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

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Технологія виявлення роботи ботнету в корпоративній мережі із застосуванням машинного навчання»

зі спеціальності *125 Кібербезпека та захист інформації*
(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

_____ Владислав САМОЙЛЕНКО
(підпис)

Виконав: здобувач вищої освіти групи БСДМ-63
САМОЙЛЕНКО Владислав
(прізвище, ім'я)

Керівник к.військ.н., доцент ГАХОВ Сергій
(науковий ступінь, вчене звання, прізвище, ім'я)

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

Київ 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ADASYN – Adaptive Synthetic Sampling

CNN – Convolutional Neural Networks

DBSCAN – Density-Based Spatial Clustering of Applications with Noise

DoS – Denial of Service

DDoS – Distributed Denial of Service

FTP – File Transfer Protocol

HTTP – Hypertext Transfer Protocol

IAT – Inter-Arrival Time

ICMP – Internet Control Message Protocol

IMAP – Internet Message Access Protocol

IoT – Internet of Things

LIME – Local Interpretable Model-agnostic Explanations

LOIC – Low Orbit Ion Cannon

HOIC – High Orbit Ion Cannon

МН – Машинне навчання

P2P – Peer-to-Peer

PCA – Principal Component Analysis

RBF – Radial Basis Function

RNN – Recurrent Neural Networks

SHAP – SHapley Additive exPlanations

SMOTE – Synthetic Minority Over-sampling Technique

SMTP – Simple Mail Transfer Protocol

SVM – Support Vector Machine

TCP – Transmission Control Protocol

TPE – Tree-structured Parzen Estimator

XGBoost – Extreme Gradient Boosting

XSS – Cross-Site Scripting

ВСТУП

Актуальність дослідження. У сучасному цифровому середовищі ботнети стали однією з найсерйозніших загроз для безпеки корпоративних мереж. Ботнети, або мережі ботів, складаються з великої кількості інфікованих пристроїв, які контролюються зловмисниками для здійснення різноманітних шкідливих дій, таких як DDoS-атаки, розсилання спаму та викрадення конфіденційних даних. З розвитком технологій та методів обходу традиційних систем захисту, ефективність сигнатурних методів виявлення ботнетів значно знизилася, що створює нагальну потребу у розробці нових, більш ефективних технологій виявлення.

Вищезазнаене підтверджує актуальність теми даної кваліфікаційної роботи, основний зміст якої становлять дослідження технології виявлення роботи ботнету в корпоративній мережі із застосуванням машинного навчання.

Об'єкт дослідження – виявлення роботи ботнету в корпоративній мережі.

Предмет дослідження – технологія виявлення роботи ботнету в корпоративній мережі із застосуванням машинного навчання.

Мета роботи – розробити та оцінити ефективну технологію виявлення ботнетів у корпоративних мережах за допомогою методів машинного навчання, що забезпечує високу точність класифікації та оперативність обробки даних.

Наукові завдання:

дослідити сутність проблеми виявлення ботнет-активності у корпоративній мережі;

проаналізувати сучасні підходи до виявлення ботнетів із застосуванням машинного навчання;

проаналізувати існуючі моделі машинного навчання для виявлення ботнетів;

проаналізувати методи попередньої обробки, балансування даних та оптимізації моделей машинного навчання;

розкрити порядок реалізації технології виявлення ботнетів у корпоративній мережі із застосуванням оптимізованих моделей машинного навчання.

Методи дослідження – аналіз літератури для вивчення існуючих підходів до виявлення ботнетів, вибір відповідних алгоритмів машинного навчання, збір та попередня обробка даних з використанням датасету CSE-CIC-IDS2018, навчання та оптимізація моделей Random Forest, XGBoost та SVM, а також оцінка продуктивності моделей за допомогою відповідних метрик.

Практичне значення одержаних результатів: запропоновано порядок застосування технології виявлення ботнетів у корпоративних мережах із застосуванням методів машинного навчання, а також розроблено рекомендації фахівцям з кібербезпеки щодо її реалізації.

Результати дослідження апробовані на VII Міжнародній науково-практичній конференції «Grundlagen der modernen wissenschaftlichen Forschung», яка відбулася 13 грудня 2024 року в м. Цюрих, Швейцарія. Результати також знаходяться на етапі публікації у науково-технічному журналі «Advanced Information Systems» (ISSN 2522-9052).

1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ВИЯВЛЕННЯ РОБОТИ БОТНЕТІВ В КОРПОРАТИВНІЙ МЕРЕЖІ

1.1. Актуальність проблеми ботнетів у корпоративних мережах

У сучасних корпоративних мережах безпека інформаційних систем є одним із найважливіших факторів для стабільної та ефективної роботи організацій. Однією з найбільш серйозних загроз у цій сфері є ботнети — мережі комп'ютерів або пристроїв, інфікованих шкідливим програмним забезпеченням та керованих зловмисниками з метою здійснення різноманітних кіберзлочинів [1-3].

Аналіз Cisco в межах щорічного Інтернет-звіту за період 2018–2023 роки демонструє тенденцію до стрімкого зростання таких атак, які, за прогнозами, до 2023 року досягнуть позначки 15,4 млн, що вдвічі більше, ніж у 2018 році. Значну частину цих атак здійснюють за допомогою ботнетів, які є ключовим інструментом у проведенні масштабних DDoS-кампаній [4] (рисунок 1.1). В Україні ситуація також залишається напруженою: Державна служба спеціального зв'язку зафіксувала 1105 кібератак у 2023 році, що на 62,5% більше порівняно з попереднім роком. Хоча більшість атак спрямовані на державний сектор, корпоративні мережі також зазнають значних збитків [5].

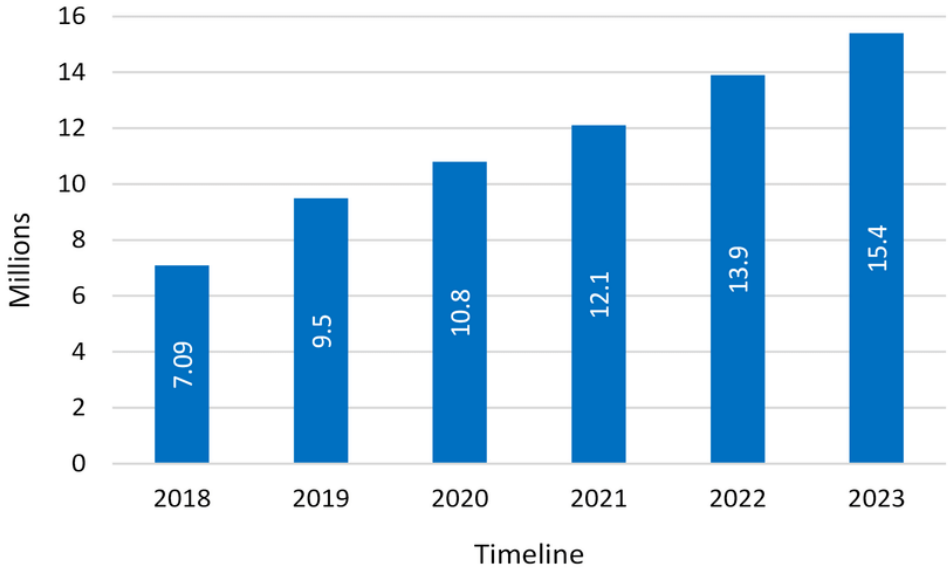


Рис. 1.1. Статистика DDoS-атак за 2018-2023 роки [4]

Ботнети складаються з кількох ключових компонентів. Зловмисник, відомий як ботмайстер, контролює мережу інфікованих пристроїв, які називаються ботами або "зомбі-комп'ютерами". Комунікація між ботмайстром і ботами здійснюється через командно-контрольні (C&C) сервери, використовуючи різні протоколи, такі як IRC, HTTP або P2P. Ця інфраструктура дозволяє ботмайстру віддалено керувати ботами, виконувати шкідливі дії, такі як розсилка спаму, крадіжка даних або проведення DDoS-атак (рисунок 1.2).

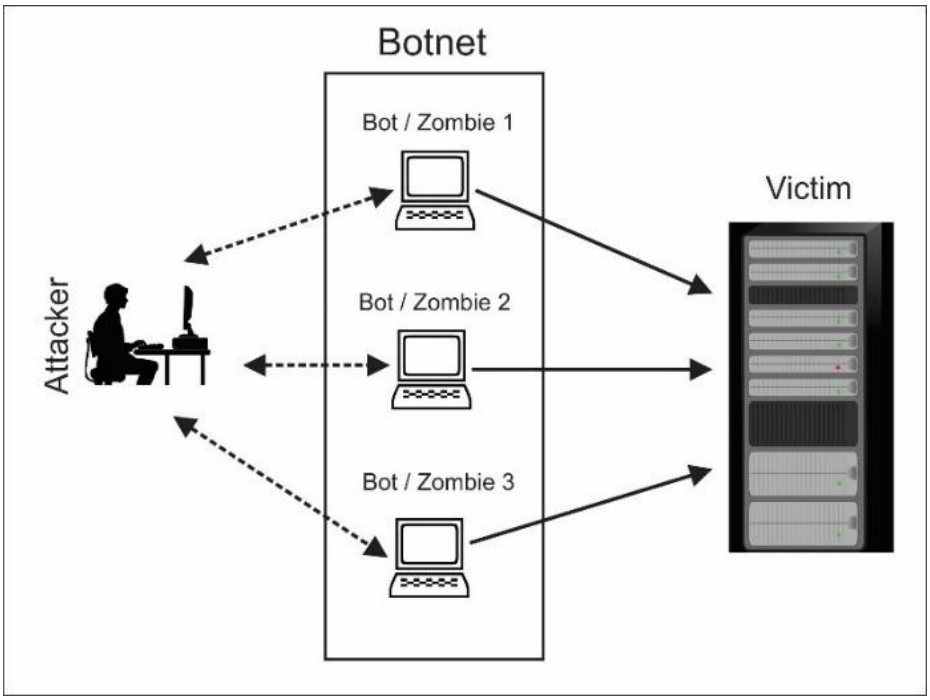


Рис. 1.2. Схема структури ботнету [6]

З розвитком технологій ботнети стають все більш масштабними та складними. Наприклад, ботнет Mirai, який у 2016 році інфікував пристрої Інтернету речей (IoT) та здійснив масивні DDoS-атаки, демонструє потенціал цих мереж для спричинення серйозних перебоїв у роботі Інтернет-сервісів [1]. Подібні випадки свідчать про здатність ботнетів адаптуватися до нових умов та використовувати різні вразливості корпоративних мереж для досягнення своїх цілей.

Ботнети можуть спричиняти значні фінансові збитки для компаній через порушення роботи систем, втрату даних та необхідність витрат на відновлення інфраструктури безпеки [5]. Крім того, інциденти, пов'язані з ботнет-атаками, можуть негативно впливати на репутацію організацій, зменшуючи довіру клієнтів

та партнерів [2]. Тому ефективне виявлення та протидія ботнетам є дуже важливим для збереження фінансової стабільності та довіри до корпоративних мереж.

Традиційні методи захисту, зокрема сигнатурні системи, стають менш ефективними у боротьбі з ботнетами через їхню здатність обходити стандартні засоби безпеки та використовувати передові техніки маскування [7]. Постійний розвиток ботнетів вимагає застосування більш динамічних та адаптивних підходів до їхнього виявлення, що підсилює актуальність досліджень у цій області.

Штучний інтелект та машинне навчання пропонують нові можливості для аналізу великих обсягів даних та виявлення аномалій, які можуть свідчити про ботнет-атаки [7]. Використання алгоритмів машинного навчання, таких як Random Forest, XGBoost та Support Vector Machine (SVM), дозволяє створювати ефективні моделі для класифікації мережевого трафіку та ідентифікації шкідливої активності [8-10]. Це відкриває нові можливості у розробці автоматизованих систем захисту, здатних оперативно реагувати на нові та невідомі загрози.

1.2. Існуючі методи виявлення ботнетів

У сфері кібербезпеки виявлення ботнетів завжди було складним завданням, особливо в умовах постійного ускладнення шкідливих мереж. Існуючі методи можна умовно розділити на традиційні підходи та сучасні, засновані на використанні штучного інтелекту.

Одним із найперших і найпоширеніших підходів є сигнатурний аналіз, який базується на зіставленні трафіку з відомими сигнатурами шкідливих програм [1]. Це дозволяє ефективно ідентифікувати ботнети, які вже були вивчені та задокументовані. Однак цей метод має кілька суттєвих обмежень:

Неможливість виявлення нових загроз. Якщо ботнет видозмінює свою структуру або використовує нові техніки маскування, сигнатурний підхід виявляється неефективним;

Висока залежність від оновлення баз даних. Постійне оновлення сигнатур вимагає значних ресурсів, і навіть невелике відставання може призвести до пропуску загроз.

Для подолання обмежень сигнатурних методів було запропоновано використовувати аномалійний аналіз, який полягає у виявленні відхилень від нормальної поведінки мережевого трафіку [2]. Цей підхід дозволяє ідентифікувати нові або невідомі ботнети, оскільки вони часто проявляють аномальні патерни в трафіку. Однак аномалійний аналіз також має свої недоліки:

Висока кількість хибнопозитивних спрацьовувань. Не всі аномалії в трафіку свідчать про наявність шкідливої активності, що може ускладнювати реагування на потенційні загрози;

Складність налаштування. Для забезпечення ефективності моделі потрібне ретельне налаштування параметрів, що потребує високого рівня експертних знань.

Традиційні методи залишаються основою багатьох систем кібербезпеки, однак їхнє поєднання із сучасними підходами дозволяє досягати значно кращих результатів [11].

Сучасні методи машинного навчання (МН) суттєво розширили можливості виявлення ботнетів. Ці технології дозволяють аналізувати великі обсяги даних і знаходити приховані шаблони, які неможливо виявити за допомогою традиційних підходів. Розглянемо основні підходи, що використовуються у цій сфері.

Кероване навчання (Supervised Learning). Цей підхід ґрунтується на використанні мічених даних, які дозволяють моделям вчитися класифікувати трафік як шкідливий або нешкідливий [3]. Алгоритми, такі як Random Forest, XGBoost та Support Vector Machine (SVM), продемонстрували високу ефективність у завданнях виявлення ботнетів. Ці алгоритми аналізують ключові характеристики трафіку, що дозволяє їм точно розділяти класи (ботнети і звичайний трафік).

Random Forest працює як ансамбль рішень, використовуючи кілька дерев класифікації для підвищення точності та стійкості моделі до перенавчання [8]. XGBoost, своєю чергою, забезпечує високу швидкість і точність завдяки ітеративному покращенню результатів кожного попереднього дерева [9]. SVM

знаходить оптимальну гіперплощину, яка розділяє класи в багатовимірному просторі, демонструючи особливо високу точність у задачах із тонкими відмінностями між класами [10].

Іншим підходом є некероване навчання. У випадках, коли мічених даних бракує, застосовуються некеровані методиметоди некерованого навчання. До них належать кластеризація та аналіз асоціативних правил. Ці методи дозволяють ідентифікувати нові типи ботнетів та аномалій у трафіку. Наприклад, кластеризація може використовуватись для групування схожих патернів трафіку, що дозволяє виявляти аномалії, які можуть свідчити про ботнет-активність. Асоціативні правила можуть виявляти взаємозв’язки між різними ознаками, які не можуть бути легко виявлені іншими методами [12].

Розглянемо підхід, що базується на глибокому навчанні. Він використовує нейронні мережі для аналізу високорозмірних даних та виявлення складних взаємозв'язків між ознаками [5]. Основні моделі включають згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN). оптимально працюють із структурованими даними, такими як мережеві потоки, перетворені у вигляді матриць або графіків. Вони дозволяють виявляти локальні шаблони в трафіку, які вказують на аномальну активність. У контексті виявлення ботнетів CNN можуть аналізувати структурні особливості трафіку, знаходячи специфічні аномалії, що характерні для шкідливих дій. Рекурентні нейронні мережі ефективні для аналізу послідовностей даних. Вони враховують часові залежності між пакетами, що дає змогу виявляти аномальні патерни поведінки ботнетів у часових інтервалах.

Для підвищення ефективності виявлення ботнетів розроблено гібридні підходи, які поєднують переваги традиційних методів із сучасними алгоритмами машинного навчання. Наприклад, поєднання сигнатурного аналізу з аномалійним аналізом дозволяє забезпечити більш повне покриття та зменшити кількість хибнопозитивних спрацьовувань [7]. Іншим прикладом є використання ансамблевих методів, які поєднують декілька алгоритмів машинного навчання для підвищення точності та стабільності моделей [13].

Дослідження на цю тему показують, що методи машинного навчання, особливо ансамблеві алгоритми, такі як Random Forest та XGBoost, перевершують традиційні сигнатурні методи у задачах виявлення ботнетів [13-15]. Однак кожен метод має свої специфічні переваги та обмеження, що робить необхідним вибір найбільш відповідного підходу залежно від конкретних умов експлуатації та вимог до системи безпеки.

З метою глибшого розуміння застосування розглянутих методів, розглянемо наукові дослідження, які демонструють їх практичне впровадження.

Одним із перших досліджень у цій галузі була робота Лівадаса та ін. [7], які застосували алгоритми машинного навчання для виявлення ботнетів, що використовують IRC-протокол для командування та управління (C&C). IRC (Internet Relay Chat) був популярним серед ботнетів через свою простоту та можливість підтримувати постійний зв'язок між зловмисником та інфікованими машинами.

У своєму дослідженні Лівадас та ін. зосередилися на аналізі статистичних ознак мережевого трафіку, таких як кількість бітів за секунду, кількість пакетів за секунду та тривалість потоків. Вони оцінили декілька алгоритмів машинного навчання, включаючи методи на основі Байєсівських мереж та метод опорних векторів (SVM). Підхід складався з двох етапів класифікації: спочатку відокремлення трафіку типу "чат", а потім розмежування шкідливого та нешкідливого чат-трафіку.

Результати дослідження показали, що використання машинного навчання дозволяє ефективно виявляти ботнети, навіть без аналізу вмісту пакетів. Проте метод мав обмеження, оскільки залежав від специфіки IRC-протоколу та був вразливим до ботнетів, що використовують інші або модифіковані протоколи.

Gu та ін. розробили систему BotMiner [16], яка стала значним кроком вперед у виявленні ботнетів незалежно від використовуваних ними протоколів командування та управління. Основна ідея BotMiner полягає в тому, що боти в мережі демонструють схожу поведінку як у комунікаціях, так і в шкідливих діях. Система розділяє аналіз на дві площини: C-площину (Communication Plane), що

аналізує комунікаційні активності між хостами, та A-площину (Activity Plane), яка відстежує шкідливі активності, такі як сканування портів або розсилання спаму.

Процес виявлення включає збір та попередню обробку даних, кластеризацію в C-площині та A-площині, а потім горизонтальну кореляцію між ними для виявлення хостів, що демонструють схожу поведінку в обох площинах. Перевагою BotMiner є незалежність від конкретних протоколів C&C та можливість виявлення невідомих ботнетів. Однак система потребує наявності декількох інфікованих хостів у мережі для ефективного виявлення, що є її обмеженням.

Tegeler та ін. запропонували систему BotFinder [17], яка фокусується на виявленні ботнетів, що використовують HTTP-протокол для комунікацій C&C. Оскільки HTTP є одним з найпоширеніших протоколів в Інтернеті, ботнети почали використовувати його для маскування свого трафіку серед легітимного. BotFinder аналізує регулярність комунікацій, спостерігаючи, що ботнети часто демонструють регулярні інтервали у своїх з'єднаннях з C&C-серверами. Використовуючи статистичні ознаки, такі як середній інтервал часу між потоками та тривалість потоків, система здатна виявляти ботнети без потреби в глибокому аналізі вмісту пакетів.

Ефективність BotFinder полягає в можливості виявлення ботнетів без використання сигнатур або глибокого аналізу трафіку. Проте її точність може бути нижчою порівняно з традиційними методами на основі сигнатур, особливо якщо ботнети адаптуються та змінюють свої патерни комунікацій для уникнення виявлення.

P2P-ботнети представляють особливу складність для виявлення через відсутність централізованого C&C-сервера та використання розподілених мережевих структур. Yen та Reiter [11] розробили алгоритм для розмежування легальних P2P-файлообмінних додатків та P2P-ботів, аналізуючи обсяг трафіку та стійкість мережевих з'єднань. Вони виявили, що боти можуть генерувати аномально великий або малий обсяг трафіку порівняно з легітимними P2P-додатками, а також мати характерні патерни з'єднань.

Zhang та ін. [18] запропонували систему для виявлення прихованих P2P-ботнетів, сфокусувавшись на ідентифікації статистичних відбитків P2P-комунікацій. Вони аналізували стійкі з'єднання з певними IP-адресами та перетинання контактованих IP-адрес між різними хостами, використовуючи методи кластеризації та аналізу графів для виявлення аномалій. Система ефективно виявляє P2P-ботнети, але може бути менш ефективною у випадках, коли боти використовують динамічні IP-адреси або змінюють свою поведінку.

Датасет CSE-CIC-IDS2018 є одним із найбільш всеосяжних для дослідження мережевих атак, включаючи ботнети. Він містить велику кількість ознак та класів, що створює виклики щодо обробки високовимірних даних та незбалансованості класів.

Hua [14] зосередився на вирішенні проблеми високої вимірності та незбалансованості класів шляхом застосування методу підвибірки (undersampling) та вбудованого відбору ознак з LightGBM-класифікатором. Після видалення непотрібних ознак та заповнення пропущених значень було отримано 77 ознак для моделі. Використовуючи LightGBM для автоматичного визначення важливості ознак, він відібрав топ-10 найбільш значущих, досягнувши точності 98,37%.

Li та ін. [15] застосували комбінований підхід для виявлення атак у режимі реального часу, використовуючи кластеризацію з Affinity Propagation та відбір ознак з Random Forest. Вони також використали автоенкодер як ненаглядний метод машинного навчання для виявлення аномалій шляхом моделювання нормальної поведінки. Цей підхід дозволив зменшити розмірність даних та підвищити ефективність моделі.

Fitni та Ramli [13] запропонували ансамблеву модель, яка поєднує декілька класифікаторів, включаючи Random Forest, Gradient Boosting та Logistic Regression. Використовуючи відбір ознак на основі коефіцієнта кореляції Спірмена та критерію χ^2, вони обрали 23 найбільш значущі ознаки. Навчання ансамблевої моделі та комбінація результатів окремих класифікаторів призвели до підвищення точності класифікації до 98,80%, що перевищує результати окремих моделей.

Зростаюча складність моделей машинного навчання піднімає питання їхньої прозорості та пояснюваності, особливо у сферах, де рішення моделі можуть мати критичні наслідки. Barbado та ін. [19] наголошують на важливості методів вилучення правил з моделей машинного навчання для покращення розуміння моделі та підвищення довіри до неї. Вони класифікують методи вилучення правил на специфічні для моделі та незалежні від моделі, що дозволяє застосовувати їх у різних контекстах.

У роботах, присвячених вилученню правил з моделей SVM, Hao та ін. [20] запропонували метод вилучення скорочених правил на основі біасного Random Forest та нечіткої SVM для ранньої діагностики діабету. Поєднання цих методів дозволило отримати інтерпретовані правила для медичних працівників, підвищуючи точність та пояснюваність моделі. Kašćelan та ін. [21] розробили метод вилучення правил для визначення впливу характеристик інвесторів на переваги торгівлі акціями, поєднуючи SVM та дерева рішень. Це дозволило отримати глибше розуміння факторів, що впливають на рішення інвесторів.

Загалом, попередні дослідження демонструють ефективність застосування методів машинного навчання для виявлення ботнетів, особливо при використанні ансамблевих методів та технік відбору ознак. Однак залишається багато перешкод, таких як розвиток ботнетів та необхідність адаптивних моделей, здатних виявляти нові та невідомі типи атак. Ця робота базується на цих підходах, пропонуючи систему, яка інтегрує балансування даних, оптимізацію моделей та аналіз важливості ознак для покращення виявлення ботнет-активності в мережевому трафіку.

1.3. Аналіз методів машинного навчання для виявлення ботнетів

Застосування машинного навчання (МН) стало важливим кроком уперед у протидії ботнетам, оскільки воно суттєво підвищує ефективність та точність у виявленні шкідливої мережевої активності. Однією з головних переваг МН є

здатність автоматизовано аналізувати великі обсяги мережевого трафіку, виявляючи приховані аномалії, патерни та зв’язки між ознаками, які залишаються непомітними для традиційних сигнатурних або евристичних підходів. Такі моделі дозволяють ідентифікувати раніше невідомі типи ботнет-активності, що особливо важливо з огляду на постійну еволюцію ботнетів і використання ними нових способів обходу систем захисту.

Однією з головниох переваг МН є його здатність адаптуватися до змінюваного середовища. Ботнети швидко еволюціонують, інтегрують нові протоколи, стратегії маскування та технології. У той час як традиційні методи, засновані на сигнатурах або жорстко запрограмованих правилах, стають менш ефективними, МН забезпечує можливість регулярного оновлення моделей. Навчання на нових даних дозволяє таким системам враховувати актуальні зміни у поведінці трафіку, підтримуючи високу ефективність виявлення навіть нових чи рідкісних типів ботнет-активності. Регулярне оновлення моделей на свіжих даних допомагає їм залишатися актуальними та ефективними в умовах динамічного кіберпростору [22].

Не менш цінною є висока точність класифікації, яку можуть продемонструвати сучасні алгоритми машинного навчання. Моделі, що належать до класу методів керованого навчання, як-от Random Forest, XGBoost або SVM, здатні досягають виняткових показників точності та повноти (recall), що особливо важливо в контексті безпеки. Виявлення ботнетів є складним завданням, оскільки шкідливий трафік може маскуватися під цілком легітимну активність, а його виявлення вимагає вміння розрізняти найтонші відмінності у поведінці пакетів, потоків або сеансів. Алгоритми машинного навчання, завдяки своїй математичній природі, можуть фіксувати дуже слабкі сигнали та обробляти багатовимірні дані, які містять сотні й тисячі ознак. Це дає змогу мінімізувати кількість хибнопозитивних тривог, коли нормальний трафік помилково ідентифікується як шкідливий, а також зменшити кількість хибнонегативних спрацювань, коли потенційно небезпечна активність залишається непоміченою.

Крім того, автоматизація процесу аналізу даних за допомогою МН дозволяє значно автоматизувати процес аналізу трафіку. Традиційні методи вимагали активної участі фахівців із кібербезпеки, які вручну перевіряли логи, аналізували зміну поведінки систем чи користувачів. МН дозволяє автоматизувати ці завдання, забезпечуючи безперервний аналіз мережевого трафіку та миттєве реагування на аномалії. Це дозволяє фахівцям зосередитися на стратегічних аспектах, таких як планування нових політик безпеки чи усунення першопричин атак.

Процес виявлення ботнетів за допомогою машинного навчання зазвичай включає кілька ключових етапів. На початковій стадії здійснюється збір даних з різних джерел, включаючи журнали подій, мережевий трафік з маршрутизаторів, комутаторів, міжмережевих екранів та спеціалізованих систем виявлення вторгнень, а також інформацію з honeypots, які слугують пастками для зловмисників. Важливо забезпечити якість та репрезентативність даних для навчання моделей машинного навчання, що включає видалення шуму, корекцію помилок та нормалізацію значень ознак [23].

Наступним етапом у процесі виявлення ботнетів є попередня обробка даних. Вона включає очищення даних від шуму та аномалій, нормалізацію числових ознак для забезпечення їхньої сумісності та видалення пропущених значень шляхом їх заміни на середні або медіанні значення. Важливим аспектом є відбір релевантних ознак, що дозволяє зменшити вимірність даних та підвищити ефективність моделей машинного навчання. Для цього використовуються методи, такі як рекурсивне вилучення ознак, метод Lasso Regression та головні компоненти аналізу (PCA).

На наступному етапі обирається та навчається модель машинного навчання, яка буде використовуватися для класифікації мережевого трафіку та ідентифікації потенційної ботнет-активності. Вибір алгоритму залежить від специфіки задачі, обсягу даних та вимог до точності та швидкості класифікації. Після навчання модель застосовується для аналізу нових даних з метою виявлення аномалій або патернів, характерних для ботнет-активності. Нарешті, результати аналізу інтерпретуються, і, за потреби, вживаються відповідні заходи реагування (рисунок 1.3).

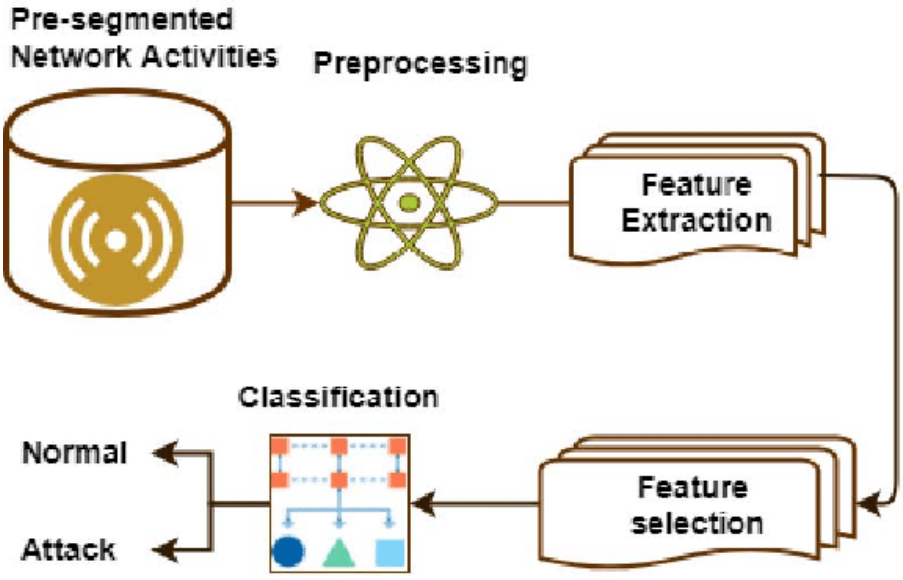


Рис. 1.3. Схема процесу виявлення ботнетів за допомогою МН [24]

Існує кілька підходів до застосування машинного навчання (МН) для виявлення ботнетів, і кожен із них має унікальні особливості, сильні сторони й обмеження. Вибір методу зазвичай залежить від доступності мічених даних, складності завдання, технічних ресурсів та вимог до точності й швидкості аналізу. Основними напрямами в цій галузі є кероване навчання (Supervised Learning), некероване навчання (Unsupervised Learning) і глибоке навчання (Deep Learning).

Кероване навчання є одним із найпоширеніших підходів для виявлення ботнетів, оскільки передбачає використання мічених даних. Це означає, що на етапі навчання модель отримує набір зразків із заздалегідь відомими класами, наприклад, "ботнет-активність" чи "нормальний трафік". На основі цих даних модель вчиться визначати шаблони, властиві кожному класу, і застосовувати ці знання до нових зразків.

Ефективність такого підходу залежить від якості мічених даних. Якщо у наборі даних є велика кількість репрезентативних прикладів, модель здатна узагальнювати закономірності та демонструвати високу точність навіть у складних сценаріях. У цьому контексті найкраще себе зарекомендували такі алгоритми, як Random Forest, XGBoost та Support Vector Machine (SVM).

Random Forest базується на методах ансамблю та використовує велику кількість дерев рішень, що працюють незалежно одне від одного. Кожне дерево навчається на випадковій підмножині даних і ознак, а підсумковий результат визначається шляхом голосування або усереднення рішень усіх дерев. Такий підхід дозволяє знизити ризик перенавчання та підвищити стійкість моделі до шумових даних.

Особливістю Random Forest є здатність аналізувати нелінійні взаємозв'язки між ознаками, забезпечуючи надійність і точність класифікації. Крім того, алгоритм надає можливість оцінити значущість кожної ознаки, що допомагає в процесі відбору найбільш інформативних характеристик.

XGBoost є одним із найбільш ефективних алгоритмів для задач класифікації та регресії. Його робота ґрунтується на принципі градієнтного бустингу, коли моделі додаються поступово. Кожна нова модель фокусується на помилках попередньої, покращуючи загальний результат. Завдяки цьому XGBoost особливо ефективний у задачах, де дані містять складні патерни або аномалії.

Алгоритм відомий своєю швидкістю та масштабованістю, що робить його ідеальним вибором для великих наборів даних. Використання регуляризації допомагає уникати перенавчання, а можливість гнучкої настройки гіперпараметрів дозволяє досягти високої продуктивності.

SVM шукає гіперплощину, яка максимально розділяє дані різних класів. Якщо дані не є лінійно роздільними, алгоритм використовує ядрові функції (наприклад, RBF), що дозволяють переносити дані у простір вищої розмірності. Це робить SVM потужним інструментом для виявлення тонких закономірностей у мережевому трафіку.

Однак SVM має обмеження: він може бути ресурсомістким при роботі з великими наборами даних і потребує ретельної оптимізації параметрів для досягнення високої точності.

Наступним підходом застосування машинного навчання є некероване навчання (Unsupervised Learning). У випадках, коли мічені дані відсутні або їх недостатньо, застосовуються некеровані методи, які виявляють приховані

структури в даних. У таких умовах алгоритми некерованого навчання дозволяють аналізувати мережевий трафік, шукаючи приховані структури, закономірності й аномалії, без потреби у попередньому визначенні класів. Цей підхід особливо цінний для виявлення нових або рідкісних типів ботнетів, які залишаються поза увагою через відсутність відповідних шаблонів або сигнатур. Основні методи включають кластеризацію та асоціативні правила.

Одним із найпоширеніших методів некерованого навчання є кластеризація. Наприклад, алгоритм K-means знаходить центри кластерів у просторі ознак і розподіляє зразки до найближчих центрів, мінімізуючи суму квадратів відстаней між точками та центроїдами. Цей підхід добре працює в умовах, коли дані мають чітко визначені групи, але може бути менш ефективним для складних, шумних або неоднорідних наборів даних.

У таких ситуаціях більш ефективним може бути DBSCAN (Density-Based Spatial Clustering of Applications with Noise), який ґрунтується на аналізі щільності даних. DBSCAN виділяє кластери як ділянки з високою щільністю, відокремлені зонами з меншою щільністю. Це дозволяє виявляти кластери довільної форми, а також ізолювати "шумові" точки, які часто вказують на рідкісні або аномальні патерни, характерні для ботнетів. Наприклад, цей метод може бути використаний для виявлення нерегулярних поведінкових моделей у мережевому трафіку, які не відповідають типовим шаблонам.

Ще одним корисним підходом у некерованому навчанні є пошук асоціативних правил. Ці алгоритми виявляють залежності між ознаками у великих масивах даних, знаходячи комбінації ознак, які часто трапляються разом. У контексті виявлення ботнетів асоціативні правила можуть використовуватись для аналізу мережевого трафіку, щоб ідентифікувати ознаки координації між ботами. Наприклад, якщо декілька IP-адрес постійно взаємодіють у певних часових інтервалах і з використанням однакових протоколів, це може вказувати на діяльність ботнету.

Іншим підходом застосування машинного навчання є глибоке навчання. Цей підхід використовує нейронні мережі для аналізу високовимірних даних та

виявлення складних патернів. Завдяки багатошаровим нейронним мережам, ці моделі можуть автоматично виділяти суттєві ознаки з даних, аналізуючи навіть ті патерни, які не піддаються виявленню за допомогою традиційних методів. Основні моделі включають згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN).

Згорткові нейронні мережі (CNN) ефективні для обробки структурованих даних, таких як зображення або перетворені мережеві потоки, дозволяючи виявляти локальні патерни. Якщо мережевий трафік представити у вигляді матриці або іншої структурованої форми, CNN здатні знаходити "узори", які вказують на ботнет-активність. Наприклад, такі моделі можуть виявляти аномальні частоти пакетів, нетипові інтервали між ними або інші специфічні характеристики, що часто пов’язані зі зловмисною активністю.

Для задач, де важливий часовий контекст, використовуються RNN (Recurrent Neural Networks), які враховують послідовність даних. Наприклад, аналіз мережевого трафіку за певний проміжок часу може допомогти виявити зміну патернів у поведінці ботнетів. Завдяки механізму внутрішньої пам’яті RNN зберігають інформацію про попередні стани, дозволяючи виявляти складні часові залежності.

Модифікації RNN, такі як LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit), оптимізовані для роботи з довгими часовими рядами. Ці архітектури дозволяють моделі запам’ятовувати важливі події протягом тривалого часу, що є важливим для ідентифікації атак із затримкою або періодичною активністю.

Кожен з методів машинного навчання має свої переваги та обмеження, що визначає їхню ефективність у конкретних умовах.

Random Forest забезпечує високу точність та стійкість до шуму, але може бути ресурсозатратним при великій кількості дерев у лісі. XGBoost відзначається швидкістю та високою продуктивністю, проте потребує ретельної оптимізації гіперпараметрів для досягнення найкращих результатів. SVM досягає високої

точності в складних задачах класифікації, але є обчислювально витратним при роботі з великими наборами даних.

Глибоке навчання, з іншого боку, є потужним інструментом для виявлення складних патернів, але потребує великих обсягів даних та значних обчислювальних ресурсів. Висока вимірність даних та проблема перенавчання також є викликами, що потребують використання методів відбору ознак та регуляризації для покращення узагальнювальної здатності моделей.

Незважаючи на високу ефективність методів МН, існують певні перешкоди, які потрібно враховувати при їх застосуванні:

Незбалансованість даних. Шкідливий трафік часто становить невелику частку від загального обсягу даних, що може призводити до упередженості моделей, які схильні ігнорувати менш представлений клас. Використання методів балансування, таких як SMOTE, допомагає вирішити цю проблему, але не завжди є достатнім для досягнення оптимальних результатів;

Висока вимірність даних. Велика кількість ознак може ускладнювати навчання моделей та призводити до перенавчання. Тому важливо проводити відбір ознак для збереження лише найбільш інформативних характеристик, що покращує продуктивність та зменшує складність моделей;

Адаптивність до нових атак. Постійний розвиток ботнетів вимагає від моделей бути гнучкими та здатними адаптуватися до нових типів атак. Це забезпечує необхідність постійного оновлення та вдосконалення алгоритмів, щоб підтримувати їхню ефективність у змінних умовах кіберпростору;

Обчислювальні ресурси. Деякі алгоритми, такі як SVM, є обчислювально витратними при роботі з великими наборами даних, що може бути недоліком у великих корпоративних мережах. Глибокі нейронні мережі також потребують значних обчислювальних ресурсів, що обмежує їхнє застосування у середовищах з обмеженими ресурсами.

Базуючись на цій інформації, можна зробити висновок, що методи машинного навчання надають потужні інструменти для ефективного виявлення ботнетів у корпоративних мережах, забезпечуючи високу точність та адаптивність

систем. Керовані методи, такі як Random Forest, XGBoost та SVM, показують високу ефективність у задачах класифікації, проте кожен з них має свої специфічні переваги та обмеження, про які детальніше буде написано у наступному підрозділі. Некеровані методи дозволяють виявляти нові та невідомі типи ботнетів, хоча вони менш точні без наявності мічених даних. Глибоке навчання розширює можливості виявлення складних патернів, але потребує великих обсягів даних та обчислювальних ресурсів.

Для успішного застосування методів машинного навчання у виявленні ботнетів також необхідно враховувати перешкоди, пов’язані з незбалансованістю даних, високою вимірністю та необхідністю адаптації до нових загроз.

2 АНАЛІЗ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ РОБОТИ БОТНЕТІВ В КОРПОРАТИВНІЙ МЕРЕЖІ

2.1. Застосовані методи машинного навчання

Як вже було зазнаено, існують різні підходи на базі методів машинного навчання для виявлення ботнетів. Кероване навчання використовує мічені датасети, де кожному зразку відповідає відома мітка класу, наприклад, "ботнет" або "нормальний" трафік. Алгоритми, навчені таким чином, здатні розпізнавати ознаки, характерні для ботнетів [25]. Некероване навчання, з іншого боку, намагається виявити приховані структури в даних без попередніх міток, що дозволяє знаходити нові або невідомі типи ботнетів [12]. Методи глибокого навчання, зі своєї сторони, використовують нейронні мережі для аналізу складних та високорозмірних даних, що може значно покращити точність виявлення [26].

Серед алгоритмів машинного навчання, що застосовуються для виявлення ботнетів, особливо виділяються Random Forest, XGBoost та Support Vector Machine (SVM), які відносяться до методів керованого навчання. Вибір цих алгоритмів обумовлений їхньою високою ефективністю у задачах класифікації та здатністю обробляти великий обсяг даних з різними типами ознак. Крім того, ці методи добре відомі своєю здатністю виявляти складні нелінійні залежності та аномалії у даних мережевого трафіку. Це робить ці моделі обумовлено доречними у використанні для виявленні роботи ботнетів.

Для задачі класифікації прогноз Random Forest визначається як мода серед прогнозів окремих дерев *(2.1):*

$$\hat{y} = mode\{h_1(x), h_2(x), \dots, h_B(x)\} \quad (2.1)$$

де: $h_1(x)$ *— прогноз* i*-го дерева,* B *— загальна кількість дерев у лісі.*

Таким чином, кожне дерево незалежно класифікує вхідний зразок, а остаточне рішення приймається голосуванням. Такий підхід дозволяє зменшити ризик перенавчання та підвищити узагальнювальну здатність моделі. Принцип роботи ансамблю дерев, де кожне дерево формує свій прогноз, а остаточний результат отримується шляхом усереднення усіх результатів, продемонстровано на рисунку 2.1. На ньому показано, як вхідні дані проходять через кілька дерев, а потім результати комбінуються для одержання остаточної відповіді.

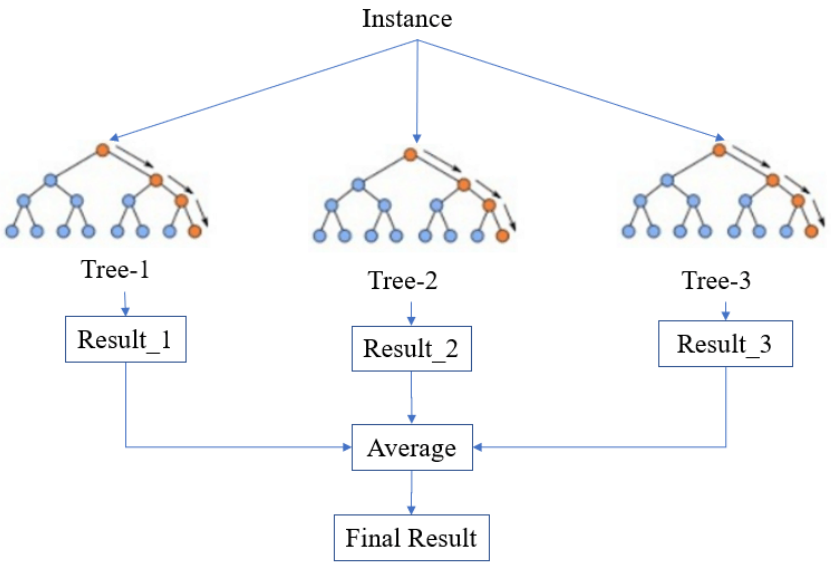


Рис. 2.1. Ансамбль методів із середнім значенням результатів [27]

XGBoost є реалізацією градієнтного бустингу, оптимізованою для швидкості та продуктивності [9]. Цей алгоритм добре масштабується на великих даних і може обробляти складні взаємозв'язки між ознаками, тобто алгоритм будує модель послідовно, додаючи нові дерева, які коригують помилки попередніх. Завдяки цьому XGBoost поступово "набирає вагу" на важких для класифікації зразках, коригуючи результати та підвищуючи загальну точність моделі.

На t-й ітерації модель оновлюється за формулою (2.2):

$$F_t(x) = F_{t-1}(x) + \eta * f_t(x) \tag{2.2}$$

де: $F_{t-1}(x)$ *— поточний прогноз,* $f_t(x)$ *— нове дерево, яке мінімізує залишкову помилку,* η *— коефіцієнт навчання.*

Функція втрат L, яку алгоритм мінімізує, розкладається в ряд Тейлора другого порядку (2.3):

$$L^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{2.3}$$

де: $g_i = \frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}$ *— перша похідна (градієнт),* $h_i = \frac{\partial^2 L(y_i, F_{t-1}(x_i))}{\partial^2 F_{t-1}(x_i)}$ *— друга похідна (гессіан),* $\Omega(f_t)$ *— регуляризаційний член, що контролює складність моделі.*

Цей підхід дозволяє XGBoost ефективно знаходити патерни навіть у складних задачах та швидко обробляти великі набори даних. Принцип бустингу можна візуально пояснити за допомогою рисунку 2.2, де перше дерево створює базовий прогноз, а наступні дерева поступово покращують його, фокусуючись на залишках від попередніх кроків. Кінцевий результат отримують шляхом сумування прогнозів усіх дерев, що забезпечує адаптацію до складних патернів та покращення точності з кожною ітерацією. Додатково, гнучкість налаштування гіперпараметрів (глибини дерев, швидкості навчання, рівня регуляризації) дає змогу налаштувати XGBoost під конкретні потреби, досягаючи ще кращих результатів у виявленні ботнет-активності.

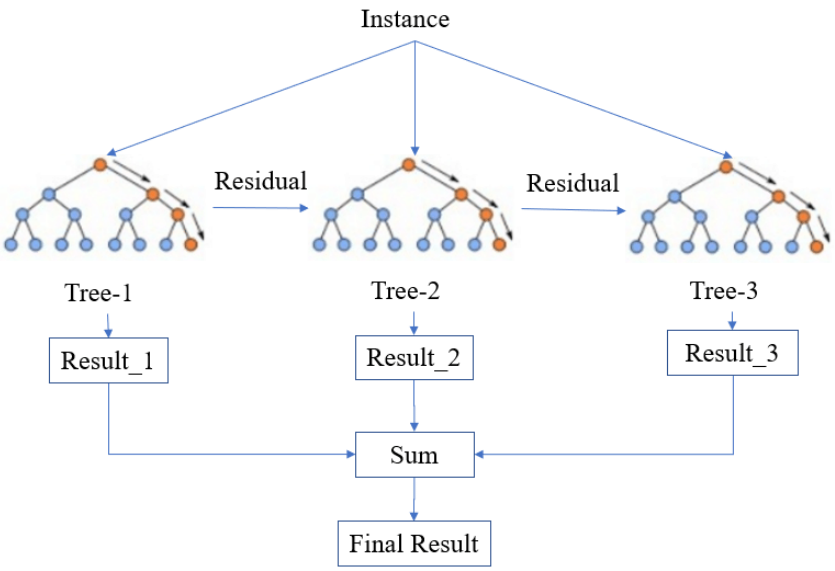


Рис. 2.2. Метод бустингу з використанням залишків [27]

SVM шукає оптимальну гіперплощину, яка максимально розділяє класи у високовимірному просторі. Така гіперплощина має не лише ділити дані на два класи, а й робити це із максимально можливим "запасом" або маржею — відстанню до найближчих точок кожного класу. При чисто лінійному розділенні це є відносно простою задачею, але коли дані не лінійно роздільні, SVM застосовує ядрові функції для переходу до більш високовимірного простору ознак, де може існувати лінійна гіперплощина розділення. Одним із найпоширеніших ядер є RBF (Radial Basis Function), яке визначається за формулою (2.4):

$$K(x^i, x^j) = \exp(-\gamma||x^i - x^j||^2) \tag{2.4}$$

де: γ — параметр, який контролює "радіус впливу" кожної навчальної точки.

З урахуванням ядра, оптимізаційна задача перетворюється з простого лінійного розділення на пошук оптимальних множників Лагранжа $\{a_i\}$, які задають положення підтримуючих векторів. Вони визначаються розв'язанням задачі оптимізації (2.5):

$$\max_a \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j K(x_i x_j) \tag{2.5}$$

з обмеженнями (2.6):

$$\sum_{i=1}^{n} a_i y_i = 0, 0 \le a_i \le C \tag{2.6}$$

де: a_i — множники Лагранжа, C — гіперпараметр, що контролює баланс між максимізацією маржі і помилками класифікації.

Графічне представлення границі рішень SVM із RBF-ядром наведено на рисунку 2.3. Тут простір класифікації поділено на дві зони, кожна з яких відповідає

своєму класу, а вигнута межа між ними ілюструє здатність SVM розмежовувати дані з використанням нелінійного перетворення.

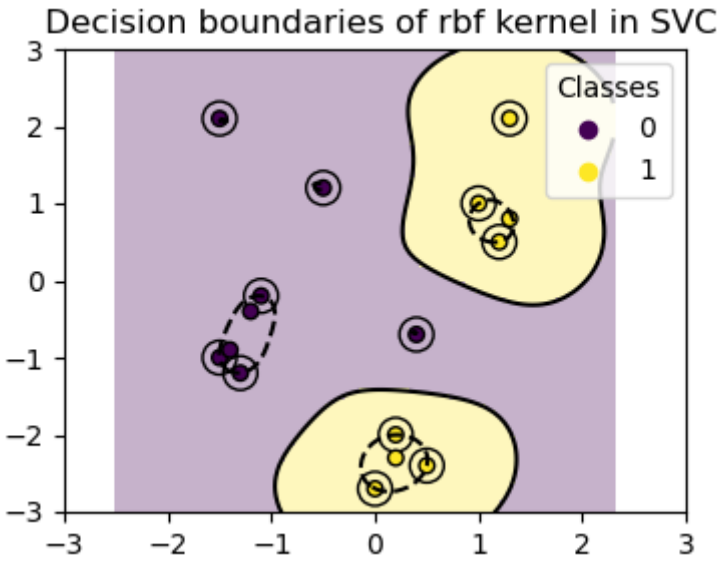


Рис. 2.3. Границя рішень SVM із RBF ядром [28]

Це дозволяє виявляти ботнети навіть у складних і динамічних мережевих умовах. Однак SVM може бути обчислювально витратним при роботі з великими наборами даних, що є важливим фактором при виборі алгоритму [10].

Вагомою проблемою при застосуванні машинного навчання до виявлення ботнетів є проблема незбалансованості даних: шкідливий трафік часто становить малу частку від загального обсягу. Це може призвести до того, що моделі навчаться добре класифікувати лише більш представлений клас, ігноруючи менш представлений. Для вирішення цієї проблеми застосовуються методи балансування даних, такі як SMOTE.

Крім того, висока вимірність даних (велика кількість ознак) може ускладнювати навчання моделей і призводити до перенавчання [29]. Тому важливо проводити відбір ознак, щоб залишити найбільш інформативні та зменшити складність моделі. Це не тільки покращує продуктивність, але й зменшує час тренування моделей. Також важливо пам'ятати, що динамічність ботнетів та їх здатність адаптуватися до нових умов вимагають від моделей бути гнучкими та здатними виявляти нові типи атак [22].

2.2. Порівняльний аналіз алгоритмів Random Forest, XGBoost та SVM

У процесі виявлення ботнетів у корпоративних мережах ключову роль відіграють алгоритми машинного навчання, здатні ефективно обробляти великі обсяги даних та виявляти складні патерни поведінки. Серед найпопулярніших алгоритмів для цієї задачі виділяються Random Forest, XGBoost та Support Vector Machine (SVM), тому вони і були обрані для цього дослідження. Проведемо детальний порівняльний аналіз цих методів, зосереджуючись на їхніх характеристиках, перевагах та обмеженнях у контексті виявлення ботнет-активності.

Random Forest забезпечує високу точність та стійкість до перенавчання, але може бути повільним. Основна концепція Random Forest полягає у створенні великої кількості незалежних дерев, кожне з яких навчається на випадково вибраній підмножині даних та ознак. Такий підхід допомагає знизити кореляцію між деревами та покращити здатність моделі до узагальнення. Випадковий вибір даних та ознак для кожного дерева зменшує ризик перенавчання, а об'єднання результатів через ансамблеве голосування робить модель більш стійкою до шуму та аномалій у даних. Крім того, оскільки дерева навчаються незалежно, алгоритм легко масштабувати за допомогою паралельних обчислень, що значно пришвидшує процес навчання на багатоядерних системах [8].

Розглянемо переваги Random Forest:

Точність класифікації. Завдяки ансамблевому підходу модель демонструє високу ефективність у задачах виявлення складних патернів у даних;

Стійкість до перенавчання. Рандомізація вибору ознак і підмножин даних мінімізує ризик перенавчання, що важливо для аналізу різноманітного мережевого трафіку;

Оцінка важливості ознак. Алгоритм дозволяє оцінювати важливість окремих ознак, що робить його зручним для вибору найбільш релевантних характеристик;

Ефективна робота з великою кількістю ознак. Добре справляється зі складними наборами ознак, що типово для аналізу мережевого трафіку.

Варто зазначити також обмеження Random Forest:

Обчислювальна складність. Велика кількість дерев збільшує час навчання й прогнозування, що може бути проблематичним для застосувань у реальному часі;

Вимоги до пам'яті. Зберігання всіх дерев потребує значних ресурсів;

Менша інтерпретованість ансамблю. Окремі дерева легко інтерпретувати, але великий ансамбль може бути складним для розуміння.

Наступною моделлю МН є XGBoost (Extreme Gradient Boosting), який характеризується високою швидкістю і точністю, добре масштабується на великі набори даних. XGBoost є ефективною реалізацією градієнтного бустингу дерев рішень, оптимізованою для високої швидкості та продуктивності. На відміну від Random Forest, дерева в XGBoost створюються послідовно, і кожне наступне дерево компенсує помилки попередніх, що дозволяє моделі фокусуватися на складних для класифікації зразках. Градієнтний бустинг оптимізує диференційовану функцію втрат шляхом додавання нових дерев, які мінімізують помилки попередніх. Включає параметри регуляризації для контролю складності моделі та запобігання перенавчанню [9].

Оптимізована реалізація з використанням паралельних та розподілених обчислень прискорює процес навчання. XGBoost має вбудовані механізми для роботи з пропущеними даними, що підвищує його гнучкість.

Звернемо увагу на переваги XGBoost:

Швидкість. Завдяки оптимізованій реалізації й можливості паралельного обчислення XGBoost є одним із найшвидших алгоритмів у своєму класі;

Висока точність. Градієнтний бустинг дозволяє моделі детально працювати з тонкими патернами, що покращує якість класифікації;

Гнучкість. Алгоритм має безліч параметрів, які можна налаштовувати під конкретну задачу;

Обробка великих наборів даних. Ефективно працює з великими наборами даних як за кількістю зразків, так і за кількістю ознак;

Проте ця модель має також певні обмеження:

Складність налаштування. Велика кількість гіперпараметрів може ускладнити процес оптимізації моделі;

Ризик перенавчання. У разі недостатньої регуляризації модель може надмірно підлаштовуватися під дані навчання;

Складність інтерпретації. Як і Random Forest, XGBoost є ансамблевим методом, що ускладнює аналіз результатів.

Наступним методом МН є SVM. Він є потужним алгоритмом для класифікації та регресії, який шукає оптимальну гіперплощину, що максимізує відстань (запас) між класами у просторі ознак. Використовуючи ядрові функції, SVM може ефективно працювати в нелінійних задачах, трансформуючи дані у простір більшої розмірності. Алгоритм прагне знайти гіперплощину, яка забезпечує максимальний запас між найближчими зразками різних класів (опорними векторами). Параметр регуляризації C контролює баланс між максимізацією запасу та помилками класифікації [10].

До переваг SVM відносяться:

Висока точність. SVM ефективно справляється із задачами, де класи добре розділені;

Гнучкість через ядрові функції. Можливість адаптуватися до різних типів даних за допомогою лінійних, поліноміальних чи RBF-ядер;

Ефективність у високовимірних даних. Особливо добре працює, коли кількість ознак перевищує кількість зразків.

До обмежень SVM відносяться:

Обчислювальна складність. При роботі з великими наборами даних процес навчання може бути дуже ресурсозатратним;

Чутливість до налаштування гіперпараметрів. Вибір ядрової функції та параметрів може значно впливати на продуктивність моделі;

Обмежена інтерпретованість. Менш прозорий у визначенні важливості ознак, особливо при використанні нелінійних ядер;

Проблеми з незбалансованими даними. Може мати труднощі у випадках, коли класи незбалансовані, що часто трапляється у задачах виявлення ботнетів.

Порівняння цих можелей вказує на те, що XGBoost часто перевершує інші моделі за точністю та швидкістю, а Random Forest залишається надійним методом із простою інтерпретацією. Random Forest та XGBoost демонструють високу точність класифікації в задачі виявлення ботнетів. Це відбувається завдяки їхнім ансамблевим підходам: вони можуть вловлювати складні патерни у даних. XGBoost часто виявляється ефективнішим за Random Forest, адже його навчання та прогнозування зазвичай швидші завдяки оптимізації для роботи з великими обсягами даних. Обидва алгоритми мають інструменти для запобігання перенавчанню, зокрема випадковий вибір ознак і даних у Random Forest та регуляризацію і техніки ранньої зупинки у XGBoost.

SVM, хоча й забезпечує високу точність у складних задачах, може бути менш ефективним при роботі з великими обсягами даних, що часто характерно для аналізу мережевого трафіку. Його обчислювальна складність та необхідність ретельного налаштування гіперпараметрів ускладнюють застосування у реальному часі.

Обчислювальна ефективність є критично важливою для виявлення ботнетів у реальному часі. XGBoost відзначається меншим часом навчання та прогнозування порівняно з Random Forest завдяки оптимізованій реалізації та паралельним обчисленням. Random Forest, у свою чергу, може вимагати більше ресурсів при роботі з великою кількістю дерев і ознак, але паралелізація частково компенсує цей недолік. SVM потребує значних обчислювальних ресурсів у великих наборах даних, що обмежує його використання у реальному часі.

Важливість ознак та інтерпретованість моделі є важливими аспектами в контексті кібербезпеки. Random Forest надає чіткі індикатори важливості ознак, що сприяє кращому розумінню моделі та допомагає у відборі найважливіших характеристик для виявлення ботнетів. XGBoost також дозволяє оцінювати важливість ознак, але складність ансамблю може ускладнювати інтерпретацію. SVM менш прозорий у визначенні важливості ознак, що може бути недоліком у контексті, де пояснюваність є дуже важливою.

У задачах виявлення ботнетів часто зустрічається проблема незбалансованих даних, де кількість нормальних зразків значно перевищує кількість шкідливих. Random Forest та XGBoost мають механізми для роботи з незбалансованими даними, такі як ваги класів та спеціальні функції втрат, що допомагає підвищити точність виявлення менш представлених класів. SVM може мати проблеми з незбалансованими даними, оскільки алгоритм прагне максимізувати запас без урахування розподілу класів, що може призвести до низької точності виявлення рідкісних подій.

Загалом, при виборі алгоритму для виявлення ботнетів у корпоративних мережах слід враховувати специфіку задачі, обсяг та характер даних, вимоги до точності та швидкості, а також можливість інтерпретації моделі. Random Forest та XGBoost є перспективними кандидатами завдяки своїм характеристикам та можливостям адаптації до складних умов мережевого трафіку. Random Forest підходить для задач, де потрібна висока точність, інтерпретованість та стійкість до перенавчання, ефективно працює з високовимірними даними та незбалансованими класами. XGBoost відзначається високою точністю та швидкістю, що робить його придатним для виявлення ботнетів у реальному часі, хоча вимагає ретельного налаштування та може перенавчатися без належної регуляризації.

SVM може бути корисним у певних сценаріях, особливо коли дані чітко роздільні, але його обмеження, такі як обчислювальна складність при великих обсягах даних та чутливість до налаштування гіперпараметрів, можуть стати перешкодою для ефективного застосування. Тому в контексті виявлення ботнетів у корпоративних мережах Random Forest та XGBoost можуть бути більш підходящими виборами [7, 23].

2.3. Опис методу виявлення ботнетів за допомогою машинного навчання

Процес побудови методу виявлення ботнетів за допомогою машинного навчання можна визначити як комплексний ланцюг взаємопов’язаних дій, кожна з

яких відіграє важливу роль у досягненні кінцевої мети — надійного та ефективного виявлення загроз, а саме ботнетів, у сучасних корпоративних мережах. Цей процес можна умовно розділити на кілька ключових етапів (рисунок 2.4):

1. Збір та попередня обробка даних. Підготовка даних для аналізу шляхом очищення та перетворення.
2. Балансування даних. Вирівнювання розподілу класів для забезпечення ефективного навчання моделей.
3. Навчання та оптимізація моделей. Побудова моделей машинного навчання з використанням оптимальних гіперпараметрів.
4. Оцінка моделей. Перевірка продуктивності моделей за допомогою відповідних метрик та крос-валідації.
5. Аналіз важливості ознак. Визначення ключових ознак, що впливають на рішення моделей.

Рис. 2.4. Схема запропонованого методу виявлення ботнетів у мережевому трафіку з використанням моделей машинного навчання

Цей підхід дозволяє розробити ефективний метод виявлення ботнетів, враховуючи особливості даних та вимоги до продуктивності.

Для навчання моделей використовується датасет CSE-CIC-IDS2018, створений у співпраці між Communications Security Establishment (CSE) та Canadian Institute for Cybersecurity (CIC) з метою моделювання реалістичних сценаріїв мережевих атак та нормального трафіку. Це дозволяє дослідникам та фахівцям з кібербезпеки розробляти та оцінювати системи виявлення вторгнень. Датасет охоплює сім різних типів атак:

1. Brute-force атаки: Спроби несанкціонованого доступу шляхом перебору можливих комбінацій логінів та паролів. Включає атаки на протоколи FTP та SSH.
2. Heartbleed: Експлуатація вразливості в бібліотеці OpenSSL, що дозволяє зловмиснику отримати доступ до конфіденційної інформації з пам'яті серверу.
3. Botnet: Мережі скомпрометованих комп'ютерів, які виконують команди зловмисника, такі як розсилання спаму або здійснення DDoS-атак.
4. DoS (Denial of Service): Атаки, спрямовані на виведення з ладу сервісу шляхом перевантаження його запитами. Включає атаки типу Hulk, GoldenEye, Slowloris та Slowhttptest.
5. DDoS (Distributed Denial of Service): Розподілені атаки на відмову в обслуговуванні, здійснювані з багатьох джерел одночасно. Включає атаки LOIC (Low Orbit Ion Cannon) та HOIC (High Orbit Ion Cannon).
6. Веб-атаки: Атаки на веб-додатки, такі як SQL-ін'єкції, XSS (Cross-Site Scripting) та інші методи компрометації веб-сайтів.
7. Інфільтрація мережі зсередини: Сценарії, де зловмисник отримує доступ до внутрішньої мережі організації, використовуючи методи соціальної інженерії або інші техніки.

Для генерації даних використовувалася концепція профілів, що містять детальні описи вторгнень та абстрактні моделі розподілу для застосунків, протоколів або нижчих рівнів мережевих сутностей. Ці профілі поділяються на два типи:

B-профілі: Описують поведінку користувачів, використовуючи методи машинного навчання та статистичного аналізу, такі як K-Means, Random Forest, SVM та J48. Вони включають розподіли розмірів пакетів протоколу, кількість пакетів на потік, певні шаблони в корисному навантаженні, розмір корисного навантаження та розподіл часу запитів протоколу. Симульовані протоколи включають HTTPS, HTTP, SMTP, POP3, IMAP, SSH та FTP;

M-профілі: Описують сценарії атак у чіткий спосіб, що дозволяє як людям, так і автономним агентам виконувати ці сценарії. Розглянуто шість різних сценаріїв атак, включаючи інфільтрацію мережі зсередини, де зловмисник надсилає шкідливий файл через електронну пошту, експлуатує вразливість додатка, виконує бекдор на комп'ютері жертви та сканує внутрішню мережу для пошуку інших вразливих систем.

Атакуюча інфраструктура складалася з 50 машин, тоді як організація-жертва мала 5 відділів, включаючи 420 комп'ютерів та 30 серверів. Це дозволило створити реалістичне середовище для моделювання різних типів атак та нормального мережевого трафіку.

Файл "Friday-02-03-2018_TrafficForML_CICFlowMeter.csv" з цього датасету містить мережевий трафік з мітками "BENIGN" та ботнетами, він і використовувався для проведення експерименту [3].

Попередня обробка даних включає кілька важливих етапів. На першому етапі завантажуються всі файли датасету, які потім об’єднуються в один великий набір даних. Це дозволяє підвищити різноманітність даних та збільшити узагальнюючу здатність моделей. Для цього використовується бібліотека pandas, що забезпечує зручність і ефективність роботи з великими обсягами даних.

Після завантаження даних здійснюється первинний аналіз структури та змісту датасету. Визначається кількість записів та ознак, типи даних у стовпцях, а також розподіл класів. Аналіз розподілу класів виявляє наявність незбалансованості, де кількість нормального трафіку значно перевищує кількість ботнет-активності.

Далі здійснюється видалення нерелевантних та дубльованих стовпців. Зокрема, видаляється стовпець з часовими мітками, оскільки в даному контексті часові мітки не сприяють виявленню ботнет-активності і можуть вносити шум у моделі [30]. Також перевіряється наявність дубльованих або постійних стовпців, які не містять варіативності, і такі стовпці видаляються для зменшення розмірності даних.

Пропущені значення можуть негативно впливати на навчання моделей, тому проводиться перевірка кожного стовпця на їх наявність. У числових ознаках пропущені значення заповнюються середніми значеннями відповідних стовпців, що дозволяє зберегти статистичні властивості даних. У категоріальних ознаках пропущені значення замінюються найбільш частим значенням або спеціальним маркером "Unknown", щоб не втратити інформацію.

Викиди або аномальні значення також можуть спотворювати навчання моделі. Для їх виявлення проводиться аналіз розподілу числових ознак за допомогою діаграм розмаху та гістограм. У випадках, коли викиди є результатом помилок збору даних, вони видаляються або обмежуються певним порогом.

Категоріальні ознаки потребують перетворення у числовий формат для використання в моделях машинного навчання. У цьому дослідженні ознака "Protocol", яка відображає типи мережевих протоколів (TCP, UDP, ICMP тощо), перетворюється за допомогою методу one-hot encoding.

One-hot encoding є методом кодування категоріальних змінних шляхом створення бінарних змінних для кожного унікального значення категорії [31]. Для кожного унікального значення ознаки "Protocol" створюється окремий стовпець. У записі, що відповідає певному протоколу, значення у відповідному стовпці встановлюється в 1, а в інших — 0.

Застосування one-hot encoding у цьому дослідженні має кілька переваг:

Відсутність введення помилкової порядкової інформації. Оскільки протоколи не мають природного порядку, one-hot encoding запобігає виникненню помилкових порядкових взаємозв'язків, які могли б вплинути на модель при використанні інших методів кодування;

Сумісність з моделями. Деякі алгоритми машинного навчання, такі як SVM та лінійні моделі, можуть неправильно інтерпретувати числові значення категорій при використанні label encoding. One-hot encoding дозволяє уникнути цієї проблеми, представляючи кожну категорію окремо;

Збереження повної інформації про категорії. Усі категорії представлені явно, що дозволяє моделям враховувати вплив кожного протоколу окремо.

Реалізація one-hot encoding здійснюється за допомогою функції get_dummies() з бібліотеки pandas. Отримані бінарні стовпці інтегруються в датасет, а оригінальна ознака "Protocol" видаляється після кодування. Кількість додаткових стовпців залежить від кількості унікальних значень ознаки "Protocol".

Проблема незбалансованості даних є поширеною у сфері виявлення ботнетів, оскільки шкідливий трафік може становити лише невелику частку від загального обсягу даних [32]. У нашому випадку, датасет CSE-CIC-IDS2018 чітко демонструє цю проблему: нормальний трафік значно переважає, що може призвести до того, що моделі машинного навчання будуть переважно класифікувати всі зразки як нормальні, ігноруючи менш представлений клас ботнет-активності. Це явище відоме як перекіс класу і може суттєво знизити ефективність системи виявлення, особливо в метриках, що стосуються менш представленого класу, таких як точність (precision) та повнота (recall).

Незбалансованість даних впливає на навчання моделей наступним чином:

Моделі можуть пристосуватися до того, щоб постійно прогнозувати найбільш представлений клас, досягаючи високої загальної точності, але не виявляючи шкідливу активність (схильність до більшості);

Менш представлений клас (ботнет-активність) може залишатися невиявленим, що є надважливим у сфері кібербезпеки (переважання помилкових негативів);

Стандартні функції втрат не враховують незбалансованість, що призводить до некоректного оновлення ваг моделі (викривлення функції втрат).

Для вирішення цієї проблеми існують різні підходи, які можна поділити на кілька категорій:

Методи вибірки даних (Resampling techniques), зокрема, oversampling (збільшення менш представленого класу), undersampling (зменшення кількості зразків більш представленого класу) та їх комбінації;

Методи на основі алгоритмів, такі як застосування вагових коефіцієнтів до функції втрат для менш представленого класу або використання спеціалізованих алгоритмів, розроблених для роботи з незбалансованими даними;

Методи на основі ансамблів, наприклад, BalancedBaggingClassifier або BalancedRandomForestClassifier, які інтегрують балансування у процес навчання.

У цьому дослідженні було обрано метод вибірки даних, зокрема oversampling за допомогою SMOTE. Цей метод було обрано через його здатність генерувати синтетичні зразки менш представленого класу без дублювання існуючих даних, що допомагає уникнути перенавчання моделей.

SMOTE (Synthetic Minority Over-sampling Technique) — це метод генерації синтетичних зразків для менш представленого класу [33]. Він працює шляхом інтерполяції між існуючими зразками меншого класу, створюючи нові, штучні зразки, які зберігають властивості оригінальних даних. Принцип роботи SMOTE полягає у виборі зразка меншого класу та визначенні його k найближчих сусідів того ж класу. Потім вибирається випадковий сусід, і створюється новий зразок шляхом лінійної інтерполяції між обраним зразком та його сусідом.

Основними параметрами SMOTE є k_neighbors (кількість найближчих сусідів для генерації синтетичних зразків, типове значення — 5), sampling_strategy (відсоток, до якого потрібно збільшити менш представлений клас) та random_state (параметр для відтворюваності результатів). У цьому дослідженні SMOTE застосовується до навчального набору після розділення на тренувальний та тестовий набори, щоб уникнути витоку інформації.

Перевагами SMOTE є збереження інформації, оскільки, на відміну від простого дублювання, він створює нові зразки, що допомагає моделі краще узагальнювати, та уникнення перенавчання. Проте, за наявності викидів у даних, SMOTE може генерувати шумові зразки. Окрім того, метод не враховує розподіл класів, що іноді призводить до збільшення кількості помилкових позитивів.

Іншим методом, який варто розглянути, є ADASYN (Adaptive Synthetic Sampling), який генерує синтетичні зразки з більшим акцентом на ті області, де менший клас менш репрезентований. Це дозволяє моделям краще розрізняти складні межі класів, проте ADASYN може збільшувати обчислювальні витрати та також ризикує створювати шумові зразки. У порівнянні з Random Oversampling, який просто дублює існуючі зразки менш представленого класу, SMOTE та ADASYN надають більш складні та інформативні синтетичні зразки, що покращує здатність моделей до узагальнення.

Варто звернути увагу на undersampling. Це метод зменшення кількості зразків більш представленого класу шляхом видалення деяких з них, щоб досягти балансу між класами без збільшення розміру датасету. Існують різні методи undersampling, такі як Random Undersampling, Cluster Centroids та NearMiss.

Перевагами цього підходу є зменшення обсягу даних, що може прискорити навчання моделі, та можливість видалення надлишкових або шумових зразків. Недоліки включають можливу втрату інформації через видалення корисних зразків та ризик перенавчання на меншому наборі даних.

Комбінація цих методів може забезпечити оптимальний баланс між класами [34].

Для оцінки продуктивності моделей дані розділяються на навчальний та тестовий набори, обираючи співвідношення 80% для навчання та 20% для тестування. Розділення здійснюється за допомогою функції train_test_split з бібліотеки scikit-learn з фіксованим параметром random_state для відтворюваності результатів. Забезпечується стратифікація, тобто однаковий розподіл класів у навчальному та тестовому наборах, щоб уникнути перекосу в даних.

Після ретельної підготовки та попередньої обробки даних здійснюється навчання моделей машинного навчання. Навчання моделей здійснюється на підготовлених даних, де було проведено кодування категоріальних ознак, масштабування числових ознак та балансування класів за допомогою SMOTE. Дані розділяються на навчальний та тестовий набори у співвідношенні 80% для навчання та 20% для тестування. Для збереження пропорцій між класами

застосовується стратифікація, що дозволяє коректно оцінювати моделі та уникати перекосу в розподілі класів.

Random Forest представляє собою ансамблевий алгоритм, який комбінує декілька дерев рішень для досягнення кращої точності та стабільності моделі [35]. Цей метод особливо корисний для обробки великих обсягів даних, де він може виявляти складні взаємозв'язки між різними ознаками. Завдяки рандомізації підмножин даних та ознак для кожного дерева цей алгоритм значно знижує ризик перенавчання, забезпечуючи при цьому узагальнення. У практичному застосуванні це робить модель ефективною навіть за наявності шуму в даних, що є важливим для вирішення завдань виявлення ботнетів.

XGBoost є потужною реалізацією градієнтного бустингу, оптимізованою для високої швидкості та продуктивності [36]. Цей алгоритм добре масштабується при роботі з великими наборами даних і здатний виявляти складні патерни завдяки послідовному додаванню дерев, кожне з яких намагається виправити помилки попередніх моделей. XGBoost має вбудовані механізми регуляризації, які дозволяють уникнути перенавчання, підтримуючи баланс між складністю моделі та її узагальнювальними можливостями. Завдяки цим характеристикам, XGBoost є ефективним інструментом для виявлення ботнет-активності в реальному часі.

Support Vector Machine (SVM) є потужним методом класифікації, який шукає оптимальну гіперплощину для розділення класів у багатовимірному просторі ознак [37]. SVM відзначається високою точністю класифікації, особливо при роботі з даними, що мають чітко роздільні класи. Алгоритм використовує ядрові функції для трансформації даних у високовимірний простір, що дозволяє ефективно працювати з нелінійними задачами класифікації. Однак, варто зазначити, що SVM може бути обчислювально затратним при обробці великих наборів даних, що обмежує його застосування у великих корпоративних мережах. Перед навчанням SVM дані масштабуються за допомогою StandardScaler, оскільки алгоритм чутливий до масштабу ознак.

Для підвищення ефективності моделей проводиться оптимізація гіперпараметрів за допомогою методу байєсової оптимізації, реалізованої через

бібліотеку Hyperopt [38]. Байєсова оптимізація дозволяє ефективно досліджувати простір гіперпараметрів та знаходити їх оптимальні значення, зменшуючи кількість необхідних ітерацій порівняно з методами випадкового або сіткового пошуку. Байєсова оптимізація була обрана через її здатність швидше знаходити оптимальні гіперпараметри з меншою кількістю ітерацій, що особливо важливо при роботі з великими наборами даних та складними моделями.

На відміну від Grid Search, який вимагає значних обчислювальних ресурсів через перебір всіх можливих комбінацій гіперпараметрів, та Random Search, який є менш ефективним у знаходженні оптимальних комбінацій, байєсова оптимізація пропонує більш ефективний підхід до оптимізації. Вона використовує ймовірнісну модель для передбачення найбільш перспективних областей простору гіперпараметрів, що дозволяє зменшити кількість необхідних ітерацій та зменшити загальні обчислювальні витрати. Це особливо важливо при роботі з великими наборами даних та складними моделями, де традиційні методи оптимізації можуть бути занадто ресурсомісткими та часозатратними.

Бібліотека Hyperopt надає можливість проводити байєсову оптимізацію, використовуючи алгоритм Tree-structured Parzen Estimator (TPE). Цей метод моделює функцію втрат та використовує інформацію з попередніх ітерацій для вибору найбільш перспективних областей простору гіперпараметрів. Це сприяє більш ефективному пошуку оптимальних значень та зменшує обчислювальні витрати.

Процес оптимізації для кожної моделі включає наступні етапи:

Визначення простору гіперпараметрів. Встановлюються діапазони значень для кожного гіперпараметра, які будуть досліджуватися під час оптимізації. Ці діапазони обираються на основі попереднього досвіду, рекомендацій з літератури та специфіки даних;

Визначення функції втрат (objective function). Створюється функція, яка буде мінімізована під час оптимізації. Вона обчислює метрику продуктивності моделі (наприклад, негативну точність або F1-міру) на валідаційному наборі даних;

Запуск процесу оптимізації. Виконується задана кількість ітерацій, під час яких Hyperopt обирає значення гіперпараметрів, тренує модель та оцінює її продуктивність. Алгоритм TPE використовує отримані результати для побудови ймовірнісної моделі та визначення наступних значень гіперпараметрів для перевірки;

Аналіз результатів. Після завершення оптимізації аналізуються знайдені гіперпараметри та їх вплив на продуктивність моделі. Вибираються найкращі значення, які забезпечують максимальну метрику на валідаційному наборі.

Оптимізація гіперпараметрів Random Forest. Для моделі Random Forest оптимізуються наступні гіперпараметри:

n_estimators: Кількість дерев у лісі. Більша кількість дерев може покращити продуктивність, але збільшує час навчання;

max_depth: Максимальна глибина дерев. Обмеження глибини запобігає перенавчанню та зменшує обчислювальну складність;

min_samples_split: Мінімальна кількість зразків, необхідних для розділення внутрішнього вузла. Допомагає контролювати складність моделі та запобігати перенавчанню;

min_samples_leaf: Мінімальна кількість зразків, необхідних для утворення листового вузла. Впливає на згладжування моделі;

max_features: Кількість ознак, які розглядаються при розділенні вузла. Вибір між 'sqrt' та 'log2' дозволяє контролювати різноманітність дерев у лісі.

Функція втрат визначається як негативне значення точності моделі на тестовому наборі. Мета — мінімізувати цю функцію, що еквівалентно максимізації точності.

Оптимізація проводиться з використанням алгоритму TPE протягом 30 ітерацій. Фіксується генератор випадкових чисел для відтворюваності результатів. Після оптимізації отримуються найкращі значення гіперпараметрів, які використовуються для побудови остаточної моделі Random Forest.

Оптимізація гіперпараметрів XGBoost. Для XGBoost оптимізуються такі гіперпараметри:

n_estimators: Кількість ітерацій бустингу;

max_depth: Максимальна глибина дерев;

gamma: Мінімальне зменшення функції втрат, необхідне для подальшого розділення вузла. Впливає на складність моделі;

reg_alpha: Параметр L1-регуляризації, який сприяє розрідженості моделі;

reg_lambda: Параметр L2-регуляризації, який допомагає уникнути великих ваг;

colsample_bytree: Частка ознак, які використовуються для кожного дерева. Допомагає зменшити кореляцію між деревами;

min_child_weight: Мінімальна сума ваг всіх зразків у листовому вузлі. Контролює складність моделі.

Оптимізація проводиться з використанням Hyperopt та TPE протягом 30 ітерацій. Після оптимізації визначаються найкращі значення гіперпараметрів для побудови остаточної моделі XGBoost.

Оптимізація гіперпараметрів SVM. Для SVM оптимізується параметр C. Це параметр регуляризації, який контролює баланс між максимізацією маржі та мінімізацією помилок класифікації.

Оптимізація проводиться протягом 10 ітерацій з використанням TPE. Отримане оптимальне значення C використовується для побудови остаточної моделі SVM.

Для SVM дані масштабуються за допомогою StandardScaler, оскільки алгоритм чутливий до масштабу ознак. Це забезпечує коректне визначення оптимальної гіперплощини розділення.

Після оптимізації гіперпараметрів остаточні моделі навчаються на повному навчальному наборі даних з використанням знайдених оптимальних значень. Це дозволяє максимально використати доступні дані для покращення продуктивності моделей.

Після навчання та оптимізації моделей Random Forest, XGBoost та Support Vector Machine (SVM) проводиться їх ретельна оцінка, щоб визначити їхню ефективність у виявленні ботнет-активності в корпоративних мережах. Оцінка

моделей дозволяє зрозуміти, наскільки добре моделі здатні узагальнювати знання на нових даних і виявляти шкідливий трафік.

Вибір відповідних метрик оцінки є найбільш важливим для коректної інтерпретації результатів моделювання. У контексті виявлення ботнет-активності важливо не лише правильно класифікувати якомога більше зразків, але й мінімізувати кількість помилкових позитивів та негативів. Для комплексної оцінки моделей були обрані наступні метрики:

Точність (Accuracy) — Відображає загальну частку правильно класифікованих зразків серед усіх зразків. Точність розраховується за формулою (2.7):

$$\text{Точність} = \frac{\text{Кількість правильних прогнозів}}{\text{Загальна кількість зразків}} \quad (2.7)$$

У контексті незбалансованих даних, як у нашому випадку, точність може бути оманливо високою, якщо модель добре класифікує більший клас, але погано — менший. Тому ця метрика повинна використовуватися разом з іншими;

Точність передбачення (Precision) — Показує, яку частку зразків, які модель класифікувала як позитивні (виявлено ботнет), дійсно є позитивними. Розраховується за формулою (2.8):

$$\text{Точність передбачення} = \frac{\text{Істинно позитивні (TP)}}{\text{Істинно позитивні (TP)} + \text{Хибно позитивні (FP)}} \quad (2.8)$$

Висока точність передбачення означає, що модель має низьку кількість помилкових спрацьовувань, що важливо для зменшення навантаження на систему безпеки через аналіз помилкових тривог;

Повнота (Recall) — Відображає, яку частку реальних позитивних зразків модель змогла правильно ідентифікувати. Розраховується за формулою (2.9):

$$\text{Повнота} = \frac{\text{Істинно позитивні (TP)}}{\text{Істинно позитивні (TP)} + \text{Хибно негативні (FN)}} \quad (2.9)$$

Висока повнота означає, що модель здатна виявити більшість ботнет-активності, що дуже важливо для безпеки мережі;

F1-міра — Гармонічне середнє між точністю передбачення та повнотою. Розраховується за формулою (2.10):

$$\text{F1} - \text{міра} = 2 * \frac{\text{Точність передбачення} * \text{Повнота}}{\text{Точність передбачення} + \text{Повнота}} \quad (2.10)$$

F1-міра є збалансованою метрикою, яка враховує як точність передбачення, так і повноту. Вона особливо корисна в умовах незбалансованих даних, де важливо знайти баланс між виявленням всіх позитивних зразків та мінімізацією помилкових тривог [39].

У задачі виявлення ботнет-активності помилкові негативи (коли шкідливий трафік не виявлено) можуть призвести до серйозних наслідків для безпеки мережі. Тому метрика повноти (Recall) має особливе значення, оскільки відображає здатність моделі виявляти всі випадки ботнет-активності.

З іншого боку, помилкові позитиви (коли нормальний трафік помилково класифікується як шкідливий) можуть спричинити непотрібні сповіщення та збільшити навантаження на аналітиків безпеки. Тому точність передбачення (Precision) також є важливою метрикою.

F1-міра поєднує ці два аспекти, забезпечуючи збалансовану оцінку моделі. Високе значення F1-міри свідчить про те, що модель добре справляється як з виявленням ботнет-активності, так і з мінімізацією помилкових тривог.

Матриця неточностей (Confusion Matrix) також використовується для візуалізації результатів класифікації, надаючи детальну інформацію про кількість істинно позитивних, хибно позитивних, істинно негативних та хибно негативних прогнозів.

Крос-валідація є статистичним методом оцінки та порівняння моделей, що використовується для уникнення перенавчання та забезпечення стабільності результатів [40]. У цьому дослідженні застосовано 5-кратну крос-валідацію, яка полягає в наступному процесі:

Розбиття даних на фолди. Початковий набір даних рівномірно розділяється на 5 непересічних підмножин (фолдів);

Навчання та тестування моделі. Модель навчається на 4 фолдах, а оцінюється на п'ятому, який слугує валідаційним набором;

Повторення процесу. Цей процес повторюється 5 разів, кожного разу використовуючи інший фолд як валідаційний;

Агрегація результатів. Після завершення всіх ітерацій обчислюються середні значення метрик оцінки, що надає більш надійну оцінку продуктивності моделі.

Вибір 5-кратної крос-валідації обумовлений балансом між обчислювальною ефективністю та надійністю оцінки моделей. Цей підхід дозволяє достатньо добре оцінити узагальнюючу здатність моделі без значних обчислювальних витрат.

Крос-валідація дозволяє:

Оцінити узагальнюючу здатність моделі. Використання різних підмножин даних для навчання та тестування допомагає оцінити, як добре модель буде працювати на невідомих даних;

Уникнути перенавчання. Перевірка моделі на різних валідаційних наборах виявляє можливе перенавчання, коли модель добре працює на навчальних даних, але погано на нових;

Забезпечити стабільність результатів. Середнє значення метрик по всіх фолдах зменшує вплив випадкових варіацій у даних.

У цьому дослідженні крос-валідація реалізується за допомогою функції cross_val_score з бібліотеки scikit-learn.

Після проведення крос-валідації отримуються п'ять значень F1-міри для кожної моделі. Середнє значення та стандартне відхилення обчислюються для оцінки стабільності моделі. Наприклад, для оптимізованої моделі Random Forest може бути отримано такі резульати:

F1 Scores: [0.98, 0.97, 0.98, 0.97, 0.98];

Середній F1 Score: 0.976;

Стандартне відхилення: 0.005

Маленьке стандартне відхилення свідчить про стабільність моделі на різних підмножинах даних.

Високе середнє значення F1-міри вказує на те, що модель ефективно виявляє ботнет-активність, зберігаючи баланс між точністю передбачення та повнотою. Стабільність результатів на різних фолдах підтверджує, що модель не перенавчена і здатна узагальнювати знання на нових даних.

Матриці неточностей для кожної моделі надають більш детальний погляд на розподіл прогнозів:

Істинно позитивні (TP) — Кількість випадків ботнет-активності, правильно ідентифікованих моделлю;

Хибно позитивні (FP) — Кількість випадків нормального трафіку, помилково класифікованих як ботнет-активність;

Істинно негативні (TN) — Кількість випадків нормального трафіку, правильно класифікованих як нормальні;

Хибно негативні (FN) — Кількість випадків ботнет-активності, які модель не виявила.

Модель з меншою кількістю хибно негативних результатів є бажаною, оскільки це означає, що шкідливий трафік не пропускається.

Після навчання та оцінки моделей машинного навчання, наступним етапом є аналіз важливості ознак, що використовуються моделями для прийняття рішень. Цей аналіз дозволяє визначити, які характеристики мережевого трафіку найбільше впливають на виявлення ботнет-активності. Наприклад, ознаки, пов'язані з портами призначення, швидкістю передачі пакетів та інтервалами між пакетами, можуть бути особливо важливими для розпізнавання ботнетів [41]. Розуміння важливості ознак не тільки підвищує інтерпретованість моделей, але й сприяє подальшому вдосконаленню систем виявлення та оптимізації набору ознак для майбутніх досліджень.

Аналіз важливості ознак є необхідним для:

Інтерпретації моделей. Допомагає зрозуміти, як модель приймає рішення, що особливо важливо в контексті кібербезпеки, де необхідно пояснювати виявлені загрози;

Виявлення ключових характеристик. Дозволяє ідентифікувати найважливіші ознаки, що впливають на класифікацію, та використовувати цю інформацію для вдосконалення системи моніторингу мережі;

Оптимізації моделі. Можливість зменшити розмірність даних, видаливши неінформативні ознаки, що може покращити продуктивність моделі та зменшити обчислювальні витрати.

У цьому дослідженні використовується кілька підходів для оцінки важливості ознак:

Важливість ознак у деревоподібних моделях. Для моделей Random Forest та XGBoost використовується внутрішня оцінка важливості ознак, заснована на тому, як часто та з яким впливом ознаки використовуються для розділення вузлів у деревах;

Метод SHAP (SHapley Additive exPlanations). Це сучасний метод, який надає можливість пояснити прогнози будь-якої моделі машинного навчання, розраховуючи внесок кожної ознаки до остаточного рішення;

Аналіз коефіцієнтів моделі. Для моделей, де це можливо, наприклад, лінійних моделей або SVM з лінійним ядром, коефіцієнти моделі можуть використовуватися для оцінки впливу ознак.

Важливість ознак у Random Forest та XGBoost. У моделі Random Forest важливість ознак оцінюється на основі зменшення критерію розщеплення при використанні конкретної ознаки. Показник важливості для ознаки розраховується як сумарне зменшення критерію розщеплення по всіх деревах, де ця ознака використовується. Це дозволяє визначити, які ознаки найбільше сприяють покращенню чистоти вузлів та, відповідно, класифікації.

У XGBoost важливість ознак може оцінюватися декількома способами:

Gain — Сумарне збільшення якості моделі при використанні ознаки для розщеплення вузлів;

Cover — Кількість зразків, які були залучені до розщеплення з використанням даної ознаки;

Frequency — Як часто ознака використовувалася для розщеплення вузлів.

У цьому дослідженні основний акцент зроблено на метриці Gain, оскільки вона відображає внесок ознаки в покращення моделі.

SHAP (SHapley Additive exPlanations) — це метод, заснований на теорії ігор, який обчислює значення Шеплі для кожної ознаки, що відображає її внесок у прогноз моделі. Переваги використання SHAP полягають у його універсальності, теоретичній обґрунтованості та можливості отримання як локальних, так і глобальних пояснень. SHAP дозволяє детальніше проаналізувати вплив кожної ознаки на прогнози моделі. Позитивні значення SHAP вказують на те, що ознака сприяє класифікації зразка як ботнет-активність, а негативні свідчать про те, що ознака сприяє класифікації зразка як нормальний трафік.

У цьому дослідженні для моделі XGBoost використовується бібліотека SHAP для розрахунку значень Шеплі. Процес включає побудову пояснювальної моделі, обчислення внеску кожної ознаки до прогнозу для кожного зразка, агрегацію результатів та візуалізацію, що надає глибоке розуміння роботи моделі.

Для моделей з лінійною структурою, таких як SVM з лінійним ядром, коефіцієнти моделі можуть використовуватися для оцінки впливу ознак. Однак у нашому випадку SVM використовував нелінійне ядро (RBF), тому пряме використання коефіцієнтів було неможливим. Тому акцент було зроблено на деревоподібних моделях та SHAP.

Розуміння важливості ознак дозволяє оптимізувати систему моніторингу, покращити моделі та виявляти нові загрози. Фокусуючись на зборі та аналізі найбільш інформативних ознак, можна зменшити обсяг даних та покращити швидкість виявлення. Видалення неінформативних або шумових ознак може підвищити точність та швидкість моделей. Розуміння, які ознаки вказують на

ботнет-активність, дозволяє створювати правила та політики безпеки для виявлення нових типів атак.

Важливо перевірити, чи залишаються визначені важливі ознаки такими ж впливовими на інших наборах даних або в реальних умовах. Ботнети постійно розвиваються, змінюючи свої патерни поведінки, тому необхідно регулярно оновлювати аналіз важливості ознак. Деякі ознаки можуть мати складні взаємодії, які важко інтерпретувати без додаткових методів аналізу. Це вимагає використання більш складних моделей пояснення або комбінування декількох методів для глибшого розуміння.

Аналіз важливості ознак є важливою частиною процесу побудови моделей машинного навчання для виявлення ботнет-активності. Використання методів визначення важливості, таких як внутрішні метрики моделей Random Forest та XGBoost, а також метод SHAP, дозволяє отримати детальне розуміння того, які характеристики мережевого трафіку є найбільш інформативними для виявлення шкідливої активності. Перспективним напрямком є використання інших методів пояснення, таких як LIME (Local Interpretable Model-agnostic Explanations), для порівняння результатів та отримання додаткових інсайтів. Дослідження взаємодій ознак та виявлення комбінацій, які разом мають значний вплив на прогноз моделі, можуть покращити точність та надійність виявлення ботнет-активності. Розробка адаптивних моделей, які можуть автоматично оновлювати важливість ознак у відповідь на зміну патернів ботнет-активності, також є важливим напрямком [41].

Під час проведення експериментів використовувався комп'ютер з такими апаратними та програмними характеристиками:

Процесор: Intel Core i5-7300HQ, 4 ядра, 2.50 ГГц;

Оперативна пам'ять: 16 ГБ;

Накопичувач: SSD SK hynix SC311 128 ГБ;

Операційна система: Windows 10 Pro, версія 22H2, збірка 19045.5131.

Робота виконувалась у середовищі Jupyter Notebook з використанням бібліотек для обробки даних та візуалізації результатів, таких як pandas, NumPy, scikit-learn, matplotlib, Seaborn, Plotly.

3 ТЕХНОЛОГІЯ ВИЯВЛЕННЯ РОБОТИ БОТНЕТУ В КОРПОРАТИВНІЙ МЕРЕЖІ ІЗ ЗАСТОСУВАННЯМ МАШИННОГО НАВЧАННЯ

3.1. Реалізація та результати експериментів

На початковому етапі роботи з даними було здійснено завантаження датасету та його первинний аналіз. Цей датасет характеризується великим обсягом мережевих потоків, які описуються числовими та категоріальними ознаками. Серед ключових характеристик виділяються розмір і довжина пакетів, проміжки між їх відправленням, типи протоколів, використані порти, швидкість передачі пакетів та інші особливості, що відображають поведінкові патерни трафіку. Такий набір параметрів забезпечує всебічний опис мережевої активності, створюючи умови для побудови складних і точних моделей.

Перевірка на наявність пропущених значень виявила, що деякі числові стовпці містять відсутні дані. Для заповнення пропущених значень у числових ознаках було застосовано метод заповнення середнім значенням стовпця. Це рішення дозволяє зберегти основні статистичні властивості ознак, знижуючи ризик викривлення даних та некоректних висновків моделі.

Стовпець "Timestamp" було видалено, оскільки він не несе суттєвої інформації про характер мережевого трафіку в контексті класифікації ботнетів. Часова мітка може бути корисною для аналізу часової динаміки атаки, але у даному разі ця ознака більше створювала шум, ніж приносила користь. Категоріальну ознаку "Protocol" перетворено у числові за допомогою методу one-hot encoding, що дозволяє моделям ефективно працювати з цими даними.

Аналіз розподілу класів показав значну незбалансованість: клас "BENIGN" значно переважає над класом ботнетів. Така незбалансованість може призвести до того, що моделі будуть схильні ігнорувати менш представлений клас, що негативно вплине на їхню здатність виявляти атаки. Для вирішення цієї проблеми застосовано

метод SMOTE (Synthetic Minority Over-sampling Technique), який синтетично збільшує кількість зразків менш представленого класу, створюючи нові зразки на основі найближчих сусідів у просторі ознак. Після застосування SMOTE кількість зразків обох класів було вирівняно, що сприяє більш ефективному навчанню моделей та покращує їхню здатність виявляти ботнети (рисунок 3.1):

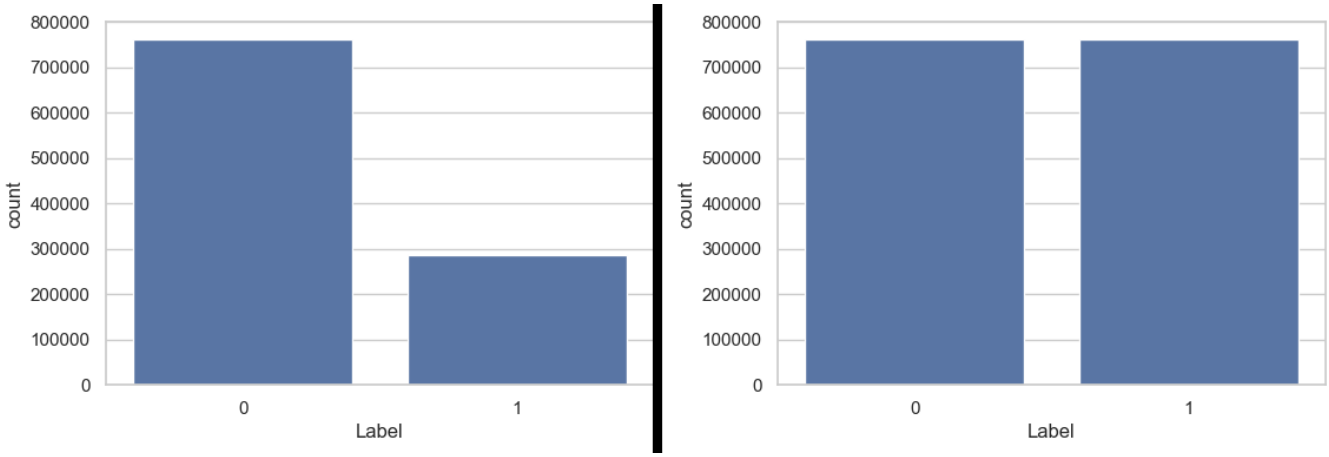


Рис. 3.1. Розподіл класів до та після застосування SMOTE

Після попередньої обробки та балансування даних, можна переходити до навчання моделей та їх оптимізація. На цьому етапі важливо не лише отримати коректну модель, але й забезпечити її високу точність, надійність та здатність до узагальнення. Першою моделлю, яку було побудовано, є Random Forest. Початкове тренування моделі з параметрами за замовчуванням дало дуже високі результати. Однак, з метою подальшого покращення продуктивності, було проведено гіперпараметричну оптимізацію.

Для підбору оптимальних гіперпараметрів Random Forest використано бібліотеку Hyperopt, яка реалізує Байєсову оптимізацію. Цей метод дозволяє ефективно досліджувати простір параметрів, автоматично обираючи наступну точку пошуку на основі попередніх результатів. Такий підхід є більш інтелектуальним та обчислювально ощадливим, ніж звичайний перебір або випадковий пошук, оскільки він швидше знаходить область з найкращими параметрами.

Оптимізовані гіперпараметри Random Forest:

n_estimators = 500: Кількість дерев у лісі. Більша кількість дерев може покращити продуктивність моделі, але збільшує час тренування;

max_depth = 25: Максимальна глибина кожного дерева. Обмеження глибини допомагає запобігти перенавчанню;

min_samples_split = 8: Мінімальна кількість зразків, необхідна для розділення внутрішнього вузла. Цей параметр контролює складність дерева;

min_samples_leaf = 1: Мінімальна кількість зразків, яка має бути у листовому вузлі. Допомагає уникнути створення вузлів з дуже малою кількістю зразків;

max_features = 'sqrt': Кількість ознак, яка розглядається при пошуку найкращого розділення. Вибір 'sqrt' означає, що використовуються квадратні корені з загальної кількості ознак.

Результати після оптимізації:

Час тренування: приблизно 1181 секунд;

Середній F1 Score з крос-валідацією: 0.999953.

Високе значення F1 Score свідчить про те, що модель добре збалансована між точністю та повнотою, ефективно виявляючи як позитивні, так і негативні випадки. Можна побачити, що матриця невідповідностей демонструє високу точність класифікації моделі Random Forest (рисунок 3.2):

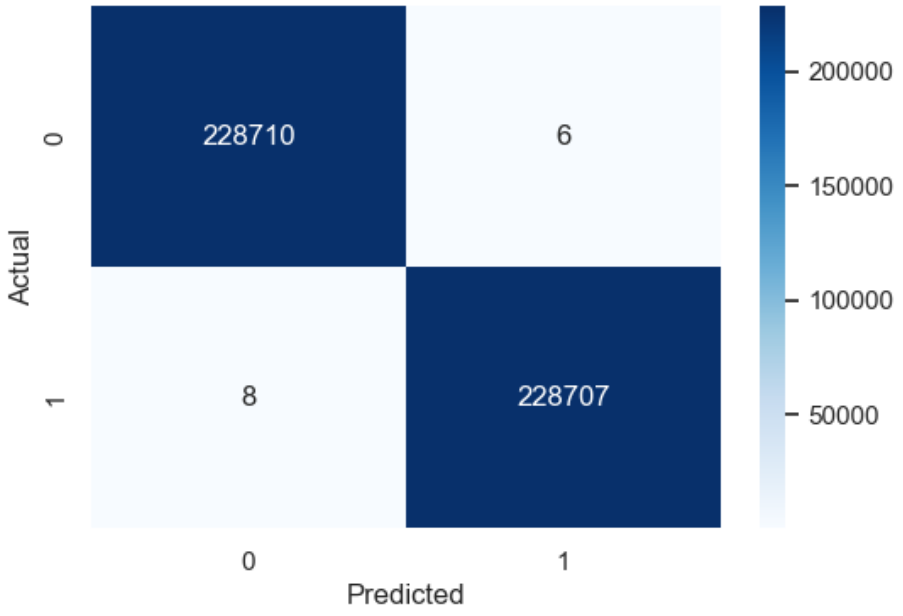


Рис. 3.2. Матриця невідповідностей для оптимізованої моделі Random Forest

Наступною моделлю, яку було побудовано, є XGBoost, яка відома своєю високою продуктивністю та швидкістю навчання на великих наборах даних. Початкове тренування з параметрами за замовчуванням показало хороші результати, але для досягнення ще кращої точності було проведено гіперпараметричну оптимізацію.

Оптимізовані гіперпараметри XGBoost:

n_estimators = 230: Кількість ітерацій бустингу. Визначає кількість дерев у моделі. Збільшення n_estimators дозволяє моделі детальніше опрацювати складні випадки, але збільшує час навчання. Оптимальний вибір забезпечує баланс між точністю та швидкістю;

max_depth = 13: Максимальна глибина дерева. Більша глибина дозволяє моделі захоплювати складніші патерни, але треба слідкувати, щоб модель не перенавчилася.;

gamma = 8.1319: Мінімальне зменшення втрат, необхідне для розділення вузла. Вищі значення роблять модель більш консервативною. Таким чином, gamma допомагає уникнути зайвих, незначних поділок, що можуть призвести до перенавчання;

reg_alpha = 40: Параметр L1-регуляризації, який штрафує ваги, зменшуючи переобладнання. Допомагає моделі фокусуватися на найбільш значущих ознаках. Велике значення регуляритизує модель, зменшуючи ваги ознак, які не вносять значного внеску у якість класифікації;

reg_lambda = 0.00626: Параметр L2-регуляризації, який також допомагає уникнути перенавчання;

colsample_bytree = 0.9937: Відсоток ознак, які вибираються випадково для кожного дерева. Значення близьке до 1 означає, що майже всі ознаки використовуються;

min_child_weight = 0: Мінімальна сума ваг усіх зразків у листовому вузлі. Менше значення робить модель чутливішою до окремих зразків. Впливає на складність моделі. Менше значення робить модель чутливішою до дрібних відмінностей між прикладами, що може бути корисним для виявлення рідкісних

або тонких патернів, характерних для ботнетів. Водночас занадто мала порогова вага може збільшити ризик перенавчання, але у поєднанні з іншими параметрами модель демонструє відмінну узагальнювальну здатність.

Результати після оптимізації:

Час тренування: 16 секунд;

Середній F1 Score з крос-валідацією: 0.999975.

Оптимізована модель XGBoost показала відмінні результати з дуже високим значенням F1 Score та значно меншим часом тренування порівняно з Random Forest. Така точність у поєднанні з коротким часом навчання робить XGBoost винятково привабливим варіантом для реальних корпоративних мереж, де даних багато, а загрози можуть швидко змінюватися. Як видно з матриці невідповідностей, модель XGBoost майже безпомилково класифікує зразки, що підтверджує її високу ефективність (рисунок 3.3):

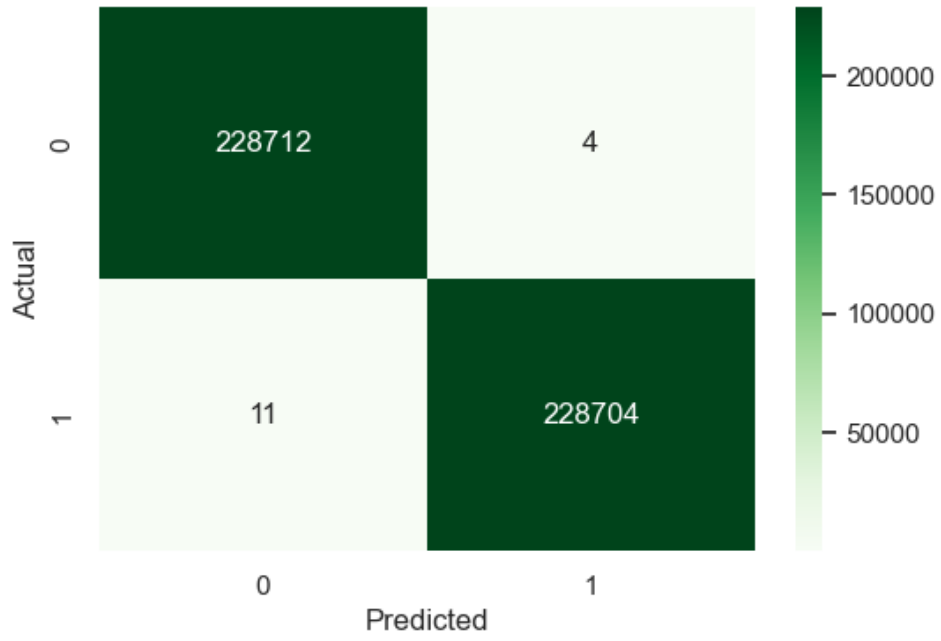


Рис. 3.3. Матриця невідповідностей для оптимізованої моделі XGBoost

Третьою моделлю є Support Vector Machine (SVM) з радіальним базисним ядром (RBF). Через великий розмір даних та складність обчислень SVM потребувала більше часу для тренування. Для оптимізації використано метод

Hyperopt з метою підбору найкращого значення параметра регуляризації C, який контролює баланс між максимізацією маржі та мінімізацією помилок класифікації.

Оптимізований параметр SVM:

C = 2.7604: Параметр регуляризації, який визначає ступінь покарання за помилки класифікації. Менші значення C роблять маржу ширшою, але можуть призвести до недообладнання, тоді як більші значення можуть спричинити перенавчання.

Результати після оптимізації:

Час тренування: 2393 секунд;

Середній F1 Score з крос-валідацією: 0.708193.

Хоча модель показала прийнятну точність, значення F1 Score значно нижче порівняно з Random Forest та XGBoost. Крім того, тривалий час тренування та високі обчислювальні витрати є вагомими недоліками у практичному застосуванні. Матриця невідповідностей на демонструє, що модель SVM має більшу кількість помилок класифікації, особливо у виявленні ботнет-атак (рисунок 3.4):

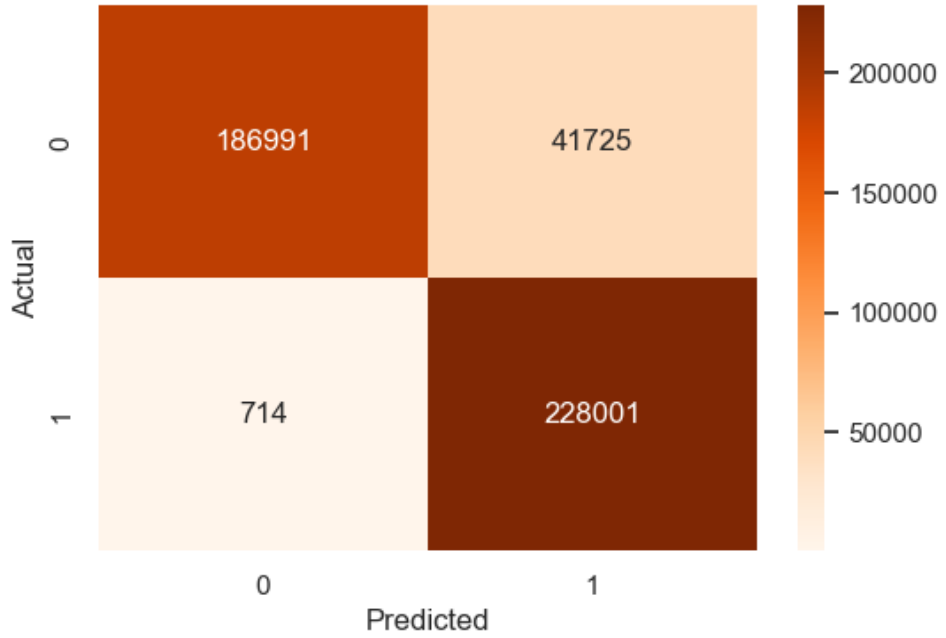


Рис. 3.4. Матриця невідповідностей для оптимізованої моделі SVM

Слід зауважити, що під час навчання SVM було використано параметр max_iter=1000 для обмеження кількості ітерацій оптимізаційного алгоритму. Це

обмеження було введено через те, що без нього модель потребувала б надто багато часу для збіжності на великому обсязі даних. Однак таке обмеження могло запобігти досягненню оптимального рішення, оскільки алгоритм не встигав повністю навчитися на даних. Це призвело до зниження точності SVM та пояснює, чому вона поступається моделям Random Forest та XGBoost за результатами. Хоча модель показала хорошу точність, проте час тренування та обчислювальні ресурси є значними, що є вагомим недоліком у практичному застосуванні.

Після тренування та оптимізації всіх моделей було проведено порівняння їхньої продуктивності за основними метриками: точність (Accuracy), точність передбачення (Precision), повнота (Recall), F1-міра без крос-валідації та час тренування (таблиця 3.1).

Таблиця 3.1.

Результати навчання моделей

Модель	**accuracy**	**precision**	**recall**	**F1 - міра**	**час тренування (с)**
Random Forest	0.9789	0.9896	0.9797	0.9828	238
Random Forest (оптимізована)	0.9999	0.9999	0.9998	0.9999	1180
XGBoost	0.9879	0.9889	0.9879	0.9856	15
XGBoost (оптимізована)	0.9999	0.9999	0.9999	0.9999	16
SVM	0.8795	0.8150	0.9819	0.8907	2414
SVM (оптимізована)	0.9072	0.8453	0.9968	0.9148	2393

З таблиці видно, що оптимізовані моделі Random Forest та XGBoost демонструють найвищі показники F1-міри та точності, досягаючи майже ідеальних значень. Це свідчить про здатність цих алгоритмів ефективно розрізняти ботнет-трафік від нормального, мінімізуючи як пропуски реальних загроз, так і хибні тривоги. Хоча Random Forest і XGBoost демонструють подібні результати за якістю, їх швидкість навчання суттєво різниться. Для тренування Random Forest

потрібно понад 1180 секунд, тоді як XGBoost досягає оптимальних налаштувань лише за 16 секунд. Така різниця має важливе практичне значення, особливо у сфері кібербезпеки, де оперативна адаптація моделей до нових загроз і даних є дуже важливою для ефективності.

Модель SVM, хоча і показує прийнятні результати, має значно нижчі метрики якості та дуже тривалий час тренування, що робить її менш придатною для практичного використання. З урахуванням масштабів даних, які потрібно обробляти, та необхідності оперативно оновлювати модель, вибір на користь SVM виявляється менш виправданим, особливо тоді, коли є ефективніші альтернативи з кращими часовими характеристиками.

Для розуміння того, які ознаки найбільше впливають на рішення моделей, було проведено аналіз важливості ознак для оптимізованих моделей Random Forest та XGBoost. Цей аналіз надає уявлення про те, які параметри найбільше сприяють точному розпізнаванню ботнетів, а також дозволяє оптимізувати подальші зусилля в області збору та обробки даних.

Random Forest визначила наступні ознаки як найбільш важливі (рисунок 3.5):

Dst Port: Номер порту призначення. Вказує на те, які сервіси використовуються і може свідчити про специфічну активність ботнету;

Flow Pkts/s: Кількість пакетів за секунду в потоці. Відображає інтенсивність трафіку;

Fwd Pkts/s: Кількість пакетів за секунду, відправлених у напрямку вперед. Вказує на активність вихідного трафіку;

Init Fwd Win Byts: Початковий розмір вікна TCP в байтах для напрямку вперед. Відображає параметри з'єднання та може вказувати на аномалії;

Flow IAT Mean: Середній інтервал між пакетами в потоці. Може виявляти регулярність або аномалії в трафіку;

Bwd Pkt Len Mean: Середня довжина пакетів у зворотному напрямку. Відображає характеристики вхідного трафіку;

Bwd Seg Size Avg: Середній розмір сегмента у зворотному напрямку. Показує особливості TCP-сегментації;

Fwd IAT Mean: Середній інтервал між пакетами у напрямку вперед. Може свідчити про патерни передачі даних;

Flow IAT Max: Максимальний інтервал між пакетами в потоці. Допомагає виявити затримки або переривання в передачі;

Flow Duration: Тривалість потоку. Довгі або короткі тривалості можуть бути характерними для ботнет-трафіку.

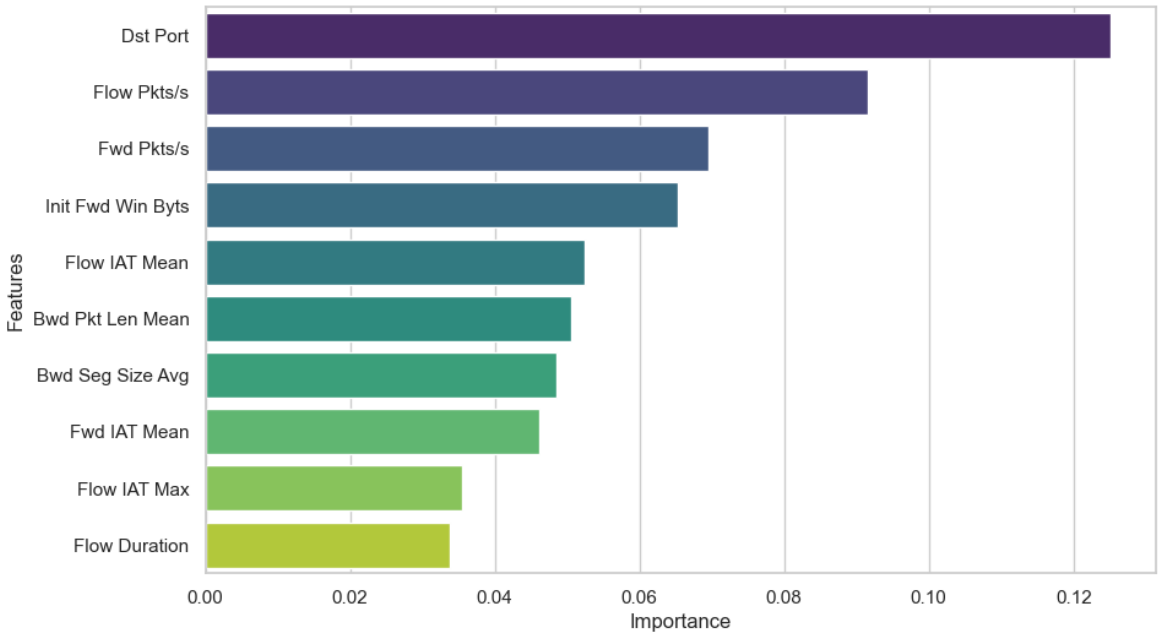


Рис. 3.5. Топ-10 важливих ознак за Random Forest

XGBoost визначила наступні ознаки як найбільш важливі (рисунок 3.6):

Dst Port: Аналогічно Random Forest, номер порту призначення є найбільш важливим показником. Ця ознака виявилась найбільш значущою для цієї моделі;

Bwd Pkt Len Mean: Середня довжина пакетів у зворотному напрямку. Важлива для характеристики вхідного трафіку;

RST Flag Cnt: Кількість встановлених прапорців RST у потоці. Вказує на скидання з'єднання та може бути ознакою аномальної активності;

Init Fwd Win Byts: Початковий розмір вікна TCP у напрямку вперед;

Fwd IAT Mean: Середній інтервал між пакетами у напрямку вперед;

Fwd IAT Std: Стандартне відхилення інтервалу між пакетами у напрямку вперед. Показує варіабельність в передачі пакетів.

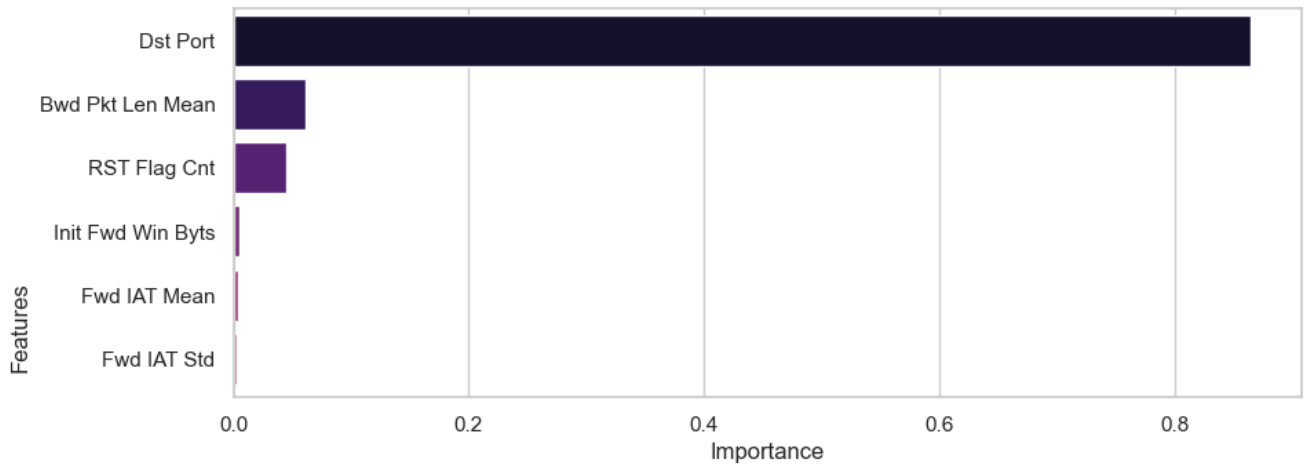


Рис. 3.6. Топ-6 важливих ознак за XGBoost

Ці результати вказують на те, що певні характеристики мережевого трафіку, такі як порти призначення, швидкість передачі пакетів та параметри TCP-з'єднань, є найбільш важливими показниками при виявленні ботнет-активності. Це узгоджується з очікуваннями, оскільки ботнети можуть генерувати аномальну кількість трафіку або мати специфічні патерни в поведінці мережевих потоків.

Оптимізовані моделі було збережено для подальшого використання у виробничому середовищі. Збереження моделей здійснено за допомогою бібліотеки joblib, що дозволяє швидко завантажувати моделі без повторного тренування. Це важливо для впровадження моделі в реальних системах моніторингу мережі, де необхідна оперативність та ефективність.

3.2. Аналіз та інтерпретація отриманих результатів

Базуючись на отриманих результатах, можна впевнено стверджувати, що методи машинного навчання є надзвичайно ефективними для виявлення ботнетів у мережевому трафіку. Оптимізовані моделі Random Forest та XGBoost продемонстрували високу точність, повноту та F1-міру, перевищуючи 99%. Це підтверджує високу здатність цих алгоритмів точно розпізнавати ботнет-активність, що робить їх актуальними у задачах захисту від мережевих загроз.

Особливої уваги заслуговує модель XGBoost, яка при оптимізованих параметрах досягла F1-міри 0.999975 з часом тренування лише 16 секунд. Для порівняння, Random Forest потребувала понад 1180 секунд. Така різниця у часі робить XGBoost привабливішою для практичного використання, де швидкість обробки даних є критично важливою. Висока продуктивність XGBoost у поєднанні зі швидкістю роботи робить її ідеальним вибором для систем моніторингу в режимі реального часу.

Одним із найважливіших аспектів успішного виявлення ботнетів є визначення найбільш значущих ознак мережевого трафіку, які сприяють точній класифікації. Аналіз важливості ознак, проведений для оптимізованих моделей Random Forest та XGBoost, виявив кілька дуже важливих важливих характеристик:

Dst Port — номер порту призначення. Random Forest визначила важливість цієї ознаки як 0.124, а XGBoost — 0.86. Ця ознака визначилась як найбільш важлива для обох моделей. Це пов'язано з тим, що ботнети часто використовують специфічні порти для комунікації зі своїми контролюючими серверами. Аналізуючи номер порту призначення, можена вити незвичайні закономірності, характерні для трафіку ботнету. Висока значимість цієї ознаки підкреслює її ключову роль у виявленні шкідливої активності;

Flow Pkts/s — кількість пакетів за секунду. Random Forest оцінила важливість ознаки як 0.092. Ця ознака каже про те, що інтенсивність трафіку є важливим показником для виявлення ботнетів, адже аномальна швидкість передачі пакетів може свідчити про підозрілу активність, таку як масові розсилання спаму або DDoS-атаки. Висока важливість цієї ознаки свідчить про її здатність виявляти підозрілі зміни в мережевому трафіку;

Flow IAT Mean — середній інтервал між пакетами). Ця ознака дозволяє аналізувати регулярність передачі пакетів у потоці. Незвичайні інтервали можуть свідчити про автоматизовану діяльність ботнетів, які намагаються обійти виявлення шляхом рандомізації інтервалів передачі даних. Виявлення таких патернів допомагає у ранньому виявленні ботнет-активності.

Порівняння оптимізованих моделей Random Forest та XGBoost показало, що обидва алгоритми здатні досягати високих показників точності та F1-міри, що перевищують 99%. Проте, XGBoost демонструє значні переваги у швидкості тренування, що робить його більш придатним для використання в системах, де важлива оперативність обробки даних. На відміну від цього, Random Forest має суттєво більший час тренування, що може бути недоліком у сценаріях, де необхідна швидка адаптація до нових даних.

Модель Support Vector Machine (SVM), хоча і показала прийнятну точність, мала значно нижчі показники F1 Score та вимагала значно більше часу на тренування порівняно з Random Forest та XGBoost. Це робить SVM менш ефективною для практичного застосування у великих корпоративних мережах, де швидкість та масштабованість є вирішальними факторами.

Отримані результати підтверджують доцільність використання методів машинного навчання, зокрема Random Forest та XGBoost, для ефективного виявлення ботнетів у мережевому трафіку. Висока точність та F1-міра цих моделей забезпечують надійний захист корпоративних мереж від складних кіберзагроз. Зокрема, XGBoost виділяється як найефективніший метод завдяки своїй здатності швидко обробляти дані та забезпечувати високу точність класифікації, що є найбільш важливим для систем моніторингу в режимі реального часу.

3.3. Розробка рекомендацій щодо покращення методів виявлення ботнетів

На основі проведеного дослідження та отриманих результатів, можна сформулювати низку рекомендацій, спрямованих на подальше покращення методів виявлення ботнетів у корпоративних мережах з використанням машинного навчання. Ці рекомендації охоплюють як технічні аспекти моделювання, так і стратегічні підходи до забезпечення безпеки мережі (рисунок 3.7).

Рис. 3.7. Структура рекомендацій для підвищення ефективності виявлення ботнетів у корпоративних мережах

Однією з ключових рекомендацій є активне використання алгоритму XGBoost у системах виявлення ботнетів завдяки його високій точності та мінімальному часу тренування. Для ефективної інтеграції XGBoost у корпоративні мережі необхідні:

Інтеграція в реальні системи моніторингу. Забезпечення безперервного збору та обробки даних у реальному часі сприятиме своєчасному виявленню аномалій та швидкому реагуванню на потенційні загрози;

Використання розподілених обчислювальних платформ. Це дасть змогу обробляти великі обсяги даних швидше та масштабувати систему відповідно до зростаючих потреб мережі.

Аналіз значущості ознак мережевого трафіку свідчить, що певні характеристики мережевого трафіку, такі як Dst Port, Flow Pkts/s та Flow IAT Mean, мають вагоме значення для точності класифікації ботнетів. Для підвищення ефективності моделей машинного навчання, рекомендується зосередитися на відборі та моніторингу ключових ознак. Це дозволить зменшити розмірність даних, що сприяє швидшій обробці та зменшенню ризику перенавчання моделей. Розробка спеціалізованих фільтрів для збору та аналізу цих ознак забезпечить більш точний аналіз трафіку та ефективніше виявлятиме аномалії, характерні для

ботнет-активності. Крім того, постійне оновлення списку важливих ознак на основі нових даних та змін у поведінці ботнетів дозволить моделям залишатися актуальними та точними.

Оскільки ботнети постійно змінюються, системи виявлення повинні залишатися адаптивними. Для цього важливо впровадити автоматизований процес регулярного оновлення моделей машинного навчання на нових даних, що відображають останні тенденції в поведінці ботнетів. Забезпечення безперервного моніторингу та аналізу змін у мережевому трафіку дозволить своєчасно виявляти нові види атак та оперативно оновлювати моделі для їх виявлення. Зберігання історичних даних для аналізу довгострокових тенденцій сприятиме ідентифікації закономірностей та передбаченню можливих напрямків розвитку ботнетів.

Щоб підвищити стабільність і тоність систем виявлення ботнетів, рекомендується використовувати ансамблеві методи, які поєднують декілька алгоритмів машинного навчання. Поєднання Random Forest та XGBoost дозволить скористатися їхніми сильними сторонами, забезпечуючи високу точність та стабільність класифікації. Використання методів бэггінгу та бустінгу зменшує варіативність моделей та покращує їхню загальну продуктивність, що особливо важливо для задач з високою складністю та розмірністю даних.

Якісна попередня обробка даних є дуже важливою для успішного навчання моделей машинного навчання. Рекомендації щодо покращення цього процесу можуть включати розширений відбір ознак та інженерію ознак. Вони дозволять створити нові, більш інформативні характеристики мережевого трафіку, що покращить здатність моделей розпізнавати складні патерни та аномалії. Використання методів зниження вимірності, таких як Principal Component Analysis (PCA) або t-SNE, допоможе зменшити розмірність даних без втрати ключової інформації, сприяючи швидшій та ефективнішій обробці. Покращення балансування даних за допомогою комбінованих методів oversampling та undersampling дозволить досягти оптимального балансу між класами, підвищуючи здатність моделей до класифікації менш представлених класів.

Методи глибокого навчання, такі як рекурентні нейронні мережі (RNN) та згорткові нейронні мережі (CNN), можуть значно покращити здатність систем виявляти складні та приховані патерни у мережевому трафіку [26]. У нашому контексті можна розглягути для впровадження такі рекомендації:

Розробка моделей глибокого навчання для аналізу тимчасових та просторових патернів. Використання RNN та CNN дозволить моделювати складні взаємозв'язки в мережевих потоках, що сприятиме точнішому виявленню ботнет-активності;

Використання автоенкодерів для виявлення аномалій та редукції вимірності. Автоенкодери можуть ефективно виявляти аномалії в даних, що є корисним для раннього виявлення ботнетів;

Інтеграція глибоких нейронних мереж з традиційними методами машинного навчання. Створення гібридних моделей дозволить поєднати переваги глибокого навчання та традиційних алгоритмів, забезпечуючи високу точність та гнучкість системи виявлення.

У корпоративних мережах вимоги до обсягу трафіку та продуктивності системи виявлення можуть відрізнятися. Тому важливо забезпечити масштабованість та гнучкість розроблених методів.

Використання розподілених обчислювальних платформ дозволить обробляти великі обсяги даних у режимі реального часу, забезпечуючи швидку реакцію на потенційні загрози. Оптимізація алгоритмів машинного навчання для роботи в середовищах з обмеженими ресурсами забезпечить високу точність класифікації навіть у умовах обмеженого апаратного забезпечення. Розробка модульної архітектури системи дозволить легко додавати нові компоненти та адаптувати систему до змінних умов мережі, підтримуючи її актуальність та ефективність у довгостроковій перспективі.

Можна підсумувати, що застосування алгоритму XGBoost у поєднанні з Random Forest, фокус на ключових ознаках мережевого трафіку, регулярне оновлення моделей, інтеграція ансамблевих методів та впровадження глибокого навчання дозволять створити ефективні та адаптивні системи виявлення ботнетів.

ВИСНОВКИ

У роботі проведено дослідження проблеми виявлення ботнетів у корпоративних мережах з використанням методів машинного навчання. Визначено мету дослідження та поставлено завдання, спрямовані на розробку ефективної технології виявлення ботнет-активності, аналіз існуючих підходів, а також оптимізацію та оцінку моделей машинного навчання для цієї задачі.

Проведено аналіз сучасних технологій виявлення ботнетів. Було розглянуто традиційні методи, такі як сигнатурний та аномалійний аналіз, а також сучасні підходи, засновані на машинному навчанні та глибокому навчанні. Виявлено, що методи машинного навчання, зокрема Random Forest та XGBoost, демонструють високу ефективність у виявленні ботнет-активності завдяки їхній здатності аналізувати великі обсяги даних та виявляти складні патерни поведінки.

Розроблено та реалізовано технологію виявлення ботнетів з використанням машинного навчання. Було здійснено збір та попередню обробку даних з датасету CSE-CIC-IDS2018, включаючи видалення нерелевантних ознак, заповнення пропущених значень та перетворення категоріальних змінних за допомогою one-hot encoding. Для вирішення проблеми незбалансованості класів застосовано метод SMOTE, що дозволило збалансувати розподіл класів та покращити навчання моделей. Навчання та оптимізація моделей Random Forest, XGBoost та Support Vector Machine (SVM) були проведені з використанням байєсової оптимізації через бібліотеку Hyperopt, що значно покращило їхню продуктивність.

Оцінено ефективність моделей за допомогою ключових метрик та крос-валідації. Оптимізовані моделі Random Forest та XGBoost досягли F1-міри понад 99%, підтверджуючи їхню високу здатність точно класифікувати ботнет-активність. Особливою перевагою є модель XGBoost, яка забезпечила високу точність при значно коротшому часі тренування порівняно з Random Forest. Модель SVM, хоча й показала прийнятну точність, мала значно нижчі показники

F1-міри та вимагала більше часу на тренування, що робить її менш придатною для практичного застосування у великих корпоративних мережах.

Визначено ключові ознаки мережевого трафіку, що впливають на виявлення ботнетів. Аналіз важливості ознак виявив, що такі характеристики, як номер порту призначення (Dst Port), кількість пакетів за секунду (Flow Pkts/s) та середній інтервал між пакетами (Flow IAT Mean), мають найбільший вплив на класифікацію ботнет-активності. Це дозволяє зосередити увагу на моніторингу цих ознак для покращення точності систем виявлення та оптимізації процесу аналізу.

Сформульовано рекомендації щодо подальшого покращення методів виявлення ботнетів. Для підвищення ефективності систем виявлення рекомендується активне використання алгоритму XGBoost завдяки його високій точності та мінімальному часу тренування, фокусування на ключових ознаках мережевого трафіку, регулярне оновлення та перенавчання моделей, а також впровадження ансамблевих методів і методів глибокого навчання для покращення здатності систем до виявлення аномалій.

Таким чином, проведене дослідження підтвердило високу ефективність застосування методів машинного навчання, зокрема Random Forest та XGBoost, для виявлення ботнетів у корпоративних мережах. Отримані результати свідчать про можливість створення надійних та швидких систем захисту від складних кіберзагроз, що значно підвищує рівень кібербезпеки організацій. Розроблені рекомендації спрямовані на подальше вдосконалення методів виявлення ботнетів, забезпечуючи їхню актуальність та ефективність у динамічному кіберпросторі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Zhang, X., Upton, O., Beebe, N.L., & Choo, K.K.R. (2020). IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers. *Forensic Science International: Digital Investigation*, 32, 300926. URL: https://doi.org/10.1016/j.fsidi.2020.300926

2. Cloudflare. (2024). DDoS Attack Trends for Q1 2024. URL: https://www.cloudflare.com/ddos/ (дата звернення: 06.11.2024).

3. Communications Security Establishment (CSE) & Canadian Institute for Cybersecurity (CIC). (2018). CSE-CIC-IDS2018 on AWS: A collaborative project. URL: https://www.unb.ca/cic/datasets/ids-2018.html (дата звернення: 10.11.2024).

4. Hussain, F., Abbas, S. G., Husnain, M., Fayyaz, U. U., Shahzad, F., & Shah, G. A. (2020). IoT DoS and DDoS Attack Detection using ResNet. 2020 IEEE 23rd International Multitopic Conference (INMIC), 1–6. URL: https://doi.org/10.1109/INMIC50486.2020.9318216

5. Kolodchak, O.M. (2012). Сучасні методи виявлення аномалій в системах виявлення вторгнень. *Науковий вісник Національного університету "Львівська політехніка"*, № 681, 98–104. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/nov/6726/16-98-104.pdf

6. Ogu, E.C., Vrakas, N., Chiemela, O., & Ajose-Ismail, B.M. (2016). On the Internal Workings of Botnets: A Review. International Journal of Computer Applications, 138(4), 39–43. URL: http://dx.doi.org/10.5120/ijca2016908797

7. Livadas, C., Walsh, R., Lapsley, D., & Strayer, W.T. (2006). Using Machine Learning Techniques to Identify Botnet Traffic. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, Tampa, FL, USA, 14–16 November 2006. URL: https://people.csail.mit.edu/clivadas/pubs/LivadasWLS06.pdf

8. Rokach, L., & Maimon, O. (2007). *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Company. URL: https://doi.org/10.1142/6604

9. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. URL: https://doi.org/10.1145/2939672.2939785

10. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. URL: https://doi.org/10.1007/BF00994018

11. Yen, T.-F., & Reiter, M.K. (2010). Are Your Hosts Trading or Plotting? Telling P2P File-Sharing and Bots Apart. In *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems* (pp. 241–252). URL: https://doi.org/10.1109/ICDCS.2010.76

12. Kaushik, S. (2016). *An Introduction to Clustering and Different Methods of Clustering*. URL: https://www.scribd.com/document/517205288/an-introduction-to-clustering-and-different-methods-of-clustering-1

13. Fitni, Q.R.S., & Ramli, K. (2020). Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems. In *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)* (pp. 118–124). IEEE. URL: http://dx.doi.org/10.1109/IAICT50021.2020.9172014

14. Hua, Y. (2020). An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM. In *2020 Information Communication Technologies Conference (ICTC)*, IEEE (pp. 125–130). URL: https://doi.org/10.1109/ICTC49638.2020.9123302

15. Li, X., Chen, W., Zhang, Q., & Wu, L. (2020). Building Auto-Encoder Intrusion Detection System Based on Random Forest Feature Selection. *Computers & Security*, 95, 101851. URL: http://dx.doi.org/10.1016/j.cose.2020.101851

16. Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Proceedings of the 17th USENIX Security Symposium*. URL: https://www.usenix.org/legacy/event/sec08/tech/full_papers/gu/gu.pdf

17. Tegeler, F., Fu, X., Vigna, G., & Kruegel, C. (2012). BotFinder: Finding Bots in Network Traffic Without Deep Packet Inspection. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies* (pp. 349–360). URL: https://dl.acm.org/doi/10.1145/2413176.2413217

18. Zhang, J., Perdisci, R., Lee, W., Luo, X., & Sarfraz, U. (2014). Building a Scalable System for Stealthy P2P-Botnet Detection. *IEEE Transactions on Information Forensics and Security*, 9(1), 27–38. URL: https://doi.org/10.1109/TIFS.2013.2290197

19. Barbado, A., Corcho, O., & Benjamins, R. (2022). Rule Extraction in Unsupervised Anomaly Detection for Model Explainability: Application to One-Class SVM. *Expert Systems with Applications*, 189, 116100. URL: http://dx.doi.org/10.48550/arXiv.1911.09315

20. Hao, J., Luo, S., & Pan, L. (2022). Rule Extraction from Biased Random Forest and Fuzzy Support Vector Machine for Early Diagnosis of Diabetes. *Scientific Reports*, 12, 9858. URL: https://doi.org/10.1038/s41598-022-14143-8

21. Kašćelan, L., Kašćelan, V., & Jovanović, M. (2014). Hybrid Support Vector Machine Rule Extraction Method for Discovering the Preferences of Stock Market Investors: Evidence from Montenegro. *Intelligent Automation & Soft Computing*, 21(4), 503–522. URL: https://www.researchgate.net/publication/277976964_Hybrid_support_vector_machine_rule_extraction_method_for_discovering_the_preferences_of_stock_market_investors_Evidence_from_Montenegro

22. Stevanovic, M., & Pedersen, J.M. (2015). An Analysis of Network Traffic Classification for Botnet Detection. *Computer Networks*, 55(5), 131–145. URL: https://doi.org/10.1109/CyberSA.2015.7361120

23. Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer. URL: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/

24. Alieyan, K., Almomani, A., & Manasrah, A. (2020). Robust Early Stage Botnet Detection using Machine Learning. In Proceedings of the 16th International

Conference on Cyber Warfare and Security (ICCWS) (pp. 1–9). IEEE. URL: http://dx.doi.org/10.1109/ICCWS48432.2020.9292395

25. Wang, P., Aslam, B., & Zou, C.C. (2010). Peer-to-Peer Botnets. In *Handbook of Information and Communication Security*, Springer (pp. 335–350). URL: https://doi.org/10.1007/978-3-642-04117-4_18

26. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, 7, 41525–41550. URL: https://doi.org/10.1109/ACCESS.2019.2895334

27. Wang, W., Chakraborty, G., & Chakraborty, B. (2021). Predicting the Risk of Chronic Kidney Disease (CKD) Using Machine Learning Algorithm. *Applied Sciences*, 11(1), 202. URL: http://dx.doi.org/10.3390/app11010202

28. Scikit-learn. (2024). Plot classification boundaries with different SVM Kernels. URL: https://scikit-learn.org/1.5/auto_examples/svm/plot_svm_kernels.html (дата звернення: 01.12.2024).

29. Omara, H., Lazaar, M., & Tabii, Y. (2018). Effect of Feature Selection on Gene Expression Datasets Classification Accuracy. *International Journal of Electrical and Computer Engineering (IJECE)*, 8(5), 3194–3203. URL: https://doi.org/10.11591/ijece.v8i5.pp3194-3203

30. Singh, R., Kumar, H., & Singla, R. (2013). Analysis of Feature Selection Techniques for Network Traffic Dataset. In *Proceedings of the 2013 International Conference on Machine Intelligence and Research Advancement*, IEEE (pp. 42–46). URL: https://doi.org/10.1109/ICMIRA.2013.15

31. Witten, I.H., Frank, E., & Hall, M.A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann. URL: https://www.elsevier.com/books/data-mining/witten/978-0-12-374856-0

32. He, H., & Garcia, E.A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), 1263–1284. URL: https://doi.org/10.1109/TKDE.2008.239

33. Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced Data Sets*. Springer. ISBN: 978-3-319-98073-7. URL: https://doi.org/10.1007/978-3-319-98074-4

34. Buda, M., Maki, A., & Mazurowski, M.A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, 249–259. URL: https://doi.org/10.1016/j.neunet.2018.07.011

35. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. URL: https://doi.org/10.1023/A:1010933404324

36. Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2015). XGBoost: extreme gradient boosting. *R package version 0.4-2*. URL: https://doi.org/10.32614/CRAN.package.xgboost

37. Hsu, C.W., Chang, C.C., & Lin, C.J. (2003). A practical guide to support vector classification. *Technical Report, Department of Computer Science, National Taiwan University*. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

38. Bergstra, J., Yamins, D., & Cox, D.D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning* (pp. 115–123). URL: https://proceedings.mlr.press/v28/bergstra13.html

39. Powers, D.M.W. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1), 37–63. URL: http://dx.doi.org/10.9735/2229-3981

40. Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2), 111–133. URL: https://www.jstor.org/stable/2984809

41. Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5), 1189–1232. URL: https://doi.org/10.1214/aos/1013203451

ДОДАТОК А

Код проекту

```
# Імпорт необхідних бібліотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    roc_curve,
    auc,
    roc_auc_score,
    matthews_corrcoef,
    precision_recall_curve,
    average_precision_score,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score
)
from xgboost import XGBClassifier
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
import joblib
import warnings
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
import time
import os

# Налаштування для візуалізації
%matplotlib inline
sns.set(style='whitegrid')

# Ігнорування попереджень
warnings.filterwarnings('ignore')

# Завантаження датасету
df = pd.read_csv('Friday-02-03-2018_TrafficForML_CICFlowMeter.csv')

# Огляд датасету
```

```
df.info()
df.head()

# Перевірка кількості пропущених значень у кожному стовпці
missing_values = df.isnull().sum()
print("Пропущені значення по стовпцях:")
print(missing_values[missing_values > 0])

# Вибір тільки числових стовпців
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

# Заповнення пропущених значень середнім для числових стовпців
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())

# Перевірка після заповнення
print(df.isnull().sum().max())
# Приклад видалення стовпців, які не використовуються
columns_to_drop = ['Timestamp']
df.drop(columns=columns_to_drop, axis=1, inplace=True, errors='ignore')
df.info()

# Перетворення категоріальних ознак у числові
df = pd.get_dummies(df, columns=['Protocol'], drop_first=True)

# Визначення ознак та міток
X = df.drop('Label', axis=1)
y = df['Label'].apply(lambda x: 0 if x.strip().upper() == 'BENIGN' else 1)  # 0 -
Benign, 1 - Botnet

# Візуалізація кількості кожного класу в Label
sns.countplot(x='Label', data=df)
plt.title("Кількість кожного класу у датасеті")
plt.show()

# Кореляційна Матриця
corr_matrix = X.corr()
plt.figure(figsize=(20, 20))
sns.heatmap(corr_matrix, cmap='coolwarm', linewidths=0.5)
plt.title('Кореляційна Матриця Числових Ознак')
plt.show()

# Вибір важливих ознак для візуалізації
important_features = ['Flow Byts/s', 'Flow Pkts/s', 'Tot Fwd Pkts', 'Tot Bwd Pkts',
'Flow Duration']

for feature in important_features:
    plt.figure(figsize=(8,4))
    sns.boxplot(x='Label', y=feature, data=df)
    plt.title(f'Розподіл {feature} за Класами')
    plt.show()
```

```
# Встановлюємо опцію для відображення всіх рядків
pd.set_option('display.max_rows', None)

# Перевірка на нескінченні значення
infinite_values = np.isinf(X).sum()
print(f"Кількість нескінченних значень у кожному стовпці:\n{infinite_values}")

# Нескінченні значення замінюємо на NaN
X.replace([np.inf, -np.inf], np.nan, inplace=True)

# Заповнення пропущених значень середнім
X.fillna(X.mean(), inplace=True)

# Балансування класів за допомогою SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Перевірка після балансування
print(pd.Series(y_resampled).value_counts())

# Поділ на тренувальний і тестовий набори даних
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
test_size=0.3, random_state=42, stratify=y_resampled)

# Масштабування числових ознак
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Ініціалізація та тренування моделей
# Ініціалізація моделі Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Вимірювання часу тренування
start_time_rf = time.time()
rf_model.fit(X_train, y_train)
end_time_rf = time.time()
print(f"Час тренування Random Forest: {end_time_rf - start_time_rf:.2f} секунд")

# Передбачення
y_pred_rf = rf_model.predict(X_test)
y_pred_rf_prob = rf_model.predict_proba(X_test)[:,1]

# Оцінка результатів
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))

# Побудова Confusion Matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(6,4))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('Confusion Matrix - Random Forest')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
cv_scores_rf = cross_val_score(rf_model, X_train, y_train, cv=5, scoring='f1')
print(f"F1 Scores для Random Forest з крос-валідацією: {cv_scores_rf}")
print(f"Середній F1 Score: {cv_scores_rf.mean():.6f}")

# Гіперпараметрична Оптимізація для Random Forest з використанням Hyperopt
def rf_objective(space):
    clf = RandomForestClassifier(
        n_estimators=int(space['n_estimators']),
        max_depth=int(space['max_depth']),
        min_samples_split=int(space['min_samples_split']),
        min_samples_leaf=int(space['min_samples_leaf']),
        max_features=space['max_features'],
        random_state=42
    )

    # Вимірювання часу тренування
    start_time = time.time()
    clf.fit(X_train, y_train)
    end_time = time.time()
    training_time = end_time - start_time
    print(f"SCORE: {clf.score(X_test, y_test)}")
    print(f"Час тренування: {training_time:.2f} секунд")

    return {'loss': -clf.score(X_test, y_test), 'status': STATUS_OK }

# Визначення простору пошуку для Random Forest
space_rf = {
    'max_depth': hp.quniform('max_depth', 5, 50, 1),
    'n_estimators': hp.quniform('n_estimators', 100, 500, 10),
    'min_samples_split': hp.quniform('min_samples_split', 2, 10, 1),
    'min_samples_leaf': hp.quniform('min_samples_leaf', 1, 4, 1),
    'max_features': hp.choice('max_features', ['sqrt', 'log2'])
}

# Ініціалізація Trials для Random Forest
trials_rf = Trials()

# Проведення оптимізації для Random Forest з виправленим генератором випадкових чисел
best_hyperparams_rf = fmin(
    fn=rf_objective,
    space=space_rf,
    algo=tpe.suggest,
    max_evals=30,
    trials=trials_rf,
    rstate=np.random.default_rng(42)
```

```
)

print("\nНайкращі параметри для Random Forest:")
print(best_hyperparams_rf)

# Перетворення знайдених параметрів на правильні типи
best_hyperparams_rf['max_depth'] = int(best_hyperparams_rf['max_depth'])
best_hyperparams_rf['n_estimators'] = int(best_hyperparams_rf['n_estimators'])
best_hyperparams_rf['min_samples_split'] =
int(best_hyperparams_rf['min_samples_split'])
best_hyperparams_rf['min_samples_leaf'] =
int(best_hyperparams_rf['min_samples_leaf'])
best_hyperparams_rf['max_features'] = ['sqrt',
'log2'][best_hyperparams_rf['max_features']]

# Побудова оптимізованої моделі Random Forest
best_rf = RandomForestClassifier(
    n_estimators=best_hyperparams_rf['n_estimators'],
    max_depth=best_hyperparams_rf['max_depth'],
    min_samples_split=best_hyperparams_rf['min_samples_split'],
    min_samples_leaf=best_hyperparams_rf['min_samples_leaf'],
    max_features=best_hyperparams_rf['max_features'],
    random_state=42
)

# Вимірювання часу тренування оптимізованої Random Forest
start_time_best_rf = time.time()
best_rf.fit(X_train, y_train)
end_time_best_rf = time.time()
training_time_best_rf = end_time_best_rf - start_time_best_rf
print(f"Час тренування оптимізованої Random Forest: {training_time_best_rf:.2f}
секунд")

# Передбачення та оцінка оптимізованої Random Forest
y_pred_best_rf = best_rf.predict(X_test)
y_pred_best_rf_prob = best_rf.predict_proba(X_test)[:,1]
print("Random Forest (Optimized) Classification Report:")
print(classification_report(y_test, y_pred_best_rf))

#Крос-валідація для Найкращої Моделі Random Forest
cv_scores_rf = cross_val_score(best_rf, X_resampled, y_resampled, cv=5, scoring='f1')
print(f"\nF1 Scores для Random Forest з крос-валідацією: {cv_scores_rf}")
print(f"Середній F1 Score: {cv_scores_rf.mean():.6f}")

#Аналіз Важливості Ознак для Оптимізованої Random Forest
feature_importances_rf = pd.Series(best_rf.feature_importances_,
index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=feature_importances_rf.values[:10],
y=feature_importances_rf.index[:10], palette='viridis')
```

```
plt.title('Top 10 Feature Importances - Random Forest (Optimized)')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.show()

# Ініціалізація моделі XGBoost
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
random_state=42)

# Вимірювання часу тренування
start_time_xgb = time.time()
xgb_model.fit(X_train, y_train)
end_time_xgb = time.time()
print(f"Час тренування XGBoost: {end_time_xgb - start_time_xgb:.2f} секунд")

# Передбачення
y_pred_xgb = xgb_model.predict(X_test)
y_pred_xgb_prob = xgb_model.predict_proba(X_test)[:,1]

# Оцінка результатів
print("XGBoost Classification Report:")
print(classification_report(y_test, y_pred_xgb))

# Побудова Confusion Matrix
cm_xgb = confusion_matrix(y_test, y_pred_xgb)
plt.figure(figsize=(6,4))
sns.heatmap(cm_xgb, annot=True, fmt='d', cmap='Greens')
plt.title('Confusion Matrix - XGBoost')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

cv_scores_xgb = cross_val_score(xgb_model, X_resampled, y_resampled, cv=5,
scoring='f1')
print(f"\nF1 Scores для XGBoost з крос-валідацією: {cv_scores_xgb}")
print(f"Середній F1 Score: {cv_scores_xgb.mean():.6f}")

#Гіперпараметрична Оптимізація для XGBoost з використанням Hyperopt
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe

def xgb_objective(space):
    clf = XGBClassifier(
        n_estimators=int(space['n_estimators']),
        max_depth=int(space['max_depth']),
        gamma=space['gamma'],
        reg_alpha=int(space['reg_alpha']),
        reg_lambda=space['reg_lambda'],
        colsample_bytree=space['colsample_bytree'],
        min_child_weight=int(space['min_child_weight']),
        use_label_encoder=False,
        eval_metric='logloss',
```

```
        random_state=42
    )

    clf.fit(X_train, y_train,
            eval_set=[(X_test, y_test)],
            early_stopping_rounds=10,
            verbose=False)

    pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, pred)
    print(f"SCORE: {accuracy}")
    return {'loss': -accuracy, 'status': STATUS_OK }

# Визначення простору пошуку
space_xgb = {
    'max_depth': hp.quniform('max_depth', 3, 18, 1),
    'gamma': hp.uniform('gamma', 1, 9),
    'reg_alpha': hp.quniform('reg_alpha', 40, 180, 1),
    'reg_lambda': hp.uniform('reg_lambda', 0, 1),
    'colsample_bytree': hp.uniform('colsample_bytree', 0.5, 1),
    'min_child_weight': hp.quniform('min_child_weight', 0, 10, 1),
    'n_estimators': hp.quniform('n_estimators', 100, 300, 10)
}

# Ініціалізація Trials
trials_xgb = Trials()

# Проведення оптимізації
best_hyperparams_xgb = fmin(
    fn=xgb_objective,
    space=space_xgb,
    algo=tpe.suggest,
    max_evals=30,
    trials=trials_xgb,
    rstate=np.random.default_rng(42)
)

print("\nНайкращі параметри для XGBoost:")
print(best_hyperparams_xgb)

# Перетворення знайдених параметрів на правильні типи
best_hyperparams_xgb['max_depth'] = int(best_hyperparams_xgb['max_depth'])
best_hyperparams_xgb['reg_alpha'] = int(best_hyperparams_xgb['reg_alpha'])
best_hyperparams_xgb['min_child_weight'] =
int(best_hyperparams_xgb['min_child_weight'])
best_hyperparams_xgb['n_estimators'] = int(best_hyperparams_xgb['n_estimators'])

# Побудова оптимізованої моделі XGBoost
best_xgb = XGBClassifier(
    n_estimators=best_hyperparams_xgb['n_estimators'],
    max_depth=best_hyperparams_xgb['max_depth'],
```

```
    gamma=best_hyperparams_xgb['gamma'],
    reg_alpha=best_hyperparams_xgb['reg_alpha'],
    reg_lambda=best_hyperparams_xgb['reg_lambda'],
    colsample_bytree=best_hyperparams_xgb['colsample_bytree'],
    min_child_weight=best_hyperparams_xgb['min_child_weight'],
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

# Вимірювання часу тренування оптимізованої XGBoost
start_time_best_xgb = time.time()
best_xgb.fit(X_train, y_train,
            eval_set=[(X_test, y_test)],
            early_stopping_rounds=10,
            verbose=False)
end_time_best_xgb = time.time()
training_time_best_xgb = end_time_best_xgb - start_time_best_xgb
print(f"Час тренування оптимізованої XGBoost: {training_time_best_xgb:.2f} секунд")

# Передбачення та оцінка оптимізованої XGBoost
y_pred_best_xgb = best_xgb.predict(X_test)
y_pred_best_xgb_prob = best_xgb.predict_proba(X_test)[:,1]
print("XGBoost (Optimized) Classification Report:")
print(classification_report(y_test, y_pred_best_xgb))

# Побудова Confusion Matrix для оптимізованої XGBoost
cm_best_xgb = confusion_matrix(y_test, y_pred_best_xgb)
plt.figure(figsize=(6,4))
sns.heatmap(cm_best_xgb, annot=True, fmt='d', cmap='Greens')
plt.title('Confusion Matrix - XGBoost (Optimized)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Додаткові Метрики для оптимізованої XGBoost
fpr_best_xgb, tpr_best_xgb, _ = roc_curve(y_test, y_pred_best_xgb_prob)
roc_auc_best_xgb = auc(fpr_best_xgb, tpr_best_xgb)
precision_best_xgb, recall_best_xgb, _ = precision_recall_curve(y_test,
y_pred_best_xgb_prob)
average_precision_best_xgb = average_precision_score(y_test, y_pred_best_xgb_prob)

# Побудова ROC Curve для оптимізованої XGBoost
plt.figure(figsize=(6,4))
plt.plot(fpr_best_xgb, tpr_best_xgb, label=f'XGBoost (AUC = {roc_auc_best_xgb:.2f})')
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - XGBoost (Optimized)')
plt.legend(loc='lower right')
plt.show()
```

```
# Крос-валідація для Найкращої Моделі XGBoost
cv_scores_xgb = cross_val_score(best_xgb, X_resampled, y_resampled, cv=5,
scoring='f1')
print(f"\nF1 Scores для XGBoost з крос-валідацією: {cv_scores_xgb}")
print(f"Середній F1 Score: {cv_scores_xgb.mean():.6f}")

# Аналіз Важливості Ознак для Оптимізованої XGBoost
feature_importances_xgb = pd.Series(best_xgb.feature_importances_,
index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=feature_importances_xgb.values[:10],
y=feature_importances_xgb.index[:10], palette='magma')
plt.title('Top 10 Feature Importances - XGBoost (Optimized)')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.show()

# Ініціалізація моделі SVM з лінійним ядром
svm_model = SVC(kernel='rbf', probability=True, random_state=42, max_iter=1000)

# Вимірювання часу тренування
start_time_svm = time.time()
svm_model.fit(X_train, y_train)
end_time_svm = time.time()
print(f"Час тренування SVM: {end_time_svm - start_time_svm:.2f} секунд")

# Передбачення
y_pred_svm = svm_model.predict(X_test)
y_pred_svm_prob = svm_model.predict_proba(X_test)[:,1]

# Оцінка результатів
print("SVM Classification Report:")
print(classification_report(y_test, y_pred_svm))

# Побудова Confusion Matrix
cm_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(6,4))
sns.heatmap(cm_svm, annot=True, fmt='d', cmap='Oranges')
plt.title('Confusion Matrix - SVM')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

cv_scores_svm = cross_val_score(svm_model, X_resampled, y_resampled, cv=5,
scoring='f1')
print(f"\nF1 Scores для SVM з крос-валідацією: {cv_scores_svm}")
print(f"Середній F1 Score: {cv_scores_svm.mean():.6f}")

# Гіперпараметрична Оптимізація для SVM з використанням Hyperopt
```

```
def svm_objective(space):
    clf = SVC(
        C=space['C'],
        kernel='rbf',
        probability=True,
        random_state=42,
        max_iter=1000
    )

    clf.fit(X_train, y_train)
    pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, pred)
    print(f"SCORE: {accuracy}")
    return {'loss': -accuracy, 'status': STATUS_OK }

# Визначення простору пошуку для SVM
space_svm = {
    'C': hp.loguniform('C', np.log(0.01), np.log(10))
}

# Ініціалізація Trials для SVM
trials_svm = Trials()

# Проведення оптимізації для SVM
best_hyperparams_svm = fmin(
    fn=svm_objective,
    space=space_svm,
    algo=tpe.suggest,
    max_evals=10,
    trials=trials_svm,
    rstate=np.random.default_rng(42)
)

print("\nНайкращі параметри для SVM:")
print(best_hyperparams_svm)

# Побудова оптимізованої моделі SVM
best_svm = SVC(
    C=best_hyperparams_svm['C'],
    kernel='rbf',  # Використовуємо RBF ядро
    probability=True,
    random_state=42,
    max_iter=1000
)

# Вимірювання часу тренування оптимізованої SVM
start_time_best_svm = time.time()
best_svm.fit(X_train, y_train)
end_time_best_svm = time.time()
training_time_best_svm = end_time_best_svm - start_time_best_svm
print(f"Час тренування оптимізованої SVM: {training_time_best_svm:.2f} секунд")
```

```
# Передбачення та оцінка оптимізованої SVM
y_pred_best_svm = best_svm.predict(X_test)
y_pred_best_svm_prob = best_svm.predict_proba(X_test)[:,1]
print("SVM (Optimized) Classification Report:")
print(classification_report(y_test, y_pred_best_svm))

# Побудова Confusion Matrix для оптимізованої SVM
cm_best_svm = confusion_matrix(y_test, y_pred_best_svm)
plt.figure(figsize=(6,4))
sns.heatmap(cm_best_svm, annot=True, fmt='d', cmap='Oranges')
plt.title('Confusion Matrix - SVM (Optimized)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Додаткові Метрики для оптимізованої SVM
fpr_best_svm, tpr_best_svm, _ = roc_curve(y_test, y_pred_best_svm_prob)
roc_auc_best_svm = auc(fpr_best_svm, tpr_best_svm)
precision_best_svm, recall_best_svm, _ = precision_recall_curve(y_test,
y_pred_best_svm_prob)
average_precision_best_svm = average_precision_score(y_test, y_pred_best_svm_prob)

# Побудова ROC Curve для оптимізованої SVM
plt.figure(figsize=(6,4))
plt.plot(fpr_best_svm, tpr_best_svm, label=f'SVM (AUC = {roc_auc_best_svm:.2f})')
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - SVM (Optimized)')
plt.legend(loc='lower right')
plt.show()

# Крос-валідація для Найкращої Моделі SVM
cv_scores_svm = cross_val_score(best_svm, X_resampled, y_resampled, cv=5,
scoring='f1')
print(f"\nF1 Scores для SVM з крос-валідацією: {cv_scores_svm}")
print(f"Середній F1 Score: {cv_scores_svm.mean():.6f}")

# Оцінка моделей
def collect_metrics(model_name, y_pred, training_time):
    return {
        'Model': model_name,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred, zero_division=0),
        'Recall': recall_score(y_test, y_pred, zero_division=0),
        'F1-score': f1_score(y_test, y_pred, zero_division=0),
        'Training Time (s)': training_time
    }

# Використання функції для кожної моделі
```

```
metrics = [
    collect_metrics('Random Forest (Default)', y_pred_rf, end_time_rf -
start_time_rf),
    collect_metrics('Random Forest (Optimized)', y_pred_best_rf,
training_time_best_rf),
    collect_metrics('XGBoost (Default)', y_pred_xgb, end_time_xgb - start_time_xgb),
    collect_metrics('XGBoost (Optimized)', y_pred_best_xgb, training_time_best_xgb),
    collect_metrics('SVM (Default)', y_pred_svm, end_time_svm - start_time_svm),
    collect_metrics('SVM (Optimized)', y_pred_best_svm, training_time_best_svm)
]

# Створення DataFrame
results = pd.DataFrame(metrics)
results

# Створення директорії для збереження моделей
models_dir = 'saved_models'
if not os.path.exists(models_dir):
    os.makedirs(models_dir)

# Список моделей для збереження
models_to_save = {
    'RandomForest_Default': rf_model,
    'RandomForest_Optimized': best_rf,
    'XGBoost_Default': xgb_model,
    'XGBoost_Optimized': best_xgb,
    'SVM_Default': svm_model,
    'SVM_Optimized': best_svm
}

# Збереження кожної моделі у окремий файл
for model_name, model in models_to_save.items():
    file_path = os.path.join(models_dir, f"{model_name}.pkl")
    joblib.dump(model, file_path)
    print(f"Модель {model_name} збережена у файл {file_path}")
```

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

Кваліфікаційна робота

на тему:

«Технологія виявлення роботи ботнету в корпоративній мережі із застосуванням машинного навчання»

Виконав: САМОЙЛЕНКО Владислав Олексійович, БСДМ-63

Керівник: ГАХОВ Сергій Олександрович, к.військ.н., доцент

Об'єкт дослідження – виявлення роботи ботнету в корпоративній мережі.

Предмет дослідження – технологія виявлення роботи ботнету в корпоративній мережі із застосуванням машинного навчання.

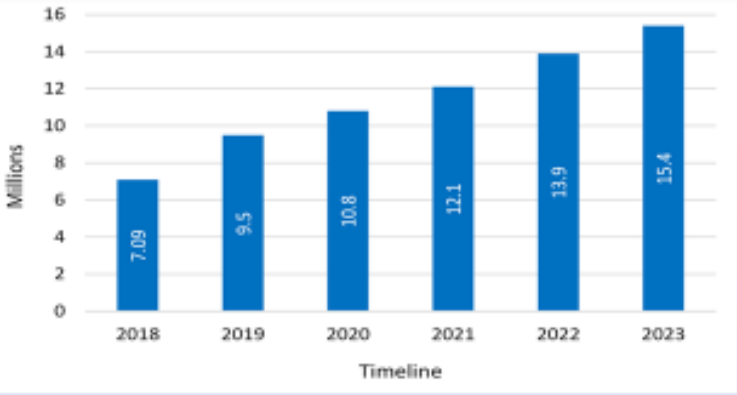
Мета роботи – розробити та оцінити ефективну технологію виявлення ботнетів у корпоративних мережах за допомогою методів машинного навчання, що забезпечує високу точність класифікації та оперативність обробки даних.

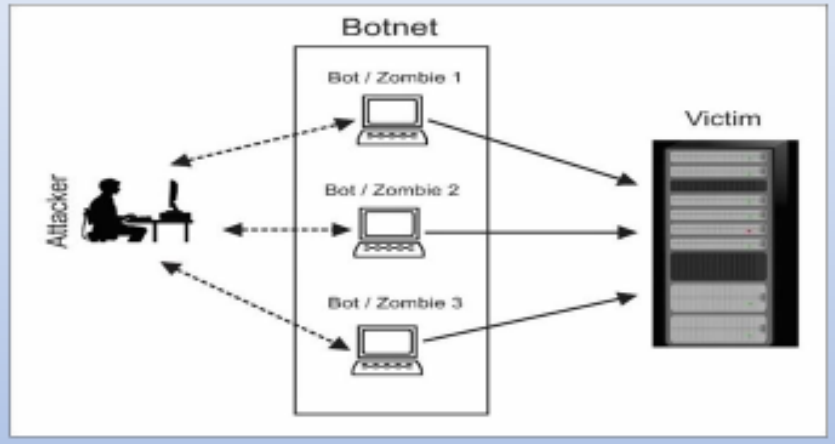
Наукові завдання:

- дослідити сутність проблеми виявлення ботнет-активності у корпоративній мережі;
- проаналізувати сучасні підходи до виявлення ботнетів із застосуванням машинного навчання;
- проаналізувати існуючі моделі машинного навчання для виявлення ботнетів;
- проаналізувати методи попередньої обробки, балансування даних та оптимізації моделей машинного навчання;
- розкрити порядок реалізації технології виявлення ботнетів у корпоративній мережі із застосуванням оптимізованих моделей машинного навчання.

Дослідження проблеми виявлення ботнетів у корпоративній мережі

У сучасному цифровому середовищі ботнети стали однією з найсерйозніших загроз для безпеки корпоративних мереж. Ботнет — це мережа скомпрометованих пристроїв (ботів), які зловмисники контролюють через командно-контрольні сервери. Ці боти можуть виконувати широкий спектр шкідливих дій, зокрема DDoS-атаки, розсилання спаму, викрадення даних та інші кібератаки. З розвитком технологій та методів обходу традиційних систем захисту, ефективність сигнатурних методів виявлення ботнетів значно знизилася, що створює нагальну потребу у розробці нових, більш ефективних технологій виявлення.



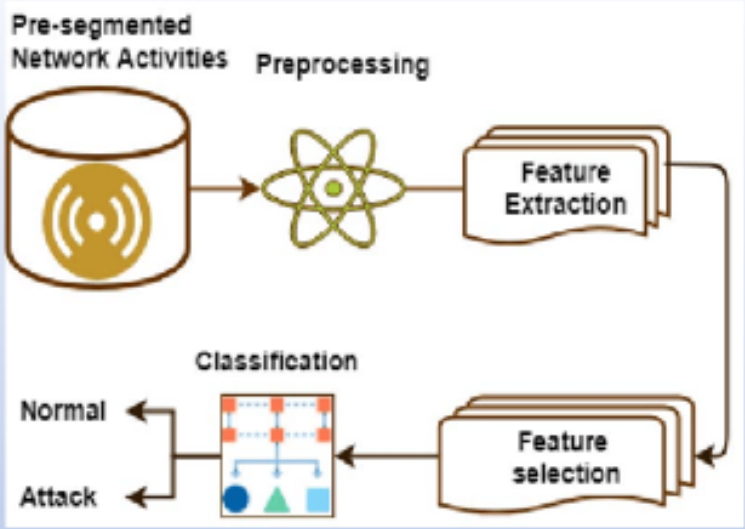


У контексті корпоративних мереж наслідки ботнет-активності можуть бути катастрофічними — від фінансових збитків до репутаційних втрат. Тому дослідження проблеми виявлення ботнетів є вкрай актуальним. Враховуючи масштаб проблеми, організації повинні запроваджувати сучасні методи захисту, що забезпечують своєчасне виявлення та реагування на ботнет-загрози. Від ефективності цих заходів залежить стійкість корпоративної мережі до інцидентів безпеки та безперервність функціонування ключових сервісів. На ілюстрації можна побачити зростання інтенсивності атак, що підтверджує актуальність питання.

Аналіз сучасних підходів до виявлення ботнетів із застосуванням машинного навчання

Сучасні підходи до виявлення ботнетів усе частіше спираються на методи машинного навчання замість традиційних сигнатурних методів. На відміну від статичних рішень, МН-алгоритми аналізують поведінкові ознаки трафіку, виділяють приховані патерни та виявляють аномалії. Це дозволяє ідентифікувати як відомі, так і нові, раніше невідомі ботнети.

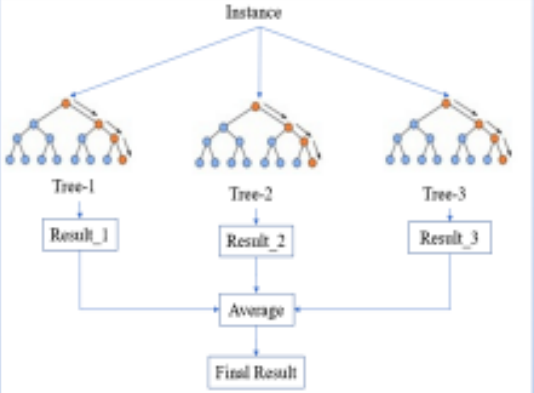
- Серед актуальних методів можна виокремити:
- Кероване навчання використовує мічені дані для навчання моделей, таких як Random Forest, XGBoost та Support Vector Machine (SVM), що досягають високих показників точності та повноти.
- Некероване навчання, зокрема методи кластеризації та асоціативних правил, дозволяє виявляти нові або рідкісні типи ботнетів без необхідності мічених даних.
- Глибоке навчання використовує нейронні мережі, такі як CNN та RNN, для аналізу високовимірних даних та виявлення складних патернів, забезпечуючи автоматичне виділення ознак і високу ефективність у складних задачах.



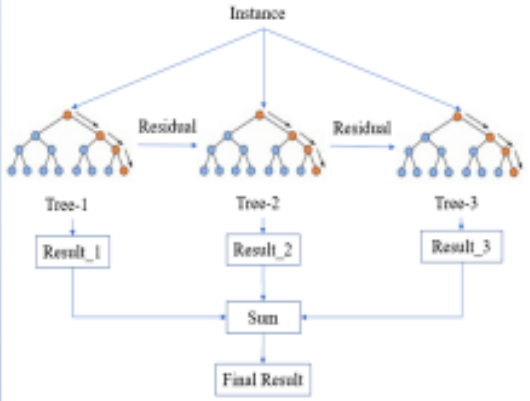
Переваги МН-підходів включають адаптивність, масштабованість та здатність до самооновлення. У міру появи нових загроз моделі можна перенавчати на актуальних даних. Це особливо важливо, оскільки ботнети постійно змінюють стратегії для уникнення виявлення.

Аналіз існуючих моделей МН для виявлення ботнетів

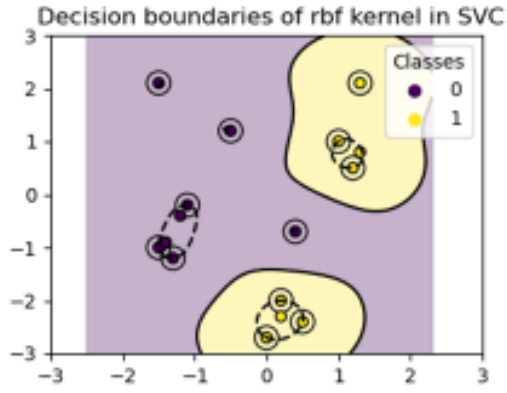
Random Forest є ансамблевим методом, який створює численні дерева рішень на випадкових підмножинах даних та ознак, що забезпечує високу точність класифікації та стійкість до перенавчання. Його здатність оцінювати важливість ознак сприяє ефективному відбору релевантних характеристик. Однак, цей алгоритм може вимагати значних обчислювальних ресурсів та пам'яті при великій кількості дерев.



XGBoost представляє собою оптимізовану реалізацію градієнтного бустингу, відзначаючись високою швидкістю навчання та прогнозування. Завдяки послідовному додаванню дерев, він ефективно коригує помилки попередніх моделей, підвищуючи точність. Проте, складність налаштування гіперпараметрів та ризик перенавчання без належної регуляризації є його недоліками.



SVM фокусується на пошуку оптимальної гіперплощини для розділення класів у високовимірному просторі, використовуючи ядрові функції для нелінійного розділення. SVM демонструє високу точність класифікації та ефективність у роботі з високовимірними даними. Однак, він є обчислювально витратним при великих наборах даних і вимагає ретельного налаштування гіперпараметрів.



Аналіз методів попередньої обробки, балансування даних та оптимізації моделей машинного навчання

Попередня обробка даних:

- Завантаження та об'єднання даних. Використання бібліотеки pandas для інтеграції файлів датасету в єдиний набір, що підвищує різноманітність та узагальнюючу здатність моделей.
- Первинний аналіз. Оцінка кількості записів, ознак, типів даних та розподілу класів, виявлення незбалансованості між нормальним трафіком та ботнет-активністю.
- Очищення даних. Видалення нерелевантних стовпців та дубльованих або постійних ознак для зменшення розмірності.
- Обробка пропущених значень. Заповнення середніми значеннями для числових ознак та використання найбільш частих значень або маркерів "Unknown" для категоріальних ознак.
- Виявлення та обробка аномалій. Аналіз розподілу числових ознак за допомогою діаграм розмаху та гістограм, видалення або обмеження викидів.

Балансування даних:

Переважання нормального трафіку може призвести до некоректної класифікації ботнет-активності. Методи балансування включають oversampling, наприклад, SMOTE, ADASYN, а також undersampling. Ці підходи дозволяють вирівняти розподіл класів, покращуючи здатність моделей розпізнавати шкідливу активність. Вибір SMOTE обумовлений генерацією синтетичних зразків менш представленого класу, що запобігає перенавчанню та покращує узагальнення моделей.

Оптимізація моделей:

Байєсова оптимізація з використанням бібліотеки Hyperopt та алгоритму Tree-structured Parzen Estimator (TPE) дозволяє ефективно знаходити оптимальні гіперпараметри. Переваги включають зменшення кількості ітерацій порівняно з Grid Search та Random Search, а також зниження обчислювальних витрат. Процес включає визначення простору гіперпараметрів, функції втрат, запуск оптимізації та аналіз результатів для моделей Random Forest, XGBoost та SVM.

Порядок реалізації технології виявлення ботнетів із застосуванням оптимізованих моделей МН

Порядок реалізації технології виявлення ботнетів із застосуванням оптимізованих моделей машинного навчання можна визначити як комплексний ланцюг взаємопов'язаних дій, кожна з яких відіграє важливу роль у досягненні кінцевої мети — надійного та ефективного виявлення загроз, а саме ботнетів, у сучасних корпоративних мережах.

Реалізація технології включає кілька етапів:

- Збір та попередня обробка даних. Підготовка даних для аналізу шляхом очищення та перетворення.
- Балансування даних. Вирівнювання розподілу класів для забезпечення ефективного навчання моделей.
- Навчання та оптимізація моделей. Побудова моделей машинного навчання з використанням оптимальних гіперпараметрів.
- Оцінка моделей. Перевірка продуктивності моделей за допомогою відповідних метрик та крос-валідації.
- Аналіз важливості ознак. Визначення ключових ознак, що впливають на рішення моделей.

Оптимізація моделей МН за допомогою гіперпараметрів

Модель	Гіперпараметр	Оптимальне значення	Опис
Random Forest	n_estimators	500	Кількість дерев у лісі
	max_depth	25	Максимальна глибина кожного дерева
	min_samples_split	8	Мінімальна кількість зразків, необхідних для розділення внутрішнього вузла
	min_samples_leaf	1	Мінімальна кількість зразків у листовому вузлі
	max_features	'sqrt'	Кількість ознак, які розглядаються при пошуку найкращого розділення
XGBoost	n_estimators	230	Кількість ітерацій бустингу
	max_depth	13	Максимальна глибина дерева
	gamma	8.1319	Мінімальне зменшення втрат, необхідне для подальшого розділення вузла
	reg_alpha	40	Параметр L1-регуляризації, що штрафує ваги, зменшуючи переобладнання
	reg_lambda	0.00626	Параметр L2-регуляризації, який допомагає уникнути перенавчання шляхом зменшення великих ваг
	colsample_bytree	0.9937	Відсоток ознак, які використовуються для кожного дерева
	min_child_weight	0	Мінімальна сума ваг усіх зразків у листовому вузлі
SVM	C	2.7604	Параметр регуляризації, який контролює баланс між максимізацією маржі та мінімізацією помилок класифікації

Результати моделей МН

Після проведення оптимізації гіперпараметрів, було отримано результати навчання моделей машинного навчання. Ці результати дозволяють оцінити ефективність кожної моделі як до, так і після оптимізації, а також порівняти їхні показники точності, повноти та часу тренування.

З таблиці видно, що оптимізовані моделі Random Forest та XGBoost демонструють найвищі показники F1-міри та точності, досягаючи майже ідеальних значень. Це свідчить про здатність цих алгоритмів ефективно розрізняти ботнет-трафік від нормального, мінімізуючи як пропуски реальних загроз, так і хибні тривоги.

Модель	accuracy	precision	recall	F1 - міра	час тренування (с)
Random Forest	0.9789	0.9896	0.9797	0.9828	238
Random Forest (оптимізована)	0.9999	0.9999	0.9998	0.9999	1180
XGBoost	0.9879	0.9889	0.9879	0.9856	15
XGBoost (оптимізована)	0.9999	0.9999	0.9999	0.9999	16
SVM	0.8795	0.8150	0.9819	0.8907	2414
SVM (оптимізована)	0.9072	0.8453	0.9968	0.9148	2393

Особливо виділяється модель XGBoost, яка після оптимізації досягла F1-міри 0.9999 та часу тренування лише 16 секунд, що робить її найкращим вибором для реальних корпоративних мереж, де важлива швидкість реагування на загрози. У порівнянні з нею, Random Forest після оптимізації також показує відмінні результати, проте потребує значно більше часу для тренування – 1180 секунд.

Модель SVM, хоча і демонструє покращення після оптимізації, має нижчі показники F1-міри (0.9148) та значно більше часу тренування (2393 секунди) порівняно з іншими моделями. Це робить її менш придатною для застосувань у великих корпоративних мережах, де обчислювальні ресурси та час є критично важливими факторами.

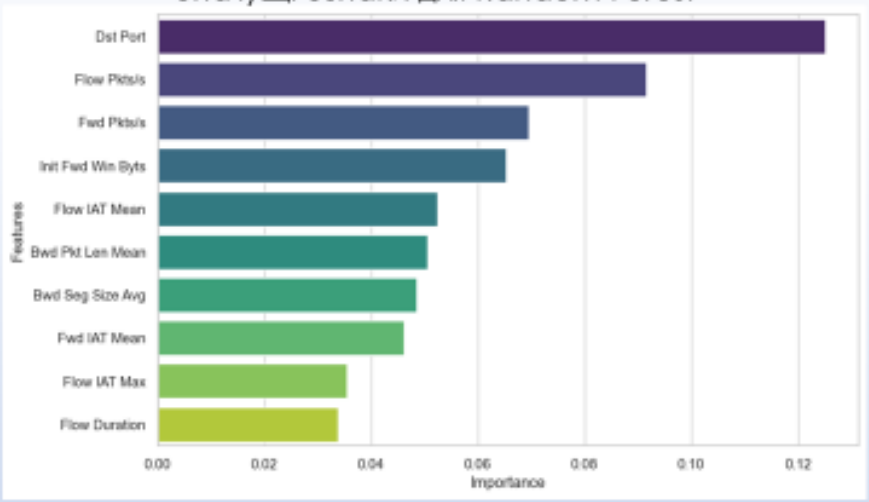
Значущість ознак

Для глибшого розуміння того, які ознаки найбільше впливають на рішення моделей, було проведено аналіз важливості ознак для оптимізованих моделей Random Forest та XGBoost. Цей аналіз дозволяє визначити ключові параметри, які найбільше сприяють точному розпізнаванню ботнетів, а також оптимізувати подальші зусилля в області збору та обробки даних. Для SVM з радіальним базисним ядром не має прямого механізму для оцінки важливості ознак, як це роблять деревоподібні моделі, такі як Random Forest та XGBoost.

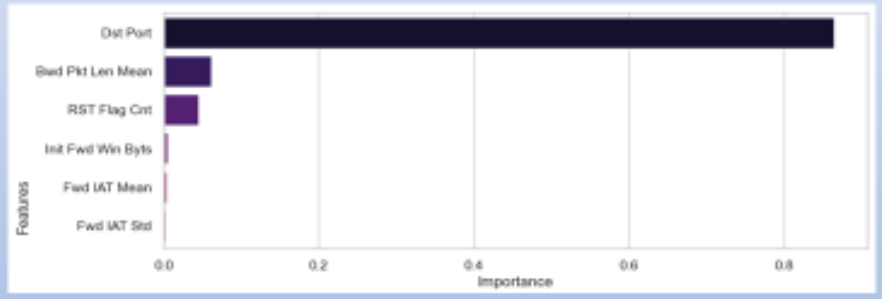
Аналіз важливості показав, що такі ознаки, як порти призначення, кількість пакетів за секунду та параметри TCP-з'єднань є визначальними для виявлення ботнетів. Наприклад, ознака Dst Port є найважливішою як для Random Forest, так і для XGBoost, що свідчить про її ключову роль у класифікації трафіку. Інші ознаки, такі як Flow Pkts/s та Bwd Pkt Len Mean, також мають високий рівень важливості, підтверджуючи їхню значущість у визначенні аномальної поведінки мережевого трафіку.

Аналіз важливості ознак не лише покращує інтерпретованість моделей, але й сприяє оптимізації систем моніторингу мережі. Визначення ключових ознак дозволяє зосередити зусилля на зборі та обробці найбільш інформативних даних, що підвищує ефективність виявлення ботнет-активності та зменшує обчислювальні витрати.

Значущі ознаки для Random Forest



Значущі ознаки для XGBoost



Рекомендації для покращення методів виявлення ботнетів

Спираючись на результати, можна сформувати конкретні рекомендації:

Висновки

- **Досліджено проблему виявлення ботнет-активності у корпоративній мережі**, визначено основні виклики та загрози, пов'язані з існуванням ботнетів. Виявлено, що ботнети можуть спричиняти значні збитки для організацій, включаючи порушення роботи мережі, витік конфіденційної інформації та зниження довіри клієнтів.
- **Проаналізовано сучасні підходи до виявлення ботнетів із застосуванням машинного навчання**, визначено їхні переваги та обмеження. Виявлено, що ансамблеві методи, такі як Random Forest та XGBoost, демонструють високу ефективність у класифікації шкідливого трафіку, тоді як методи, засновані на Support Vector Machine (SVM), мають обмежену застосовність через високу обчислювальну складність та нижчу точність.
- **Оцінено існуючі моделі машинного навчання для виявлення ботнетів**, визначено, що оптимізовані моделі Random Forest та XGBoost забезпечують найвищі показники точності та F1-міри. Зокрема, XGBoost відзначається найкращими показниками за швидкістю тренування та точністю класифікації, що робить його найбільш придатним для використання в реальних корпоративних мережах.
- **Проаналізовано методи попередньої обробки, балансування даних та оптимізації моделей машинного навчання**, підтвердивши важливість використання технік, таких як SMOTE для балансування класів та байєсова оптимізація за допомогою Hyperopt для налаштування гіперпараметрів. Ці методи значно покращують продуктивність моделей, забезпечуючи високу точність та стабільність результатів.
- **Розроблено та реалізовано технологію виявлення ботнетів у корпоративній мережі із застосуванням оптимізованих моделей машинного навчання**. Результати показали, що запропонована технологія забезпечує високу точність класифікації та оперативність обробки даних, що є критично важливим для ефективного реагування на загрози у реальному часі. Оптимізовані моделі Random Forest та XGBoost продемонстрували здатність ефективно розрізняти ботнет-трафік від нормального, мінімізуючи як пропуски реальних загроз, так і хибні тривоги.