МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації
_____ Михайло СТЕПАНОВ
«_____»__ _____ 2023 р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Метод застосування Random Forest для розпізнавання кібератак

Виконавець: Кіріл БУРЛАКА
Керівник: д.т.н., доцент Людмила ТЕРЕЙКОВСЬКА
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА
Нормоконтролер: д.т.н., доцент Людмила ТЕРЕЙКОВСЬКА

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп’ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації
_____ Михайло СТЕПАНОВ
«__»_____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
здобувача вищої освіти Бурлаки Кіріла Руслановича

1. Тема: *Метод застосування Random Forest для розпізнавання кібератак* затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: проаналізувати існуючі системи та методи розпізнавання кібератак; на основі аналізу виділити вхідні і вихідні параметри, завдяки яким можливо провести порівняння існуючих методів розпізнавання кібератак, виявити їх переваги і недоліки; розробити метод застосування Random Forest для розпізнавання кібератак та програмне забезпечення системи розпізнавання.
4. Зміст пояснювальної записки: аналіз засобів розпізнавання кібератак; розробка методу застосування Random Forest для розпізнавання кібератак; система розпізнавання кібератак.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	19.09.2023	*Виконано*
2.	Аналіз літературних джерел	20.09.2023	*Виконано*
3.	Обґрунтування вибору рішення	22.09.2023	*Виконано*
4.	Збір інформації	24.09.2023	*Виконано*
5.	Дослідження сучасних систем і методів розпізнавання кібератак	15.10.2023	*Виконано*
6.	Розробка методу застосування Random Forest для розпізнавання кібератак	22.11.2023	*Виконано*
7.	Розробка програмного забезпечення та проведення експериментальної верифікації отриманих результатів	07.12.2023	*Виконано*
8.	Апробація роботи на II науково-технічній конференції “Modern research in science and education”	23.10.2023	*Виконано*
9.	Перевірка на антиплагіат	11.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	15.11.2023	*Виконано*
11.	Оформлення презентації	16.11.2023	*Виконано*
12.	Отримання рецензій	17.11.2023	*Виконано*

Здобувач вищої освіти Кіріл БУРЛАКА
(підпис, дата)

Керівник кваліфікаційної роботи Людмила ТЕРЕЙКОВСЬКА
(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків і має 98 сторінки основного тексту, 27 рисунків, 7 таблиць, 9 сторінок додатків. Список використаних джерел містить 30 найменування і займає 4 сторінки. Загальний обсяг роботи 117 сторінок.

Метою роботи є підвищення ефективності засобів розпізнавання кібератак за рахунок використання запропонованого методу.

В роботі вирішено задачу побудови методу застосування Random Forest для розпізнавання кібератак.

На основі розробленого методу розроблено структуру системи розпізнавання кібератак та відповідне програмне забезпечення.

Розроблений метод та програмне забезпечення відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності від кібератак.

Можливі напрямки розвитку цієї роботи пов’язані із розширенням номенклатури кібератак.

Ключові слова: метод, розпізнавання кібератак, машинне навчання, Random Forest, датасет, система розпізнавання.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний світ невперше зіткнувся з кіберзагрозами, але з кожним роком їхні обсяги та складність продовжують зростати. Кібератаки, що впливають на комп'ютерні системи, мережі та дані, стали невід'ємною частиною нашого цифрового життя. Вони можуть завдати серйозної шкоди індивідам, компаніям, урядам та загалом суспільству. Щоб захистити цифрові ресурси та інфраструктуру, необхідно не тільки виявляти кібератаки, але і розпізнавати їх у реальному часі. У цьому контексті методи машинного навчання стають все більш важливими, оскільки вони можуть забезпечити ефективну оборону від кіберзагроз. Один з таких методів - алгоритм Random Forest, який відзначається високою точністю та здатністю працювати зі складними наборами даних. Використання Random Forest для розпізнавання кібератак стає актуальною задачею у сфері кібербезпеки.

Відомі підходи до вирішення поставленої задачі. Для боротьби з кібератаками було розроблено різні підходи та методи, такі як сигнатурний аналіз, виявлення аномалій, та машинне навчання. Однак, метод Random Forest, що базується на ансамблю дерев рішень, надає нові можливості для розпізнавання кібератак та підвищення ефективності захисту.

Метою роботи є підвищення ефективності засобів розпізнавання кібератак. Для досягнення поставленої мети необхідно вирішити такі задачі:

- аналізувати існуючі рішення в області розпізнавання кібератак;
- розробити метод застосування Random Forest для розпізнавання кібератак;
- розробити програмне забезпечення та провести експериментальну верифікацію отриманих результатів.

Галузь застосування. Розроблений метод та програмне забезпечення можуть бути використані в галузі кібербезпеки для забезпечення надійного захисту від кібератак.

Об'єктом дослідження є процеси розпізнавання кібератак.

Предметом дослідження є моделі та методи розпізнавання кібератак на основі застосування Random Forest.

Методи дослідження базуються на теорії інформації, системному аналізі (для аналізу предметної області досліджень), на машинному навчанні (для розробки методу застосування Random Forest для розпізнавання кібератак), на методах теорії алгоритмів, об'єктно-орієнтованого проєктування, програмування (для програмної реалізації розробленої системи розпізнавання).

Новизна одержаних результатів полягає в наступному:

отримав подальший розвиток метод застосування Random Forest для розпізнавання кібератак, що за рахунок адаптації параметрів алгоритма Random Forest дозволяє підвищити ефективність розпізнавання кібератак. Найвища точність моделі, отримана під час тестування системи, складає 0.9991639716272454 з 1. Модель показала лише 1 помилково-позитивний та 6 помилково-негативних результатів на тестових даних, що є вражаючим результатом.

Практична цінність отриманих результатів:

- розроблено інструмент для розпізнавання кібератак із застосуванням застосуванням Random Forest.
- розроблена методика проведення експерименту з використанням розробленого програмного засобу розпізнавання кібератак.

Апробація. Основні положення роботи доповідалися та обговорювалися на таких конференціях: МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ «ЖИВУЧІСТЬ ТА РЕЗИЛЬЄНТНІСТЬ – 2023» (Survivability & Resilience – 2023.

РОЗДІЛ 1. АНАЛІЗ ЗАСОБІВ РОЗПІЗНАВАННЯ КІБЕРАТАК

1.1. Проблематика розпізнавання кібератак

На сьогоднішній день багато галузей мають потребу у використанні алгоритмів, що дозволяють класифікувати дані або будувати прогнози подальшої зміни певних значень. Безпосередньо в сфері кібербезпеки так алгоритми використовуються у переважній більшості для виявлення атак. Тобто класифікації деяких вхідних даних такими, що означають атаку певного роду. Найбільш незахищеними у даній ситуації є кінцеві точки, що відносяться до інтернету речей.

Проблема забезпечення кібербезпеки для таких пристроїв, які відносяться до IoT, полягає у особливостях цієї галузі. Наприклад використання технологій, які вже є застарілими. Такі технології зазвичай мають проблеми з конфіденційністю та загалом з безпекою, бо на таких пристроях оновлення програмного забезпечення(ПЗ) відбувається досить рідко, або взагалі не відбувається.

Кібератаки представляють собою зловмисні дії, спрямовані на комп'ютерні системи, мережі та дані з метою завдання шкоди або отримання несанкціонованого доступу. У цьому розділі розглядаються основні типи кібератак, включаючи віруси, черви, троянці, фішинг, DDoS-атаки та інші. Визначаються загрози, які створюють кібератаки для інформаційної безпеки.

Кібератаки - це зловмисні дії, спрямовані на комп'ютерні системи, мережі та дані з метою нанесення шкоди, викрадення інформації або порушення нормального функціонування цих систем. Кібератаки можуть бути виконані різними способами і мати різні мети. Ось деякі поняття, типи та загрози, пов'язані з кібератаками:

Типи кібератак:

1) Віруси і черв'яки. Це програми, які можуть самореплікуватися та поширюватися через комп'ютерні мережі, завдяки чому вони можуть інфікувати багато систем.

2) Фішинг. Це спроби обману користувачів, які зазвичай включають в себе підроблені веб-сайти та повідомлення, що стимулюють користувачів виконувати дії, які вигідні атакуючому.

3) DDoS-атаки. Атаки на забруднення послуг (Denial of Service) спрямовані на перевантаження цільової системи трафіком, щоб вона перестала працювати нормально.

4) Ретельна перехоплення інформації (Man-in-the-Middle). Атакувач перехоплює трафік між двома сторонами і може відстежувати або змінювати передачу даних.

5) Витік інформації (Data Breach). Незаконне здобуття доступу до конфіденційної інформації, такої як особисті дані користувачів або бізнес-секрети.

Таблиця 1.1

Типи кібератак та їх характеристики

Кібератаки:	**Опис:**
DDoS	Атаки, які спрямовані на перевантаження серверів або мережі.
Фішинг	Спроби здійснити шахрайські атаки, використовуючи піддельні веб-сайти або електронні листи.
Малвара	Зловмисне програмне забезпечення, що вивчає конфіденційні дані або завдає шкоди системі.
SQL ін'єкції	Атаки, які використовують вразливості SQL для незаконного доступу до баз даних.
XSS атаки	Атаки, які вбудовують зловмисний код у веб-сторінки, що виконується на браузері користувача.
Інші атаки	Інші різновиди кібератак, які можуть включати в себе атаки на підтримку сесій, введення через форму тощо.

Загрози та мотивації:

1) Фінансова мотивація. Атаки можуть бути спрямовані на вигоду, зокрема на крадіжку грошей або інших цінних ресурсів.

2) Шпигунство та розвідка. Деякі атаки проводяться для отримання конфіденційної інформації, такої як військова або комерційна розвідка.

3) Активізм та політичні мотиви. Атаки можуть бути здійснювані для виразу політичних поглядів або з метою дестабілізації політичних режимів.

4) Сайбервійна інфраструктура. Атаки можуть бути спрямовані на знищення або нарушення роботи критичних інфраструктурних об'єктів, таких як електростанції або системи водопостачання.

Заходи захисту:

1) Антивірусне програмне забезпечення. Встановлення і оновлення антивірусного програмного забезпечення для виявлення і блокування шкідливого коду.

2) Фаєрволи. Використання фаєрволів для контролю доступу до мережі та фільтрації трафіку.

3) Шифрування. Використання шифрування для захисту конфіденційної інформації під час передачі та зберігання даних.

4) Оновлення програмного забезпечення. Регулярне оновлення операційних систем, програм та патчів для закриття вразливостей.

5) Навчання та навчання персоналу. Просвічування користувачів та персоналу щодо заходів безпеки та небезпек.

Кібератаки можуть мати серйозні наслідки для індивідів, компаній та національної безпеки, тому важливо бути завжди пильним і застосовувати відповідні заходи захисту.

Проблема розпізнавання кібератак є актуальною і важливою в сучасному цифровому світі з кількох причин:

1) Зростання кількості кібератак. За останні роки кількість кібератак інтенсивно збільшилася. Це стосується як кількості окремих атак, так і їх складності. Кіберзлочинці постійно розвивають свої методи, щоб обходити захист та залишити менше слідів.

2) Зростання кількості кіберзлочинів. Злочинці, які використовують кіберпростір для виконання різних видів злочинів, стали більш активними і

вдосконалили свої техніки. Кібератаки можуть призвести до втрати конфіденційної інформації, завдати фінансових втрат, а також порушити роботу критичних інфраструктурних об'єктів, таких як енергетичні системи чи медичні заклади.

4) Серйозні наслідки для організацій і осіб. Кібератаки можуть призвести до втрати даних, фінансових збитків, порушення конфіденційності та репутації, а також загрозити національній безпеці. Це стосується не лише компаній, але і урядових організацій, критично важливих інфраструктур, та звичайних користувачів.

5) Залежність від технологій. З кожним роком наша залежність від інформаційних технологій росте. Велика частина наших особистих, корпоративних та державних процесів здійснюється онлайн. Тому загрози для кібербезпеки стають більш серйозними.

6) Комплексність атак. Кіберзлочинці використовують різні методи та вектори атак для досягнення своїх цілей. Це може бути фішинг, DDoS-атаки, використання вразливостей в програмному забезпеченні, розповсюдження шкідливих програм, та інші техніки. Часто вони комбінують декілька методів, щоб зробити атаку ще складнішою для виявлення.

7) Економічні наслідки. Кібератаки можуть призвести до значних економічних втрат. Компанії втрачають гроші на відновлення інфраструктури та компенсації клієнтам за втрати. Крім того, загрози для кібербезпеки можуть впливати на стійкість фінансових ринків.

8) Необхідність реагування в реальному часі. Для успішного уникнення або обмеження збитків від кібератаки, необхідно виявляти атаки в реальному часі або навіть перед їхнім виконанням. Це вимагає ефективних систем розпізнавання кібератак та аналізу поведінки систем.

9) Збереження конфіденційності. Захист конфіденційної інформації є критично важливим, особливо для компаній та урядових організацій. Кібератаки можуть призвести до розкриття конфіденційних даних, які можуть бути

використані для різних злочинів, включаючи шантаж та крадіжку особистої інформації.

10) Використання штучного інтелекту і машинного навчання. Щоб розпізнавання кібератак було більш ефективним, сучасні системи використовують технології штучного інтелекту та машинного навчання. Вони допомагають аналізувати величезні обсяги даних та виявляти аномалії в реальному часі.

11) Національна безпека. Деякі кібератаки можуть бути спрямовані на державні структури або критичну інфраструктуру, що створює загрозу національній безпеці. Зокрема, атаки на об'єкти критичної інфраструктури, такі як енергетичні системи чи транспорт, можуть призвести до серйозних наслідків.

12) Законодавчий тиск і вимоги щодо забезпечення кібербезпеки. Багато країн вводять строгі вимоги щодо кібербезпеки для організацій та урядових установ. Це включає вимоги до розпізнавання та повідомлення про кібератаки, а також заходи для запобігання їм.

Таблиця 1.2

Методи розпізнавання кібератак, їх переваги та недоліки

Метод розпізнавання	**Переваги**	**Недоліки**
Сигнатурні методи	Висока точність, особливо для відомих атак.	Неефективні проти нових та невідомих атак.
Аномалійні методи	Виявлення невідомих атак та змін у системі.	Велика кількість хибних позитивів, складнощі у визначенні порогових значень.
Методи машинного навчання	Адаптуються до нових атак та паттернів.	Вимагають великої кількості даних для навчання.

DDoS атаки.

Атаки прикладного рівня - це одна з найпоширеніших форм відмови в обслуговувані додатку на сьогоднішній день заснована на безкінечній відправці http(s)-повідомлень GET на порт 80(443) для завантаження веб-сервера, що робить його нездатним обробляти інші запити. Часто об’єктом флуду є не корінь веб-додатку, а одна з його частин, що виконує ресурсомістке завдання або та, що використовує базу даних. Надзвичайно швидке зростання кількості логів веб-серверів означало б, що атака почалася.

Контрзаходи проти HTTP-flood включають налаштування веб-серверів і баз даних для зменшення впливу атак, а також перевірку на наявність DoS- ботів за допомогою різних методів. По-перше, потрібно збільшити максимальну кількість підключень до бази даних за раз. По-друге, встановлення легкого та ефективного nginx перед головним веб-сервером значно підвищило б відмовостійкість, адже він буде гешувати(hashing) запити та видавати статичні дані, коли головний веб сервер витрачав свої ресурси на більш важливі задачі.

Також ресурсоємні частини сайту можуть та мають бути захищені перевіркою так званої «людяності» користувача, наприклад за допомогою використання CAPTCHA.

Детектування такого роду атаки є непростим завданням, адже відрізнити легітимні HTTP запити від тих, які причетні до DDoS атаки, нетривіальна задача. Ця проблема вирішується при використанні машинного навчання, адже між трафіком зловмисника та трафіком легітимного користувача є відмінності, але не завжди помітні людиною через втому, неуважність, недосвідченість і т. п.

Атаки мережевого рівня - це дуже поширений спосіб для DDoS за допомогою SYN пакетів, що надсилаються на певну машину. Мета цієї атаки це забити канал зв’язку та перевести мережевий стек операційної системи в стан, коли він більше не прийматиме нові запити на з’єднання. Заснована на спробі ініціювати велику кількість одночасних TCP-з'єднань шляхом відправки пакета SYN з неіснуючої адресою повернення.

Більшість операційних систем ставлять у чергу таке з’єднання де після кількох спроб відповісти на пакет ACK не прийшла відповідь. Тільки після n- ї спроби отримати відповідь на пакет ACK операційна система закриває з'єднання. Через великий трафік пакетів ACK, що намагаються закінчити встановлення з’єднання, черга швидко заповнюється, і ядро відмовляється намагатися відкрити нове з’єднання. Найрозумніші DoS-боти також аналізують систему перед атакою, надсилаючи лише запити на вже відкриті порти.

Атаки вичерпання полоси пропускання - UDP flood полягає в надсиланні до комп’ютера жертви великої кількості UDP пакетів, що призводить до захаращення смуги пропускання.

ICMP flood простий та ефективний метод забивання смуги через легкість реалізації.

DNS підсилення або DNS відбиття – метод що дозволяє за допомогою одного атакуючого комп’ютера та багатьох DNS серверів третіх сторін, генерувати великий об’єм трафіку. Така атака має в собі особливість невеликої кількості пропускної здатності атакуючого, заповнити набагато більшу полосу пропускання жертви. Це можливо за рахунок генерації такого запиту до DNS, відповідь до якого буде набагато більша у розмірі. Але відповідь отримає не атакуючий, а жертва, через підміну IP адреси у запиті.

Отже одна з цілей (D)DoS атаки полягає в заповненні полоси пропускання трафіку жертви. Середній та малий бізнес зазвичай використовують сервери з пропускною здатністю 1Gbps або 10 Gbps, що в свою чергу коштує в середньому 50 та 300 доларів США за місяць відповідно. Виснаження полоси пропускання зробить його непридатним до обслуговування легітимних запитів.

Іншою ціллю може бути вичерпання всіх доступних ресурсів, таких як RAM(Random Access Memory), використання всієї потужності ЦП, тощо.

Наслідками такої атаки можуть бути або недоступність сервера, або настання такого стану, який є потенційно небезпечним та може відкривати нові вразливості, що притаманні саме цьому стану системи та можуть призвести до компрометації сервера або даних.

Існує достатня кількість статичних і динамічних методів протидії таким атакам:

• Створити план протистояння DDoS атакам.

• Забезпечити високий рівень мережевої безпеки – використовувати брандмауери, антивіруси, засоби контролю вхідного та вихідного трафіку, сегментація мережі.

• Створення надлишковості.

• Використання CDN.

• Постійний моніторинг за системою.

Як можна побачити, є достатня кількість засобів захисту, які можна використовувати, щоб не стати жертвою DDoS атаки, або щоб вдало їй протистояти. Однак навіть в такому випадку після початку атаки до її виявлення може пройти певний час. Все залежить від того, як налаштований процес моніторингу та реакції на інциденти. Зловмисники намагаються збільшити час для виявлення атаки різними методами, тому навіть два три інженери можуть бути не в змозі надати оцінку показникам, що вони спостерігають на екранах. Тож гостро стоїть проблема саме детектування DDoS атаки вчасно, щоб максимально зменшити вплив на інфраструктуру. Звісно DDoS можна зафіксувати дивлячись на графік, наприклад такий як видно на рисунку 1.1

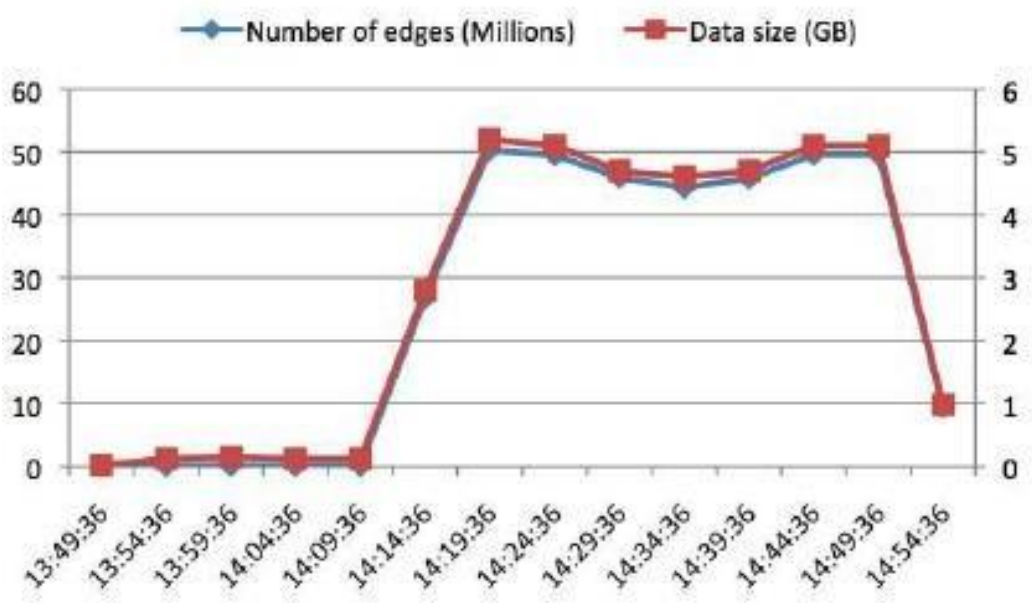

Рис. 1.1.Приклад графіку DDoS атаки

Проте крок дискретизації тут у 5 хвилин, це означає що таку атаку можуть розпізнати лише в проміжку від 1 до 5 хвилин, коли трафік збільшився в 30-50 разів.

У випадках, коли даних дуже багато і треба швидко дати їм оцінку, найбільш вдало справляються комп’ютери, адже вони вміють за короткий час проаналізувати надані дані. Проте аби-який метод, особливо статистичний, не дасть нам коректної оцінки. Статистичний метод, що створений для детектування атак може бути корисним і надавати правильну інформацію, але при зміні природи атаки, чи певних її складових такий метод нічого не помітить, а переписати його на льоту достатньо нетривіальна задача. Проблему з детектуванням різних видів атак причому з класифікацією до потоків пакетів може вирішити машинне навчання. Цей метод гнучкий, може використовуватись для багатьох сфер діяльності, в тому числі і у виявлені атак.

Фішинг атаки.

Фішинг (або "phishing") - це вид кібератаки, при якій атакуючий намагається обманути людину або користувача комп'ютерної системи, вивівши від неї конфіденційну інформацію, таку як паролі, особисту інформацію або фінансові дані. Фішинг може приймати різні форми і методи, включаючи:

1). Email Фішинг. Спам-повідомлення: Атакуючий надсилає масові спам-листи, в яких закликає одержувачів перейти на фішинговий веб-сайт або надіслати особисту інформацію.

Спуфінг: Атакуючий підроблює адресу електронної пошти від імені довірених організацій або осіб, щоб здати себе за них і викликати довіру.

2). Сайт Фішинг. Фішинговий сайт: Зловмисник створює підроблений веб-сайт, який схожий на легітимний сайт, такий як банк або соціальна мережа. Він закликає користувачів ввести свої логіни та паролі.

Хостинг фішингового контенту: Зловмисники можуть розміщувати фішинговий контент на вірусних або компрометованих веб-сайтах.

3). СМС Фішинг (або "Smishing"). Зловмисники надсилають користувачам SMS-повідомлення з проханням відправити особисту інформацію або перейти за посиланням.

4). Телефонний Фішинг (або "Vishing"). Атакуючий використовує голосовий зв'язок, намагаючись переконати людину надати конфіденційну інформацію по телефону. Це може включати в себе вимоги про відправлення грошей, пред'явлення паролів або кредитних карток.

5.) Соціальний Фішинг. Атакуючий використовує соціальні мережі або інші онлайн-платформи, щоб вивести інформацію зі сторінки користувача, яка могла б стати об'єктом атаки.

6). Фішинг через USB (USB Phishing або "USB Baiting"). Атакуючий залишає інфіковані USB-пристрої в областях, де вони можуть бути знайдені, і сподівається, що хтось їх підключить до свого комп'ютера. Це може призвести до відкриття вірусів або інших шкідливих програм.

7). Фішингові спроби або "Spear Phishing". В цьому випадку атака спрямована на конкретну особу або організацію, і атакуючий вивчає інформацію про свою жертву для створення більш реалістичних фішингових повідомлень.

8). Бізнес-фішинг або "Business Email Compromise" (BEC). В цьому випадку атака спрямована на бізнеси, і зловмисники намагаються видати себе за вищих керівників або постачальників з метою переказу грошей або конфіденційної інформації.

Це лише кілька з типів фішингу, і кожен з них може мати різні варіації та методи атаки. Усі ці види фішингу мають спільну мету - отримати доступ до конфіденційної інформації або фінансових ресурсів користувача. Для захисту від фішингу важливо бути обережним і надійним, не надавати особисту інформацію на ненадійних веб-сайтах або через ненадійні канали зв'язку.

Віруси і черв'яки.

Віруси і черв'яки є двома типами шкідливих програм, які можуть атакувати комп'ютерні системи і мережі, але вони відрізняються за способом поширення і функціональністю. Ось короткий огляд кожного типу:

1) Віруси (Computer Viruses)

Спосіб поширення: Віруси є програмами, які прикріплюються до інших програм або файлів і розповсюджуються, коли ці програми або файли виконуються або відкриваються.

Дія: Віруси можуть впроваджувати свій код в інші файли або програми і виконувати певні дії під час їхнього запуску. Ці дії можуть включати в себе видалення файлів, перекривлення роботи комп'ютера або поширення на інші системи через заражені файли або пристрої.

Приклади: Знаменитий вірус "ILOVEYOU", який поширювався через електронну пошту і завдає шкоди файлам на комп'ютері.

2) Черв'яки (Computer Worms)

Спосіб поширення. Черв'яки є автономними програмами, які можуть розповсюджуватися самостійно через комп'ютерні мережі без необхідності прикріплення до інших програм або файлів. Черв'яки призначені для пошуку уразливих систем у мережі та автоматичного внедренняся в них. Після цього вони можуть виконувати різні дії, такі як розповсюдження себе на інші системи, викрадання інформації, завади роботі мережі тощо. Черв'як "Conficker", який швидко поширився через Інтернет і інфікував мільйони комп'ютерів.

Обидва типи програм можуть завдавати серйозної шкоди комп'ютерним системам і мережам, і їхнім виявленням та захистом від них присвячено багато зусиль в області кібербезпеки. Будь-яка комп'ютерна система повинна бути належно захищеною за допомогою антивірусних програм, оновлених програм і брандмауера для зменшення ризику зараження вірусами та черв'яками.

Типи витоку інформації

Витік інформації - це ситуація, коли конфіденційна, особиста або захищена інформація стає доступною третім сторонам без дозволу або згоди власника цієї інформації. Це може призвести до серйозних наслідків, таких як порушення приватності, фінансові втрати, репутаційні проблеми та інші проблеми. Витік інформації може приймати різні форми та спричиняти різні наслідки. Ось декілька типів витоків інформації:

1) Технічний витік інформації (Technical Data Breach). Цей тип витоку інформації стосується порушення безпеки даних в комп'ютерних системах або мережах. Включає в себе незаконний доступ до баз даних, крадіжку паролів, вибірку даних з серверів тощо.

2) Фізичний витік інформації (Physical Data Breach). В цьому випадку конфіденційна інформація може витікати через фізичний доступ до пристроїв або документів. Наприклад, крадіжка комп'ютера, USB-накопичувача або документів зі секретною інформацією.

3) Витік даних через соціальну інженерію (Social Engineering Data Breach). Цей тип витоку інформації відбувається через обман або маніпуляцію людьми, щоб вони надали конфіденційну інформацію атакуючим. Це може включати в себе фішинг, відправку шкідливих посилань або телефонну обману.

4) Витік даних через внутрішні загрози (Insider Data Breach). Коли співробітники або особи з внутрішнього оточення підлягають витоку інформації. Це може бути навмисними діями або недбалістю.

5) Компрометація хмарних послуг (Cloud Data Breach). Витік даних, які зберігаються у хмарних послугах або інших онлайн-платформах, через незаконний доступ або порушення безпеки цих послуг.

6) Витік даних через атаку "ман-в-середнього" (Man-in-the-Middle Data Breach). Коли атакуючий вставляється між двома комунікуючими сторонами і перехоплює або змінює передачу даних між ними.

7) Витік фінансової інформації (Financial Data Breach). Включає в себе виток інформації про кредитні картки, банківські реквізити або іншу фінансову інформацію.

8) Медичний витік інформації (Medical Data Breach). Витік медичних записів або інших особистих медичних даних пацієнтів.

9) Витік інтелектуальної власності (Intellectual Property Data Breach). Витік конфіденційних технологічних чи дизайнерських розробок, які можуть завдати збитків компанії.

Витік інформації може мати серйозні наслідки для організацій та осіб, і захист від нього вимагає ретельного вивчення потенційних загроз та впровадження ефективних заходів безпеки. Загалом, проблема розпізнавання кібератак є актуальною і складною завданням, яке вимагає поєднання технологічних рішень, навчання та постійного моніторингу для забезпечення кібербезпеки в сучасному світі. Узагальнюючи, проблема кібербезпеки є надзвичайно важливою в сучасному світі, і ефективні методи розпізнавання кібератак є ключовим елементом для захисту інформації, ресурсів та національної безпеки. Визначення потенційних проблем та викликів при використанні Random Forest для розпізнавання кібератак.

Таблиця 1.3

Виклики та перспективи удосконалення розпізнавання кібератак

Виклики:	**Перспективи удосконалення:**
Відсутність чітких паттернів для нових атак	Використання глибокого навчання та нейронних мереж для виявлення складних атак.
Велика кількість хибних позитивів	Розробка алгоритмів для зменшення кількості хибнопозитивних виявлень.
Складність в навчанні моделей на реальних даних	Використання симуляцій та віртуальних середовищ для навчання та тестування моделей.

Random Forest - це потужний алгоритм машинного навчання, який може бути використаний для розпізнавання кібератак. Однак при його використанні можуть виникнути деякі потенційні проблеми та виклики:

1) Невиразність моделі. Випадковий Ліс може бути дуже складним, і інтерпретація його результатів може бути непростою. Це може стати перешкодою при виявленні причин або змісту конкретної кібератаки.

2) Підгонка моделі. Якщо модель навчання випадкового лісу налаштована

надто точно на навчальних даних, вона може бути переоснащеною. Це означає, що модель може добре впоратися з навчальними даними, але буде погано справлятися з новими атаками або змінами в структурі даних.

3) Обсяг інформації. Велика кількість функцій або атрибутів може призвести до перенавчання або втрати важливої інформації. Превелика кількість ознак може зробити модель надто громіздкою та невикористовуваною.

4) Незбалансованість класів. У кібербезпеці часто спостерігається нерівновага між класами, коли кількість нормальних подій значно перевищує кількість атак. Це може призвести до того, що модель буде погано розпізнавати атаки через домінування нормальних подій.

5) Оновлення моделі. Кібератаки постійно змінюються, тому модель, побудована за допомогою Random Forest, потребує постійного оновлення та навчання на нових даних для забезпечення ефективного виявлення нових загроз.

6) Обробка категорійних даних. Якщо вхідні дані містять категорійні атрибути, то їх потрібно відповідним чином кодувати або векторизувати перед використанням Random Forest.

7) Відсутність можливості визначення причин: Random Forest - це метод, спрямований на класифікацію і прогнозування, але він не завжди надає змогу визначити причини кібератаки або ідентифікувати зловмисників.

8) Час виконання: Випадковий Ліс може бути обчислювально витратним, особливо при обробці великих обсягів даних. Важливо забезпечити оптимальний обробки час при розгляді реального часу на виявлення кібератак.

Загалом, Random Forest може бути потужним інструментом для розпізнавання кібератак, але його використання вимагає уважної підготовки даних, контролю за перенавчанням та постійного оновлення для ефективного захисту від сучасних кіберзагроз.

1.2. Методи розпізнавання кібератак

Розпізнавання кібератак - це процес виявлення зловмисних дій або аномалій в комп'ютерних системах, мережах та даних. Існують різні методи

розпізнавання кібератак, які використовуються для виявлення потенційно небезпечних активностей. Основні методи включають:

1). Сигнатурний аналіз. Сигнатурний аналіз використовує заздалегідь відомі патерни (сигнатури) для ідентифікації конкретних типів кібератак. Ці сигнатури можуть бути ключовими словами, хешами файлів або конкретними послідовностями байтів. Математично цей метод може бути представлений як порівняння поточних даних із сигнатурами. Якщо збіг виявлено, то це може свідчити про кібератаку.

2). Виявлення аномалій. Цей метод спрямований на виявлення аномалій або несправедливих дій, які не відповідають нормальному стану системи. Для цього аналізуються статистичні характеристики поведінки системи. Математичні моделі, такі як моделі машинного навчання (наприклад, нейронні мережі або алгоритми кластеризації), можуть використовуватися для побудови нормального профілю системи. Аномалії виявляються як відхилення від цього профілю.

3). Виявлення вразливостей. Цей метод полягає в пошуку вразливостей в програмному забезпеченні або мережах, які можуть бути використані для здійснення атаки. Математичні аналізи включають в себе перевірку коду на вразливості, аналіз даних введення, перехресні атаки та інші методи для виявлення слабких місць у системі.

4). Виявлення атак на основі журналів. Цей метод включає аналіз журналів подій із систем та мереж, щоб виявити незвичайну або підозрілу активність. Математичний аналіз полягає у використанні правил і порогових значень для виявлення незвичайних подій у журналах.

5). Модельовані атаки (хакерський тестувальник). Цей метод включає в себе симуляцію атак на систему, щоб перевірити, наскільки вона захищена і чи можливо розпізнати ці атаки.

Таблиця 1.4

Сигнатурні методи

Метод	**Опис**
Статичні сигнатури	Використовують фіксовані паттерни атак, що базуються на відомих вразливостях та характеристиках атак.
Динамічні сигнатури	Вивчають дії та зміни в системі для виявлення відомих атак, а також зміни відомих атак на льоту.

Математичні методи можуть використовуватися для порівняння результатів симуляції з очікуваними патернами поведінки.

Ці методи розпізнавання кібератак можуть використовувати різні математичні підходи, включаючи статистику, машинне навчання, аналіз даних і шифрування. Важливо відзначити, що для ефективного розпізнавання кібератак часто використовують комбінацію цих методів та системи, які постійно оновлюються для виявлення нових загроз.

Аналізується сучасний стан методів розпізнавання кібератак, включаючи правила, евристичні та статистичні методи, які використовуються в кібербезпеці. Особлива увага приділяється методам машинного навчання, їх перевагам та недолікам. Математика в контексті методів розпізнавання кібератак грає важливу роль, оскільки багато з цих методів ґрунтуються на статистиці, алгоритмах класифікації та інших математичних концепціях. Обрано один із ключових математичних аспектів, який можна включити в розділ "Методи розпізнавання кібератак:

Алгоритми та машинне навчання

Класифікація. Застосування алгоритмів класифікації, таких як Random Forest, Support Vector Machines (SVM) або Neural Networks, для визначення, чи деякий трафік є кібератакою чи ні.

Кластеризація. Використання алгоритмів кластеризації, наприклад, k-Means, для виявлення аномалій або незвичайних патернів у трафіку.

Коли розглядаємо алгоритми та методи машинного навчання для розпізнавання кібератак, математична формула грає важливу роль у визначенні та поясненні цього методу.

Формула для прогнозування у Random Forest може бути зведена до голосування дерев рішень:

$$y^{\wedge} = \frac{1}{N_{trees}} \sum_{i=1}^{N_{trees}} f_i\,(x) \tag{1.1}$$

де y^ - передбачений клас;

N_{trees}- кількість дерев у лісі;

$f_i(x)$- прогноз дерева *i* для вихідних даних *x*.

Розпізнавання кібератак - це важливий аспект забезпечення кібербезпеки, який дозволяє виявити та відповісти на потенційно шкідливі дії в мережі та комп'ютерних системах. Існує багато методів і технологій для розпізнавання кібератак. Ось деякі з них:

1) Системи виявлення вторгнень (IDS) і системи запобігання вторгнень (IPS):

- IDS аналізують мережевий трафік та журнали подій, шукаючи вказівки на потенційні кібератаки.

- IPS можуть надавати захист в реальному часі, блокуючи атаки або запобігаючи їхньому розповсюдженню.

2) Аналіз журналів та реєстрація подій. Моніторинг та аналіз журналів подій комп'ютерних систем може виявити надзвичайні події, несправності або спроби несанкціонованого доступу.

3) Аналіз мережевого трафіку. Моніторинг та аналіз мережевого трафіку може розкрити незвичайні або аномальні паттерни, які вказують на кібератаку.

4) Машинне навчання та аналітика. Використання алгоритмів машинного навчання для створення моделей, які можуть виявляти аномалії в системах та мережах.

5) Системи детекції зловмисного програмного забезпечення. Виявлення шкідливих програм, таких як віруси, черв'яки та троянські коні, за допомогою антивірусного та антишпигунського програмного забезпечення.

6) Аналіз поведінки. Системи, які аналізують поведінку користувачів та систем, можуть виявляти аномалії або незвичайні дії, що можуть свідчити про кібератаку.

7) Інтелектуальні системи аналізу вмісту (Content Analysis). Аналіз текстового або вмістового матеріалу для виявлення загроз, таких як фішингові повідомлення або шкідливий код у вкладеннях.

8) Системи аналізу вразливостей. Сканування систем на вразливості та слабкі місця, які можуть бути використані зловмисниками для атак.

9) Аналіз коду та інтеграція з джерелами загроз. Аналіз програмного коду на наявність вразливостей та інтеграція із відомими джерелами загроз може допомогти виявити потенційні ризики.

10) Системи кореляції подій (SIEM). Системи SIEM збирають, аналізують та кореляціюють дані з різних джерел для виявлення аномалій та загроз у реальному часі.

Розпізнавання кібератак - це складний і постійно еволюціонуючий процес, і важливо поєднувати різні методи та технології для максимальної ефективності виявлення потенційних загроз.

Відбір ознак, обробка великої кількості даних та навчання на нових типах атак є важливими аспектами розпізнавання кібератак за допомогою машинного навчання. Давайте розглянемо ці питання детальніше:

1) Відбір ознак (Feature Selection). У задачах кібербезпеки може бути багато різних ознак, і не всі вони можуть бути корисними для виявлення атак. Важливо відібрати ті признаки, які дійсно важливі для моделі, щоб покращити її точність та швидкість роботи. Методи відбору ознак включають у себе статистичні та експертні підходи, такі як взаємна інформація, аналіз важливості ознак, методи відбору на основі дерев рішень, ансамблів тощо.

2) Велика кількість даних (Big Data). Збільшення обсягу даних може

вимагати використання потужних обчислювальних ресурсів та оптимізованих алгоритмів для ефективного навчання.

Робота з розподіленими даними. У кібербезпеці дані можуть бути розподілені по різних джерелах. Важливо мати систему обробки та збору цих даних.

3) Навчання на нових типах атак. Актуалізація навчання - спрощений спосіб навчання моделі та заборона на нові типи атак може виявитися недостатнім. Важливо періодично актуалізувати модель, використовуючи нові дані та нові признаки для виявлення нових загроз. Аналіз атак - важливо аналізувати нові типи атак і розуміти їх характеристики для ефективного виявлення. Це може включати в себе аналіз аномалій та навчання на тестових даних з новими атаками.

4) Ефективність моделі. Оцінка моделі - після навчання моделі важливо провести оцінку її ефективності на тестових даних та метрики, такі як точність, відновлення, F1-мера, ROC-крива тощо. Це допоможе визначити, наскільки добре модель справляється з розпізнаванням атак. Ансамблі та стекінг - для покращення ефективності моделі можна використовувати ансамблі моделей або стекінг (комбінування різних моделей).

Загальний підхід до цих питань полягає в поєднанні докладної підготовки даних, вибору правильних алгоритмів, оцінці та постійному оновленні моделі для виявлення кібератак. Також важливо мати команду експертів, які можуть аналізувати та реагувати на нові загрози та атаки, що розвиваються.

1.3. Характеристика методу Random Forest

Метод "Random Forest" (від англійського "випадковий ліс") є потужним алгоритмом машинного навчання, який використовується для завдань класифікації і регресії. Він є частиною сімейства ансамблевих методів, і його головною ідеєю є поєднання декількох рішень прийняття рішень, отриманих від окремих дерев рішень, для покращення точності та стабільності моделі.

Розширення алгоритму було запропоновано Лео Брейманом і Аделем Катлером, «Random Forests» є їхньою торговою маркою. Алгоритм поєднує в собі дві основні ідеї: метод беггінга Бреймана і метод випадкових підпросторів, запропонований Tin Kam Ho.

Таблиця 1.5

Характеристики методу Random Forest

Характеристика:	**Опис:**
Тип методу	Ансамбль дерев рішень, що використовується для класифікації, регресії та інших завдань машинного навчання.
Принцип роботи	Кожне дерево у випадковому лісі навчається на випадковій підвибірці даних та голосує за класифікацію або регресію.
Вибір параметрів	Включає кількість дерев, глибину дерев, критерій розбиття та інші параметри, які впливають на точність моделі.
Підсумовування відповідей дерев	Може використовувати голосування більшості (для класифікації) або середнє значення (для регресії) відповідей всіх дерев.
Стійкість до перенавчання	Зазвичай менше схильний до перенавчання порівняно зі звичайними деревами рішень завдяки випадковому вибору даних для навчання.

Обробка відсутніх даних	Здатний ефективно обробляти відсутні дані без необхідності їх попередньої обробки.
Інтерпретованість	Менше інтерпретований порівняно зі стандартними деревами рішень, але можливість отримання важливості ознак.
Розмір даних	Здатний ефективно обробляти великі обсяги даних завдяки паралельному навчанню багатьох дерев.

Основні характеристики методу "Random Forest":

1) Ансамбль дерев рішень. Метод "Random Forest" базується на використанні ансамблю дерев рішень. Зазвичай використовуються багато дерев (сотні або тисячі), кожне з яких навчається на випадковому підмножині даних із заміною (bootstrap samples).

2) Випадковість при навчанні. Випадковість входить у два аспекти методу "Random Forest". Випадковий вибір підмножини даних: кожне дерево рішень навчається на випадковому підмножині навчальних даних. Це дозволяє кожному дереву бачити лише обмежену частину даних, що допомагає уникнути перенавчання і підвищити стійкість моделі. Випадковий вибір ознак: під час побудови кожного вузла дерева випадково вибираються підмножини ознак для подальшого розгляду. Це зменшує кореляцію між деревами та робить модель більш різноманітною.

3) Багато дерев. Random Forest використовує багато дерев, і для прийняття рішення модель агрегує (середнє для регресії або голосування більшості для класифікації) прогнози всіх дерев.

4) Стійкість до перенавчання. Велика кількість дерев і випадковість в навчанні допомагають знизити перенавчання моделі на тренувальних даних.

5) Важливість ознак. Random Forest може оцінювати важливість кожної ознаки в процесі навчання, що допомагає визначити, які ознаки найбільше впливають на прогнози моделі.

6) Використання в різних задачах. Метод "Random Forest" може використовуватися як для задач класифікації (визначення класу або категорії) так і для задач регресії (прогнозування числового значення).

7) Достатня точність. Random Forest відомий своєю високою точністю і здатністю робити надійні прогнози, що робить його популярним алгоритмом в багатьох областях, включаючи медицину, фінанси, інтернет-рекламу і багато інших.

В цілому, метод "Random Forest" є потужним і важливим інструментом у сфері машинного навчання та аналізу даних, завдяки своїй здатності до ансамблю дерев рішень та стійкості до перенавчання.

Математична модель використання алгоритму машинного навчання з Random Forest може бути представлена наступним чином.

1. Навчання Random Forest

1.1 Вибір дерев рішень (n_trees): Випадково обираємо кількість дерев у лісі (n_trees).

1.2 Вибір підмножини даних для кожного дерева: Для кожного дерева вибираємо випадковий набір даних із заміщенням. Це може бути підмножина навчальних даних.

1.3 Вибір випадкових ознак для кожного розгалудження: Для кожного розгалудження вибираємо випадковий набір ознак для дерева з загальної кількості ознак.

1.4 Побудова дерева рішень. Для кожного дерева в Random Forest побудова дерева рішень відбувається на вибраних підмножинах даних та ознак.

2. Прогнозування з Random Forest

2.1. Класифікація. Кожне дерево голосує за клас. Передбачений клас для Random Forest визначається голосуванням більшості дерев:

$$y\hat{} = mode\left(f_1(x), f_2(x), \ldots, f_{n_{trees}}(x)\right), \quad (1.2)$$

де $y\hat{}$- передбачений клас;

n_{trees}- кількість дерев у лісі;

$f_i(x)$- передбачення дерева *i* для вхідних даних *x*.

2.2. Регресія. Прогноз у випадку регресії визначається середнім значенням передбачень усіх дерев:

$$y\hat{} = \frac{1}{n_{trees}} \sum_{i=1}^{n_{trees}} f_1(x) \quad (1.3)$$

де $y\hat{}$- передбачене числове значення;

n_{trees}- кількість дерев у лісі;

$f_i(x)$- передбачення дерева *i* для вхідних даних *x*.

3. Важливість ознак

Random Forest може надати важливість кожній ознаці. Це може бути виміряно за допомогою міри важливості ознак, яка визначається на основі того, як часто ознака використовується для розгалужень у всіх деревах у лісі.

Ця математична модель відображає базовий процес навчання та передбачення з Random Forest, і може бути використана для розуміння та визначення, як цей алгоритм працює у ваших дослідженнях.

Random Forest як ансамбль методів машинного навчання, що використовується для класифікації та регресії, базується на ідеї комбінування декількох дерев рішень для отримання більш точних та стійких прогнозів. Random Forest виник у відповідь на проблеми перенавчання, які можуть виникнути при роботі з одиночним деревом рішень.

Основні принципи роботи Random Forest

1) Вибір вибіркової підвибірки (Bootstrap). Спочатку формується вибіркова підвибірка з навчального набору даних. Це означає, що з одного набору даних вибираються випадково обрані записи з поверненням для створення набору даних для навчання кожного дерева.

2) Побудова дерев рішень. Для кожної вибіркової підвибірки будується окреме дерево рішень. Під час побудови дерева вибираються випадково обрані

ознаки (атрибути) з навчального набору даних для розгляду на кожному рівні дерева.

3) Голосування багатьма деревами. Після побудови всіх дерев Random Forest проводиться класифікація або регресія кожного вхідного прикладу (спостереження) для кожного дерева. У випадку класифікації для визначення кінцевого класу береться більшість голосів дерев. У випадку регресії - береться середнє значення прогнозів дерев.

4). Зменшення перенавчання. Один з ключових принципів Random Forest - зменшення перенавчання. Це досягається завдяки випадковому вибору ознак на кожному рівні дерева та використанню багатьох дерев, що дозволяє зменшити вплив шуму та варіативності даних.

5). Оцінка важливості ознак. Random Forest надає можливість оцінювати важливість ознак, використовуючи інформацію з дерев.

Переваги Random Forest включають високу точність, здатність працювати з великими наборами даних та здатність визначати важливість ознак. Він також добре підходить для роботи з даними, що містять багато категоріальних ознак. **Основним недоліком** може бути більший час навчання порівняно з деякими іншими алгоритмами. Random Forest має кілька переваг у порівнянні з іншими методами машинного навчання, особливо у контексті розпізнавання кібератак та в інших областях. Ось деякі з них:

1). Висока точність. Random Forest відомий своєю високою точністю в прогнозуванні і класифікації. Він може добре впоратися з різними видами даних та завдань, включаючи класифікацію атак та виявлення аномалій.

2). Важливість признаків. Random Forest надає оцінки важливості признаків, що дозволяє ідентифікувати найбільш впливові атрибути в даних. Це корисно для розуміння, які фактори важливі для рішення.

3). Випадковість та стійкість до перенавчання. Випадковий вибір підмножини даних і признаків під час побудови кожного дерева допомагає уникнути перенавчання. Це робить метод стійким до шуму в даних.

4). Взаємодія з категоріальними та числовими признаками. Random Forest ефективно працює з різними типами даних, включаючи категоріальні та числові признаки, без потреби в передпроцесінгу.

5). Можливість роботи з великою кількістю даних. Random Forest може ефективно обробляти великі обсяги даних без значного погіршення швидкості роботи. Це важливо в контексті кібербезпеки, де потрібно аналізувати велику кількість мережевого трафіку і журналів подій.

6) Стійкість до втрати даних. Random Forest може продовжувати працювати і навчатися навіть в тому випадку, якщо в даних є пропуски чи втрата частини інформації.

7) Вбудована оцінка важливості ознак. Random Forest автоматично рахує важливість ознак, що допомагає відокремити суттєві атрибути від менш важливих.

8) Добра відтворюваність результатів. Результати Random Forest відтворюються, що робить його надійним і можливим для використання в наукових дослідженнях.

9) Можливість виявлення взаємодії між признаками: Random Forest може виявляти складні взаємодії між признаками, що може бути важливим у випадку виявлення нових атак.

10) Широкий спектр застосувань. Random Forest може бути використаний в різних галузях, включаючи кібербезпеку, медицину, фінанси, рекомендаційні системи та інші.

Хоча Random Forest має численні переваги, важливо також враховувати його обмеження і вибирати метод машинного навчання відповідно до конкретного завдання та характеристик даних. Random Forest є потужним і дуже ефективним для багатьох завдань машинного навчання, але він також має деякі недоліки та обмеження:

1) Час навчання та передбачення. Побудова великої кількості дерев може вимагати багато обчислювальних ресурсів і тривалого часу. Особливо це стосується великих наборів даних та глибоких дерев.

2). Складність інтерпретації. Модель Random Forest складна для інтерпретації, оскільки вона використовує багато дерев і може бути складною для розуміння, яким чином приймається кожне окреме рішення.

3). Велика кількість гіперпараметрів. У Random Forest є кілька гіперпараметрів, які потребують налаштування, такі як кількість дерев, глибина дерев, критерії розгалуження і інші. Неправильна настройка може призвести до перенавчання або недонавчання.

4). Обмеженість для регресії. Random Forest не завжди добре підходить для завдань регресії, особливо якщо залежність між ознаками та цільовою змінною не лінійна або складна.

5). Велика кількість дерев може призвести до перенавчання. Незважаючи на стійкість до перенавчання, яку надає Random Forest, якщо використовувати дуже велику кількість дерев, може виникнути перенавчання на тренувальних даних, що призведе до гірших результатів на нових даних.

6). Необхідність у великій кількості даних. Щоб Random Forest давав гарні результати, зазвичай потрібно багато даних. Для деяких завдань, особливо в області глибокого навчання, це може бути обмеженням.

7). Важко робити екстраполяцію за межі діапазону даних. Random Forest не завжди добре підходить для прогнозування поза діапазоном даних, на яких він був навчений.

Незважаючи на ці недоліки, метод Random Forest залишається одним з найпопулярніших та ефективних алгоритмів машинного навчання, і в багатьох випадках, правильно налаштований та знацьований, він може забезпечувати вражаючі результати.

Використання Random Forest для розпізнавання кібератак є актуальним напрямом досліджень у галузі кібербезпеки. Дослідники та професіонали в цій галузі вивчають можливість застосування ансамблю дерев рішень, таких як Random Forest, для виявлення кібератак та виявлення аномальних поведінок в мережі.

Наведемо область застосування Random Forest:

1) Виявлення фішингу. Random Forest використовується для аналізу різних ознак, що характеризують фішингові веб-сайти та повідомлення, щоб визначити їх як легітимні або підозрілі.

2) Виявлення вторгнень в системи. Random Forest може бути використаний для виявлення аномальних дій в комп'ютерних системах та мережах, що може свідчити про вторгнення.

3) Аналіз мережевого трафіку. Random Forest дозволяє виявляти аномалії в мережевому трафіку, такі як DDoS-атаки, атаки на переповнення буфера та інші загрози.

4) Класифікація зловмисних програм. Дослідження використання Random Forest для виявлення шкідливого програмного забезпечення, такого як віруси, черв'яки та троянські коні.

5) Виявлення аномальних поведінок користувачів. Random Forest може бути застосований для виявлення аномальних дій користувачів в системах, що може свідчити про несанкціонований доступ або порушення безпеки.

6) Прогнозування загроз (threat prediction). Використання Random Forest для прогнозування майбутніх загроз та атак на основі аналізу історичних даних та паттернів.

7) Системи виявлення зловмисності в реальному часі. Застосування Random Forest у реальному часі для виявлення зловмисності та аномалій у мережах та системах.

Дослідження та розробка алгоритмів, які використовують Random Forest для кібербезпеки, є активною галуззю, оскільки цей метод може виявляти складні та неочікувані атаки, надійно виявляти аномальність та допомагати в реагуванні на кібер загрози. Random Forest може бути використаний для виявлення кібератак у комп'ютерних системах і мережах завдяки своїм можливостям аналізу даних та класифікації. Ось способи, якими він може бути застосований у цьому контексті:

1) Класифікація атак та нормального трафіку. Random Forest може бути навчений на наборі даних, який включає в себе інформацію про нормальний

мережевий трафік та відомі види кібератак. Модель будує ансамбль дерев рішень, які визначають, чи є певний трафік аномальним (кібератака) чи нормальним. Це дозволяє виявляти незвичайні або підозрілі активності в реальному часі.

2) Виявлення аномалій. Окрім класифікації, Random Forest може використовуватися для виявлення аномалій. Модель навчається на нормальних даних та створює базовий профіль для нормальної активності. Всі події, які суттєво відрізняються від цього профілю, вважаються аномаліями і можуть вказувати на кібератаки або інші загрози.

3) Виявлення нових атак. Random Forest може бути корисним для виявлення нових типів кібератак, які раніше не були відомі. Оскільки модель аналізує дані та взаємодіє з ансамблем дерев, вона може виявити аномалії в новому трафіку чи подіях, що може вказувати на нові загрози.

4) Виявлення вторгнень (IDS). Random Forest може бути використаний для створення системи виявлення вторгнень (IDS), яка спрямована на виявлення незаконного доступу до системи. Він може аналізувати вхідні дані та сповіщати про можливі вторгнення.

5) Аналіз журналів подій (Log Analysis). В багатьох організаціях реєструються журнали подій, що містять інформацію про активність користувачів та системи. Random Forest може бути використаний для аналізу цих журналів та виявлення незвичайних подій, які можуть вказувати на кібератаки.

6) Виявлення спаму та фішингу. У сфері електронної пошти та веб-сайтів Random Forest може бути використаний для виявлення спаму та фішингових повідомлень, аналізуючи текстові та структурні ознаки листів чи сторінок.

7) Захист від зловмисних програм. Random Forest може аналізувати характеристики програм та файлів, що виконуються, та виявляти зловмисні програми чи віруси на комп'ютерах.

Random Forest демонструє високу ефективність у виявленні кібератак завдяки своїм здатностям до аналізу даних та виявлення незвичайних патернів.

Важливо відпрацьовувати параметри та підготовку даних для досягнення оптимальних результатів.

1.4. Висновки до розділу

Проблематика розпізнавання кібератак є надзвичайно актуальною і важливою в наш час. Швидкість та розмір зростання кіберзагроз зробили цю проблему критично важливою для багатьох організацій, користувачів та навіть країн. Кібератаки можуть завдати серйозних фінансових, економічних, технічних та політичних збитків, а також загрожувати безпеці та конфіденційності даних. Тому розпізнавання кібератак стало критичною складовою сфери кібербезпеки та вимагає постійного дослідження та розробки нових технологій та методів. Проведений аналіз літературних джерел підкреслює необхідність постійного удосконалення та розширення інструментарію для розпізнавання кібератак, щоб забезпечити безпеку в цифровому світі та мінімізувати їхні наслідки. Було проведено порівняльний аналіз методів розпізнавання кібератак, а також аналіз математичних моделей та застосування їх у сучасній кібербезпеці. Методи розпізнавання кібератак, розглянуті в дослідженні, включають в себе різноманітні техніки, які базуються на аналізі даних та використовують принципи для виявлення аномальної чи ворожої активності: нейронні мережі, методи статистики, алгоритми машинного навчання - для розв'язання задач класифікації, кластеризації, виявлення аномалій. Досліджено та надано опис методу Random Forest, що є ефективним у вирішенні задачі розпізнавання кібератак та аномалій в цифровому середовищі. Основні переваги методу Random Forest - це висока точність, стійкість до перевантаження та здатність до роботи з великими обсягами даних та багатьма атрибутами. Крім того, він дозволяє оцінити важливість кожного атрибуту у процесі прийняття рішення, що може бути корисним для аналізу кібератак.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ЗАСТОСУВАННЯ RANDOM FOREST ДЛЯ РОЗПІЗНАВАННЯ КІБЕРАТАК

2.1 Модель визначення вхідних параметрів

Модель визначення вхідних параметрів - це процес вибору та формування набору ознак (features), які будуть використовуватися як вхідні дані для методу розпізнавання або прогнозування. Вибір вхідних параметрів є ключовим етапом в розробці будь-якої моделі машинного навчання чи статистичного аналізу, оскільки від них залежить якість та ефективність результуючої моделі. Нижче наведено кроки та методи, які можна використовувати для визначення вхідних параметрів.

1. Розуміння задачі. Перед визначенням вхідних параметрів важливо ретельно розібратися у задачі. Розуміння цілей та контексту задачі допоможе нам визначити, які ознаки можуть бути важливими для прогнозування чи класифікації.

2. Експертна думка. Залучення експертів у відповідній області може допомогти нам визначити ключові ознаки. Експерти можуть запропонувати знання про те, які параметри можуть бути важливими для вирішення задачі.

3. Експлоративний аналіз даних. Вивчення даних за допомогою графіків, статистичних метрик та кореляцій може розкрити зв'язки між різними ознаками. Це може допомогти визначити, які параметри мають суттєвий вплив на результати.

4. Відбір ознак. Використання методів відбору ознак, таких як взаємна інформація, приріст інформації, коефіцієнт кореляції тощо, може допомогти вибрати найбільш важливі параметри.

5. Збалансованість та зайві ознаки. Важливо забезпечити баланс між кількістю ознак та їхньою важливістю. Зайві чи корельовані ознаки можуть призвести до перенавчання, тому їхній вибір потрібно ретельно взважити.

6. Нормалізація та стандартизація. Ознаки можуть мати різний масштаб та розподіл. Нормалізація (приведення значень до одного масштабу) та

стандартизація (вирахування середнього значення та стандартного відхилення) можуть бути важливими для коректної роботи деяких методів машинного навчання.

Бази даних KDD Cup 99 і NLS-KDD є двома різними наборами даних, які використовуються для вивчення виявлення вторгнень в комп'ютерних мережах. KDD Cup 99 - це один із найвідоміших та найчастіше використовуваних наборів даних для вивчення вторгнень. Він був створений для конкурсу з виявлення вторгнень KDD Cup 1999. NLS-KDD - це розширена версія KDD Cup 99, яка містить додаткові дані та призначена для поліпшення якості навчання моделей для виявлення вторгнень.

Ось деякі основні відмінності між цими базами даних.

Кількість даних

• KDD Cup 99. Оригінальний набір містить приблизно 5 мільйонів записів, але він має деякі проблеми, такі як несбалансованість класів та відсутність деяких типів вторгнень.

• NLS-KDD. Розширена версія містить більше записів, які дозволяють створити більш точні та надійні моделі.

Категорії вторгнень

• KDD Cup 99. Зазвичай використовується для розпізнавання бінарних класів (нормальні з'єднання та вторгнення).

• NLS-KDD. Містить розширений перелік категорій вторгнень, що дозволяє навчати моделі для розпізнавання більш різноманітних вторгнень.

Розмірність даних

• KDD Cup 99. Містить обмежену кількість ознак, що може обмежити можливості моделі.

• NLS-KDD. Має більшу кількість ознак, які можуть забезпечити більш точне навчання моделей.

Додаткові дані

• KDD Cup 99. Він включає в себе тільки основні характеристики з'єднань та типи вторгнень.

• NLS-KDD. Містить додаткові дані, які можуть бути корисними для додаткового аналізу та вивчення.

KDD CUP 99

При розробці моделі для виявлення вторгнень на основі бази даних KDD Cup 99 важливо ретельно обрати вхідні параметри для навчання. Визначення вхідних параметрів для цієї задачі:

1. Числові параметри:

• `duration`: тривалість з'єднання.

• `src_bytes`: кількість байтів, відправлених від джерела до призначення.

• `dst_bytes`: кількість байтів, відправлених від призначення до джерела.

• `count`: загальна кількість з'єднань до системи протягом певного часового періоду.

• `srv_count`: кількість з'єднань до окремого сервісу протягом певного часового періоду.

• `serror_rate`: відсоток з'єднань, які мають 'SYN' eppop.

• `srv_serror_rate`: відсоток з'єднань до сервісу, які мають 'SYN' eppop.

• `rerror_rate`: відсоток з'єднань, які мають 'REJ' eppop.

• `srv_rerror_rate`: відсоток з'єднань до сервісу, які мають 'REJ' eppop.

2. Категоріальні параметри:

• `protocol_type`: тип протоколу (tcp, udp, icmp тощо).

• `service`: вид сервісу (http, ftp, ssh тощо).

• `flag`: статус прапорця (SF, S0, REJ тощо).

3. Бінарні параметри:

• Наявність або відсутність певних подій, наприклад, 'is_guest_login', 'is_host_login' тощо.

4. Інші параметри:

• Інші характеристики, такі як час доби, день тижня тощо, які можуть бути важливими для виявлення вторгнень.

5. Цільова змінна:

• Помічаємо, які з'єднання є нормальними (не вторгненнями) та які

вважаються вторгненнями. Це буде нашою цільовою змінною, яку ми намагаємося передбачити.

6. Нормалізація та кодування:

• Нормалізуємо числові параметри для приведення їхніх значень до одного масштабу.

• Застосовуємо One-Hot Encoding для категоріальних та бінарних параметрів, щоб перетворити їх у бінарні вектори.

Після вибору та підготовки вхідних параметрів можна використовувати їх для навчання різних моделей машинного навчання (в приклад, Random Forest, SVM, або нейронні мережі) для виявлення вторгнень в систему.

NLS-KDD:

При створенні моделі для виявлення вторгнень на основі бази даних NLS-KDD, яка є розширеною версією датасету KDD Cup 99, важливо ретельно обирати параметри для навчання. Для цієї задачі визначаються наступні типи параметрів:

1. Числові параметри

• `duration`: тривалість з'єднання.

• `src_bytes`: обсяг байтів, відправлених від джерела до призначення.

• `dst_bytes`: обсяг байтів, відправлених від призначення до джерела.

• `count`: загальна кількість з'єднань до системи протягом певного періоду.

• `srv_count`: кількість з'єднань до конкретного сервісу протягом певного періоду.

• `serror_rate`: відсоток з'єднань із помилкою 'SYN'.

• `srv_serror_rate`: відсоток з'єднань до сервісу із помилкою 'SYN'.

• `rerror_rate`: відсоток з'єднань із помилкою 'REJ'.

• `srv_rerror_rate`: відсоток з'єднань до сервісу із помилкою 'REJ'.

2. Категоріальні параметри

• `protocol_type`: тип протоколу (tcp, udp, icmp і т.д.).

• `service`: тип сервісу (http, ftp, ssh і т.д.).

• `flag`: статус прапорця (SF, S0, REJ і т.д.).

3. Бінарні параметри

• Наявність або відсутність певних подій, таких як 'is_guest_login', 'is_host_login' і т.д.

4. Інші параметри

• Додаткові характеристики, такі як час доби, день тижня і т.д., які можуть бути важливими для виявлення вторгнень.

5. Цільова змінна

• Позначає, які з'єднання є нормальними (без вторгнень) та які вважаються вторгненнями. Це є цільовою змінною, яку намагаємося передбачити.

6. Нормалізація та кодування

• Числові параметри нормалізуються для приведення їхніх значень до єдиного масштабу.

• Застосовується One-Hot Encoding для категоріальних та бінарних параметрів, перетворюючи їх у бінарні вектори.

Після відбору та підготовки вхідних параметрів можна використовувати їх для навчання різних моделей машинного навчання (наприклад, Random Forest, SVM або нейронні мережі) для виявлення вторгнень у систему. Цей підхід дозволяє виявляти неправомірні дії та забезпечує безпеку мережі або системи.

Обираючи між двома базами даних, «KDD Cup 99» і «NLS-KDD», важливо враховувати конкретні потреби нашого дослідження та завдання. Якщо точність та різноманітність вторгнень є важливим, то «NLS-KDD» може бути кращим вибором, оскільки він надає більше різноманітних та деталізованих даних. Слід також враховувати розмір даних та обробку невідповідностей в даних для обраної бази даних. Отже, більш детально розгянемо базу даних NLS-KDD, далі будемо її використовувати.

NLS-KDD включає в себе оригінальні дані KDD Cup 1999, а також нові марковані дані, які були позначені більш точно для вторгнень. Цей набір даних був створений для того, щоб вирішити деякі проблеми оригінального KDD Cup 1999, такі як несбалансованість класів, низька якість маркування та відсутність деяких типів вторгнень. NLS-KDD став дуже популярним у спільноті

дослідників інформаційної безпеки для експериментів та вивчення методів виявлення вторгнень в комп'ютерних мережах. Цей набір даних містить реальні дані про мережевий трафік, які можуть бути використані для навчання та тестування моделей машинного навчання в галузі інформаційної безпеки.

В датасеті NLS-KDD, який використовується для вивчення виявлення вторгнень, зазвичай наявні різні параметри, які можна використовувати як вхідні ознаки для методів розпізнавання кібератак. Деякі з найпоширеніших параметрів, які можна розглядати як вхідні ознаки для моделей машинного навчання, включають такі категорії:

1. Статистичні параметри мережевого трафіку

• Кількість пакетів (packets)

• Кількість байтів (bytes)

• Тривалість з'єднання (duration)

• Кількість унікальних з'єднань (unique_connections)

2. Категоріальні параметри, пов'язані із з'єднаннями

• Протокол (protocol_type)

• Порт входу (service)

• Флаги (flags)

3. Параметри, пов'язані із спробами вторгнень

• Тип вторгнення (label)

4. Статистика пакетів для конкретних протоколів

• Кількість ICMP пакетів

• Кількість TCP пакетів

• Кількість UDP пакетів

5. Інші параметри

• Час доби (великий вплив на активність)

• IP-адреса джерела та призначення

• Розмір пакетів

Ці параметри можна використовувати як ознаки для навчання моделей машинного навчання з метою виявлення кібератак. Перед використанням даних

слід їх аналізувати та вибирати найбільш інформативні та корисні параметри для конкретного завдання виявлення вторгнень. Цей вибір може варіюватися в залежності від типу аналізу, який ми хочемо виконати (в приклад, виявлення аномалій чи класифікація вторгнень).

CTU-13:

Мета створення цього набору даних полягала в охопленні значного обсягу реального трафіку ботнету, який був змішаний із звичайним і фоновим трафіком. Набір даних CTU-13 складається з тринадцяти захоплень (сценаріїв), представляючи різні зразки ботнетів. Загалом налічується 349 архівів із різноманітними зразками шкідливого програмного забезпечення. В кожному сценарії автори виконували конкретне шкідливе програмне забезпечення, яке використовувало різні протоколи та виконувало різноманітні дії. Кожен сценарій був записаний у файл pcap, що містить усі пакети трьох типів трафіку. Ці файли pcap були оброблені для отримання іншої інформації, такої як NetFlows, WebLogs і т.д. У першому аналізі набору даних CTU-13, описаному та опублікованому у статті "Емпіричне порівняння методів виявлення бот-мереж", використовувалися односпрямовані NetFlows для представлення трафіку та нанесення міток.

Параметри, які подаються на вхід методу Random Forest для розпізнавання кібератак з використанням бази даних CTU-13, можуть бути ретельно вибрані, щоб забезпечити оптимальну ефективність моделі. Параметри, які можна розглядати для використання в Random Forest для аналізу даних CTU-13:

n_estimators:

• Кількість дерев, які будуть побудовані в ансамблі. Більше дерев може покращити стійкість та точність, але при занадто великому значенні може виникнути перенавчання.

max_depth:

• Максимальна глибина кожного дерева. Встановлення обмежень на глибину може допомогти уникнути перенавчання та збільшити швидкодію моделі.

min_samples_split та min_samples_leaf:

• Мінімальна кількість зразків, необхідних для розділення вузла та для листка. Ці параметри допомагають управляти процесом розділення та можуть зменшити ризик перенавчання.

max_features:

• Кількість ознак, які слід розглядати при кожному розділенні. Вибір "auto" або "sqrt" дозволяє вибирати квадратний корінь від загальної кількості ознак, що є типовим вибором.

bootstrap:

• Чи використовувати bootstrap samples при побудові дерев. Встановлення цього параметра в True дозволяє використовувати bootstrap samples, що є стандартним варіантом для Random Forest.

class_weight:

• Встановлення ваг класів, особливо корисне, якщо ви працюєте з набором даних, де класи незбалансовані.

Це лише загальний перелік параметрів. Важливо експериментувати та налаштовувати ці параметри з урахуванням конкретності завдання та характеристик даних CTU-13. Використання методів крос-валідації може допомогти знаходженню оптимальних значень параметрів.

Нормалізація даних у базі даних CTU-13 є важливим етапом для стандартизації та оптимізації вхідних даних перед їх використанням у моделях машинного навчання або аналітиці. Етапи нормалізації для бази даних CTU-13:

1. Видалення непотрібних атрибутів:

• Перевіряємо атрибути бази даних та видаляємо непотрібні, чи непотрібно збільшуючі розмір бази даних.

2. Обробка пропущених значень:

• Якщо база даних містить пропущені значення, вирішуємо, як їх краще заповнити або видалити, щоб не спотворювати результати аналізу.

3. Кодування категоріальних змінних:

• Якщо база даних містить категоріальні змінні (наприклад, типи атак чи

пристроїв), використовуємо методи, такі як one-hot encoding або label encoding, для перетворення їх у числовий формат.

4. Масштабування числових атрибутів:

• Застосовуємо масштабування до числових атрибутів для забезпечення однакового діапазону значень. Можна використати стандартизацію (наприклад, з використанням z-оцінки) або нормалізацію (наприклад, мін-макс масштабування).

5. Робота з часовими атрибутами:

• Якщо в базі даних є часові атрибути, розглядаємо можливість розбиття їх на окремі складові (рік, місяць, день, година тощо), щоб врахувати важливість різних періодів часу.

6. Перетворення текстових даних:

• Якщо є текстові атрибути, використовуємо методи векторизації (наприклад, TF-IDF або Word Embeddings) для перетворення їх у числовий формат.

7. Перевірка балансу класів (для класифікації):

• Враховуємо баланс класів, щоб уникнути перекосу в моделі. Можливо, застосувати методи, такі як oversampling чи undersampling.

Ці етапи нормалізації допомагають підготувати дані для подальшого використання у моделях машинного навчання та аналітиці.

MCFP:

Ці набори даних були отримані в Університеті CTU в Чеській Республіці. У кожній папці, де зберігається конкретний набір даних, міститься розширена інформація, така як файли NetFlow, HTTP-журнали та дані DNS. Ці файли періодично оновлюються при зборі нової інформації. Вони розв'язують проблему розрізнення між клієнтом і сервером, і включають в себе більше інформації, а також мають більш детальні мітки. Папка, де зберігається кожен набір даних, також містить іншу корисну інформацію, таку як файли NetFlow, HTTP-журнали та дані DNS, які регулярно оновлюються при отриманні нових даних.

Звичайний датасет складається з файлів .pcap, які записані на хості, включає опис сценарію датасету, а також IP-адресу хоста та сервера. Крім того, датасет може містити кілька файлів статистичної інформації, таких як:

• capture20110810.pcap: Запис усього трафіку (фонового, звичайного та ботнету). Цей файл не публікується через вміст приватної інформації про користувачів мережі. Захоплено на головному маршрутизаторі університетської мережі.

• botnet-capture-20110810-neris.pcap: Запис тільки трафіку ботнету. Цей файл публікується і був захоплений на інтерфейсі зараженої віртуальної машини.

• capture20110810.pcap.netflow.labeled: Файл із мережевими потоками, створеними за допомогою односпрямованого argus.

• Bro: Папка із файлами для системи виявлення вторгнень Bro-IDS.

Детальні-двонаправлені-потоки-мітки.html: Графічна html-сторінка, створена за допомогою HTTP-запитів CapTipper у захопленні.

• Json: Файл, необхідний для html-сторінки.

• *.binetflow.2format

2.2 Математичне забезпечення методу

Метод Random Forest базується на концепції ансамблювання (ensemble learning) та використовує декілька дерев рішень (decision trees) для прийняття рішення. Основні математичні аспекти цього методу включають в себе випадковий вибір підвибірки даних і випадковий вибір підмножини ознак при побудові кожного дерева рішень, а також комбінування результатів для отримання кінцевого прогнозу або класифікації.

Ось деякі математичні аспекти методу Random Forest:

• **Випадковий вибір підвибірки даних (Bootstrap Sampling):**

Для кожного дерева рішень створюється випадкова підвибірка (вибірка з поверненням) з вихідного набору даних. Це означає, що одни й ті ж самі елементи можуть потрапити до підвибірки декілька разів, а інші можуть бути пропущені.

- **Випадковий вибір підмножини ознак (Random Feature Selection):**

При кожній розгалужені дерева рішень випадковим чином вибираються лише певні ознаки з підмножини всіх доступних ознак. Це допомагає забезпечити різноманітність між деревами та уникнути перенавчання.

- **Побудова Дерев Рішень (Decision Trees):**

На основі вибірки даних та вибраних ознак для кожного дерева рішень будується дерево рішень. При кожному вузлі дерева вибирається оптимальна ознака для розділення даних на дві підгрупи.

- **Голосування або Усереднення Результатів:**

У випадку класифікаційних задач, кожне дерево голосує за клас, до якого належить вхідний приклад, і результати комбінуються за допомогою голосування більшості (для бінарних класифікаційних задач) або за допомогою усереднення класів, які вибираються кожним деревом. У регресійних задачах результати кожного дерева усереднюються для отримання остаточного результату.

- **Важливість Ознак (Feature Importance)**

Random Forest надає можливість визначити важливість кожної ознаки. Це можна зробити, спостерігаючи, як змінюється якість моделі при заміщенні кожної ознаки.

Ці математичні аспекти дозволяють Random Forest бути потужним і надійним алгоритмом для класифікації та регресії в різних застосуваннях машинного навчання.

Random Forest використовує декілька математичних понять та алгоритмів для свого функціонування. Основні математичні концепції включають:

1. Бутстреп-вибірка:

Використовується для випадкового вибору наборів даних з повторенням для навчання кожного дерева у випадковому лісі.

2. Дерева рішень:

Багато дерев приймають рішення на основі рекурсивного розгалуження та критеріїв якості, таких як індекс Джині чи інші метрики, які вимірюють чистоту

розгалуження.

3. Випадковий вибір ознак (Feature Selection)

Використовується для вибору випадкового піднабору ознак на кожному рішенні вузла дерева.

4. Голосування та середнє значення

Для класифікації випадковий ліс використовує голосування для визначення кінцевого класу, що набрав найбільше голосів. Для регресії використовується середнє значення прогнозів дерев.

5. Нормалізація

Використовується для масштабування ознак та приведення їх до одного масштабу.

Ці математичні поняття допомагають методу Random Forest забезпечити різноманітність та стабільність моделі, а також зменшити ймовірність перенавчання. Рандомізований підхід до вибору даних та ознак, а також ансамбль дерев рішень, роблять цей метод потужним і ефективним для класифікації та регресії.

Розглянемо математичні формули для двох ключових аспектів, які використовуються в Random Forest: дерева рішень та бутстреп-вибірки.

Дерево рішень:

1. Критерій вибору поділу вузла:

• Нехай D - вибірка даних у вузлі, а $X_j - j$ – та ознака. Критерій вибору поділу може бути визначений, наприклад, за допомогою коефіцієнта Джині:

$$G(D) = 1 - \sum_i p_i^2 \tag{2.1}$$

де p_i – ймовірність того, що об'єкт вибірці належить класу i.

2. Формула для розрахунку індексу вузла:

• Під час побудови дерева важливо визначити оптимальний спосіб поділу вузла. Формула може мати вигляд:

$$Q(D, i, t) = G(D) - \frac{m_L}{m} G(D_L) - \frac{m_R}{m} G(D_R) \tag{2.2}$$

де D_L та D_R – підвибірки даних, що розділені за умовою $X_j \leq t$, m – загальна

кількість прикладів в вузлі, m_L та m_R - кількість прикладів у лівому та правому підвузлі відповідно.

Бутстреп-вибірка:

Бутстреп-вибірка використовується для вибору випадкового піднабору даних для навчання кожного дерева в Random Forest.

1. Формула для обчислення ймовірності вибору кожного прикладу:

• Під час бутстреп-вибірки кожен приклад має ймовірність $1 - \frac{1}{n}$ бути вибраним, де *n* – кількість прикладів у вибірці:

$$P(\text{вибір прикладу}) = 1 - \frac{1}{n} \quad (2.3)$$

• Потім ймовірність того, що конкретний приклад не буде вибраним (його вилучено) - це:

$$P(\text{не вибір прикладу}) = \frac{1}{n} \quad (2.4)$$

Ці формули допомагають визначити, як будуються дерева рішень та як створюються бутстреп-вибірки для навчання кожного дерева в Random Forest.

Розпізнавання кібератак за допомогою Random Forest зазвичай включає в себе класифікацію прикладів на "нормальні" (не атаки) та "вторгнення". Для цього можна використовувати математичні формули, що відображають процес побудови та використання моделі Random Forest. Нижче представлені загальні концепції.

1. Будування окремого дерева рішень

При побудові кожного дерева рішень T_i на основі бутстреп-вибірки D_i випадково вибирається підмножина ознак F_i. Ми можемо використовувати індекс *«j»* для представлення конкретної ознаки та порог *«t»* для визначення розділення вузла. Критерій якості (наприклад, коефіцієнт Джині або ентропія) визначається як *«Q(D,j,t)»*.

$$Q(D, i, t) = \frac{m_L}{m} G(D_L) + \frac{m_R}{m} G(D_R) \quad (2.5)$$

де D – вибірка даних у вузлі;

D_L та D_R – підвибірки даних, що розділені за умовою $X_j \leq t$;

G - критерій якості (наприклад, Джині або ентропія);

m - загальна кількість прикладів в вузлі;

• m_L та m_R - кількість прикладів у лівому та правому підвузлі відповідно.

2. Прогноз для ансамблю

Прогноз для нового прикладу

«x» визначається як мода (найчастіше зустрічаючеся значення) прогнозів окремих дерев:

$$\hat{y} = mode\big(T_1(x), T_2(x), \dots, T_n(x)\big), \tag{2.6}$$

де $\hat{y}$ – прогноз ансамблю;

$T_i(x)$ – прогноз i-го дерева.

Таким чином, за допомогою цих формул можна виразити процес побудови та використання Random Forest для завдання розпізнавання кібератак. Важливо враховувати, що конкретні деталі можуть варіюватися в залежності від конкретної реалізації та налаштувань.

<u>Випадковий вибір ознак (Feature Selection)</u>

Математичне забезпечення для випадкового вибору ознак в методі Random Forest представлено наступним чином:

1. Кількість ознак для врахування при побудові кожного вузла:

Нехай m – загальна кількість ознак у наборі даних.

Кількість ознак для врахування при побудові кожного вузла може бути визначена як k, де k – це параметр, який може бути встановлений користувачем:

$$k = \sqrt{m} \text{ або } k = \frac{m}{3} \tag{2.7}$$

2. Випадковий вибір ознак при побудові кожного вузла:

При побудові кожного вузла випадковим чином обирається k ознак із загальної кількості m.

$$\text{Вибір ознак} = rand() \times m \ , \tag{2.8}$$

де $rand() -$ випадкове число між 0 і 1.

Ці вирази забезпечують введення елементу випадковості в процес

побудови кожного дерева випадкового лісу, що робить його стійким до перенавчання та ефективним для роботи з різноманітними даними.

Голосування та середнє значення

В контексті методу випадкового лісу часто використовують голосування для класифікаційних задач і середнє значення для задач регресії.

1. Голосування для класифікації

Нехай у випадковому лісі є *T* дерев, і кожне дерево видає прогноз класу c_i для вхідних даних.

Голосування визначає прогнозований клас як клас, який отримав найбільше голосів серед усіх дерев.

Формула для голосування:

$$\text{Прогнозований клас} = argmax_c \sum_{i=1}^{T} II\,(c_i = c) \qquad (2.9)$$

де II(•) - індикаторна функція, яка дорівнює 1, якщо умова вказана в дужках є правдивою, і 0 в іншому випадку.

2. Середнє значення для регресії

У випадковому лісі кожне дерево видає числовий прогноз для вхідних даних.

Прогнозоване значення для регресії обчислюється як середнє значення прогнозів усіх дерев.

$$\text{Прогнозоване значення} = \frac{1}{T}\sum_{i=1}^{T} y_i \qquad (2.10)$$

де y_i – прогноз регресії, який надає *i*-те дерево.

У цих формулах *T* - це кількість дерев у випадковому лісі. Ці прості операції голосування та обчислення середнього значення дозволяють використовувати ансамбль дерев для класифікаційних і регресійних завдань.

Нормалізація

Нормалізація в контексті машинного навчання і обробки даних використовується для приведення атрибутів чи функцій до стандартних діапазонів або норм. Декілька популярних методів нормалізації включають мінімакс нормалізацію, z-нормалізацію (стандартизацію) і нормалізацію за сумою.

1. Міні-макс нормалізація

Призначена для перетворення значень атрибутів у заданий діапазон, зазвичай [0, 1].

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (2.11)$$

де x' - нормалізоване значення, x – оригінальне значення, $\min(x)$ – мінімальне значеннння атрибута, $\max(x)$ – максимальне значення атрибута.

2. Z-нормалізація (стандартизація):

Призначена для перетворення значень атрибутів так, щоб вони мали середнє значення 0 і стандартне відхилення 1.

$$z = \frac{x - \mu}{\sigma} \quad (2.12)$$

де z – стандартизоване значення;

x – оригінальне значення;

μ – середнє значення атрибута;

σ – стандартне відхилення атрибута.

3. Нормалізація за сумою

Призначена для приведення значень атрибутів так, щоб їх сума була фіксованою (наприклад, 1).

$$x' = \frac{x}{\sum_i x_i} \quad (2.13)$$

де x_i – нормалізоване значення;

x_i – оригінальне значення атрибута, а сума береться по всім значенням атрибута.

Ці методи допомагають стандартизувати або нормалізувати дані, зробити їх більш придатними для алгоритмів машинного навчання та забезпечити більш стійке та ефективне навчання моделей.

У методі Random Forest для визначення вихідних параметрів виявлення кібератак з використанням бази даних CTU-13 вхідні параметри включають атрибути чи ознаки, які визначають кожен елемент даних. Нижче описано, як використовуються та оброблюються ці вхідні параметри:

1. Вибір атрибутів:

Визначення, які атрибути чи ознаки з бази даних CTU-13 будуть використані для навчання моделі. Це може включати інформацію про мережевий трафік, характеристики пакетів, дані про час і т.д.

2. Нормалізація:

Якщо атрибути мають різні діапазони значень, вони можуть бути нормалізовані, наприклад, шляхом стандартизації (застосування z-оцінки) або мін-макс масштабування, щоб усі атрибути були в однаковому діапазоні значень.

3. Робота з категоріальними атрибутами:

Якщо база даних містить категоріальні атрибути, такі як типи атак або пристроїв, потрібно вирішити, як їх перетворити у формат, зрозумілий для моделі. Це може включати one-hot encoding, label encoding або інші методи.

4. Обробка пропущених значень:

Якщо в базі даних є пропущені значення, потрібно вирішити, як їх краще заповнити або видалити, щоб це не впливало на навчання моделі.

5. Відокремлення міток та ознак:

Розділення даних на мітки (цільова змінна, наприклад, чи є атака чи ні) та ознаки (вхідні параметри), які будуть використані для навчання моделі.

6. Використання Random Forest:

Підготовані дані подаються на вхід в ансамбль дерев Random Forest, який будує кілька різних дерев рішень, використовуючи випадкові підвибірки даних та випадкові підмножини ознак для кожного дерева.

7. Прогнозування:

Коли модель Random Forest навчена, вона може бути використана для прогнозування нових даних, включаючи визначення того, чи є вони частиною кібератаки або ні.

8. Оцінка результатів:

Результати визначення кібератак можуть бути оцінені за допомогою метрик якості, таких як точність, чутливість, специфічність і інші.

Загальна ідея полягає в тому, щоб ефективно використовувати різні атрибути та їх комбінації для навчання моделі та визначення кібератак на основі цих даних.

Математичний опис використання методу Random Forest для виявлення кібератак з використанням параметрів CTU-13 включає опис процесу навчання та прогнозування моделі. Більшість процесів у машинному навчанні описуються за допомогою алгоритмів та формул, нижче подано загальний опис:

1. Навчання (тренування) Random Forest

Позначимо набір даних CTU-13 як D

де D містить пари (X_i, y_i), де X_i - вектор ознак (атрибутів) для і-го спостереження, а y_i - мітка класу (цільова змінна), яка вказує, чи є це спостереження частиною кібератаки ($y_i = 1$) або ні ($y_i = 0$).

Кожен Random Forest має параметри, такі як кількість дерев ($n_{estimators}$), максимальна глибина дерев (max_{depth}), мінімальна кількість зразків для розділення вузла ($min_{samples_{split}}$), тощо.

Побудова Random Forest включає створення $n_{estimators}$ дерев рішень. Кожне дерево будується на випадковій підвибірці даних, і для кожного вузла вибирається випадковий піднабір ознак.

Кожне дерево навчається максимізувати функцію індексу Джині чи ентропії для розділення вузлів та прийняття рішень.

Кожне дерево прогнозує вихідну мітку для нового вектора ознак X.

2. Прогнозування для нових даних

Коли модель Random Forest навчена, вона може бути використана для прогнозування нових даних.

Для нового вектора ознак X, кожне дерево в ансамблі робить прогноз, і вибірка міток кожного дерева може вважатися голосом або рішенням для ансамблю.

Таким чином, результат Random Forest визначається як голосування більшості або усереднення прогнозів дерев.

Формально, для бінарної класифікації:

$$prediction = sign\left(\sum_{i=1}^{n_{estimators}} f_i\,(X)\right) \quad (2.14)$$

де $f_i(X)$ – проноз і-го дерева, а $sign\,(\bullet)$ – функція знаку.

3. Оцінка якості моделі

Оцінка якості моделі може включати використання різних метрик, таких як точність, чутливість, специфічність, F1-мера тощо.

Важливо враховувати, які саме метрики важливі для конкретного випадку виявлення кібератак та налаштовувати модель для оптимальної продуктивності.

В методі Random Forest вхідні параметри, такі як ті, які використовуються в наборі даних NLS-KDD для визначення вихідних параметрів для виявлення кібератак, використовуються для побудови дерев рішень та прийняття рішень на основі цих дерев. Основна ідея полягає в тому, що кожне дерево вирішує, чи є даний вхідний приклад "нормальним" чи "вторгненням", і ансамбль дерев приймає рішення на основі більшості голосів дерев.

Розглянемо, як використовуються вхідні параметри NLS-KDD для визначення вихідних параметрів за допомогою Random Forest, ці параметри ми описували раніше в пункті 2.1.

1. Вибір вхідних параметрів

На основі вхідних параметрів, які наведені в NLS-KDD, будується дерево рішень для кожного дерева в ансамблі Random Forest.

2. Побудова дерев рішень

Кожне дерево рішень в Random Forest обирає підмножину вхідних параметрів для кожного вузла, а також визначає пороги для розділення даних. Це відбувається на основі критеріїв якості, таких як коефіцієнт Джині або ентропія.

3. Класифікація прикладів

Кожне дерево приймає рішення про те, чи належить даний приклад класу "нормальний" чи "вторгнення". Результатом є прогноз для кожного дерева.

4. Агрегація прогнозів

Прогноз для ансамблю визначається на основі більшості голосів дерев. Наприклад, у випадку класифікації, якщо більше половини дерев визначають приклад як "вторгнення", то і ансамбль також визначить його як "вторгнення".

Математичні формули можуть бути виражені за допомогою критеріїв якості для вибору поділу вузла та механізмів підрахунку голосів для агрегації прогнозів, але конкретні формули будуть залежати від конкретної реалізації Random Forest. Найчастіше для класифікації використовуються критерії якості, такі як коефіцієнт Джині або ентропія, для прийняття рішення при кожному розділенні вузла.

Для математичного опису використання методу Random Forest з параметрами NLS-KDD для виявлення кібератак, розглянемо конкретні кроки, які включає цей процес:

1. Оголошення ансамблю Random Forest

Нехай *RF* буде ансамблем дерев рішень, де T_1,T_2,…,T_n – окремі дерева в ансамблі.

2. Оголошення вхідних параметрів

Нехай *X* - вектор вхідних параметрів для нового прикладу, який включає в себе параметри NLS-KDD.

3. Прогноз для кожного дерева

Для кожного дерева T_i, де *i* від 1 до *n*, визначимо прогноз як $T_i\ (X)$. Це може бути 0 або 1, де 0 вказує на «нормальний» стан, а 1- на «вторгнення».

4. Голосування в ансамблі

Прогноз ансамблю RF ($\hat{Y}$) визначається голосуванням більшості. Якщо більше половини дерев класифікують приклад як «вторгнення», то ансамбль також класифікує його як «вторгнення». Математично це може бути виражено так:

$$\hat{Y} = mode(T_1(X), T_2(X), \dots, T_n(X)), \qquad (2.15)$$

де *mode* - операція визначення моди (найчастіше зустрічаючеся значення).

5. Визначення порогу для класифікації

При необхідності можна визначити поріг, який вказує, скільки дерев повинно класифікувати приклад як "вторгнення", щоб весь ансамбль прийняв таке рішення.

Таким чином, метод RF використовує вхідні параметри NLS-KDD для навчання окремих дерев рішень та вирішення задачі класифікації для нових прикладів на основі голосування в ансамблі. Визначення «вторгнення» або «нормальності» здійснюється на підставі більшість прогнозів дерев.

2.3 Етапи виконання методу

Основна ідея полягає в тому, щоб побудувати кілька дерев прийняття рішень та об'єднати їх, щоб отримати більш точні та стійкі результати. Нижче наведено етапи, які входять до методу Random Forest:

1. Вибір випадкової підвибірки:

Вибирається випадкова підвибірка з набору даних з поверненням (bootstrap sample).

Це означає, що кожен дерево буде навчатися на власному унікальному піднаборі даних, іншими словами, частина даних може бути включена декілька разів, а деяка — ні.

2. Побудова дерева рішень:

На кожному вузлі дерева випадковим чином вибирається підмножина ознак для врахування при подальшому розділенні.

Дерево будується до певного глибокого рівня або до вичерпання варіантів розділення.

3. Повторення:

Кроки 1 і 2 повторюються для кожного дерева в ансамблі.

4. Голосування або усереднення:

Для класифікації: Клас, який найчастіше визначається усіма деревами, вважається прогнозом Random Forest.

Для регресії: Вираховується середнє або медіанне значення прогнозів всіх дерев.

Цей алгоритм виявляється ефективним завдяки комбінації випадкового вибору при оформленні даних та ансамблю дерев рішень, що зменшує перенавчання та робить його стійким до шуму в даних

Метод Random Forest використовує кілька дерев рішень для вирішення проблем класифікації або регресії. Загальна структура методу Random Forest включає кілька ключових етапів:

1. Створення Бутстреп-вибірок:

Згенерувати випадковий набір бутстреп-вибірок із навчального набору даних. Це означає вибір прикладів з повторенням. Кожна бутстреп-вибірка використовується для навчання окремого дерева в лісі.

2. Побудова Дерев Рішень:

Для кожної бутстреп-вибірки створюється окреме дерево рішень. Під час побудови дерева вибираються випадкові підмножини ознак для кожного вузла. Критерій вибору поділу може бути, наприклад, коефіцієнт Джині або ентропія.

3. Прогнозування Кожним Деревом:

Кожне дерево в лісі дає свій прогноз для кожного прикладу з тестового набору або нового прикладу.

4. Голосування або Усереднення:

Для задачі класифікації використовується голосування більшості: вибирається клас, який отримав більше всього голосів серед всіх дерев. У випадку регресії може використовуватися середнє значення прогнозів дерев.

5. Зменшення Кореляції Моделей:

Щоб дерева в лісі були різноманітними, важливим є випадковість вибору ознак для кожного дерева і випадковість вибору бутстреп-вибірок. Це сприяє зменшенню кореляції між деревами, що робить ліс більш стійким та ефективним.

6. Оцінка Точності та Важливості Ознак:

Визначення точності моделі на тестовому наборі для оцінки її ефективності. Також розраховується важливість ознак, що дозволяє визначити, які параметри були найбільш істотними для прийняття рішень.

Така структура дозволяє Random Forest виявляти складні залежності в

даних, покращувати загальну точність та зменшувати ризик перенавчання. Метод Random Forest широко використовується для вирішення завдань класифікації та регресії в області машинного навчання.

Метод Random Forest включає кілька етапів обробки інформації для ефективного виявлення кібератак. Основні перетворення даних на кожному етапі включають:

1. Вибір вхідних параметрів (Feature Selection)

Метод Random Forest використовує вибір вхідних параметрів (Feature Selection) на кожному вузлі при побудові кожного дерева. Основна ідея полягає в тому, щоб вибирати випадковий піднабір ознак для розгляду на кожному етапі розгалуження. Це допомагає забезпечити різноманітність і стабільність моделі. Вибір вхідних параметрів може бути реалізований декількома способами, і одним з них є випадковий вибір піднабору ознак для кожного вузла.

Розглянемо цей процес детальніше за допомогою математичних формул які вже раніше згадувались в пункті 2.2:

• Вибір піднабору ознак:

На кожному вузлі в дереві для Random Forest обирається випадковий піднабір ознак. Позначимо множину ознак як F, а кількість ознак у піднаборі як m. Тоді випадковий піднабір ознак для вузла може бути обраний так:

$$F_{\text{піднабір}} = \text{випадковий вибір}(F, m) \quad (2.16)$$

де $F_{\text{піднабір}}$- випадковий підбір ознак;

F – повний набір ознак;

m – кількість ознак у піднаборі.

• Індекс Джині чи Інші критерії:

Наступний крок - визначення індексу Джині чи іншого критерію якості для кожного можливого розгалуження у вузлі на основі вибраного піднабору ознак. Нехай $\text{Джині}_{\text{піднабір}}$буде індексом Джині для випадкового піднабору ознак $F_{\text{піднабір}}$.

$$\text{Індекс Джині}_{\text{піднабір}} = 1 - \sum_k (p_k)\, 2 \quad (2.17)$$

де p_k- ймовірність того що об’єкт належить до класу k у вибірці для вузла з використанням випадкового піднабору ознак.

• Вибір оптимального розгалуження:

Оптимальне розгалуження визначається таким чином, щоб максимізувати індекс Джині чи інший критерій якості. Обчислення індексу Джині може бути проведено для кожного розгалуження у вузлі.

• Побудова дерева:

Рекурсивно проводиться процес вибору піднабору ознак та побудови розгалужень для кожного вузла, аж до досягнення критеріїв зупинки, таких як максимальна глибина дерева чи мінімальна кількість об'єктів у вузлі.

• Опис: Вибір конкретних характеристик або ознак для використання у моделі. Цей етап дозволяє вибрати тільки значущі та інформативні параметри, що покращує ефективність моделі та зменшує обсяг даних.

• Перетворення: Вибір конкретних параметрів NLS-KDD, таких як числові, категоріальні, бінарні та інші, для використання під час навчання моделі.

Цей підхід до вибору вхідних параметрів у методі Random Forest допомагає створювати різноманітні та стабільні моделі, оскільки кожне дерево в ансамблі навчається на різних піднаборах ознак.

2. <u>Нормалізація</u>

Важливим етапом підготовки даних для методу Random Forest може бути нормалізація, яка допомагає привести значення ознак до одного масштабу. Це може бути важливим для моделей, які базуються на відстанях чи градієнтах, таких як метод Random Forest. Один із загально використовуваних методів нормалізації - це масштабування ознак до діапазону від 0 до 1. Давайте розглянемо цей процес із застосуванням математичних формул які ми описували в пункті 2.2:

• Мінімаксні нормалізація (Min-Max Scaling):

Припустимо, маємо ознаку $\boldsymbol{X}$ зі значення від $X_{\text{мін}}$ до $X_{\text{макс}}$. Мінімаксна нормалізація перетворює кожне значення X у нове значення $X_{\text{нормалізоване}}$ в діапазоні від 0 до 1 за допомогою наступної формули:

$$X_{\text{нормалізоване}} = \frac{X - X_{\text{мін}}}{X_{\text{макс}} - X_{\text{мін}}} \quad (2.18)$$

де $X_{\text{нормалізоване}}$ – нормалізоване значення ознаки;

X – оригінальне значення ознаки;

$X_{\text{мін}}$ – мінімальне значення ознаки;

$X_{\text{макс}}$ – максимальне значення ознаки.

• Застосування до всіх ознак:

Цей процес повторюється для кожної ознаки в даних, забезпечуючи, що всі ознаки знаходяться в одному масштабі.

• Опис: Приведення числових параметрів до одного масштабу для забезпечення стабільності та швидкості навчання моделі. Це важливо, оскільки деякі алгоритми можуть бути чутливі до масштабу різних параметрів.

• Перетворення: Застосування нормалізації до числових параметрів, таких як `duration`, `src_bytes`, `dst_bytes`, і т.д.

Нормалізація допомагає забезпечити, що всі ознаки мають приблизно однаковий вплив на модель, що може поліпшити швидкість навчання та стабільність моделі, особливо для методів, які використовують відстані чи градієнти, таких як метод Random Forest.

3. <u>One-Hot Encoding</u>

One-Hot Encoding є методом кодування категоріальних змінних, що перетворює кожну категорію вектором бінарних (0 або 1) ознак. Цей підхід широко використовується в методі Random Forest, оскільки багато алгоритмів вимагають числових значень. Розглянемо процес One-Hot Encoding та математичні формули, які його описують.

Припустимо, маємо категоріальну ознаку, яку потрібно закодувати за допомогою One-Hot Encoding. Нехай ця ознака приймає C унікальних значень. Тоді кожне унікальне значення буде перетворене в вектор бінарних ознак довжиною C.

• Одиничний код (One-Hot Encoding) для кожної категорії:

Нехай X – категоріальна ознака, і X_j- i – те унікальне значення цієї ознаки.

Кожне нікальне значення X_i перетворюється в вектор V_i за допомогою One-Hot Encoding:

$$V_i = [0,0,\dots,1,\dots,0,0] \quad (2.19)$$

де 1 розміщено на *i*-тій позиції вектора що відповідає унікальному значенню X_i, а всі інші елементи вектора рівні 0.

• Кінцевий вектор для всіх категорій:

Кінцевий One-Hot Encoded вектор для ознаки X формується шляхом об'єднання всіх векторів V_i для кожного унікального значення:

$$One - Hot\ Encoded\ X = [V_1, V_2, \dots, V_C] \quad (2.20)$$

Кількість позицій у кінцевому векторі визначається кількістю унікальних значень *C*.

• Опис: Перетворення категоріальних параметрів у бінарні вектори для правильного використання їх алгоритмами машинного навчання.

• Перетворення: Застосування One-Hot Encoding до категоріальних параметрів, таких як `protocol_type`, `service`, `flag`, для їхнього представлення у вигляді бінарних функцій.

Цей процес створює представлення категоріальних даних у вигляді числового вектора, що можна використовувати в методах, які працюють з числовими даними, такими як Random Forest.

4. Бутстреп-вибірка (Bootstrap Sampling)

Бутстреп-вибірка (Bootstrap Sampling) - це метод, що використовується в методі Random Forest для створення різних навчальних вибірок шляхом випадкового вибору елементів з оригінального набору даних з повторенням. Цей процес створює багато різних "бутстреп-вибірок", які використовуються для навчання кожного дерева в ансамблі. Процес бутстреп-вибірки, виразимо його за допомогою математичних формул, котрі ми вже описували раніше в пункті 2.2:

• Створення Бутстреп-вибірки:

Нехай маємо оригінальний набір даних з N спостережень (об'єктів). Бутстреп-вибірка створюється шляхом випадкового вибору N спостережень з оригінального набору з повторенням. Кожен об'єкт може бути вибраний більше

одного разу або залишитися необранним.

$$D_{\text{бутстреп}} = \{(X_i, y_i)\} \mid i \in \text{вибрані індекси}\} \quad (2.21)$$

де $D_{\text{бутстреп}}$ – Бутстреп-вибірка;

$(X_i, y_i) - i$-те спостереження з оригінального набору;

вибрані індекси – випадково вибрані індекси з повторенням.

• Використання в ансамблі:

Кожне дерево в ансамблі Random Forest навчається на своїй власній бутстреп-вибірці. В результаті кожне дерево може бачити різні підмножини даних, що сприяє різноманітності та стійкості моделі.

$$Random\ Forest = \{\text{Дерево}_1, \text{Дерево}_2, \dots, \text{Дерево}_k\} \quad (2.22)$$

де кожне $\text{Дерево}_{\boldsymbol{k}}$ навчається на своїй власній бутстреп-вибірці.

• Опис: Випадковий вибір підмножини прикладів даних з повторенням для навчання кожного окремого дерева.

• Перетворення: Формування бутстреп-вибірки, де кожен екземпляр може бути вибраний з ймовірністю $1 - \frac{1}{n}$, де n – кількість прикладів у вибірці.

Цей процес бутстреп-вибірки допомагає забезпечити різноманітність моделі Random Forest та зменшує ймовірність перенавчання (overfitting), так як кожне дерево має можливість бачити різні підмножини даних під час навчання.

5. <u>Побудова дерева рішень</u>

Побудова дерева рішень в методі Random Forest включає в себе кілька етапів, таких як вибір вузла для розгалуження, визначення розгалуження та рекурсивне побудова лівого та правого піддерева. Розглянемо цей процес більш детально та виразимо його за допомогою математичних формул, які вже описувались в пункті 2.2:

• Вибір вузла для розгалуження:

На кожному етапі побудови дерева обирається вузол, за який дані будуть розгалужуватися. Це робиться на основі певного критерію якості, такого як індекс Джині чи інший критерій, який вимірює чистоту розгалуження. Нехай *X* – множина ознак, *T* – підмножина даних у вузлів, а *Q(T,X)* – критерій якості для

вибору вузла:

$$\text{Вузол} = \ argmin_v \in \ x^Q(T, v) \tag{2.23}$$

де Вузол – означає вибір ознаки для розгалуження;

Q(T,v) – критерій якості для вузла з ознакою *v*.

• Розгалуження вузла:

Після вибору вузла для розгалуження визначається, які об'єкти підпадають під це розгалуження. Нехай $T_{\text{лівий}}$ та $T_{\text{правий}}$ – підмножини об’єктів, які підходять та не підходять для розгалуження:

$$T_{\text{лівий}} = \{(X_i, y_i) \in \ T \ | \ X_i[\text{Вузол}] \leq \text{поріг}\} \tag{2.24}$$

$$T_{\text{правий}} = \{(X_i, y_i) \in \ T \ | \ X_i[\text{Вузол}] > \text{поріг}\} \tag{2.25}$$

де $X_i[\text{Вузол}] -$ значення ознаки вибору вузла для *i* – того об’єкта;

поріг – значення порогу для розгалуження.

• Рекурсивна побудова лівого та правого піддерева:

Для кожного з підмножин $T_{\text{лівий}}$ та $T_{\text{правий}}$ рекурсивно виконується процес побудови нових вузлів, використовуючи лише підмножину даних, яка відповідає даному розгалуженню.

$$\text{ліве піддерево} = \text{Побудова дерева } (T_{\text{лівий}}) \tag{2.26}$$

$$\text{праве піддерево} = \text{Побудова дерева } (T_{\text{правий}}) \tag{2.27}$$

• Опис: Використання бутстреп-вибірки для навчання кожного окремого дерева рішень.

• Перетворення: Застосування алгоритму розбиття дерева для побудови дерева рішень, де вибір поділу базується на оптимізації певного критерію, такого як коефіцієнт Джині чи ентропія.

Такий процес рекурсивно побудови дерева рішень в Random Forest веде до створення багато дерев, кожне з яких навчається на своїй власній підмножині даних.

6. <u>Голосування або середнє значення</u>

Останній етап методу Random Forest - це голосування або обчислення середнього значення передбачень дерев в ансамблі. Зазвичай використовується

голосування для класифікації та середнє значення для регресії. Давайте розглянемо ці процеси більш детально та виразимо їх за допомогою математичних формул, які ще раніше згадувались в пункті 2.2.

• Голосування для класифікації:

Нехай *M* – кількість дерев у Random Forest, а $H_i(x) -$ клас, передбачений *i*-тим деревом для вхідного об’єкта *x*. Голосування для класифікації може бути виражене так:

$$\text{Голосування(x)} = argmax_c\left(\sum_{i=1}^{M} I\left(H_i(x) = c\right)\right) \quad (2.28)$$

де *c* – клас;

I (•) – індикаторна функція, яка приймає значення 1, якщо вираз усереднення істинний, і 0 в інших випадках;

Це означає, що об'єкт *x* класифікується як клас, який набрав найбільше "голосів" серед усіх дерев.

• Середнє значення для регресії:

У випадку регресії, де ми передбачаємо числові значення, середнє значення обчислюється для кожного об'єкта $\boldsymbol{x}$:

$$\text{Середнє значення}(x) = \frac{1}{M}\sum_{i=1}^{M} H_i(x) \quad (2.29)$$

де $H_i(x) -$ числове значення, передбачене *i*-тим деревом для вхідного об’єкта *x*.

• Опис: Об'єднання прогнозів всіх дерев для кінцевого визначення класу для нового прикладу.

• Перетворення: Використання голосування більшості або середнього значення прогнозів для визначення кінцевого результату.

Цей процес голосування або усереднення забезпечує об'єднане рішення всього ансамблю Random Forest. Кінцевий результат визначається на основі більшості голосів у випадку класифікації або шляхом усереднення у випадку регресії.

Ці етапи допомагають забезпечити ефективне та точне виявлення кібератак за допомогою методу Random Forest.

2.4 Висновки до розділу 2

Отже, ми описали які параметри подавались на вхід методу Random Forest для розпізнавання кібератак. Використовували бази даних, такі як NLS-KDD, CTU-13, MCFP також описали нормалізацію параметрів. Параметри набору даних грають важливу роль у визначенні вхідних параметрів для методу Random Forest, який використовується для розпізнавання кібератак. Описуючи ці параметри та їх нормалізацію, ми визначаємо ключові аспекти підготовки даних для ефективного використання моделі. Серед цих датасетів, найбільш інформативними та актуальними є набори даних MCFP. Отримав подальший розвиток інтеграції параметрів з набору даних CTU-13 у метод Random Forest для виявлення кібератак. Врахування різноманітних характеристик мережевого трафіку та їх адекватна нормалізація дозволяють створювати більш точні та стійкі моделі виявлення аномалій в кіберпросторі. Крім того, можливість динамічної адаптації моделі до нових типів атак або змін у мережевому середовищі сприяє підвищенню ефективності та гнучкості системи виявлення кіберзагроз. Використали математичний апарат процедури розпізнавання за допомогою Random Forest. Розглянули та описали математичні формули методу Random Forest. Описали, як використовуються і оброблюються в Random Forest вхідні параметри NLS-KDD, MCFP, CTU-13 для визначення вихідних параметрів.

Метод Random Forest визначається своїм математичним апаратом, що використовується на кожному етапі, починаючи від побудови дерева рішень і закінчуючи голосуванням та усередненням прогнозів.

Отримав подальший розвиток методу Random Forest для вирішення специфічних завдань, таких як виявлення кібератак. Можливість використання методу з базою даних MCFP дозволяє ефективно розпізнавати нові шаблони атак та адаптувати модель до змін у кіберпросторі, що є важливим для забезпечення кібербезпеки та захисту важливої інформації. Розширення можливостей методу та вдосконалення його ефективності в сфері кібербезпеки може призвести до

нових стратегій виявлення та запобігання кібератак. Метод Random Forest є потужним і ефективним інструментом у сфері машинного навчання, який використовує ансамбль дерев рішень для розв'язання різноманітних завдань, включаючи класифікацію та регресію. Процес його виконання включає кілька ключових етапів, кожен з яких виконує певні перетворення інформації для покращення якості та ефективності моделі.

Отримав подальший розвиток метод Random Forest з метою покращення точності та робастності моделі. Можливе розширення застосування методу для вирішення нових завдань та адаптації до різноманітних даних.

РОЗДІЛ 3. СИСТЕМА РОЗПІЗНАВАННЯ КІБЕРАТАК

3.1 Вибір мови програмування

Python - це високорівнева, інтерпретована мова програмування, яку відзначає простота читання коду та високий рівень продуктивності. Вона була створена Гвідо ван Россумом і вперше випущена у 1991 році. Основні особливості Python:

1. Простота читання та синтаксис:

Python відомий своїм простим і лаконічним синтаксисом, який робить його легким для вивчення та використання. Відсутність фігурних дужок та використання відступів для визначення блоків коду сприяють зрозумілості коду.

2. Високорівнева мова програмування:

Python дозволяє високорівневий стиль програмування, що спрощує розробку та зменшує кількість рядків коду, потрібних для вирішення завдань.

3. Інтерпретованість:

Мова Python є інтерпретованою, що означає, що ви можете виконувати код без необхідності компіляції. Це полегшує швидку розробку та експерименти.

4. Мультипарадигмальність:

Python підтримує об'єктно-орієнтований, процедурний та функціональний підходи до програмування.

5. Велика стандартна бібліотека:

Python поставляється з великою кількістю корисних бібліотек та модулів, які спрощують взаємодію з файловою системою, мережею, введенням/виведенням та іншими аспектами програмування.

6. Активна спільнота:

Python має велику та активну спільноту розробників, що сприяє швидкому розвитку та розширенню екосистеми мови.

Python широко використовується у різних галузях, включаючи веб-розробку, наукове моделювання, штучний інтелект, обробку даних, аналіз даних, розробку ігор та багато іншого.

Використання Python для розпізнавання кібератак за допомогою методу Random Forest має декілька переваг:

1. Богата екосистема бібліотек:

Python має широкий вибір бібліотек для роботи з машинним навчанням, включаючи scikit-learn, яка включає в себе реалізацію алгоритму Random Forest. Готові бібліотеки значно спрощують розробку та реалізацію моделі.

2. Простота використання та читабельність коду:

Синтаксис Python є простим та зрозумілим, що робить його дуже зручним для розробки та збереження читабельності коду. Це особливо важливо у великих проектах, де розуміння та підтримка коду можуть стати складними завданнями.

3. Ефективність та продуктивність:

Python має високорівневий стиль програмування та вбудовані засоби для ефективної роботи з даними, що дозволяє розробникам швидше створювати та тестувати моделі машинного навчання.

4. Велика спільнота та підтримка:

Python має велику та активну спільноту розробників, яка регулярно поновлює та вдосконалює бібліотеки для машинного навчання. Це забезпечує доступ до нових інструментів, багато документації та можливість отримання підтримки від інших розробників.

5. Широкі можливості візуалізації даних:

Python має ряд бібліотек для візуалізації даних, таких як Matplotlib, Seaborn та Plotly, що дозволяє аналізувати результати та взаємодіяти з великим обсягом даних.

6. Зручна робота зі статистикою та науковими обчисленнями:

Python є популярним серед дослідників та науковців, оскільки в ньому легко використовувати для наукових обчислень та статистичного аналізу даних, що часто потрібно в задачах розпізнавання кібератак.

Отже, Python стає популярним вибором для реалізації систем розпізнавання кібератак за допомогою методу Random Forest завдяки своїм зручностям, ефективності та розширеній підтримці у сфері машинного навчання.

Хоча Python є потужним та зручним інструментом для багатьох завдань, включаючи розпізнавання кібератак за допомогою методу Random Forest, він також має свої обмеження та недоліки. Є кілька аспектів, які можуть бути важливими в цієї задачі:

1. Швидкодія:

Python, як інтерпретована мова, може бути менш ефективною з точки зору швидкодії порівняно з компільованими мовами, такими як C++ або Java. У випадках, де важлива швидкодія, це може бути недоліком.

2. Потребує великої кількості пам'яті:

Деякі алгоритми машинного навчання, зокрема Random Forest, можуть вимагати значної кількості пам'яті для навчання та використання моделі. Хоча це менш критично з точки зору сучасних обчислювальних ресурсів, але це важливо враховувати при роботі з обмеженими ресурсами.

3. Невідповідність для важкого обчислення:

Python може не бути ідеальним вибором для завдань, які вимагають важких обчислень, таких як реалізація складних криптографічних алгоритмів або оптимізація для великих масштабів даних.

4. Неідеальна підтримка паралельних обчислень:

В порівнянні з деякими іншими мовами, такими як C++ чи Java, Python може мати обмежену підтримку паралельних обчислень, що може вплинути на продуктивність у випадках, де потрібні інтенсивні обчислення.

5. Важкість інтеграції з іншими мовами:

Хоча існують інструменти для інтеграції Python з іншими мовами, такими як C, іноді виникають складнощі, особливо якщо вам потрібно поєднати Python-код з великими та ефективними бібліотеками, написаними на інших мовах.

6. Великі вимоги до обсягу пам'яті для великих даних:

Якщо ви працюєте з великими обсягами даних, Python може вимагати значної кількості пам'яті, що може стати обмеженням для обробки великих даних.

Не зважаючи на ці недоліки, багато розробників використовують Python з успіхом для задач розпізнавання кібератак, зокрема з використанням методу Random Forest, завдяки його простоті, ефективності та широкої підтримки в галузі машинного навчання.

Python – це мій власний вибір для реалізації системи розпізнавання кібератак з використанням методу Random Forest. Python має широкий спектр бібліотек для машинного навчання і аналізу даних, включаючи scikit-learn, який має ефективну реалізацію алгоритму Random Forest.

Ось що можна виконати з використанням Python та бібліотеки scikit-learn:

Встановлення бібліотек:

Переконаємося, що у нас встановлені необхідні бібліотеки. Якщо ми використовуємо ***pip***, ми можемо встановити їх так:

pip install scikit-learn

Імпорт бібліотек:

Імпортуємо необхідні бібліотеки у нашому Python-скрипті:

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report

Підготовка даних:

Завантажуємо та підготовуємо дані для навчання моделі. Розділіляємо дані на тренувальний та тестовий набори:

 # Завантаження та обробка даних

X, y = load_data()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Створення та навчання моделі:

Створюємо модель Random Forest та навчаємо її на тренувальних даних:

 # Створення моделі

model = RandomForestClassifier(n_estimators=100, random_state=42)

 # Навчання моделі

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

Оцінка результатів:

Оцінюємо ефективність моделі на тестовому наборі:

```
# Прогнозування класів на тестовому наборі
y_pred = model.predict(X_test)
# Оцінка результатів
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{report}')
```

Це загальний підхід до реалізації системи розпізнавання кібератак з використанням методу Random Forest у Python. Зверніть вашу увагу на налаштування параметрів моделі, таких як `n_estimators` (кількість дерев) та інші, для досягнення оптимальних результатів в конкретному випадку.

3.2 Архітектура системи

Архітектура системи розпізнавання кібератак може включати декілька ключових етапів та компонентів для ефективного виявлення та відповіді на потенційні загрози. Нижче наведено загальний опис архітектури такої системи:

1. Збір Даних:

Сенсори та Джерела Даних: Використання сенсорів, системи реєстрації подій, файрволів та інших джерел для збору мережевого та системного трафіку.

2. Обробка та Нормалізація Даних:

Процесори та Агенти: Обробка та нормалізація отриманих даних для стандартизації форматів та виявлення аномалій.

3. Виявлення Аномалій:

Системи Виявлення Вторгнень (IDS): Використання IDS для виявлення аномальних або підозрілих змін у мережевому та системному трафіку.

4. Аналіз та Класифікація:

Моделі Машинного Навчання: Застосування алгоритмів машинного навчання для аналізу та класифікації паттернів, що вказують на можливі кібератаки.

5. Відповідь та Захист:

Системи Захисту: Введення заходів захисту, таких як блокування атак або ізоляція вразливих систем.

Системи Автоматизованої Відповіді: Розробка систем, які можуть автоматично взаємодіяти з загрозами та застосовувати відповідні заходи.

6. Моніторинг та Аудит:

Системи Моніторингу: Надзорові та аудиторські системи для відстеження подій та оновлення архітектури з врахуванням нових загроз.

7. Візуалізація та Звітність:

Інтерфейси та Панелі Керування: Створення інтерфейсів для візуалізації стану безпеки та створення звітів для адміністраторів.

8. Оновлення та Підтримка:

Механізми Оновлення: Забезпечення системи механізмами оновлення для врахування нових загроз та вдосконалення роботи системи.

9. Інтеграція з Іншими Системами:

API та Протоколи: Розробка інтерфейсів та протоколів для інтеграції з іншими системами безпеки.

10. Аналіз Вихідних Даних:

Системи Аналізу Вихідних Даних: Оцінка ефективності та роботи системи на основі зібраних вихідних даних.

Ця архітектура розпізнавання кібератак дозволяє створювати комплексні та реактивні системи для захисту мережевих інфраструктур від різноманітних кіберзагроз.

Архітектура системи розпізнавання кібератак на основі датасету MCFP включає наступні ключові компоненти та етапи:

1. Збір та Підготовка Даних:

Збір Даних: За допомогою інструментів, таких як Wireshark та Zeek IDS, здійснюється захоплення мережевого трафіку.

Підготовка Даних: Датасет MCFP обробляється для створення вхідних даних для моделі, включаючи розбиття на навчальний та тестовий набори.

2. Навчання Моделі:

Вибір Моделі: Вибір ефективної моделі машинного навчання, наприклад, моделі випадкового лісу.

Навчання: Модель навчається на навчальному наборі, використовуючи вхідні дані та відповіді (мітки).

3. Оптимізація та Підбір Гіперпараметрів:

GridSearchCV: Використання оптимізованої таблиці гіперпараметрів для оптимізації моделі та забезпечення кращої продуктивності.

4. Оцінка Моделі:

Тестування: Модель перевіряється на тестовому наборі для оцінки її точності та ефективності.

Метрики: Використання різних метрик, таких як точність, відкликання, F1-оцінка, матриця плутанини, для кращого розуміння продуктивності моделі.

5. Збереження та Відновлення Моделі:

Збереження: Навчена модель може бути збережена у файловому форматі, наприклад, .pkl, для подальшого використання.

Відновлення: Можливість відновлення моделі з файлу для використання без повторного навчання.

6. Використання Моделі в Системі:

Інтеграція: Розробка системи, яка використовує навчену модель для розпізнавання кібератак у реальному часі.

Моніторинг: Постійний моніторинг та оновлення моделі для виявлення нових типів кіберзагроз.

7. Відображення та Аналіз Результатів:

Візуалізація: Виведення результатів роботи системи, включаючи графіки, таблиці та інші візуальні елементи для зручного аналізу.

Аналіз: Оцінка продуктивності системи та виявлення можливих шляхів її покращення.

Ця архітектура забезпечує повний цикл розпізнавання кібератак на основі машинного навчання та визначає ключові етапи впровадження та оптимізації системи.

Пропонується наступна структурна система (рис. 3.1) для розпізнавання кібератак, яка складається з 12 основних підсистем.

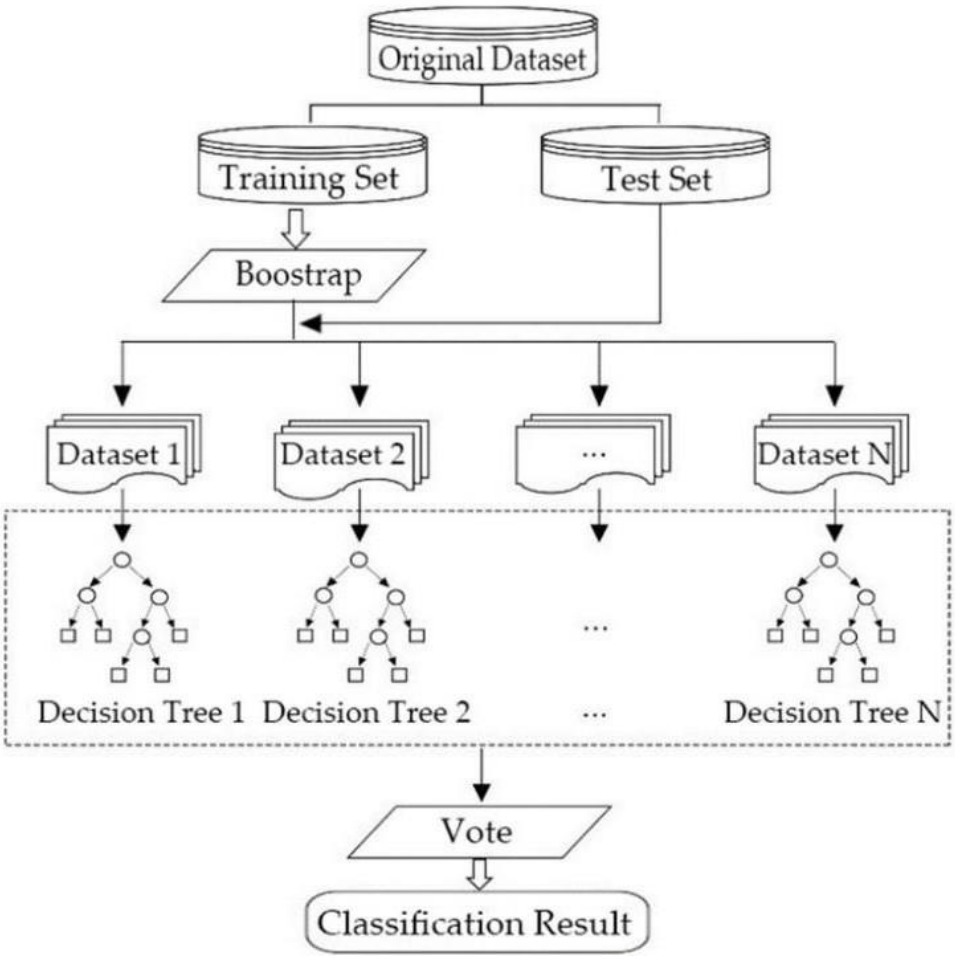

Рис. 3.1. Архітектура системи

Призначення підсистем:

"Original Dataset" - має на меті забезпечити доступ і роботу з оригінальним датасетом (набором даних), який використовується для тренування, валідації та тестування моделі машинного навчання для виявлення кібератак. Ця підсистема є важливою складовою системи розпізнавання кібератак, оскільки якість та

коректність оригінального датасету безпосередньо впливає на ефективність тренування та роботу моделі.

"Training Set" - має на меті забезпечити ефективне тренування моделі машинного навчання для виявлення кібератак. Ця підсистема є ключовою частиною процесу розробки та оптимізації моделі машинного навчання для виявлення кібератак, оскільки вона надає основу для ефективного навчання та налаштування моделі.

"Test Set" - призначена для оцінки та валідації ефективності навченої моделі машинного навчання для виявлення кібератак. Вона важлива для об'єктивної оцінки та валідації роботи моделі, забезпечуючи детальний аналіз її продуктивності на нових даних, які не використовувалися під час тренування.

"Bootstrap" - грає ключову роль у вдосконаленні навчання моделі та забезпеченні її стійкості та генералізації на різних варіаціях даних.

"Dataset 1" - відіграє важливу роль у підготовці та організації даних для подальшого використання в системі розпізнавання кібератак на основі моделі машинного навчання.

"Dataset 2" - це забезпечення додаткового датасету, який використовується для тестування та перевірки ефективності моделі. Це дозволяє враховувати різноманітність та загальну адаптивність моделі до різних умов та сценаріїв.

"Dataset N" - ця підсистема відіграє ключову роль у забезпеченні комплексності та високої ефективності системи розпізнавання кібератак.

"Decision Tree 1" - ця підсистема є важливою складовою архітектури для виявлення та класифікації кібератак, використовуючи методи машинного навчання.

"Decision Tree 2" - відповідає за побудову та використання другого рішучого дерева в системі. Це може бути ще одна модель машинного навчання для класифікації даних та виявлення кібератак.

"Decision Tree N" - відповідає за побудову та використання N-го рішучого дерева в системі. Це може бути додаткова модель машинного навчання для класифікації даних та виявлення кібератак.

"Vote" - інтегрує різні джерела інформації, роблячи систему більш адаптивною та надійною в процесі виявлення кібератак.

"Classification Result" - відіграє ключову роль у визначенні статусу системи та прийнятті рішення про наявність кібератаки на основі агрегованої інформації від різних джерел.

Алгоритм "Ліс рішень", що формується за допомогою методу Random Forest, зазвичай навчається за допомогою методу "мішків". Дерева рішень мають здатність прогнозувати з певною точністю, але коли їх об'єднують в ліс, вони стають значно надійнішими в прогнозуванні. Збільшення кількості дерев у лісі призводить до підвищення точності та запобігає проблемі перенавчання. Алгоритми ансамблю мають кілька характеристик:

• Метод пакування (Bootstrap Aggregation або Bagging) - це мета-алгоритм ансамблю. Ідея полягає в тому, щоб об'єднати прогнози кількох базових моделей для отримання більш точних результатів;

• Гіперпараметри використовуються в моделі випадкового лісу для підвищення прогностичної здатності або прискорення моделі.

Налаштування оточення для навчання.

Першим кроком, необхідним для тренування моделі випадкового лісу, є налаштування робочого оточення. Процес навчання був виконаний на комп'ютері з такими характеристиками:

• Шестиядерний процесор Intel Core i5-750.

• Відеокарта AMD Sapphire Nitro GDDR 5 8Gb.

• 16 гігабайт оперативної пам'яті.

• Операційна система Windows 10.

Для тренування моделі випадкового лісу, необхідно втілити алгоритм у його оригінальній формі або скористатися бібліотеками, такими як scikit-learn.

Розглянемо останній варіант:

• Використовується Zeek (також відомий як Bro IDS) в ролі препроцесора для обробки трафіку.

• Встановлена мова програмування Python.

• Встановлено середовище розробки Visual Code.

• Встановлені модулі numpy, pandas, math, matplotlib, sklearn.

• Завантажені датасети MCFP та проведена їх обробка.

• Здійснено тренування моделі.

Підготовка наборів даних для виявлення шкідливого програмного забезпечення (ПЗ).

Набори даних складаються з файлів .pcap, що містять зафіксовані пакети, та допоміжних файлів від Zeek. Файли .pcap повинні бути перетворені в масштабований вектор, який включає список корисних функцій, експортованих з потоку.

MCFP має набори даних, які включають записи звичайного трафіку та записи трафіку, що містить відкрите шкідливе ПЗ. Таким чином, цей набір даних дозволяє застосовувати методологію "навчання з учителем", де в якості учителя виступає залежна змінна, що вказує алгоритму, чи є трафік шкідливим, чи ні.

Для підготовки датасету MCFP були виконані такі кроки:

• Завантажені архіви з офіційного сайту MCFP , які включають файли .pcap та допоміжні файли зі статистикою.

• Розроблено та виконано скрипт на мові програмування Python, який експортує таблицю .csv у формат, придатний для моделі.

Кожен рядок у вихідній таблиці (матриці) є вектором, який можна переглядати, редагувати та використовувати для оцінки рішення системи. Він містить всі функції, що можна експортувати з мережевих дампів (файлів .pcap).

На табл. 3.1 показано оброблений датасет, де витягнуто 439 функцій з потоку, взятих з файлів .pcap.

Таблиця 3.1

Підготовлений набір даних, отриманий із MCFP бази.

Src.Port	Dst_Port	Bytes_in	…	isMalware
1337	443	12222	…	1
43303	443	5432	…	0

На рис. 3.2 представлено розподіл між шкідливими та стандартними потоками у наборі даних:

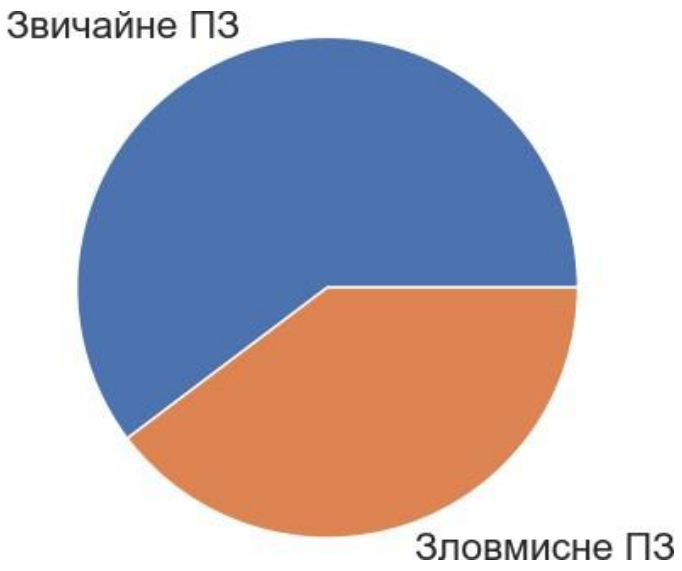

Рис. 3.2. Підготовлений датасет MCFP.

Особливості набору даних:

• Звичайне програмне забезпечення: 24,999 прикладів.

• Шкідливе програмне забезпечення: 16,467 прикладів.

• Загалом: 41,466.

• Кількість функцій: 439, від Src_Port до isMalware.

• Типи даних: dtypes: float64(3), int64(436).

• Використання пам'яті: 138.9 мегабайт.

На рис. 3.3. представлено кілька екземплярів з завантаженого набору.

Характеристики датасету:

	Src_Port	Dst_Port	Bytes_in	...	ec_pts_0	ec_pts_1	isMalware
0	49754	443	6856	...	1	0	1
1	49769	449	1578	...	1	0	1
2	49777	449	1541	...	1	0	1
...	...	...	...	...	...	...	...
41463	43303	443	1813	...	1	0	0
41464	46723	443	1813	...	1	0	0
41465	43075	443	1674	...	1	0	0

[41466 rows x 439 columns]

Рис. 3.3. Характеристики датасету

З рис. 3.4. та по рис. 3.9., відображені різні параметри обробленого набору даних.

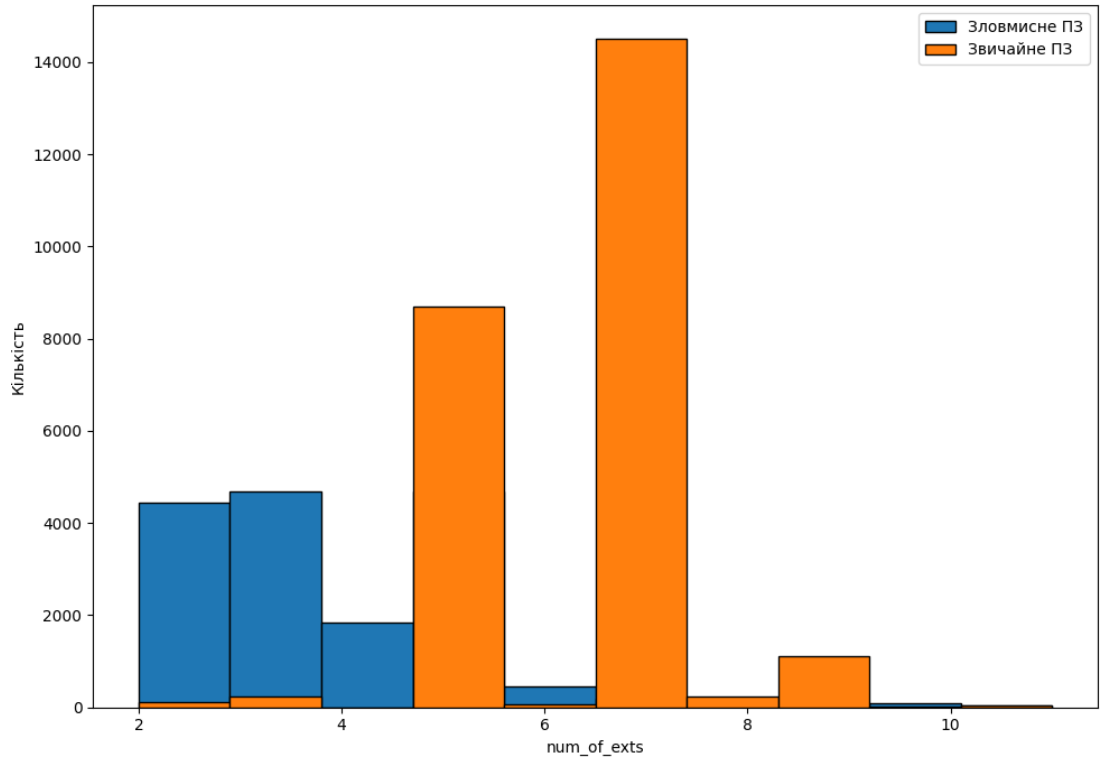

Рис. 3.4. Статистика параметру: кількості розширень

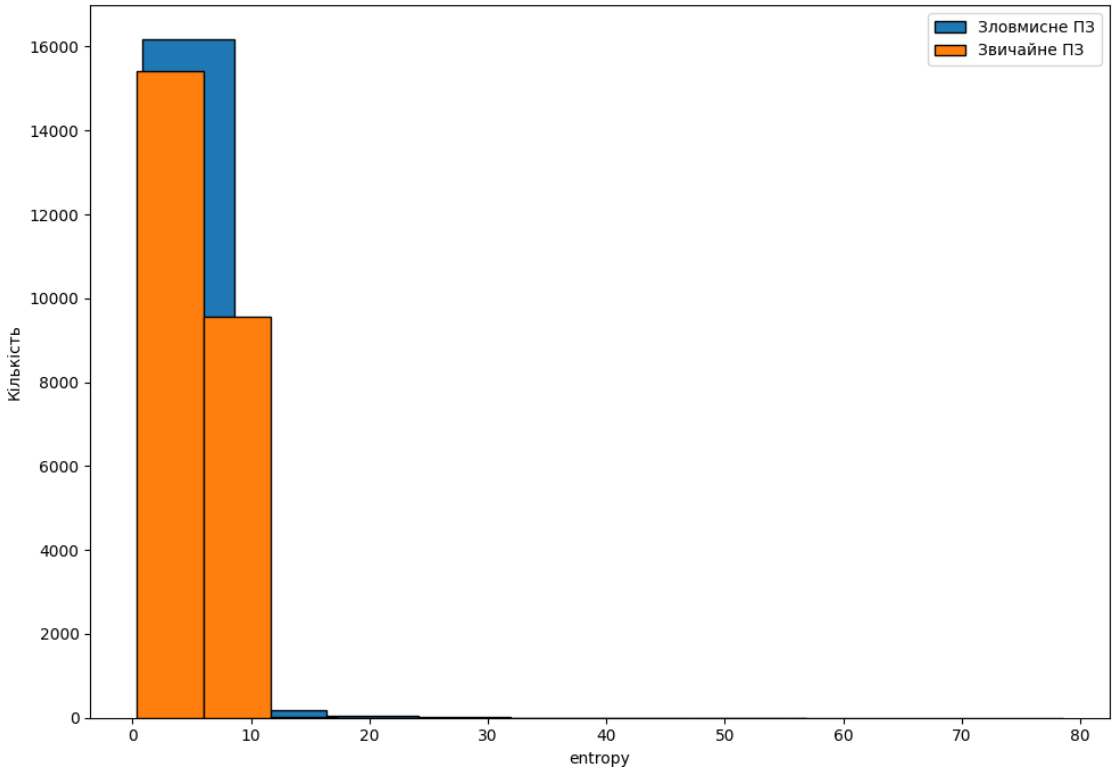

Рис. 3.5. Статистика параметру: ентропія сеансу

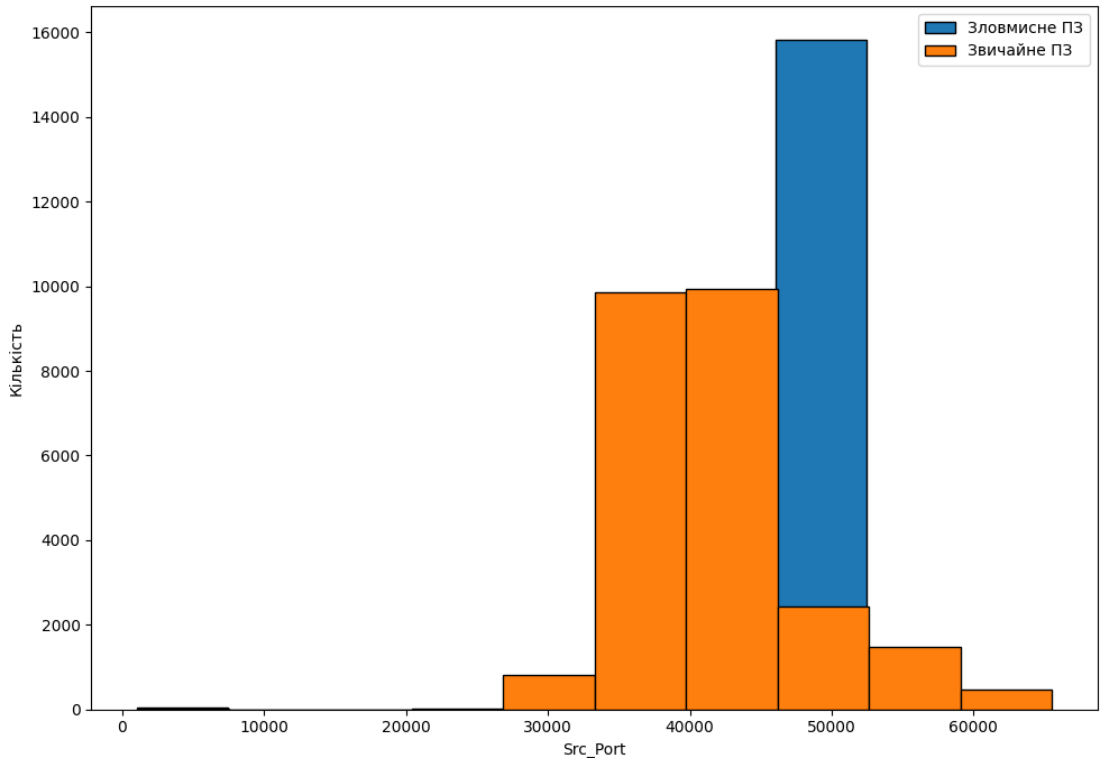

Рис. 3.6. Статистика параметру: вхідний порт

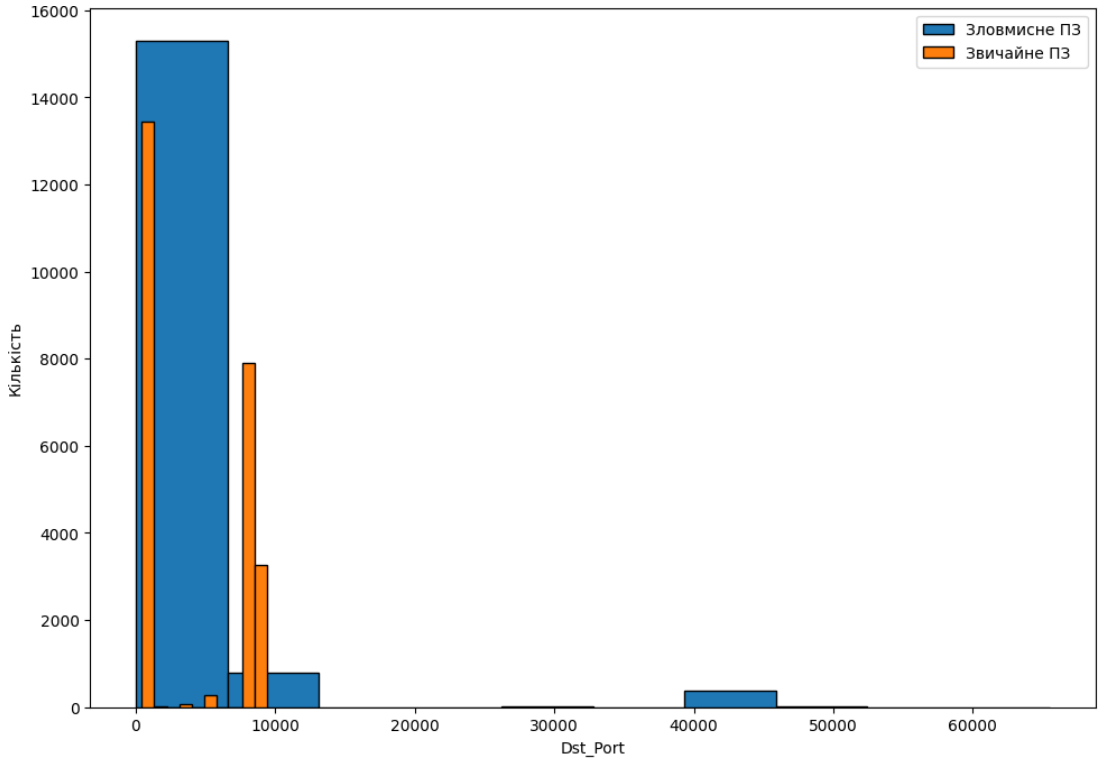

Рис. 3.7. Статистика параметру: вихідний порт

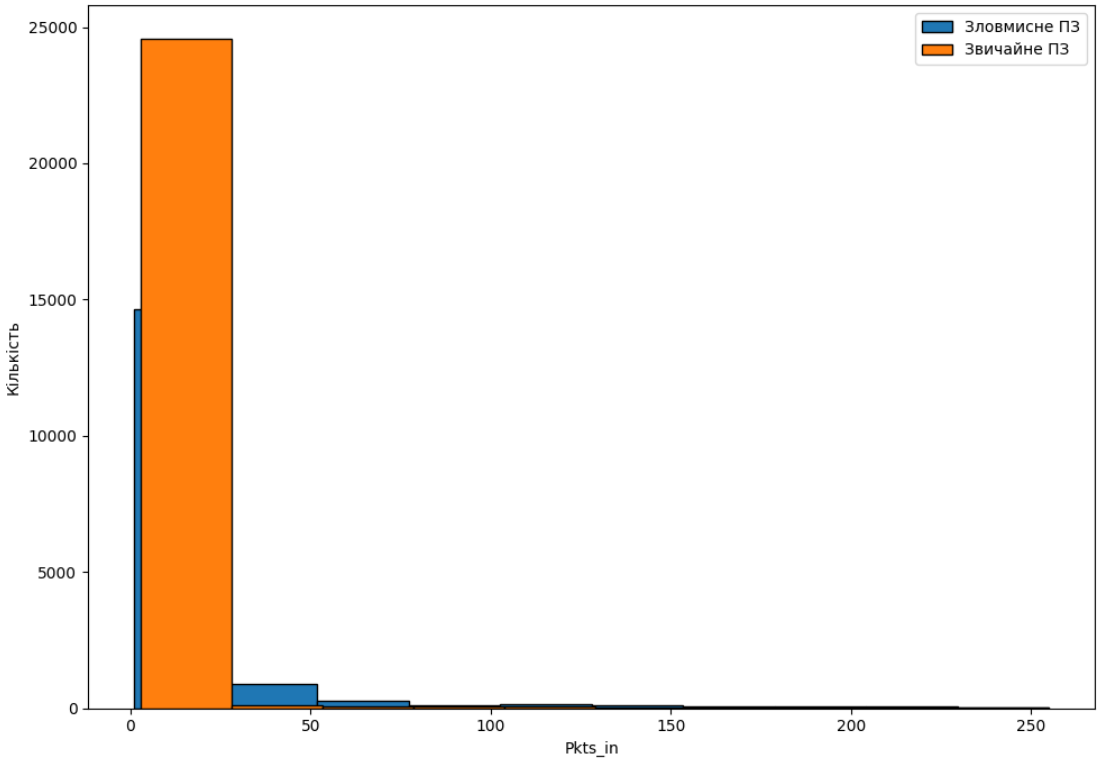

Рис. 3.8. Статистика параметру: пакетів отримано

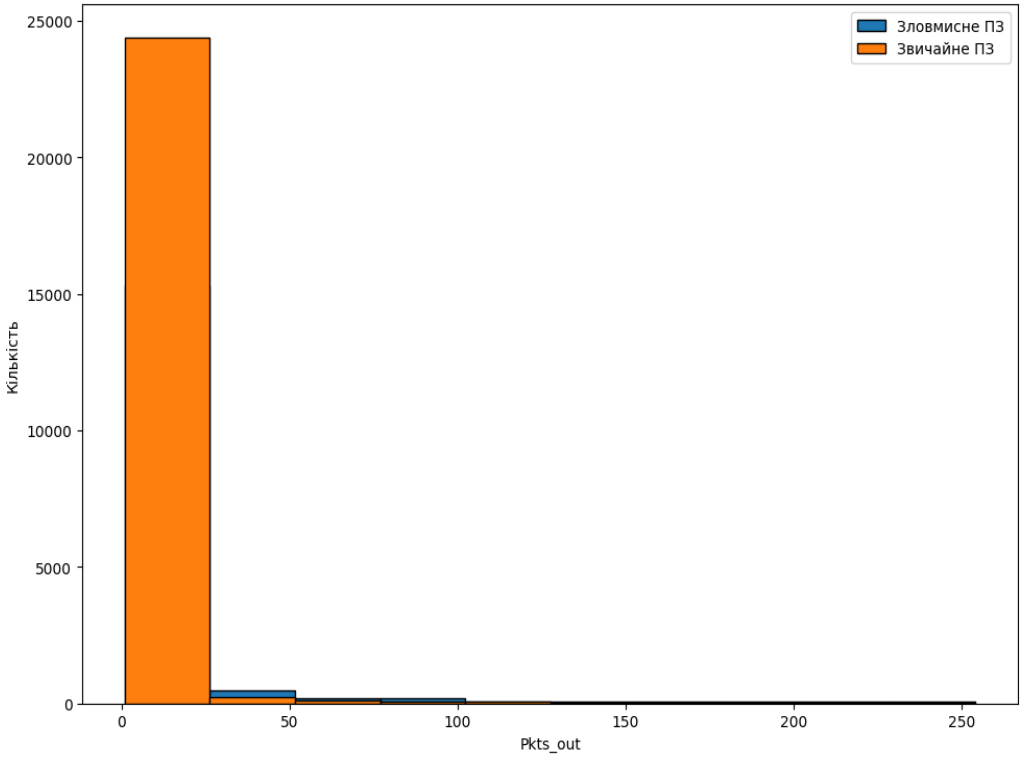

Рис. 3.9. Статистика параметру: пакетів відправлено

Конфігурація середовища для розробки системи включала в себе такі технології:

• Використання мови програмування Python 3.10 для самостійної реалізації функціоналу препроцесінгу трафіку, та Zeek для автоматичного препроцесінгу трафіку.

• Використання бібліотеки sklearn для створення та управління моделлю Random Forest.

• Використання бібліотек nmap та Wireshark для захоплення та аналізу трафіку.

Конфігурація середовища для впровадження системи передбачає використання операційної системи Windows 10 для тестування. Для правильної роботи класифікатору слід виконати такі кроки:

• Забезпечити підтримку мови програмування Python (версії 3.10).

• Встановити необхідні бібліотеки, такі як sklearn, numpy, pandas, seaborn, joblib, matplotlib.

• Встановити Wireshark для аналізу файлів .pcap.

• Встановити Zeek IDS для ручної інспекції пакетів.

3.3 Експериментальні дослідження

Після підготовки датасетів та встановлення бібліотеки sklearn, було проведено навчання, використовуючи оптимізовану таблицю гіперпараметрів за допомогою GridSearchCV. На початку виокремимо залежну змінну isMalware, яка вказує алгоритму, чи є програмне забезпечення шкідливим у розглядуваному наборі даних.

Лістинг 1. Відокремлення залежної змінної:

```
reduced_y = mixed_Dataframe['isMalware']
reduced_x = mixed_Dataframe.drop(['isMalware'], axis=1); train_X, test_X, train_Y, test_Y = train_test_split(reduced_x, reduced_y, test_size=0.25)
```

Модель тренується на випадковій підмножині даних, перевіряється на іншому відокремленому наборі даних, а потім тестується на даних, які були

відокремлені від основного набору даних. Це може створити певні труднощі, оскільки відокремлюючи тестовий набір даних, ми можемо випадково вилучити частину спостережень, які були б ключовими для оптимального тренування моделі. Збереження частини даних для тестування допомагає моделі тренуватися більш ефективно. Додатково, для поліпшення цього методу може використовуватися перехресна валідація, яка розділяє вхідний набір даних на випадкові групи, тримає одну групу для тестування та тренує модель на інших групах. Цей процес повторюється k-разів, використовуючи кожну групу як тестовий набір.

GridSearchCV - це метод пошуку найкращих значень параметрів з визначеної сітки параметрів. В основному це метод перехресної перевірки. Модель та параметри вводяться, визначаються оптимальні значення параметрів, і потім робляться прогнози. Якщо не використовувати гіперпараметри, можливе перенавчання, коли модель занадто специфічно навчається на тренувальному наборі даних, що може призвести до помилок при тестуванні на інших наборах даних. Також може виникнути недостатнє навчання, коли модель недостатньо тренується або не враховує виняткових випадків в тренувальних даних, що може також призвести до помилок при тестуванні.

Лістинг 2. Створення класифікатора з використанням гіперпараметрів:

```
objRF = RandomForestClassifier() params = {
"n_estimators": [7,20,50,100,200],
"max_depth": [10,30,50,90,120],
"min_samples_leaf" : [2, 4, 6, 8, 12]
}
```

Навчання моделі, використовуючи таблицю гіперпараметрів clf = GridSearchCV(рис. 3.10) (objRF, param_grid=params, cv=5) clf.fit(train_X,train_Y)

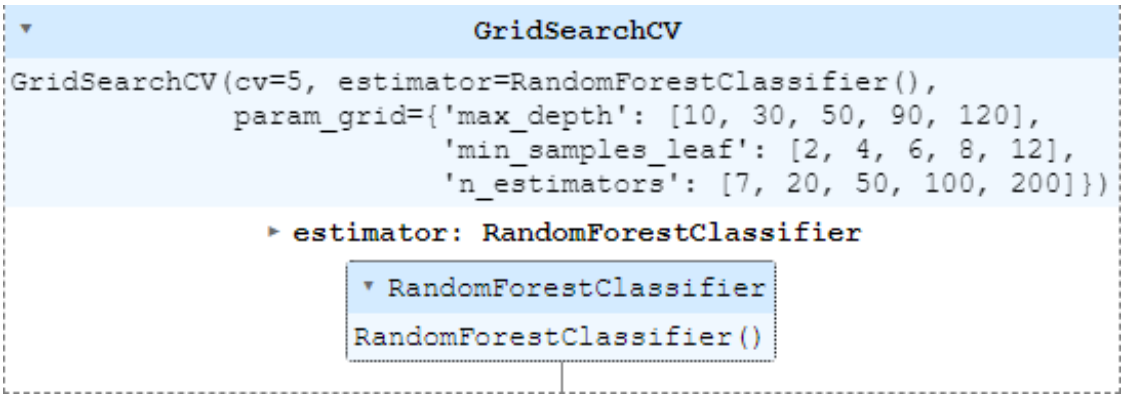

Рис 3.10. Параметри таблиці GridSearchCV

В результаті було визначено оптимальні гіперпараметри для моделі:

{'max_depth': 30, 'min_samples_leaf': 2, 'n_estimators':

50}

Після успішного тренування можна вивчити ваги моделі (рис. 3.11.), яка показує найважливіші параметри трафіку, які можуть бути експортовані у відкритому текстовому чи hex-форматі.

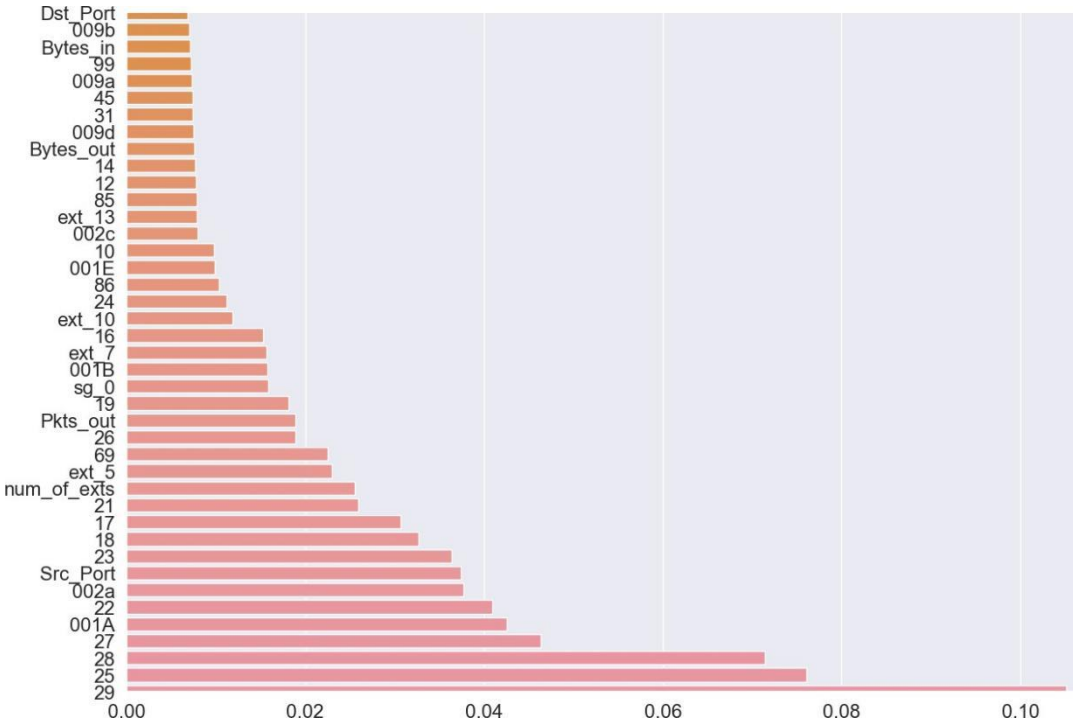

Рис. 3.11. Найвагоміші параметри навченої моделі

Для створення моделі було використано мову програмування Python. Давайте розглянемо таблицю імпорту:

Лістинг 3. Таблиця імпорту:

import numpy as np import pandas as pd from math import sqrt; import seaborn as sns

import joblib as joblib

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.tree import plot_tree

from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import ConfusionMatrixDisplay,

accuracy_score, r2_score, confusion_matrix, mean_absolute_error, mean_squared_error, f1_score, log_loss, classification_report, confusion_matrix

from sklearn.model_selection import GridSearchCV, train_test_split

З використанням бібліотеки pandas ми зчитуємо набір даних з csv-файлу та перетворюємо його у DataFrame.

Лістинг 4. Завантаження датасету та характеристики:

mixed_Dataframe = pd.read_csv('mixed_flows.csv') print('Характеристики датасету: ')

with pd.option_context('display.max_rows', 7,

'display.max_columns', 7,

'display.precision', 5,):print(mixed_Dataframe) mixed_Dataframe.isna().sum()

Статистика

plot_dataset(mixed_Dataframe)

Робимо прогноз на тестових даних.

Лістинг 5. Матриця плутанини та предикт на тестових даних:

testPredict = clf.predict(test_X) print(classification_report(test_Y, testPredict))

print('Точність на тестовій вибірці:',

accuracy_score(test_Y, testPredict))

cm = confusion_matrix(test_Y, testPredict) print('Матриця плутанини:') sns.heatmap(cm,annot=True,fmt="d")

Лістинг 6. Найважливіші функції та їх вага:

```
print('На:')
plot_features(clf.best_estimator_.feature_importances_, train_X.columns)
```

Лістинг 7. Найкращі гіперпараметри моделі:

```
print('Найкращі гіперпараметри моделі:', clf.best_params_)
```

Вивчена модель може бути збережена у файлі формату .pkl і потім відновлена з цього файлу для подальшого використання.

Лістинг 8. Збереження й завантаження моделі:

```
print('Збережено модель: ') print(clf)
joblib.dump(clf, "trained_model_1.pkl")
print('Завантажено модель: ')
model=joblib.load("trained_model_1.pkl")
print(model.best_estimator_.feature_names_in_)
```

Після завантаження моделі ми можемо провести перевірку на доступні функції. Рис. 3.12. показує результати збереження та відновлення моделі.

```
Збережено модель:
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [10, 30, 50, 90, 120],
                         'min_samples_leaf': [2, 4, 6, 8, 12],
                         'n_estimators': [7, 20, 50, 100, 200]})
Завантажено модель:
['Src_Port' 'Dst_Port' 'Bytes_in' 'Bytes_out' 'Pkts_in' 'Pkts_out'
 'entropy' 'byte_dist_std' 'byte_dist_mn' 'num_of_exts' '0' '1' '2' '3'
 '4' '5' '6' '7' '8' '9' '000a' '000b' '000c' '000d' '000e' '000f' '10'
 '11' '12' '13' '14' '15' '16' '17' '18' '19' '001A' '001B' '001E' '001F'
 '20' '21' '22' '23' '24' '25' '26' '27' '28' '29' '002a' '002b' '002c'
 '002d' '002e' '002f' '30' '31' '32' '33' '35' '36' '37' '38' '39' '003a'
 '003b' '003c' '003d' '003e' '003f' '40' '41' '42' '43' '44' '45' '46'
 '67' '68' '69' '006a' '006b' '006c' '006d' '84' '85' '86' '87' '88' '89'
 '008a' '008b' '008c' '008d' '008e' '008f' '90' '91' '92' '93' '94' '95'
 '96' '97' '98' '99' '009a' '009b' '009c' '009d' '009e' '009f' '00a0'
 '00a1' '00a2' '00a3' '00a4' '00a5' '00a6' '00a7' '00a8' '00a9' '00aa'
 '00ab' '00ac' '00ad' '00ae' '00af' '00b0' '00b1' '00b2' '00b3' '00b4'
 '00b5' '00b6' '00b7' '00b8' '00b9' '00ba' '00bb' '00bc' '00bd' '00be'
 '00bf' '00c0' '00c1' '00c2' '00c3' '00c4' '00c5' '00c6' '00c7' '00ff'
 '1301' '1302' '1303' '1304' '1305' '5600' 'c001' 'c002' 'c003' 'c004'
 'c005' 'c006' 'c007' 'c008' 'c009' 'c00a' 'c00b' 'c00c' 'c00d' 'c00e'
 'c00f' 'c010' 'c011' 'c012' 'c013' 'c014' 'c015' 'c016' 'c017' 'c018'
 'c019' 'c01a' 'c01b' 'c01c' 'c01d' 'c01e' 'c01f' 'c020' 'c021' 'c022'
 'c023' 'c024' 'c025' 'c026' 'c027' 'c028' 'c029' 'c02a' 'c02b' 'c02c'
...
 'ext_46' 'ext_47' 'ext_48' 'ext_49' 'ext_50' 'ext_51' 'ext_52' 'sg_0'
 'sg_1' 'sg_2' 'sg_3' 'sg_4' 'sg_5' 'sg_6' 'sg_7' 'sg_8' 'sg_9' 'sg_10'
 'sg_11' 'sg_12' 'sg_13' 'sg_14' 'sg_15' 'sg_16' 'sg_17' 'sg_18' 'sg_19'
 'sg_20' 'sg_21' 'sg_22' 'sg_23' 'sg_24' 'sg_25' 'ec_pts_0' 'ec_pts_1']
```

Рис. 3.12. Збереження та відновлення моделі з файлу формату .pkl та відображення функцій завантаженої моделі

Також розглянемо корисну функцію збереження вагових коефіцієнтів функцій моделі. На рис. 3.13. відображено вихідну таблицю з усіма функціями та їх вагами.

Лістинг 9. Збереження таблиці ваг функцій:

```
fi_df.to_csv('features.csv')
```

No	feature_names	feature_importance
49	29	0.105150138
45	25	0.076049082
48	28	0.07137013
47	27	0.046368955
36	001A	0.042558746
42	22	0.040926007
50	002a	0.037680586
0	Src_Port	0.037428
43	23	0.036373176
34	18	0.03269597
33	17	0.030707575
41	21	0.025910425
9	num_of_exts	0.025531605
362	ext_5	0.022943907

Рис. 3.13. Запис функції моделі та їх ваг у файл формату .csv

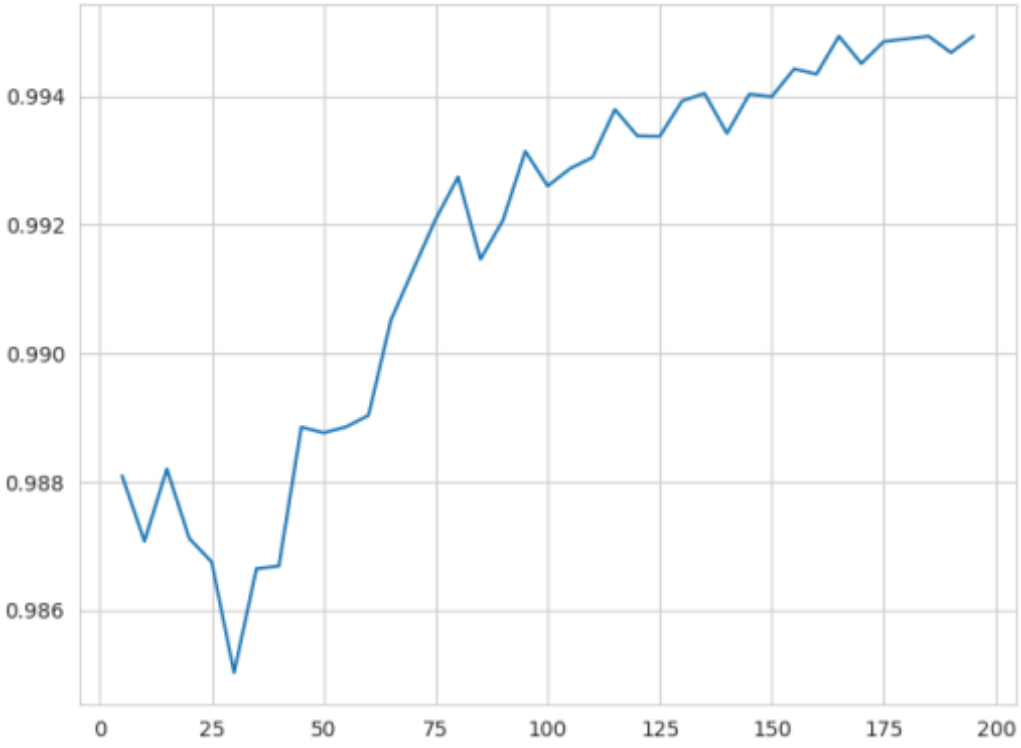

Рис. 3.14. Графік навчання моделі на датасеті MCFP

Під час створення комп'ютерної системи було розроблено систему для виявлення та обробки кібератак з використанням моделі Random Forest. На рис. 3.14. представлений графік точності навчання моделі на наборі даних MCFP.

Як видно, коефіцієнт точності моделі досягає 0.992, враховуючи всього 75 параметрів, отриманих з захопленого трафіку.

Матриця плутанини – це таблиця, що містить інформацію про те, наскільки точно алгоритм класифікації визначає класи в наборі даних. У матриці плутанини представлені наступні значення:

• **Істинно позитивне (TP) та справжньо негативне (TN)** – це значення, які алгоритм правильно визначив. Модель виявила 6248 істинно позитивних значень та 4112 справжніх негативних значень.

• **Помилково позитивне (FP) та помилково негативне (FN)** – це значення, які класифікатор неправильно визначив.

Найвищий показник точності моделі склав 0.9991639716272454. Модель показала 1 помилково позитивний та 6 помилково негативних результатів. На рис. 3.15 представлена матриця плутанини.

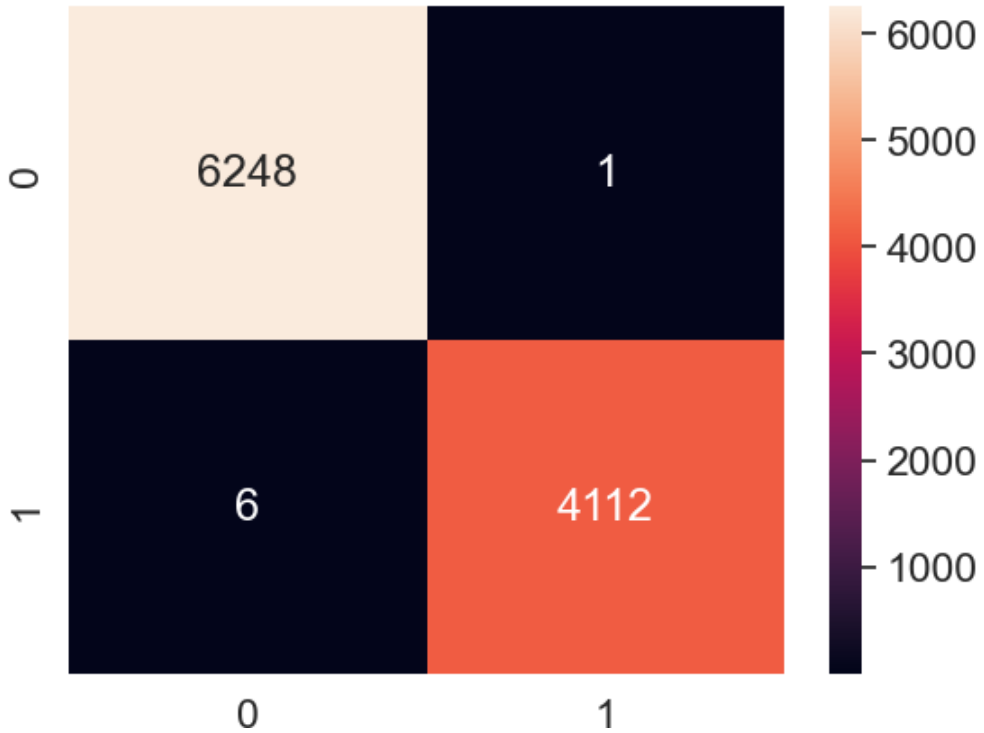

Рис. 3.15. Матриця плутанини

На рис. 3.16. наведено класифікаційний звіт моделі, який відображає показники точності, чутливості (відкликання), F1-мери та підтримки моделі.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6249
1	1.00	1.00	1.00	4118
accuracy			1.00	10367
macro avg	1.00	1.00	1.00	10367
weighted avg	1.00	1.00	1.00	10367

Рис. 3.16. Класифікаційний звіт

Precision (точність) відображає відсоток правильних прогнозів та визначається як відношення істинно позитивних (TP) до суми істинно позитивних і хибно позитивних (FP) спрацювань. Точність = TP / (TP + FP). Цей показник визначає здатність класифікатора не помічати екземпляр як позитивний, який насправді є негативним.

Recall (відкликання) відображає відсоток позитивних випадків, які модель виявила, і визначається як відношення істинно позитивних (TP) до суми істинно позитивних і хибно негативних (FN) спрацювань. Відкликання = TP / (TP + FN). Цей показник визначає здатність класифікатора знаходити всі позитивні екземпляри.

F1 Score (оцінка F1) визначає відсоток правильних позитивних прогнозів та представляє собою середньозважене гармонійне значення точності. F1 Score = 2 * (Recall * Precision) / (Recall + Precision). Цей показник варіюється від 0.0 (найгірший) до 1.0 (найкращий).

Support (підтримка) визначає кількість фактичних входжень класу в зазначений набір даних. Незбалансована підтримка в навчальних даних може свідчити про структурні слабкі місця в заявлених класифікатором балах.

Підтримка не змінюється між моделями, але вказує на процес оцінки.

У результаті була створена модель машинного навчання. У табл. 3.2 представлені назви та ваги перших 10 найважливіших функцій цієї моделі.

Таблиця 3.2

Таблиця функцій моделі

Назва параметру hex / string	Вага
0029 (TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5)	10.5
0025 (TLS_KRB5_WITH_IDEA_CBC_MD5)	7.60
0028 (TLS_KRB5_EXPORT_WITH_RC4_40_SHA)	7.13
0027 (TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA)	4.63
001A (TLS_DH_anon_WITH_DES_CBC_SHA)	4.25
0022 (TLS_KRB5_WITH_DES_CBC_MD5)	4.09
0002 (TLS_RSA_WITH_NULL_SHA)	3.76
Src_Port	3.74
23 (TLS_KRB5_WITH_3DES_EDE_CBC_MD5)	3.6
18 (TLS_DH_anon_WITH_RC4_128_MD5)	3.2

З рис. 3.17. по 3.26. ми розглянемо взаємозв'язки між найбільш впливовими функціями у датасеті для визначення відмінностей між зловмисним та звичайним програмним забезпеченням.

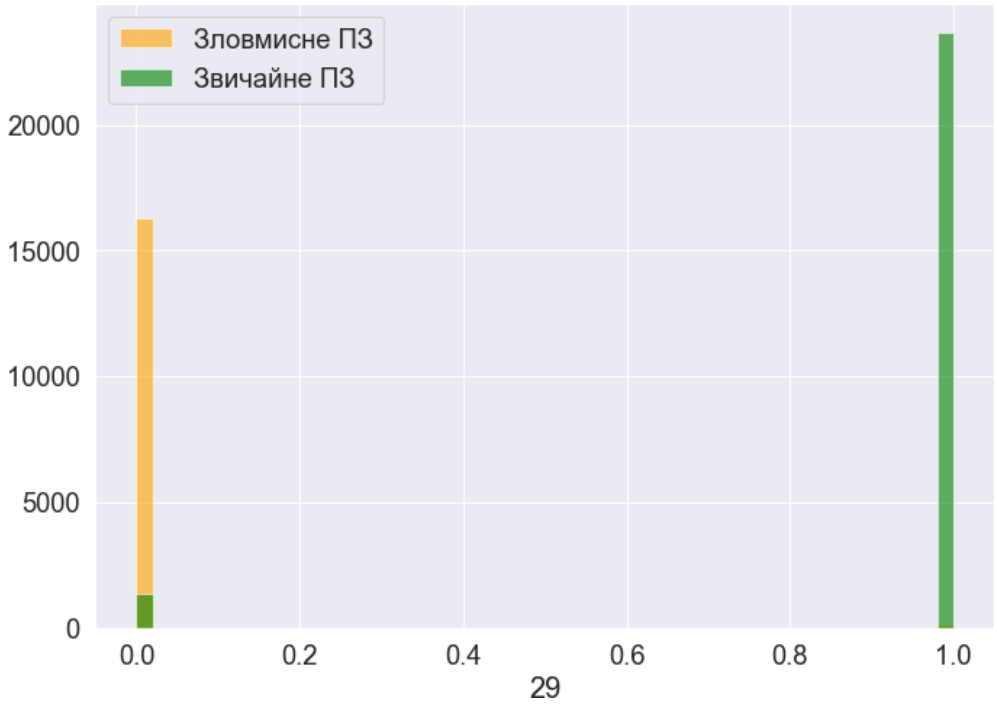

Рис. 3.17. Параметр навченої моделі: шифр TLS_KRB5_EX-PORT_WITH_DES_CBC_40_MD5

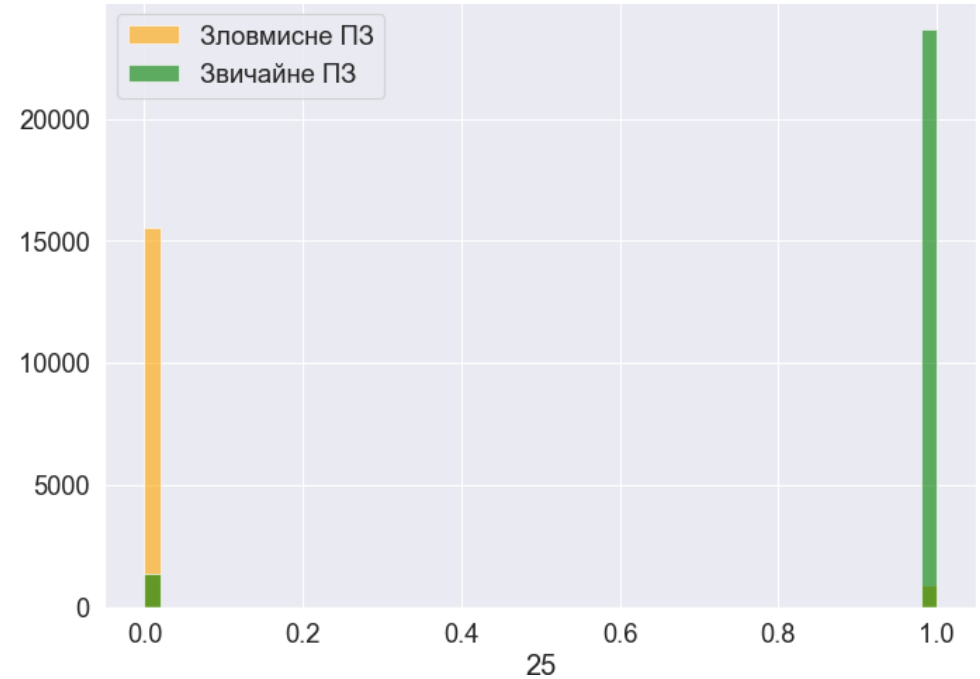

Рис 3.18. Параметр навченої моделі: шифр TLS_KRB5_WITH_IDEA_CBC_MD5

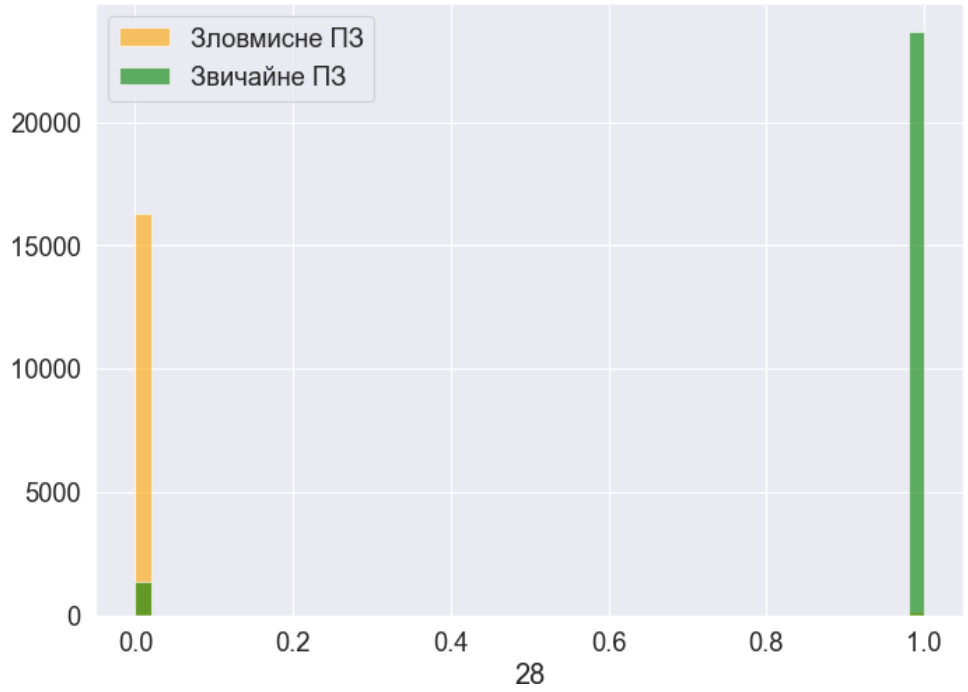

Рис. 3.19. Параметр навченої моделі: шифр TLS_KRB5_EX-PORT_WITH_RC4_40_SHA

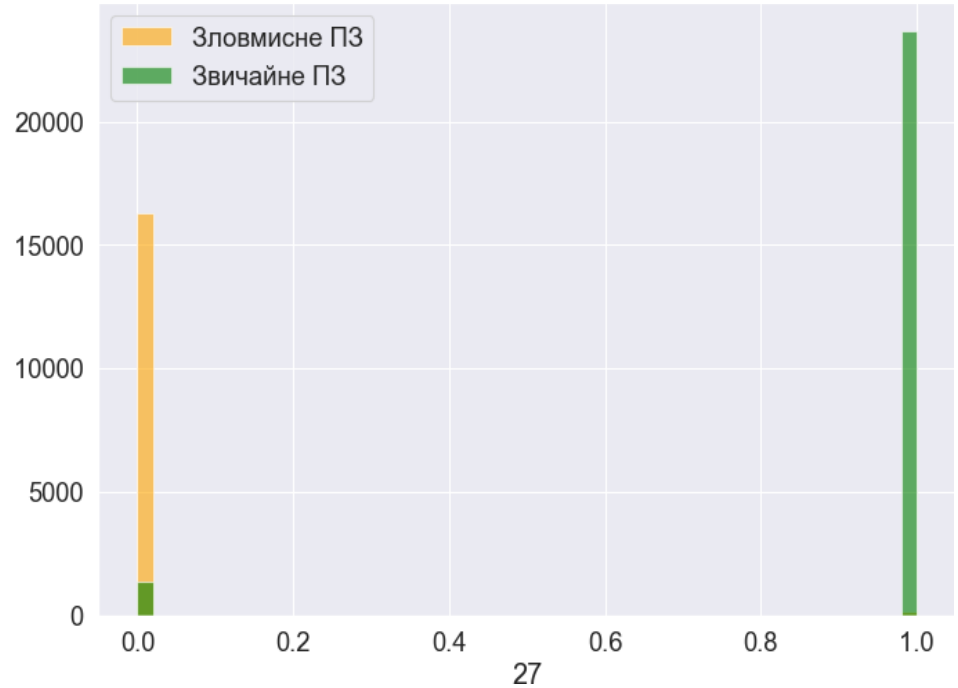

Рис. 3.20. Параметр навченої моделі: шифр TLS_KRB5_EX-PORT_WITH_RC2_CBC_40_SHA

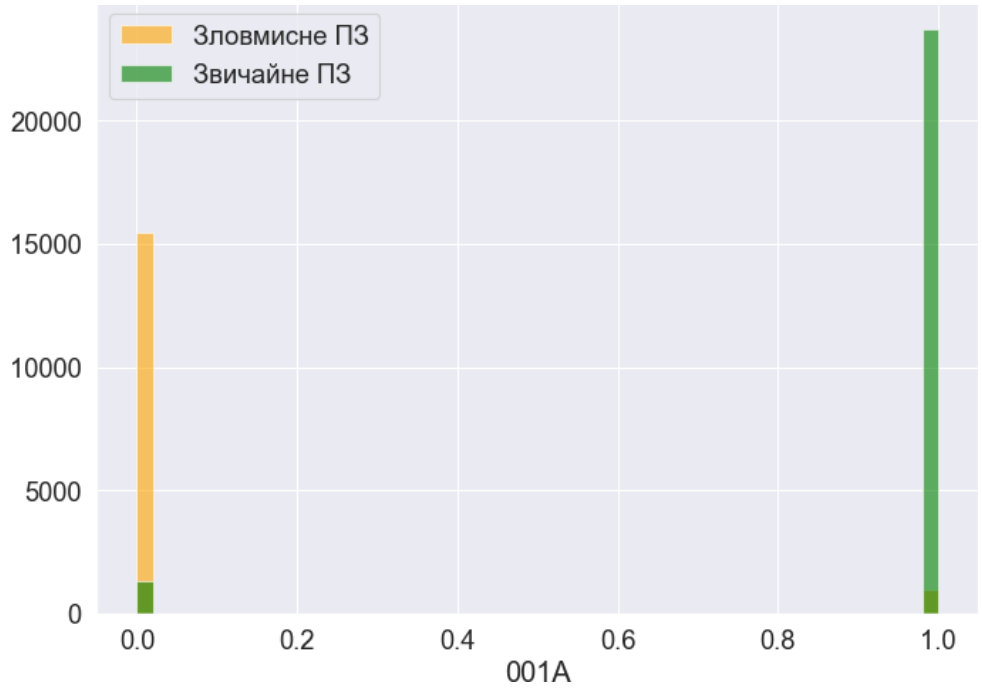

Рис. 3.21. Параметр навченої моделі: шифр TLS_DH_anon_WITH_DES_CBC_SHA

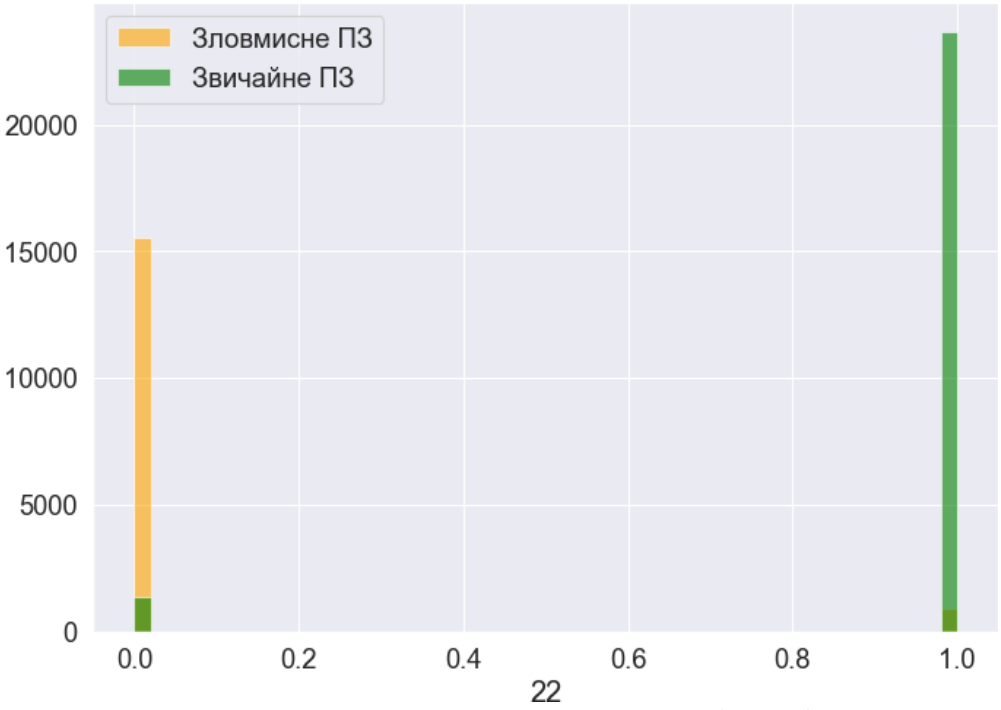

Рис. 3.22. Параметр навченої моделі: шифр TLS_KRB5_WITH_DES_CBC_MD5

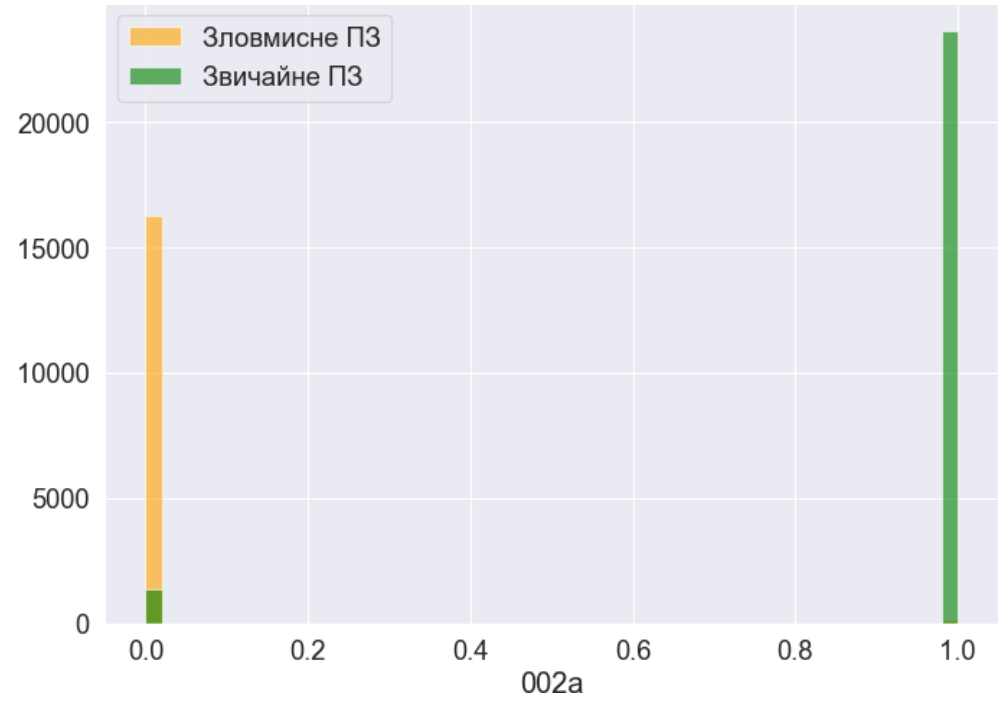

Рис. 3.23. Параметр навченої моделі: шифр TLS_RSA_WITH_NULL_SHA

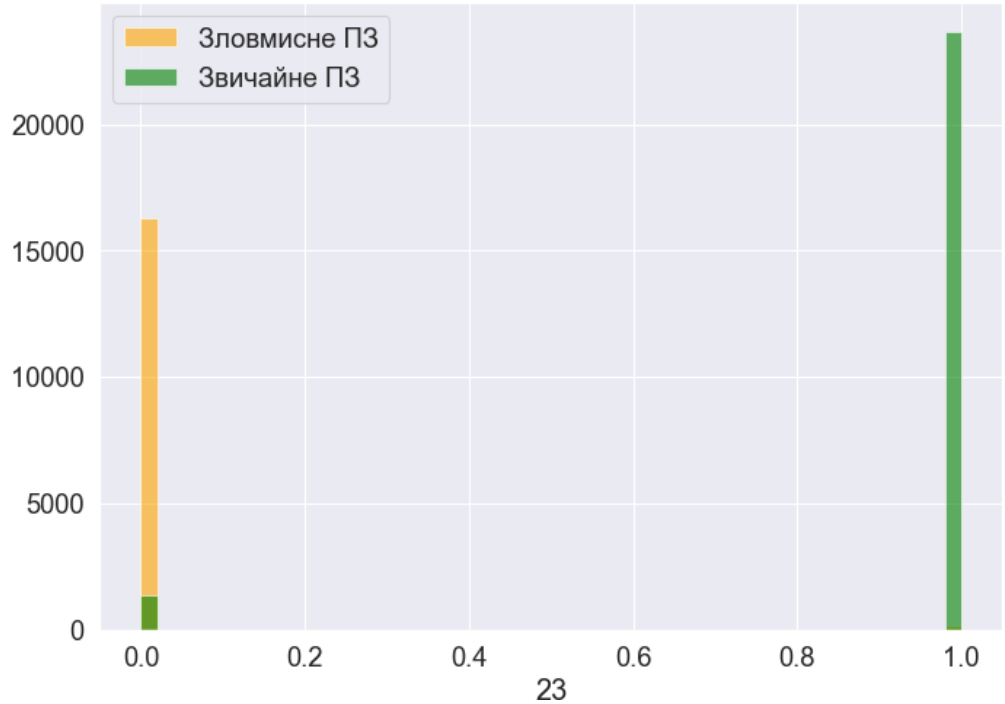

Рис. 3.24. Параметр навченої моделі: шифр TLS_KRB5_WITH_3DES_EDE_CBC_MD5

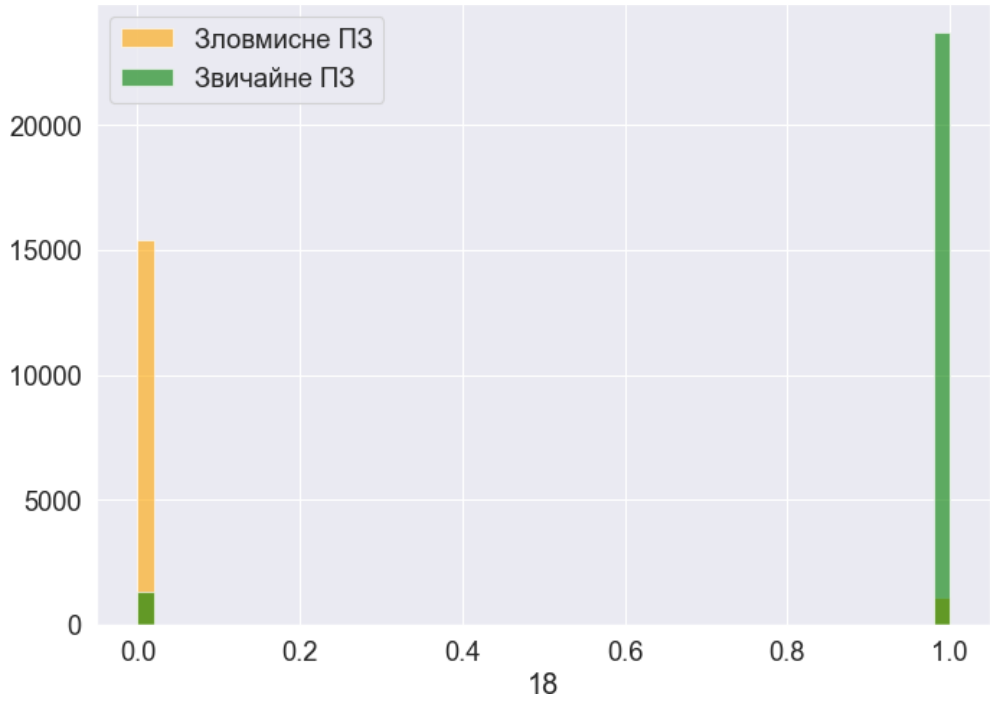

Рис. 3.25. Параметр навченої моделі: шифр TLS_DH_anon_WITH_RC4_128_MD5

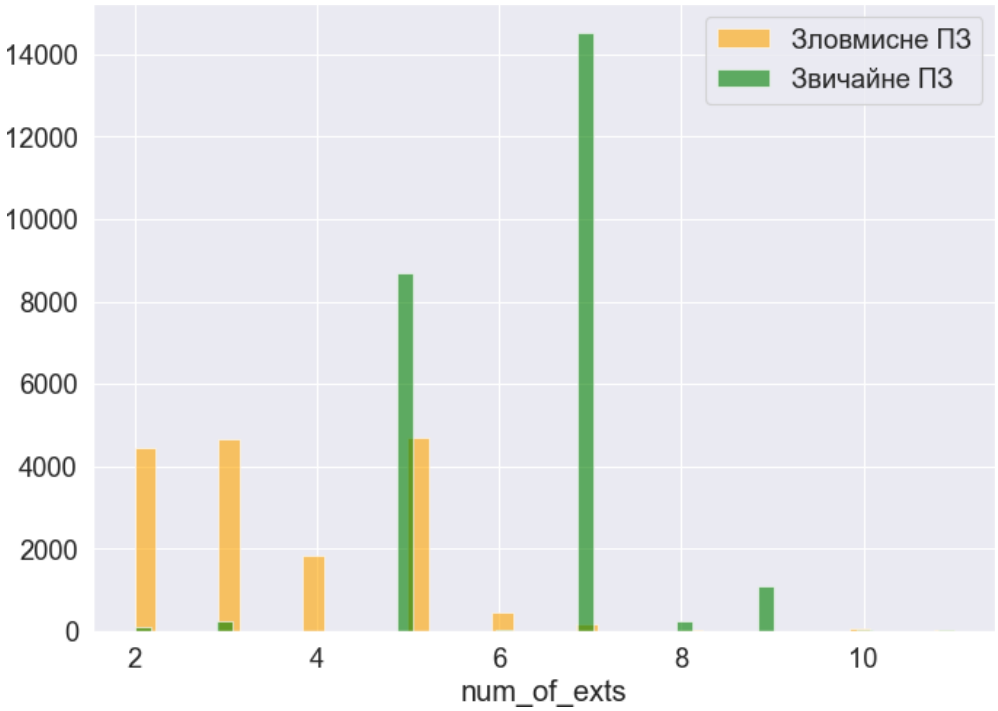

Рис. 3.26. Параметр навченої моделі: num_of_extension

Навчання моделі на опрацьованому датасеті забирає приблизно одну хвилину. Розмір самого датасету складає 191 мегабайт у комп'ютерній пам'яті. При використанні таблиці гіперпараметрів з крос-валідаційною валідацією, час навчання помітно збільшується, але такий підхід є необхідним для досягнення вищої якості моделі.

Ефективність роботи розробленої системи в основному залежить від якості навчання моделі машинного навчання. Взагалі, можна висунути наступні рекомендації для вдосконалення моделі:

• Експериментувати з різними датасетами та гіперпараметрами моделі.

• Автоматизувати збір файлів захоплення, як доброякісних, так і шкідливих.

• Використовувати інший набір функцій для підвищення надійності та TPR.

• Застосовувати функції інших протоколів (DNS, HTTP) для підвищення надійності та TPR.

• Порівнювати класифікатор з іншими типами класифікаторів.

3.4 Висновки до розділу

Ми розгорнули модель машинного навчання, яка пройшла навчання на датасеті MCFP та продемонструвала високу точність під час навчання.

Ми представили характеристики та найбільш важливі функції моделі.

Для підвищення точності роботи навчених моделей існують рекомендації:

• Експериментувати з різними датасетами та гіперпараметрами моделі.

• Автоматизувати збір файлів захоплення, як доброякісних, так і шкідливих.

• Використовувати інший набір функцій для підвищення надійності та TPR.

• Застосовувати функції інших протоколів (DNS, HTTP) для підвищення надійності та TPR.

• Порівнювати класифікатор з іншими типами класифікаторів.

Найвища точність моделі, отримана під час тестування системи, складає 0.9991639716272454 з 1. Модель показала лише 1 помилково-позитивний та 6 помилково-негативних результатів на тестових даних, що є вражаючим результатом.

Застосований підхід до навчання моделі та використання таблиці гіперпараметрів моделі підтверджує свою ефективність у вирішенні завдань класифікації трафіку.

РОЗДІЛ 4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

4.1 Екологічна інформація

Екологічна інформація охоплює всі аспекти, пов'язані з отриманням, аналізом, зберіганням та поширенням даних про стан навколишнього середовища. Ця інформація включає в себе дані про якість повітря, води, ґрунту, рослинний і тваринний світ, а також вплив людської діяльності на екосистеми.

Роль екологічної інформації надзвичайно важлива. Вона допомагає науковцям, урядам, активістам та громадськості зрозуміти стан довкілля, виявити проблеми та ризики для природи та здоров'я людей, і виробляти стратегії для збереження та охорони природи.

Екологічна інформація зазвичай включає в себе різноманітні дані, такі як:

1. Моніторинг забруднення: Вимірювання рівнів різних забруднювачів у повітрі, воді та ґрунті.

2. Біорізноманіття: Дані про видовий склад рослин і тварин, їх поширення та стан популяцій.

3. Загрози та виклики: Інформація про кліматичні зміни, втрату природних ресурсів, знищення екосистем тощо.

4. Стратегії охорони: Дані щодо зон охорони, ефективних методів управління природними ресурсами та збереження біорізноманіття.

5. Екологічні ініціативи: Інформація про проекти та програми, спрямовані на збереження природи та зменшення впливу людської діяльності на довкілля.

Отримання доступу до цієї інформації є критичним для прийняття обґрунтованих рішень у сфері екології та створення стійких стратегій розвитку. Інформованість громадськості, наявність доступу до даних та їх аналіз є ключовими факторами у забезпеченні збалансованого розвитку, який береже природні ресурси для майбутніх поколінь.

Просвіта широкої громадськості й підтримка добровільної діяльності повинні стати головними завданнями. Слід також мати на увазі участь у

дискусіях із великих проектів, що мають значення для довкілля, а також у пошуках їх найкращих здійснень.

Ці принципи потрібно враховувати при розгляді інформації на міжнародному рівні. Недооцінка цих факторів призводить до неадекватного сприйняття матеріалів на екологічну тематику, а при відхиленні від певних норм навіть викликає несподівану для авторів реакцію.

Подібні ситуації виникали в початковий період висвітлення Чорнобильської катастрофи, коли внаслідок когнітивного дисонансу споживач інформації відчував дискомфорт. Почуття протиріччя виникає у людини в тому разі, коли вона одночасно володіє двома психологічно несумісними фактами щодо того чи іншого предмету.

Одразу після трагедії 1986 року дію ЗМК було направлено, з одного боку, на применшення масштабів і наслідків того, що сталося. А з іншого, звучав заклик суворо й ретельно дотримуватися різноманітних заходів профілактики радіоактивного забруднення організму людини й довкілля. Населення потрапило в складну психологічну ситуацію, адже головним джерелом інформації для нього були саме ЗМК.

Український журналістикознавець В. Бугрим, аналізуючи це явище, визначив принципи журналістики екстремальних подій. "Дослідження діяльності засобів інформації з висвітлення Чорнобильської катастрофи дає підстави сформулювати деякі науковоприкладні фактори журналістики екстремальних подій:

1. Прогностична ймовірність, що охоплює передбачення кризових ситуацій і обов'язково готовність властей і населення до стохастичнооповіщувальних ефектів. Чорнобильською Касандрою стала Л. Ковалевська зі своєю публікацією в "Літературній Україні" 27 березня 1986 р.

2. Інформаційнокомунікативна адекватність матеріалів засобів інформації реальності явища, що передбачає максимальну оперативність, достовірність і повноту повідомлень.

3. Соціальнопсихологічна оптимізація матеріалів засобів інформації, під якою розуміється задоволення найважливіших інформаційних потреб населення щодо безпеки, здоров'я і життєдіяльності та формування емоційного стану людини. З'ясування "ризику" ситуації та компенсаційних дій.

4. Управлінськопрагматична функціональність, що означає своєчасне інформування людей про способи й засоби розв'язання неординарних проблем, хід ліквідації наслідків катастрофи.

5. Плюралістичне відображення екстремальної події всіма каналами засобів інформації.

6. Координація взаємодії засобів комунікації (редакції, програми, служби, агентства) з метою уникнення дублювання і розподілення сил для всебічного охоплення явища, системність матеріалів".

У контексті журналістики екстремальних подій цікаві дослідження екологічної комунікації з елементом страху. Німецький дослідник Луман вважає, що нові екологічні теми змінили напрям диференціації проблем від схеми "другворог" до "системаперспектива довкілля". Нові теми, наповнені страхом, потребують нових властивостей.

На думку вченого, не треба боятися показати страх. Коли він починає обговорюватися, то набуває моральної сутності. Вона у свою чергу робить необхідними турботи про стан довкілля, що й стає попереджуючим фактором, з усіма моральними ризиками. Екологічна комунікація перетворюється в такому випадку на моралізовану та наповнену страхом. Та тільки майбутнє може показати, наскільки цей страх був виправданий.

Для того, аби розтлумачити широким масам громадськості певну проблему, її рішення, журналістам потрібно самим розумітися на висвітлюваних питаннях. Якщо цього немає, то помилкові дії, а тим паче заклик до них, завдає великої шкоди як довкіллю, так і суспільству взагалі. Особливо болюче питання екотематики на газетних шпальтах - порушення журналістської етики. Дослідник О. К. Мелещенко наводить один із таких прикладів у Запоріжжі.

4.2 Висновки до розділу

Загалом, екологічна інформація відіграє критичну роль у збереженні природних ресурсів та вирішенні екологічних проблем, тому необхідно забезпечити її доступність, точність та використання для досягнення сталого розвитку.

1. Необхідність доступу до точної інформації: Екологічна інформація - ключовий ресурс сучасності. Забезпечення доступу до точних даних про стан навколишнього середовища є необхідною передумовою для розробки ефективних стратегій охорони природи та збереження біорізноманіття.

2. Залучення громадськості: Посилення участі громадськості у зборі та аналізі екологічної інформації сприяє більшій увазі до проблем довкілля та формуванню відповідального ставлення до природи.

3. Посилення міжнародного співробітництва: Екологічна інформація потребує глобального підходу. Міжнародне співробітництво у зборі та обміні даними є важливим для вирішення глобальних екологічних проблем.

4. Наука та інновації: Інформаційні технології, нові методи досліджень та аналізу даних стають ключовими інструментами для отримання, обробки та розповсюдження екологічної інформації.

5. Подальший розвиток і освіта: Постійний розвиток системи збору, аналізу та поширення екологічної інформації, а також формування екологічної свідомості через освіту, є важливими факторами для створення стійкого майбутнього.

ВИСНОВКИ

Результатом виконаної роботи є вирішення задачі побудови методу застосування Random Forest для розпізнавання кібератак.

У процесі виконання роботи отримані наступні результати:

1. Показано, що перспективним шляхом підвищення ефективності розпізнавання кібератак в комп'ютерних системах, є впровадження в них методу Random Forest для поставленої задачі. Також визначено доцільність застосування в системі розпізнавання кібератак актуального датасету.

2. Розроблена структура та математичне забезпечення методу Random Forest. Застосовані рішення які дозволили пристосувати метод Random Forest до поставленої задачі розпізнавання кібератак. Було проведено навчання моделі, включаючи налаштування середовища для навчання, підготовку вхідних даних для навчання та тестування, а також виконання самого навчання й тестування.

3. Розроблена комп'ютерна система розпізнавання, яка адаптована до застосування розробленого математичного забезпечення і виявляє зловмисне програмне забезпечення, через процеси навчання та тренування моделі.

4. Розроблене програмне забезпечення системи розпізнавання, що базується на використані створеного математичного та інформаційного забезпечення. Це дає нам можливість підвищити рівень розпізнавання кібератак в інформаційних системах та веб-додатках. Модель навчалася на датасетах MCFP, які містять близько 400 прикладів захопленого трафіку від зловмисного програмного забезпечення, і виявляє близько 18 сімейств ботнетів та інших типів зловмисного ПЗ на навчених прикладах.

5. В результаті проведених комп'ютерних експерементів показана доцільність застосування методу Random Forest та розробленої системи розпізнавання кібератак, яка працює дуже добре, особливо при використанні тих самих сімейств зловмисного ПЗ, на яких модель була навчена. Розроблене програмне забезпечення розпізнавання рекомендується використовувати в інформаційних системах та веб-додатках загального призначення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Random forest. URL: https://uk.wikipedia.org/wiki/Random_forest

[2] Random Forest Modeling for Network Intrusion Detection System. URL: https://www.sciencedirect.com/science/article/pii/S1877050916311127

[3] Network Attacks and Their Detection Mechanisms. URL: https://www.researchgate.net/publication/263052997_Network_Attacks_and_Their_Detection_Mechanisms_A_Review

[4] Understanding Random Forest. How the Algorithm Works and Why it Is So Effective. URL: https://towardsdatascience.com/understanding-random- forest-58381e0602d2

[5] Canadian Institute for Cybersecurity. URL: https://www.unb.ca/cic/datasets/

[6] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32. URL: https://www.scirp.org/(S(czeh2tfqw2orz553k1w0r45))/reference/referencespapers.aspx?referenceid=1734556

[7] Zhou, Z. H. (2012). Ensemble methods: Foundations and algorithms. CRC Press.URL:https://www.routledge.com/Ensemble-Methods-Foundations-and-Algorithms/Zhou/p/book/978143983003

[8] Mittal, D., & Gupta, N. (2014). A survey on feature selection methods in intrusion detection system. Computers & Electrical Engineering, 41, 12-24.1. URL: https://www.researchgate.net/publication/317543749_A_STUDY_OF_FEATURE_SELECTION_METHODS_IN_INTRUSION_DETECTION_SYSTEM_A_SURVEY

[9] Kolter, J. Z., & Maloof, M. A. (2006). Learning to detect and classify malicious executables in the wild. The Journal of Machine Learning Research, 7, 2721-2744. URL: https://www.jmlr.org/papers/volume7/kolter06a/kolter06a.pdf

[10]Roli, F., & Giacinto, G. (2003). Design of effective neural network ensembles for image classification purposes. Image and vision computing, 21(12), 1035-1044.URL:https://www.researchgate.net/publication/243784622_Design_of_effective_neural_network_ensembles_for_image_classification

[11] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63(1), 3-42. URL: https://www.researchgate.net/publication/220343368_Extremely_Randomized_Trees

[12] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22. URL: https://www.researchgate.net/publication/228451484_Classification_and_Regression_by_RandomForest

[13] Gomes, H. M., Papa, J. P., & de Albuquerque, V. H. C. (2019). A comprehensive review on feature selection and feature extraction. Expert Systems with Applications, 138, 112830. URL: https://www.researchgate.net/publication/369355042_A_Comprehensive_Review_of_Feature_Selection_and_Feature_Selection_Stability_in_Machine_Learning

[14] Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. Bioinformatics, 23(19), 2507-2517. URL: https://academic.oup.com/bioinformatics/article/23/19/2507/185254.

[15] Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. arXiv preprint arXiv:1802.04269. URL: https://www.scitepress.org/papers/2018/66398/66398.pdf

[16] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications 2009 (pp. 53-58). IEEE. URL: https://www.ecb.torontomu.ca/~bagheri/papers/cisda.pdf

[17] Mahbod, M. A., Mashinchi, R. B., & Mahdavi-Nasab, H. (2011). A new approach for feature selection in intrusion detection systems by using fisher criterion as a feature ranking method. In 2011 Fourth International Symposium on Computational Intelligence and Design (pp. 215-220). IEEE. URL: https://www.researchgate.net/publication/348637315_Feature_selection_for_intrusion_detection_systems

[18] Raman, M., & Alazab, M. (2019). A review of cyber security datasets for machine learning. Journal of Information Security and Applications, 50, 102419. URL:

https://www.researchgate.net/publication/339617181_MACHINE_LEARNING_DATASETS_FOR_CYBER_SECURITY_APPLICATIONS

[19] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182. URL: https://www.researchgate.net/publication/221996079_An_Introduction_of_Variable_and_Feature_Selection

[20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830. URL: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[21] Yadav, B., & Kumar, S. (2020). Cyber threat detection using machine learning algorithms: A review. Journal of Ambient Intelligence and Humanized Computing, 11(3), 1253-1270. URL: https://www.researchgate.net/publication/344704519_Cyber_Threat_Detection_Using_Machine_Learning_Techniques_A_Performance_Evaluation_Perspective

[22] Zhang, J., Zulkernine, M., & Haque, A. (2007). Intrusion detection using profile-based representation and runtime testing. IEEE Transactions on Computers, 56(5), 619-633. URL: https://www.researchgate.net/publication/350322679_Intrusion_Detection_for_Secure_Social_Internet_of_Things_Based_on_Collaborative_Edge_Computing_A_Generative_Adversarial_Network-Based_Approach

[23] Tang, W., Sheng, Y., & Cao, Q. (2009). Applying random forest to intrusion detection system. In International Conference on Computational Intelligence and Security (pp. 201-205). IEEE. URL: https://www.researchgate.net/publication/332396674_Intrusion_Detection_System_using_Random_Forest

[24] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 15. URL: https://www.researchgate.net/publication/220565847_Anomaly_Detection_A_Survey

[25] Kim, H., & Choi, B. (2010). Intrusion detection system using random forest classification. In International Conference on Computational Science and Its Applications (pp. 597-606). Springer. URL: https://www.researchgate.net/publication/332396674_Intrusion_Detection_System_using_Random_Forest

[26] Garcia, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An extensive evaluation of seven machine learning methods for hand gesture recognition. Expert Systems with Applications, 41(9), 4655-4667. URL: https://www.researchgate.net/publication/340993863_Real Time_Hand_Gesture_Recognition_Using_Surface_Electromyography_and_Machine_Learning_A_Systematic_Literature_Review

[27] Nguyen, H. N., & Hong, T. P. (2016). Hybrid intelligent intrusion detection system using snort and random forests. Journal of Ambient Intelligence and Humanized Computing, 7(1), 69-78. URL: https://www.researchgate.net/publication/325851146_Intelligent_Intrusion_Detection_System_Through_Combined_and_Optimized_Machine_Learning

[28] Varun, & Singh, V. (2018). Machine learning approaches for intrusion detection system: A review. Procedia Computer Science, 132, 809-816. URL: https://www.researchgate.net/publication/336663511_Review_of_intrusion_detection_systems_based_on_deep_learning_techniques_coherent_taxonomy_challenges_motivations_recommendations_substantial_analysis_and_future_directions

[29] Abhishek, K., Tanwar, S., & Kumar, Y. (2019). A systematic review of machine learning in cyber security. Journal of King Saud University-Computer and Information Sciences. URL: https://www.mdpi.com/2076-3417/13/14/8056

[30] Yadav, B., & Kumar, S. (2020). Cyber threat detection using machine learning algorithms: A review. Journal of Ambient Intelligence and Humanized Computing, 11(3), 1253-1270. URL: https://www.researchgate.net/publication/367511636_MACHINE_LEARNING_ALGORITHMS_FOR_DETECTION_OF_CYBER_THREATS_USING_LOGISTIC_REGRESSION

ДОДАТОК А

Поля в наборі даних

Source Port
Destination Port
Protocol
Flow Duration
Total Fwd Packets
Total Backward Packets
Total Length of Fwd Packets
Total Length of Bwd Packets
Fwd Packet Length Max
Fwd Packet Length Min
Fwd Packet Length Mean
Fwd Packet Length Std
Bwd Packet Length Max
Bwd Packet Length Min
Bwd Packet Length Mean
Bwd Packet Length Std
Flow Bytes/s
Flow Packets/s
Flow IAT Mean
Flow IAT Std
Flow IAT Max
Flow IAT Min
Fwd IAT Total
Fwd IAT Mean
Fwd IAT Std
Fwd IAT Max
Fwd IAT Min
Bwd IAT Total
Bwd IAT Mean
Bwd IAT Std
Bwd IAT Max
Bwd IAT Min
Fwd PSH Flags
Bwd PSH Flags
Fwd URG Flags
Bwd URG Flags
Fwd Header Length
Bwd Header Length
Fwd Packets/s
Bwd Packets/s
Min Packet Length
Max Packet Length
Packet Length Mean
Packet Length Std
Packet Length Variance
FIN Flag Count
SYN Flag Count
RST Flag Count
PSH Flag Count
ACK Flag Count
URG Flag Count
CWE Flag Count
ECE Flag Count
Down/Up Ratio
Average Packet Size
Avg Fwd Segment Size
Avg Bwd Segment Size
Fwd Avg Bytes/Bulk
Fwd Avg Packets/Bulk
Fwd Avg Bulk Rate
Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk
Bwd Avg Bulk Rate
Subflow Fwd Packets
Subflow Fwd Bytes
Subflow Bwd Packets
Subflow Bwd Bytes
Init_Win_bytes_forward
Init_Win_bytes_backward
act_data_pkt_fwd
min_seg_size_forward
Active Mean
Active Std
Active Max
Active Min
Idle Mean
Idle Std
Idle Max
Idle Min
Label

ДОДАТОК Б

Код програми

```
from functools import reduce import math
import sys
from threading import Thread
from typing import Dict, List, Tuple import joblib
from sklearn.preprocessing import LabelEncoder from tqdm import tqdm
import numpy as np import pandas as pd
from pandas import DataFrame import requests
from sklearn.ensemble import RandomForestClassifier import warnings
import logging
from timeit import default_timer as timer
from sklearn.model_selection import cross_val_score from sklearn import metrics
logging.basicConfig(stream=sys.stdout, level=logging.INFO)
CLF_FILE = "RandomForestTrained.joblib" TEST_DATA_URL =
"http://205.174.165.80/CICDataset/CICDDoS2019/Dataset/CSVs/CSV-03-11.zip"
def merge_classifiers(clf1, clf2):
n1, n2 = clf1.n_estimators, clf2.n_estimators clf1.estimators_ += clf2.estimators_
clf1.n_estimators = len(clf1.estimators_)
logging.debug(f"Combined two forests: {n1} + {n2} = {clf1.n_estimators}") return clf1
def strip_dataset_columns(dataset):
"eliminate spaces before and after in column names: ' Label ' -> 'Label'" logging.debug("Stripping dataset")
dataset.columns = [x.strip() for x in dataset.columns]
def drop_columns(labels: List[str], dataset): for label in labels:
try:
```

logging.debug(f"Trying to drop column {label}") dataset = dataset.drop(label, axis=1) logging.info(f"Column '{label}' dropped successfully")

except KeyError as e:

logging.warning(f"column '{label}' was not deleted.") return dataset

def fill_none(dataset):

#mitigate None values in dataset for col in dataset.columns:

try:

logging.debug(f"Trying to fill None values in {col}") dataset[col].fillna(dataset[col].median().round(1), inplace=True) logging.debug(f"Successfully filled Nones in {col}")

except:

logging.warning(f"'{col}' Failed fill 'None' value with median")

def replace_infinity(dataset): for col in dataset.columns:

#Elimintaion of Infinities with the largest real number logging.debug(f"replacing infinities in column {col} dataset") largest_number_in_col = dataset[col][dataset[col] != np.inf].max() dataset[col] = dataset[col].replace(np.inf, largest_number_in_col)

def split_test_training_data(dataset, frac=0.8) -> Tuple[DataFrame, DataFrame]: """Splint DataFrame to smaller datasets

Default sizes:

training dataset -- 80% training dataset -- 20%

return: Tuple[training dataset, testing dataset]""" logging.debug(f"Splitting test and training data with {frac=}") training_data = dataset.sample(frac=frac, random_state=25) testing_data = dataset.drop(training_data.index)

return (training_data, testing_data,)

def X_Y_data_split(dataset, y_col) -> Tuple[DataFrame, DataFrame]: logging.debug("Trying to separate X and Y")

x = dataset.drop(y_col, axis=1)

y = pd.DataFrame(dataset[y_col], columns=[y_col]) logging.debug("X and Y separated successfully") return (x, y)

```
def _fit_clf(clf: RandomForestClassifier, X, Y): clf.fit(X,Y)
def _predict_clf(clf: RandomForestClassifier, X: DataFrame, thread_id: int, output: Dict):
output[thread_id] = list(clf.predict(X))
def time_measure(func):
def wrapper(*args, **kwargs)  : start = timer()
ret = func(*args, **kwargs) end = timer()
logging.info(f"time measurment: {end - start}") return ret
return wrapper
@time_measure
def create_clf_synchronycaly(X, Y, X_test, Y_test):
clf = RandomForestClassifier(n_estimators=100, max_depth=20) clf.fit(X, Y)
logging.info('Forest is ready. Trees:' + str(len(clf.estimators_))) logging.info(f"Score: {clf.score(X_test, Y_test)}")
return clf
@time_measure
def create_clf_paralelly(X, Y, X_test, Y_test, n_jobs=10) -> RandomForestClassifier:
small_forest_clfs = [RandomForestClassifier(n_estimators=10, max_depth=20) for _ in range(n_jobs)]
threads: List[Thread] = [Thread(target=_fit_clf,args=[clf, X, Y]) for clf in small_forest_clfs]
for thread in threads: thread.start()
for thread in threads: thread.join()
logging.debug(f"Merging '{n_jobs}' classifiers") clf = reduce(merge_classifiers, small_forest_clfs)
logging.info('Forest is ready. Trees:' + str(len(clf.estimators_))) logging.info(f"Score: {clf.score(X_test, Y_test)}")
return clf
def split_dataset(data, parts):
```

```
rows_count = math.ceil(len(data) / parts)
return [data[rows_count*i:rows_count*(i+1)] for i in range(parts)]
def paralel_predict(clf, X, n_jobs=10): data_parts = split_dataset(X, n_jobs) predicted = {}
threads = [Thread(target=_predict_clf, args=[clf, data_parts[i], i, predicted]) for i in range(n_jobs)]
@time_measure
def _start_threads(threads): for thread in threads:
thread.start()
for thread in threads: thread.join()
_start_threads(threads=threads)
out = []
for i in range(len(predicted)): out += predicted[i]
return out
@time_measure
def synchronical_predict(clf: RandomForestClassifier, X): return clf.predict(X)
def preprocess_dataset(dataset, labels_to_drop=None): "Preprocessing data"
if labels_to_drop is None:
labels_to_drop = ['Unnamed: 0', 'Flow ID', 'Source IP', 'Destination IP', 'Timestamp', 'SimillarHTTP', 'Dst IP', 'Src IP',
'Fwd Header Length.1', 'Inbound'] strip_dataset_columns(dataset)
dataset = drop_columns(labels_to_drop, dataset) fill_none(dataset)
replace_infinity(dataset) return dataset
def question():
return input(f"Download Test Data from \"{TEST_DATA_URL}\"?(y/N): ")
def download_data():
local_filename = TEST_DATA_URL.split('/')[-1] logging.info(f"Downloading file from \"{TEST_DATA_URL}\"") with requests.get(TEST_DATA_URL, stream=True) as r:
r.raise_for_status()
```

```
total_size_in_bytes= int(r.headers.get('content-length', 0)) block_size = 128*1024
progress_bar = tqdm(total=total_size_in_bytes, unit='iB', unit_scale=True) with open(local_filename, 'wb') as f:
for chunk in r.iter_content(chunk_size=block_size): progress_bar.update(len(chunk))
f.write(chunk)
logging.info(f"File {local_filename} has been downloaded. Please unarchive. Exitting...")
def confusion_matrix(Y_test, Y_predicted):
print(pd.crosstab(Y_test, Y_predicted))
def score(clf, X, Y):
accuracy = cross_val_score(clf, X, Y, cv=10, scoring='accuracy') print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf, X, Y, cv=10, scoring='precision') print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf, X, Y, cv=10, scoring='recall') print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf, X, Y, cv=10, scoring='f1')
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
warnings.filterwarnings("ignore") logging.basicConfig(stream=sys.stdout, level=logging.INFO)
def main(): try:
file = sys.argv[1] logging.info(f"Using {file}")
except IndexError:
file = '03-11/Small-Syn-.csv'
logging.info(f"Not given any file to the program. Using {file}.")
try:
dataset = pd.read_csv(file, low_memory=False)
except FileNotFoundError:
```

```
logging.error(f"Not found default file: {file}. Exitting...") if question().lower() == 'y':
download_data() exit()
dataset = preprocess_dataset(dataset)
#Split data
training_data, testing_data = split_test_training_data(dataset)
X_testing_data, Y_testing_data = X_Y_data_split(testing_data, 'Label') X_training_data, Y_training_data = X_Y_data_split(training_data, 'Label')
print("Training Data:") print(X_training_data.shape) print(Y_training_data.shape)
print("Testing Data:") print(X_testing_data.shape) print(Y_testing_data.shape)
# Encode 'Y' column le = LabelEncoder()
Y_testing_data = le.fit_transform(Y_testing_data['Label']) Y_training_data = le.fit_transform(Y_training_data['Label'])
logging.info("---Creating Forest Synchronycally---")
clf = create_clf_synchronycaly(X_training_data, Y_training_data, X_testing_data, Y_testing_data)
logging.info("---Creating Forest in Paralel---")
clf_paralel = create_clf_paralelly(X_training_data, Y_training_data, X_testing_data, Y_testing_data, 10)
logging.info("---Predicting Forest Synchronycally---") predicted = synchronical_predict(clf, X_testing_data)
logging.info("---Predicting Forest in Paralel---") for i in range(2, 51):
print(f'Threads: {i}. ', end='')
predicted_paralel = paralel_predict(clf_paralel, X_testing_data, i)
print("--- Analyze Synchronical ---") score(clf, X_testing_data, Y_testing_data) confusion_matrix(Y_testing_data, predicted) print("--- Analyze Paralel ---")
score(clf_paralel, X_testing_data, Y_testing_data) confusion_matrix(Y_testing_data, predicted_paralel)
```

```
        predicted = le.inverse_transform(predicted) predicted_paralel =
le.inverse_transform(predicted_paralel)
        logging.info("Synchronically predicted:\n" +
str(DataFrame(predicted).value_counts()))
        logging.info("Predicted in Paralel:\n" +
str(DataFrame(predicted_paralel).value_counts()))
        joblib.dump(clf_paralel, CLF_FILE, compress=3)
        if name == ' main ':
        main()
```