттєво знижує рівень хибних спрацювань.

У масштабних корпоративних середовищах необхідно реалізувати централізований збір, агрегацію та нормалізацію трафіку з різних джерел. Це забезпечує повноцінну візуалізацію мережевої активності та дозволяє системі ефективно працювати навіть у великих розподілених інфраструктурах. Прикладом такої масштабованої архітектури є Darktrace, яка може обробляти тисячі агентів по всьому світу одночасно.

Особливо цінною є можливість автоматизованої обробки та аналізу інцидентів. Cyber AI Analyst від Darktrace дозволяє автоматично формувати логіку подій, інтегрувати з MITRE ATT&CK таксономією та формувати пояснення для SOC-команди, зменшуючи навантаження на аналітиків та пришвидшуючи час реагування[11].

Ще більш ефективною система стає при інтеграції з додатковими захисними шарами, зокрема[]:

- *WAF* (ModSecurity, AWS WAF, Azure Front Door) — для глибшого контролю HTTP-рівня, фільтрації SQLi/XSS;
- *SIEM-системами* (Splunk, ELK, QRadar) — для централізованої кореляції подій;
- *SOAR-платформами* (Cortex XSOAR, IBM Resilient) — для автоматизації реагування на інциденти.

Окрім Kentik і Darktrace, важливими гравцями в області NDR також є *Vectra AI, Corelight, ExtraHop та інші*, які пропонують рішення з глибокою ML-аналітикою, сигнатурними базами, адаптивним базуванням і можливістю інтеграції з хмарною, гібридною та фізичною інфраструктурою. Саме такий підхід — поєднання алгоритмів, поведінкового аналізу, аналітики та адаптації — сьогодні є домінуючим у сфері сучасного захисту мереж.

ВИСНОВКИ

У роботі проведено дослідження проблеми виявлення аномалій у мережевому трафіку та використання методів машинного навчання для цього. Визначено доцільність застосування алгоритмів машинного навчання як основи для побудови адаптивних систем безпеки мереж. Також проаналізовано сучасні підходи до класифікації аномалій та архітектури систем виявлення вторгнень. Розглянуто особливості застосування машинного навчання у цій сфері, зокрема supervised, unsupervised та semi-supervised підходів. На практиці проведено підготовку та обробку даних CICIDS2017, зокрема очищення, нормалізацію, генерацію ознак та формування фінального набору для тренування моделей. Було побудовано декілька моделей класифікації трафіку, проведено їх тестування та порівняння. Алгоритм XGBoost показав найвищу точність серед протестованих, що стало підставою для його використання у практичній частині. Результатом реалізації стала базова система виявлення мережевих аномалій (NIDS), яка здатна обробляти трафік у реальному часі, виявляти загрози та зберігати результати у вигляді логів та сповіщень.

Система була протестована в умовах згенерованого трафіку з атак і продемонструвала успішну класифікацію аномалій із високим рівнем достовірності. Окрім виявлення, реалізовано логування інцидентів, що створює основу для подальшого реагування або автоматизації — наприклад, створення правил блокування чи інтеграції з іншими системами захисту. Запропонована реалізація може бути масштабована до рівня повноцінної системи NDR або IPS при наявності механізмів автоматичного реагування. У роботі надано практичні рекомендації щодо покращення системи.

Загалом, результати дипломної роботи підтверджують ефективність використання методів машинного навчання для виявлення аномалій, підвищення точності, швидкості та автоматизації процесу виявлення мережевих загроз у сучасних умовах, зберігаючи гнучкість і можливість масштабування під конкретні потреби організації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bhattacharyya D. K., Kalita J. K. Network anomaly detection: a machine learning perspective. Taylor & Francis Group, 2013. 366 p. (дата звернення: 15.03.2025).

2. Bhattacharyya S. 10 popular ML algorithms for solving classification problems. _Medium_. URL: https://medium.com/@howtodoml/10-popular-ml-algorithms-for-solving-classification-problems-b3bc1770fbdc (дата звернення: 20.03.2025).

3. Bozdag E. Anomaly-based Intrusion Detection System using unsupervised ML approach. _Medium_. URL: https://medium.com/hootsuite-engineering/anomaly-based-intrusion-detection-system-using-machine-learning-a18e88694ce0 (дата звернення: 20.04.2025).

4. Contributors to Wikimedia projects. K-means clustering - Wikipedia. _Wikipedia, the free encyclopedia_. URL: https://en.wikipedia.org/wiki/K-means_clustering (дата звернення: 10.05.2025).

5. GeeksforGeeks. Difference between HIDs and NIDs - GeeksforGeeks. _GeeksforGeeks_. URL: https://www.geeksforgeeks.org/difference-between-hids-and-nids/ (дата звернення: 14.04.2025).

6. GitHub - anacletu/ml-intrusion-detection-cicids2017: Machine Learning-based Intrusion Detection System (IDS) tailored for resource-constrained networks. _GitHub_. URL: https://github.com/anacletu/ml-intrusion-detection-cicids2017 (дата звернення: 11.05.2025).

7. IDS 2017 | datasets | research | canadian institute for cybersecurity | UNB. _University of New Brunswick | UNB_. URL: https://www.unb.ca/cic/datasets/ids-2017.html (дата звернення: 10.05.2025).

8. Model evaluation: assessing the performance of machine learning models. _Coursera_. URL: https://www.coursera.org/articles/model-evaluation (дата звернення: 14.05.2025)

9. Network anomaly detection: a comprehensive guide. _Kentik_. URL: https://www.kentik.com/kentipedia/network-anomaly-detection/ (дата звернення: 10.05.2025).

10. Rahalkar C. How to build a real-time intrusion detection system with python and open- sourcelibraries. _freeCodeCamp.org_. URL: https://www.freecodecamp.org/ news/build-a-real-time-intrusion-detection-system-with-python/ (дата звернення: 15.05.2025).

11. Lesot M. J., Rifqi M. Anomaly-based network intrusion detection:

Techniques, systems and challenges. International Journal of Knowledge Engineering and Soft Data Paradigms. 2009. Vol. 1, no. 1. P. 63–84 (дата звернення: 15.05.2025).

12. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey. ACM Computing Surveys. 2009. Vol. 41, no. 3. P. 15–38. (дата звернення: 15.05.2025).

13. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Incorporated, 2022. p 513 (дата звернення: 15.03.2025)

14. E. E. Abdallah, W. Eleisah, and A. F. Otoom, “Intrusion detection systems using supervised machine learning techniques: A survey,” Procedia Computer Science, vol. 201, pp. 125–132, 2022, doi: 10.1016/j.procs.2022.03.029 (дата звернення: 10.05.2025)

15. P. Prakriti, “Cyber threat detection using machine learning,” Int. J. Sci. Res. Eng. Manag. (IJSREM), Apr. 2024, vol. 4, no. 4, pp. 1–6, doi: 10.55041/IJSREM36799 (дата звернення: 13.05.2025)

16. W. Song, M. Beshley, K. Przystupa, H. Beshley, O. Kochan, A. Pryslupskyi, D. Pieniak, and J. Su, “A Software Deep Packet Inspection System for Network Traffic Analysis and Anomaly Detection,” Sensors, vol. 20, no. 6, p. 1637, Mar. 2020, doi: 10.3390/s20061637 (дата звернення: 15.05.2025)