Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: "Паралелізація алгоритму класифікації Random Forest для пришвидшення виявлення кібератак"

Виконав: студент (ка) IV курсу, групи СБ-42

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Параїл Олександр Володимирович
підпис (прізвище та ініціали)

Керівник Козак Р.О.
підпис (прізвище та ініціали)

Нормоконтроль Дроздова Т. В.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Параїлу Олександру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Паралелізація алгоритму класифікації Random Forest для пришвидшення виявлення кібератак

Керівник роботи Козак Руслан Орестович, к.т.н., доцент
доцент кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 02 » 05 2025 року № 4/7-361

2. Термін подання студентом завершеної роботи 15.06.2025

3. Вихідні дані до роботи Літературні джерела про предметну область

4. Зміст роботи (перелік питань, які потрібно розробити)
Теоретичні засади розподілених атак (DDoS)
Базові поняття машинного навчання
Практична реалізація паралельної обробки в алгоритмі рандом форест
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Мариненко С. Ю., к.т.н. доцент кафедри МТ		

7. Дата видачі завдання 29.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.01 - 02.02	Виконано
2.	Підбір джерел в галузі дослідження	29.01 - 05.02	Виконано
3.	Опрацювання джерел	06.02 - 20.02	Виконано
4.	Аналіз атак та їхньої реалізації	22.02 - 12.03	Виконано
5.	Розробка програмного коду	15.03 - 25.03	Виконано
6.	Тестування роботи програми	25.02 - 10.04	Виконано
7.	Оформлення розділу "Теоретичні засади розподілених атак (DDoS)"	10.02 - 05.03	Виконано
8.	Оформлення розділу "Базові поняття машинного навчання "	26.03 - 04.05	Виконано
9.	Оформлення розділу "Практична реалізація паралельної обробки в алгоритмі рандом форест "	12.04 - 20.04	Виконано
10.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	25.04 - 10.05	Виконано
11.	Оформлення кваліфікаційної роботи	23.05 - 08.06	Виконано
12.	Нормоконтроль	10.06 - 15.06	Виконано
13.	Перевірка на плагіат	20.06 - 22.06	Виконано
14.	Попередній захист кваліфікаційної роботи	14.06 - 15.06	Виконано
15.	Захист кваліфікаційної роботи	26.06.2025	

Студент _____ Параїл О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Козак Р.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Паралелізація алгоритму класифікації Random Forest для пришвидшення виявлення кібератак // Кваліфікаційна робота ОР «Бакалавр» // Параїл Олександр Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБ-41 // Тернопіль, 2025 // С. ____, рис. - ____, табл. - ____, ліст. - _____, додат. - ____.

Ключові слова: машинне навчання, випадковий ліс, точність, паралелізація

У роботі досліджено проблему виявлення атак типу DDoS, що становлять загрозу інформаційній безпеці сучасних мережевих інфраструктур. Розглянуто природу, класифікацію та існуючі методи боротьби з подібними атаками, а також обґрунтовано доцільність застосування алгоритмів машинного навчання для підвищення точності їх детектування. Основну увагу зосереджено на використанні методу Random Forest для класифікації мережевого трафіку. Запропоновано підхід до паралельної реалізації алгоритму, що дозволяє суттєво скоротити час обробки великих обсягів даних. Проведено експериментальне порівняння ефективності класифікації у послідовному та багатопотоковому режимах. Отримані результати підтверджують переваги паралельної обробки та вказують на перспективність подальших досліджень із застосуванням розподілених обчислень у сфері кібербезпеки.

ABSTRACT

Parallelization of the Random Forest Classification Algorithm to Speed Up Cyberattack Detection // Thesis of educational level "Bachelor" // Oleksandr Parajil // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, Group SB-41 // Ternopil, 2025 // ____ pages, figures - _____, tables - __, listings – ___, appendices – ____..

Keywords: Machine Learning, Random Forest, Accuracy, Paralelization

This thesis addresses the challenge of detecting distributed denial-of-service (DDoS) attacks, which pose significant threats to modern network security. The work explores the nature, classification, and mitigation strategies of such attacks, emphasizing the limitations of traditional statistical detection methods. A machine learning-based approach is proposed to improve detection accuracy, specifically through the use of the Random Forest algorithm for network traffic classification. A parallel implementation of the algorithm is developed to optimize processing time and enhance system performance. Experimental results demonstrate the efficiency of the proposed model, comparing single-threaded and multithreaded execution. Based on these findings, a copyright application has been submitted. Additionally, the study outlines the potential for distributed computing in handling large-scale datasets, offering a foundation for further academic exploration in the field of cybersecurity.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT – Internet of Things

DoS – Denial of Service

DDoS –Distributed Denial of Service

ML – Machine Learning.

TP – True Positive.

FP – False Positive.

TN – True Negative

FN – False Negative.

ВСТУП

У сучасному цифровому просторі безпека комп’ютерних мереж і систем стикається з постійно зростаючими загрозами. Кібератаки, зокрема розповсюдження шкідливого програмного забезпечення, фішинг, а також атаки типу відмови в обслуговуванні (DoS/DDoS), стають дедалі складнішими та агресивнішими. Відтак, виникає потреба у створенні нових інструментів захисту, які здатні ефективно реагувати на змінювану тактику зловмисників. Успішне забезпечення кіберзахисту базується на своєчасному виявленні загроз, тому актуальним залишається вдосконалення існуючих механізмів і розробка нових підходів до протидії атакам.

Об’єктом дослідження є системи захисту від DDoS-атак.

Предмет дослідження – методика паралельної класифікації з використанням алгоритму Random Forest.

Мета дослідження полягає у створенні ефективної методики виявлення DDoS-атак для підвищення рівня кібербезпеки.

Під час роботи застосовувалися такі методи: огляд наукових джерел за тематикою дослідження, розробка підходів до перевірки достовірності результатів, аналіз отриманих даних, а також оцінювання ефективності алгоритмів машинного навчання для вибору найрезультативнішого способу детектування загроз.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ РОЗПОДІЛЕНИХ АТАК (DDoS)

1.1 Актуальність проблеми

На сьогоднішній день численні галузі активно застосовують алгоритми для аналізу даних, класифікації інформації та прогнозування змін певних параметрів. У сфері кібербезпеки такі алгоритми відіграють ключову роль у виявленні загроз, дозволяючи ідентифікувати потенційні атаки на основі аналізу вхідних даних. Зокрема, особливо вразливими до кіберзагроз є пристрої, що входять до екосистеми Інтернету речей (IoT).

Забезпечення безпеки пристроїв IoT є складним завданням через особливості цієї технології. Однією з основних проблем є використання застарілих технологій, які мають суттєві недоліки у сфері конфіденційності та захисту даних. Крім того, оновлення програмного забезпечення для таких пристроїв відбувається нерегулярно або взагалі не здійснюється, що збільшує ризики їх компрометації. Саме тому удосконалення методів кіберзахисту та впровадження нових алгоритмічних рішень є критично важливими для забезпечення надійної безпеки в IoT-системах.

1.2 Різновиди мережевих атак

Мережеві атаки можна розподілити на дві основні категорії:

- Активні атаки. Цей тип атак безпосередньо впливає на функціональність або працездатність системи. Наприклад, шкідливі програми можуть змінювати або знищувати дані на комп’ютері, а атаки на веб-застосунки–модифікувати їхній вміст. До цієї категорії також належать DoS-атаки, які спричиняють відмову в обслуговуванні, блокуючи доступ до ресурсів. Відмінною рисою активних атак є залишення в системі слідів шкідливого впливу.
- Пасивні атаки. Основна мета пасивних атак–збір інформації про систему або її користувачів без прямого втручання у її роботу. До них

належать, наприклад, програми-перехоплювачі (sniffers), які аналізують мережевий трафік. Такі атаки складно виявити, оскільки вони не залишають явних слідів у системі.

Основні мотиви кіберзлочинців. Хакери можуть здійснювати атаки з різною метою:

- Отримання доступу до конфіденційних даних.
- Відмова в обслуговуванні цільового ресурсу, блокуючи його роботу.
- Збір інформації про систему–так звані розвідувальні атаки.

Розвідувальні атаки зазвичай є першим етапом підготовки до масштабного вторгнення. Вони дозволяють зловмисникам визначити слабкі місця в інфраструктурі та обрати відповідні інструменти для подальших атак.

Одним із найпоширеніших методів порушення працездатності системи є атаки відмови в обслуговуванні (DoS). Вони унікальні тим, що їхня мета полягає не у крадіжці даних, а в блокуванні доступу до мережевих ресурсів або їх перевантаженні. Якщо атака здійснюється із залученням великої кількості пристроїв, то вона класифікується як розподілена атака (DDoS).

1.3 Основні форми DDoS-атак

1.3.1 Атаки на прикладному рівні

Одним із найпоширеніших видів атак відмови в обслуговуванні (DoS) є HTTP-flood, який спрямований на перевантаження веб-серверів через безперервні запити GET на порти 80 або 443. Такий потік запитів призводить до того, що сервер втрачає можливість обробляти легітимні запити користувачів. Часто атака націлена не на головну сторінку веб-додатку, а на його ресурсомісткі компоненти або модулі, що активно взаємодіють із базою даних. Швидке зростання кількості логів на сервері може свідчити про початок атаки.

Методи протидії HTTP-flood. Для мінімізації впливу таких атак використовуються кілька стратегій:

- Оптимізація роботи веб-серверів та баз даних – підвищення максимальної кількості одночасних підключень до бази даних для зменшення впливу навантаження.
- Використання nginx як проксі-сервера – встановлення легкого та швидкого переднього сервера, який кешує запити та видає статичні дані, дозволяючи головному серверу обробляти критично важливі завдання.
- Захист від ботів за допомогою CAPTCHA – впровадження перевірки «людяності» користувача на сторінках із високими витратами ресурсів, щоб запобігти автоматизованим атакам.

Використання машинного навчання для виявлення атак. Однією з головних проблем боротьби з HTTP-flood є складність розпізнавання шкідливих запитів серед легітимного трафіку. Людське око може не помітити відмінності через втому або недосвідченість, тому для аналізу патернів запитів ефективно застосовується машинне навчання. Алгоритми аналізують поведінку користувачів та виявляють аномальні потоки даних, що можуть бути пов’язані з DDoS-атаками.

1.3.2 Атаки на мережевому рівні

Одним із найпоширеніших методів DDoS-атак є SYN-flood, що спрямований на перевантаження цільової машини через масовану кількість SYN-пакетів. Основна мета такої атаки–виснажити пропускну здатність каналу зв’язку та довести мережевий стек операційної системи до стану, коли вона перестає приймати нові запити на встановлення з’єднання.

Ця атака працює шляхом ініціації великої кількості TCP-з’єднань, надсилаючи SYN-запити з підробленими або неіснуючими адресами відправника. Більшість операційних систем тимчасово додають такі запити до черги очікування, сподіваючись отримати відповідь ACK. Однак через підроблені адреси відповідь не надходить, і після кількох невдалих спроб OS закриває з’єднання. При великому потоці таких SYN-пакетів черга з’єднань швидко заповнюється, внаслідок чого ядро системи перестає приймати нові

запити, фактично блокуючи доступ до сервера.

Сучасні DoS-боти здатні аналізувати стан системи перед початком атаки. Замість безладної масової розсилки запитів вони спрямовують атаки тільки на відкриті порти, підвищуючи ефективність перевантаження сервера.

1.3.3 Атаки на виснаження пропускної здатності

UDP-flood є одним із методів атаки відмови в обслуговуванні (DDoS), що ґрунтується на надсиланні великої кількості UDP-пакетів до цільового пристрою, що призводить до перевантаження смуги пропускання. ICMP-flood– простий, але ефективний спосіб створення надмірного мережевого трафіку через високу частоту ICMP-запитів, що ускладнює нормальне функціонування мережі.

DNS-ампліфікація (DNS-відбиття) – атакувальний метод, який використовує сторонні DNS-сервери для створення надлишкового трафіку. Зловмисник надсилає невеликий запит, відповідь на який значно перевищує його розмір. При цьому IP-адреса відправника підроблена, і замість атакувальника отримувачем відповіді стає жертва атаки.

Масштабні DDoS-атаки спрямовані на:

- Заповнення пропускної здатності – сервери малого та середнього бізнесу зазвичай мають пропускну здатність 1 Gbps або 10 Gbps, а виснаження доступного ресурсу робить їх недоступними для легітимного трафіку.
- Виснаження апаратних ресурсів – перевантаження оперативної пам’яті (RAM), процесорних потужностей та інших критичних компонентів. Це може спричинити не тільки збої у роботі сервера, а й створити нові вразливості для подальших атак.

Існує ряд ефективних стратегій протидії таким атакам:

- Розробка плану реагування на DDoS.
- Зміцнення мережевої безпеки (брандмауери, антивіруси, контроль трафіку, сегментація мережі).

- Створення надмірності ресурсів для витримки підвищеного навантаження.
- Впровадження CDN для розподілу трафіку.
- Моніторинг у режимі реального часу.

Незважаючи на широкий спектр заходів захисту, час реагування на атаку залишається вирішальним фактором. Хакери використовують тактики, що дозволяють приховати атаки протягом тривалого часу, а інженери, навіть працюючи у команді, можуть не помітити загрози вчасно. Тому вдосконалення методів автоматичного детектування DDoS є критично важливим для забезпечення безпеки мережевих інфраструктур.

На рисунку 1.1 наведено приклад графіку DDoS-атаки.

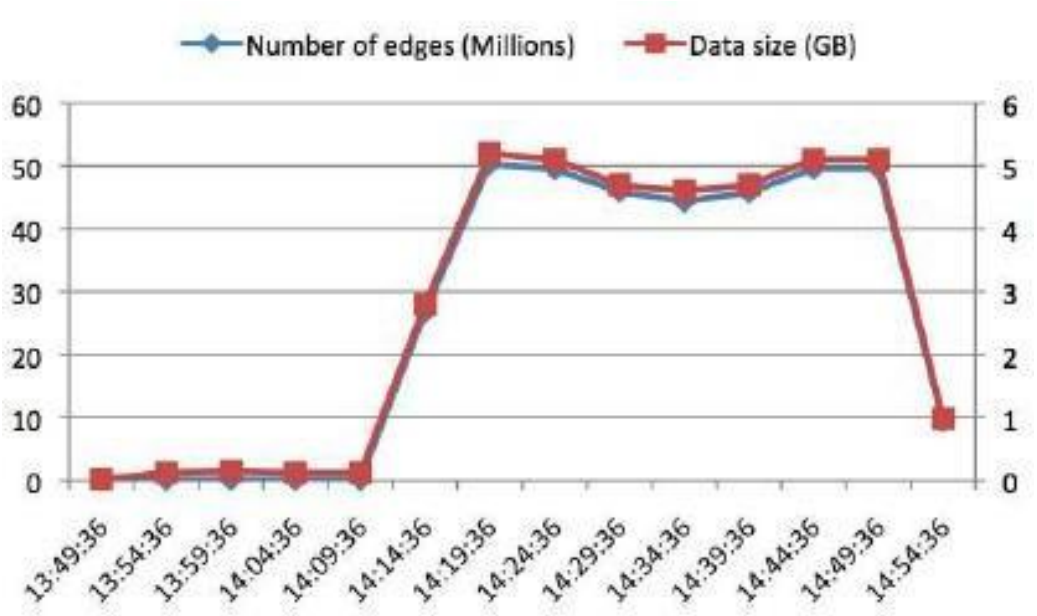


Рисунок 1.1 – Приклад графіку DDoS-атаки

Однак крок дискретизації становить 5 хвилин, що означає, що така атака може бути розпізнана лише у проміжку від 1 до 5 хвилин після значного зростання трафіку (у 30–50 разів). Це створює ризик затримки реагування на загрозу, тому впровадження ефективних систем моніторингу є критично важливим. В умовах величезного обсягу даних саме комп’ютерні системи здатні швидко їх обробляти та оцінювати ситуацію. Однак жоден алгоритм, особливо статистичний, не гарантує повної точності.

Статистичні методи виявлення DDoS-атак можуть бути ефективними, але

мають обмеження:

- Зміна природи атаки або її параметрів може зробити статистичний аналіз неактуальним.
- Переписати алгоритм в реальному часі–нетривіальне завдання.
- Людський фактор також відіграє роль–можливі помилки в оцінці даних.

Детальний аналіз цієї проблеми представлено у статті [13], де розглядаються межі статистичних методів у виявленні загроз. Головна перевага машинного навчання у боротьбі з DDoS-атаками–гнучкість та можливість адаптації до нових атакувальних стратегій. Ці алгоритми здатні класифікувати потоки пакетів, виявляти аномалії та швидко аналізувати патерни трафіку. Машинне навчання використовується у багатьох сферах, і кібербезпека–одна з них. Його впровадження дозволяє значно покращити детектування атак та зменшити ризик компрометації мережевих ресурсів.

РОЗДІЛ 2. БАЗОВІ ПОНЯТТЯ МАШИННОГО НАВЧАННЯ

2.1 Загальні принципи машинного навчання

Машинне навчання–це один із напрямів штучного інтелекту, який дозволяє програмному забезпеченню аналізувати дані, класифікувати їх і прогнозувати значення на основі наявної інформації. Суть цього підходу полягає у навчанні комп'ютера приймати рішення на основі виявлених закономірностей, подібно до того, як це робить людина.

Основні типи машинного навчання:

- Навчання з учителем (Supervised ML) – алгоритм тренується на вже класифікованих даних, встановлюючи чіткі відповідності між вхідними параметрами та очікуваними результатами. Після навчання модель здатна ідентифікувати нові, раніше невідомі дані. Це один із найпростіших та найефективніших підходів.
- Навчання без учителя (Unsupervised ML) – метод, що працює з некласифікованими даними. Він дозволяє знаходити приховані закономірності та використовувати великі обсяги інформації без попередньої розмітки.
- Посилене навчання (Reinforcement Learning) – процес навчання на основі зворотного зв'язку. Алгоритм проходить серію випробувань, отримуючи «винагороду» за успішні рішення або «покарання» за помилки. Це допомагає моделі оптимізувати свої дії для досягнення цілей.

Для досягнення цілей цієї роботи доцільно застосовувати підхід навчання з учителем, адже він дає змогу ефективно здійснювати класифікацію даних і виявляти взаємозв'язки між вхідними та вихідними характеристиками.

Припустимо, існує модель, яка виконує класифікацію інформації. У цьому випадку задачу можна сформулювати наступним чином:

- X – це матриця вхідних параметрів,
- Y – матриця цільових (очікуваних) результатів.

Між цими множинами даних X та Y є певна закономірність, яку можна дослідити та використати для підвищення точності роботи моделі.

$$X = \begin{pmatrix} x_{11} \dots x_{1n} \\ \dots \cdots \dots \\ x_{m1} \dots x_{mn} \end{pmatrix} \quad Y = \begin{pmatrix} y_{11} \\ \dots \\ y_{m1} \end{pmatrix} \tag{2.1}$$

Припустимо, існує функція f*f*, яка перетворює матрицю X на матрицю Y. Для навчання моделі використаємо навчальні множини X1 та Y1. Якість роботи моделі оцінюватиметься на тестових множинах X2 та Y2.

$$X_1 = \begin{pmatrix} x_{11} \dots x_{n1} \\ \dots \cdots \dots \\ x_{1m} \dots x_{nm} \end{pmatrix} \quad Y = \begin{pmatrix} y_{11} \\ \dots \\ y_{m1} \end{pmatrix} \tag{2.2}$$

$$X_2 = \begin{pmatrix} x'_{11} \cdots x'_{n1} \\ \dots \cdots \dots \\ x'_{1m} \cdots x'_{nm} \end{pmatrix} \quad Y = \begin{pmatrix} y'_{11} \\ \dots \\ y'_{m1} \end{pmatrix} \tag{2.3}$$

Мета машинного навчання – знайти функцію f*f*, яка задовольняє умову f(X1)=Y1*f*(*X*1)=*Y*1. Ефективність моделі на нових даних оцінюється за допомогою коефіцієнта відхилення між Y′*Y*′ та Y2*Y*2, де Y′=f(X2)*Y*′=*f*(*X*2). Тестові дані X2*X*2 повинні максимально відрізнятися від навчальних даних X1*X*1. Нехай коефіцієнт точності (accuracy) позначений як a∈[0;1]*a*∈[0;1]. Формула для розрахунку коефіцієнта відхилення:

$$a = \frac{\text{Кількість правильно класифікованих даних}}{\text{Загальна кількість даних}} \tag{2.4}$$

Якщо модель має низьку точність, це означає, що вона часто робить помилки при класифікації даних. З іншого боку, висока точність вказує на ефективну роботу моделі. Хоча коефіцієнт точності є базовим показником, у науковому аналізі часто використовуються більш складні метрики, такі як F-міра, точність (precision) та повнота (recall).

$$\text{Recall} = \text{Повнота} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2.5)$$

$$\textit{P}\text{recision} = \text{Влучність} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positieves}} \quad (2.6)$$

$$f_{measure} = f_{\text{міра}} = 2\frac{\text{Влучність} * \text{Повнота}}{\text{Влучність} + \text{Повнота}} \quad (2.7)$$

Ключові показники якості класифікації:

- Істинно позитивні *(True Positives)* – це кількість випадків, коли система класифікує об'єкт як позитивний, і це дійсно відповідає істині.
- Істинно негативні (*True Negatives*) – ситуації, коли модель правильно визначає об'єкт як негативний.
- Хибно позитивні (*False Positives*) – випадки, коли система вважає об'єкт позитивним, хоча він насправді є негативним.
- Хибно-негативні (*False Negatives*) – це випадки, коли модель неправильно класифікує позитивний об'єкт як негативний.
- Повнота (*Recall*) вимірює, який відсоток фактично позитивних результатів було правильно визначено моделлю.
- Точність (*Precision*) показує, який відсоток позитивних результатів, передбачених моделлю, дійсно є позитивними.
- F-міра (*F1-score*) об'єднує точність та повноту в один показник, надаючи загальну оцінку якості класифікації.

У дослідженні Гальчинського та Грайворонського [11] було проаналізовано ефективність різних моделей машинного навчання для класифікації:

- Алгоритм найближчих сусідів (k-Nearest Neighbors, KNN).
- Наївний баєсівський класифікатор (Naive Bayes).
- Метод випадкового лісу (Random Forest).
- Логістична регресія (Logistic Regression).
- Класифікаційне дерево рішень (Decision Tree).

У таблиці 2.1 представлено результати оцінки цих алгоритмів, використовуючи метрики точність, повнота та F-міра на різних наборах даних.

Таблиця 2.1 – Результати оцінки цих алгоритмів, використовуючи метрики точність, повнота та F-міра на різних наборах даних.

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

2.2 Методика дерев рішень

Одним із підходів у машинному навчанні є метод дерев прийняття рішень (Decision Trees). Цей метод вартий уваги, адже саме на його основі створюється більш складна модель – ліс рішень (Random Forest). Щоб краще зрозуміти, як працює дерево рішень, доцільно розглянути простий приклад.

У нас є сім кущів, і кожен має свої особливості. Тепер ми можемо працювати з цими кущами як з об'єктами в задачі класифікації або створити

умовну модель оцінки врожайності, смаку, адаптивності до клімату тощо. Наприклад, можна уявити, що ці характеристики – це "ознаки", а ми хочемо класифікувати, які кущі найкраще підходять для вирощування в заданих умовах.

Можемо навіть побудувати дерево рішень, щоб визначати за набором властивостей, до якого типу належить кущ. Хочеш спробувати? Я можу допомогти побудувати умовну модель або придумати історію, як ці кущі використовуються в практиці. Спираючись на ці властивості, можна поставити серію запитань, на які відповідь буде лише "так" або "ні". Суть роботи дерева полягає у поетапному розподілі об'єктів за відповідями на подібні запитання – саме так алгоритм приймає рішення. Це нагадує повсякденний вибір, наприклад, між ложкою і виделкою – рішення ґрунтується на оцінці кількох простих критеріїв.

Отже, маючи набір даних із різними типами кущів, ми можемо виділити певні вимірювані параметри, які дозволяють згрупувати ці об'єкти за схожими ознаками. Системі, як і людині, не потрібно знати точну назву об'єкта, щоб здійснити класифікацію – достатньо базуватись на спостережуваних відмінностях.

Однією з перших і найпростіших характеристик, яка привертає увагу, є колір ягід. У нашому прикладі присутні кущі з червоними та синіми ягодами. Це дозволяє поставити перше запитання: «Чи є ягоди на кущі червоного кольору?». Відповідь на нього вже дозволяє поділити всі кущі на дві основні групи. Якщо поглянути на одну з цих груп (припустимо, ту, де ягоди сині), можна побачити, що всі елементи в ній ідентичні – класифікацію завершено. Інша ж група (з зеленими ягодами) передбачає додаткового аналізу. Наступною важливою ознакою може стати розмір ягід. Вимірявши діаметри та обчисливши середнє значення, ми можемо розмежувати кущі ще на дві підгрупи: з ягодами більшого та меншого розміру.

Хоча ми інтуїтивно користуємось подібними механізмами щодня, не замислюючись над їх формальною структурою, дерево рішень відтворює ці процеси логічно та послідовно. Завдяки таким простим критеріям, як колір або

розмір, алгоритм здатен ефективно класифікувати об’єкти, як у нашому прикладі з ягодами.

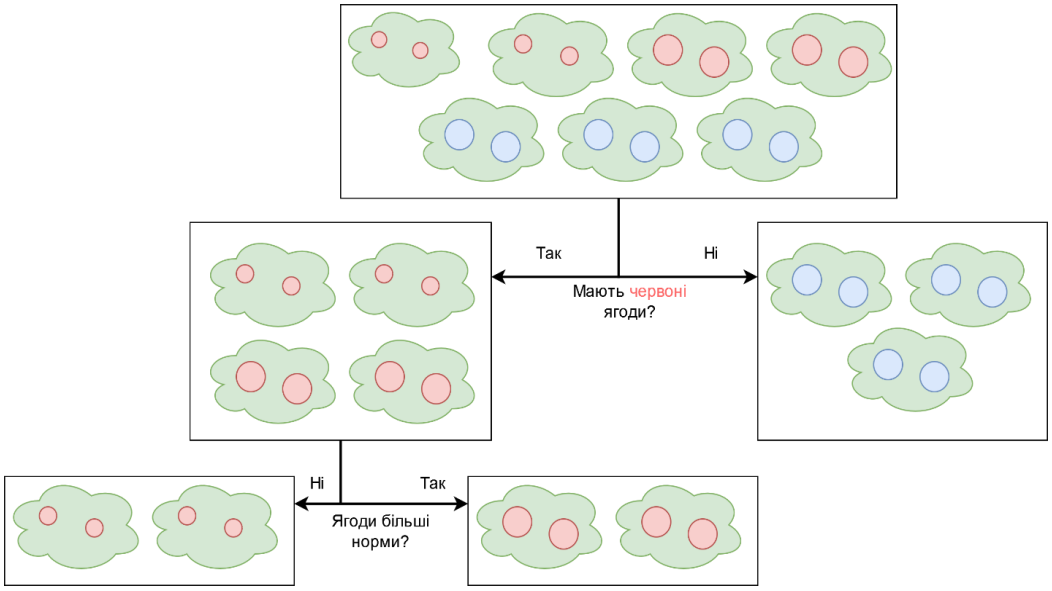


Рисунок 2.1 – Візуалізація структури дерева рішень.

2.3 Паралельна реалізація Random Forest для ефективного виявлення мережевих загроз

Random Forest базується на використанні багатьох дерев рішень, побудованих з певними відмінностями. Алгоритм Random Forest дійсно складається з великої кількості окремих дерев рішень, кожне з яких самостійно приймає рішення на основі випадкової вибірки даних і ознак. Кожне дерево є слабким класифікатором, але в сукупності – "ліс" дає сильне й стійке рішення.

Кожне дерево передбачає клас, і клас, який отримує найбільшу кількість "голосів", визначає загальний прогноз моделі. Схематичне зображення цього процесу представлено на рисунку 2.2.

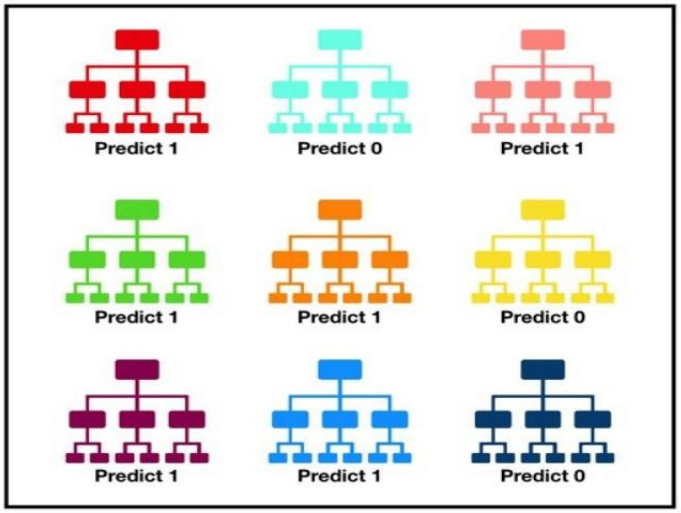


Рисунок 2.2 – Принцип колективного ухвалення рішення деревами в алгоритмі Random Forest

Методу випадкових лісів базується на принципі колективного розуму: об’єднання багатьох незалежних, хоча і відносно простих, моделей (дерев рішень) створює потужнішу й надійнішу загальну модель. Такий підхід забезпечує високу точність і стійкість, оскільки кожне дерево приймає рішення самостійно, а фінальний прогноз формується через голосування всіх дерев.

Ключовим фактором ефективності є низький рівень кореляції між окремими деревами. Цей принцип нагадує роботу збалансованого інвестиційного портфеля, у якому поєднано фінансові інструменти з різними рівнями кореляції (наприклад, цінні папери та державні облігації). Завдяки такому підходу знижується загальний рівень ризику й підвищується стабільність доходу, оскільки несприятливі коливання одного активу можуть бути компенсовані стабільністю іншого.Так само і в випадковому лісі: якщо одне дерево помиляється, інші можуть компенсувати цю помилку, формуючи загальний правильний результат. Навіть якщо частина дерев дає хибні прогнози, загальна система працює точніше, ніж будь-яке окреме дерево.

Щоб модель працювала ефективно, необхідно виконати дві умови:

- Дані повинні містити реальний сигнал, а не випадковий шум.
- Щоб підвищити загальну точність моделі, важливо забезпечити мінімальну подібність результатів окремих дерев у складі ансамблю. Інакше кажучи, необхідно зменшити взаємну залежність (кореляцію) між прогнозами

дерев.

Досягти цієї незалежності дозволяють два ключові механізми:

- Навчання на випадкових підмножинах даних. Кожне дерево навчається на окремому випадковому наборі з основного масиву даних із повтореннями (так званий бутстрепінг). Наприклад, із масиву [1, 2, 3, 4, 5, 6] одне дерево може отримати [1, 2, 4, 4, 4, 6], інше – [1, 1, 3, 3, 4, 6] і т.д. Таким чином дерева бачать подібні, але не однакові набори прикладів, що сприяє їх різноманітності.
- Випадковий вибір ознак при поділі. Кожне дерево під час побудови використовує лише випадкову підмножину доступних ознак на кожному кроці поділу. Наприклад, якщо маємо 4 ознаки, одне дерево може працювати лише з ознаками 2 і 3, а інше – з 1 і 4 (див. рисунок 2.3). Це запобігає утворенню ідентичних дерев і сприяє зниженню кореляції між ними.

Завдяки цим двом підходам випадковий ліс формує узгоджений, точний і стійкий до шуму прогноз, який ґрунтується на багатьох незалежних судженнях.

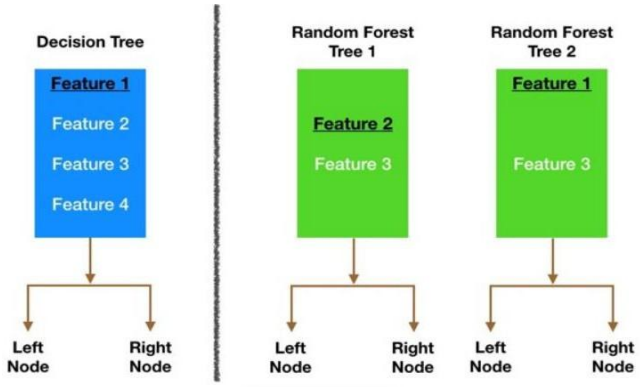


Рисунок 2.3 – Процес вибору критеріїв для побудови окремих дерев

Використовуючи описані вище дерева, можна побудувати "ліс" з відносно незалежними деревами, кожне з яких може приймати власне рішення на основі вхідних даних.

2.4 Паралельна обробка: переваги та обмеження

Паралельна обробка даних означає здатність комп'ютерної системи

виконувати обчислення одночасно в кількох потоках. Ця можливість характерна для більшості сучасних комп’ютерів, що мають принаймні два процесорні ядра.

До основних позитивних аспектів використання паралельного обчислення належать:

- помітне зменшення часу, необхідного для виконання обчислень або навчання моделей машинного навчання;
- підвищення продуктивності, що особливо актуально при роботі з великими масивами даних.

Водночас паралельна обробка має свої виклики:

- збільшується споживання апаратних ресурсів, оскільки декілька потоків функціонують одночасно;
- ускладнюється управління потоками, що вимагає додаткової координації та синхронізації;
- розробка багатопотокових програм є технічно складнішою та потребує глибших знань щодо конкурентного виконання.

У типових сценаріях машинного навчання паралельно реалізуються два критично важливі етапи: процес навчання моделі та етап прогнозування (інференції). Перший з них – побудова класифікатора – може займати значний час, особливо коли навчальний датасет має велику кількість прикладів. Час навчання зростає пропорційно до обсягу даних, що потребує застосування методів прискорення.

Рисунок 2.4 ілюструє багатопотокову реалізацію побудови моделі. На ньому зображено приклад із трьома незалежними потоками, кожен з яких отримує однакові дані та застосовує той самий алгоритм формування Random Forest. Завдяки різним випадковим ініціалізаціям структура дерев у кожному потоці відрізняється, що гарантує різноманітність та незалежність рішень. Об’єднання результатів усіх потоків дозволяє сформувати єдину ансамблеву модель, яка поєднує в собі унікальні дерева з різною структурою, проте спільними вхідними характеристиками.

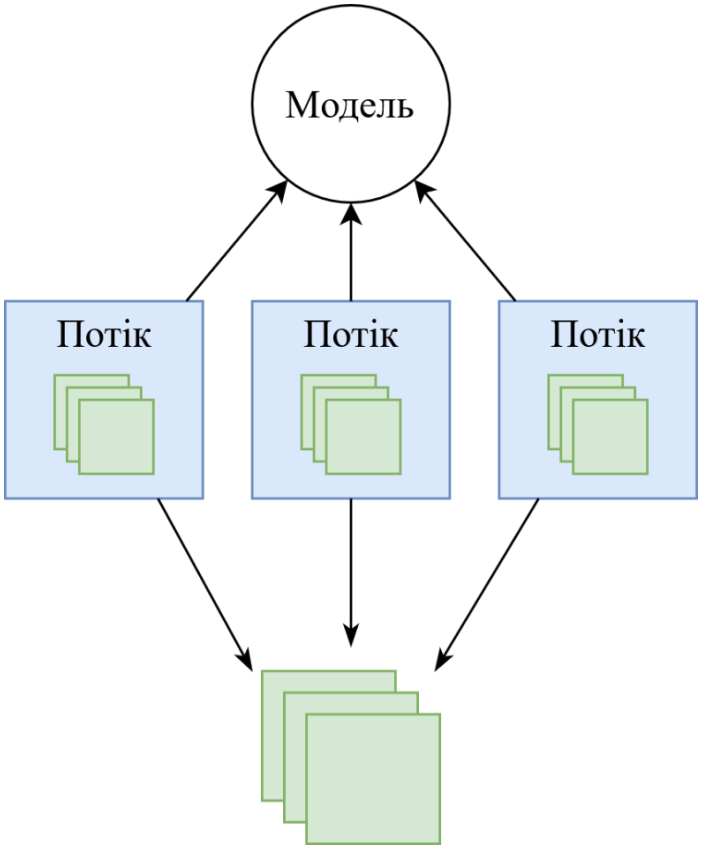


Рисунок 2.4 – Багатопотокова реалізація побудови моделі

У випадку, коли створення моделі не є основною метою, оскільки воно зазвичай здійснюється заздалегідь із застосуванням перевірених наборів даних, основна увага переміщується на швидкість класифікації інформації. Саме оперативність цього етапу є критичною для своєчасного виявлення загроз. Організація паралельного виконання завдань можлива як мінімум двома підходами.

Один із них передбачає використання однієї моделі, яка застосовується до різних частин вхідних даних. У цьому підході потоки існують поза самою моделлю, працюючи з її копією та окремими фрагментами даних (див. рисунок 2.5). Весь масив інформації ділиться на однакові частини, які розподіляються між паралельними потоками. Після завершення їх обробки результати об'єднуються в єдиний масив відповідей, що представляє собою структурований стовпець класифікаторних рішень.

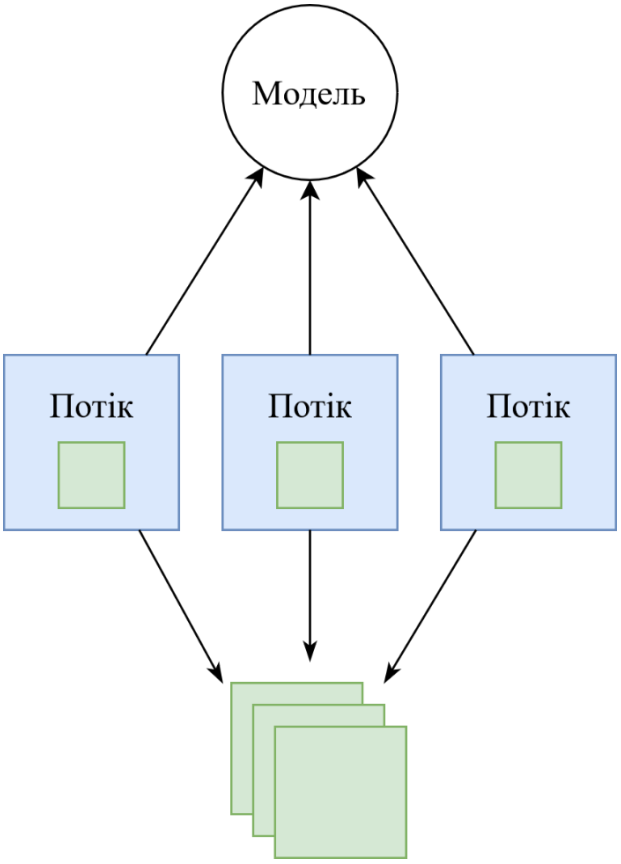


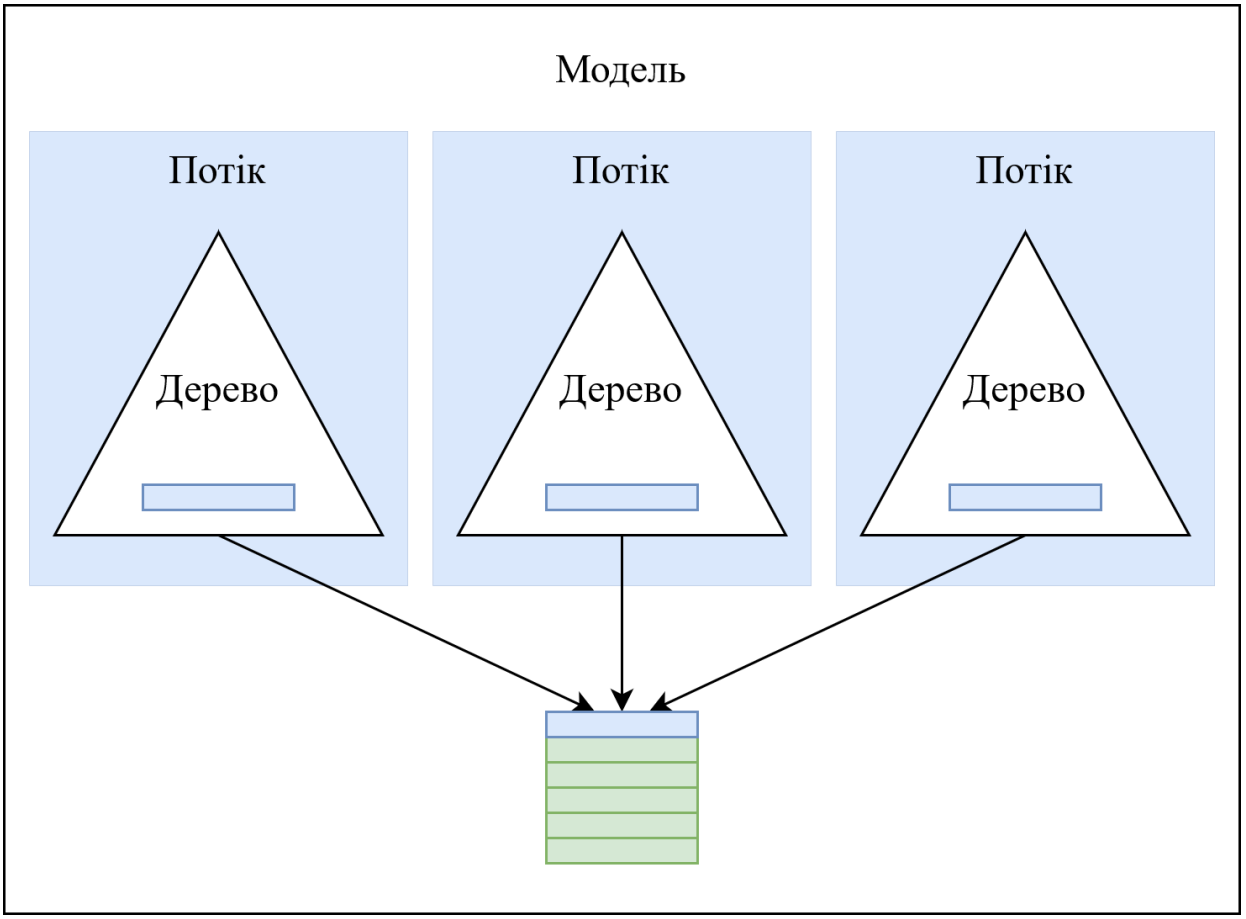
Рисунок 2.5 – Базовий варіант класифікації даних на основі одного потоку

Другий підхід до паралельної реалізації класифікатора передбачає інтеграцію багатопоточності безпосередньо у внутрішню архітектуру моделі, що супроводжується модифікацією функціональних можливостей використовуваної бібліотеки. Суть цього методу полягає в тому, що для кожного дерева класифікації виділяється окремий потік, у межах якого дерево автономно виконує свої обчислення, працюючи синхронно з іншими деревами.

Як зображено на рисунку 2.6, усі дерева мають спільний доступ до одного набору вхідних даних, які вони обробляють паралельно. Після завершення класифікації результати від кожного потоку синхронізуються, і на основі колективного голосування формується остаточне рішення про належність об’єкта до певного класу. Однак, одним із суттєвих недоліків такого підходу є можливість виникнення затримок: якщо хоча б одне дерево виконує обчислення повільніше за інші, це може призвести до блокування загального процесу ухвалення рішення, тим самим знижуючи продуктивність моделі в цілому.

Альтернативна стратегія полягає у формуванні кожним деревом окремого вектора рішень незалежно від роботи інших. Після того як усі дерева

завершують обробку вибірки, їх відповіді – представлені у вигляді векторів (Y_i) – об’єднуються у фінальний результат. Хоча такий підхід потенційно вимагає більших обсягів оперативної пам’яті, він може забезпечити вищу швидкість обробки в умовах великого навантаження та кількості потоків.



Ричунок 2.6 – Приклад багатопоточної класифікації.

Слід зазначити, що виділення окремого потоку на кожне дерево у складі Random Forest не завжди є оптимальним рішенням. У певних випадках доцільніше застосовувати об’єднання дерев у групи, які обслуговуються окремими потоками. Наприклад, замість створення 100 потоків для 100 дерев, доцільно організувати 10 потоків, кожен з яких паралельно обробляє групу з 10 дерев – як це умовно продемонстровано на рисунку 2.7. Такий підхід дозволяє більш раціонально використовувати апаратні ресурси, зменшити витрати на синхронізацію й уникнути перевантаження системи через надмірну кількість одночасно активних потоків.

При цьому визначення оптимальної кількості потоків для кожної

конкретної задачі класифікації залежить від ряду факторів – зокрема, від потужності доступного процесора, розміру даних, конфігурації моделі та типу обробки. Це питання потребує емпіричного дослідження та тестування, з урахуванням продуктивності в обраних умовах.

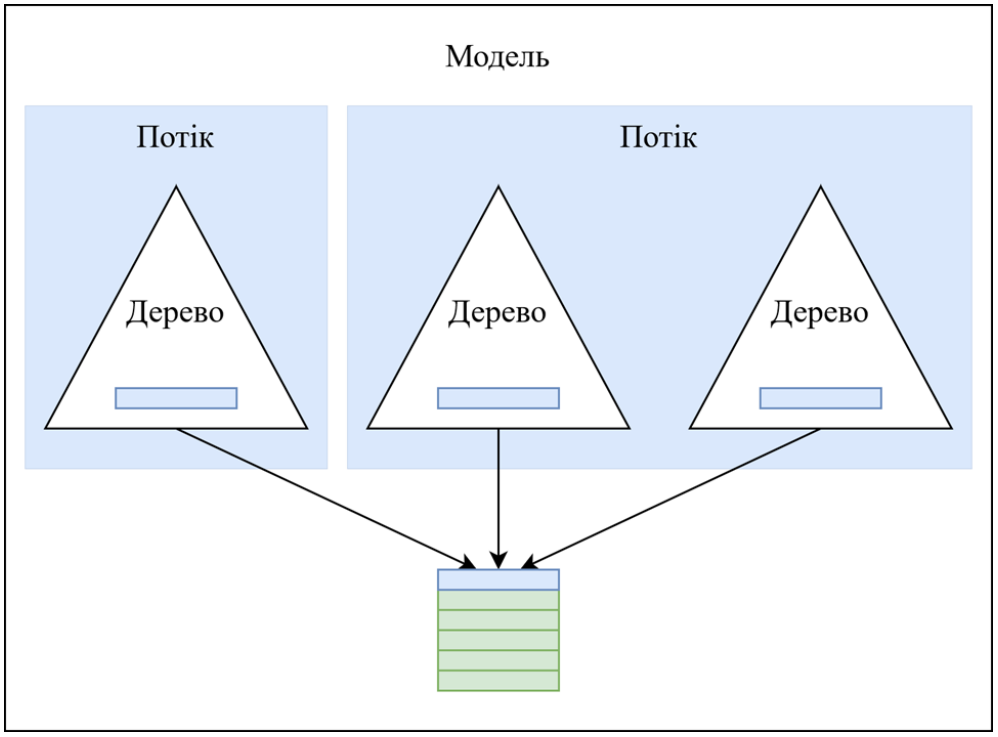


Рисунок 2.7 – Групування дерев у потоки під час паралельної обробки в моделі Random Forest

2.5 Розподілені обчислення: сильні та слабкі сторони

Концепція розподілених обчислень базується на використанні кількох незалежних обчислювальних вузлів, які спільно працюють над вирішенням однієї комплексної задачі. У контексті алгоритму RandomForest, що реалізований у Python-бібліотеці scikit-learn, наразі підтримується паралельна обробка даних на одному пристрої. Однак для роботи з великими обсягами інформації доцільно розширити можливості алгоритму, застосовуючи кілька фізичних машин для підвищення продуктивності.

Переваги цього підходу включають:

- Прискорене виконання обчислень завдяки розподілу навантаження.
- Масштабованість, що дозволяє обробляти більші обсяги даних за

рахунок додавання нових серверів.

- Зростання стійкості системи до збоїв – архітектура без єдиного критичного елементу, вихід з ладу якого міг би повністю припинити роботу всієї моделі.
- Резервування інформації через реплікацію даних у межах кластера.

Проте є й недоліки:

- Ускладнене технічне обслуговування та пошук причин збоїв.
- Складніша архітектура розробки й інтеграції.

У контексті систем, що здійснюють аналіз мережевого трафіку в режимі реального часу, надзвичайно важливо забезпечити їхню здатність або працювати безпосередньо на маршруті проходження трафіку (inline), миттєво реагуючи на підозрілі пакети, або взаємодіяти з окремим компонентом, який на основі отриманих класифікаційних результатів ініціює відповідні контрдії.

Фундаментальною вимогою до таких платформ є забезпечення неперервного оброблення вхідного потоку даних без затримок або втрати продуктивності. Для досягнення високої ефективності розподілу навантаження між обчислювальними вузлами, як правило, застосовуються спеціалізовані балансувальні механізми, що розподіляють трафік згідно з певними критеріями – наприклад, обсягом потоку, джерелом запитів, типом протоколу або рівнем пріоритету пакетів.

- Циклічне надання даних кожному обробнику.
- Врахування кількості вже оброблених пакетів.
- Динамічний розподіл на основі швидкості обробки конкретного вузла.

Кожен із підходів має свої сильні та слабкі сторони. Одні з найефективніших реалізацій таких механізмів пропонуються компанією F5.

Рисунок 2.8 схематично зображено механізм надходження трафіку до обробників. Серед основних елементів: потік вхідних даних, керуючий вузол, що координує процес, та виконавчі одиниці – робітники.

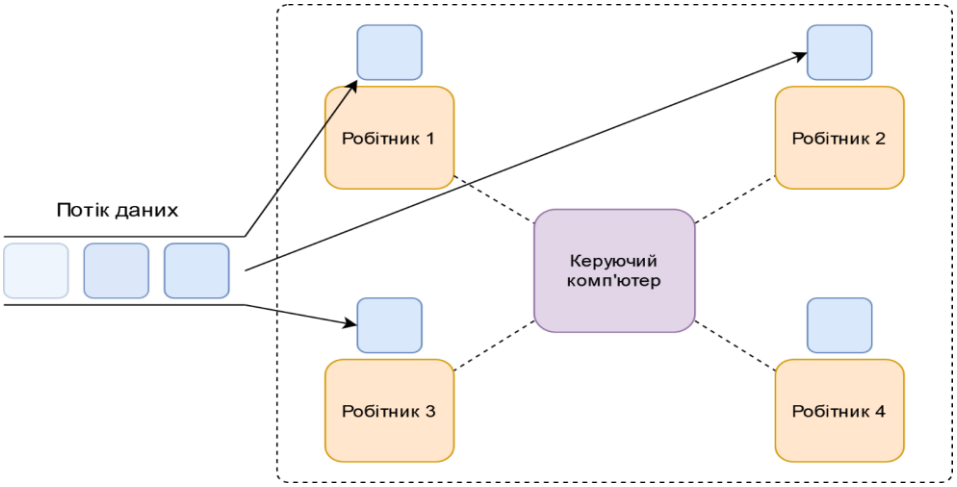


Рисунок 2.8 – Механізм надходження трафіку до обробників

Запропонована модель розподіленої обробки розглядається як потенційно ефективний напрямок для реалізації. Проте в межах цієї дипломної роботи її впровадження не передбачається через обмеження обсягу та практичної реалізації. Водночас дана концепція становить значний науковий інтерес і може бути всебічно розглянута в рамках магістерської дисертації на наступному етапі дослідження.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОЇ ОБРОБКИ В АЛГОРИТМІ РАНДОМ ФОРЕСТ

З метою реалізації завдань, визначених у межах дослідження, було обрано мову програмування Python версії 3.10.0. Цей інструмент відзначається високою гнучкістю та функціональністю, що робить його придатним для створення комплексних програмних рішень. Значною перевагою Python є наявність широкого спектра готових модулів та бібліотек, які спрощують процес розробки та дозволяють зосередитися на логіці алгоритму. Особливу роль у цьому проєкті відіграє бібліотека scikit-learn, що надає зручний інтерфейс для реалізації алгоритмів машинного навчання і повністю відповідає вимогам поставленої задачі.

3.1 Підготовка вхідних даних

Джерелом вхідних даних для реалізації експериментальної частини дослідження виступає офіційний репозиторій Канадського інституту кібербезпеки (Canadian Institute for Cybersecurity) [4]. Надані цим інститутом набори даних є авторитетними у професійному середовищі, оскільки моделюють сучасні сценарії DDoS-атак із високою ступінню відповідності реальному трафіку. Завдяки цьому вони широко використовуються як у наукових дослідженнях, так і у практичних розробках систем кіберзахисту в усьому світі.

На офіційному сайті інституту доступні різні формати даних – від PCAP-файлів до вже оброблених CSV-таблиць, готових до подальшого аналізу. Особливу цінність цієї інформації становить наявність великої кількості атрибутів, що дозволяють ефективно ідентифікувати DDoS-атаки. Повний список полів, що входять до відповідного датасету, наведено у Додатку [1].

Для подальшої роботи над моделюванням було обрано набір CIC-DDoS2019 (DDoS Evaluation Dataset). Однак перш ніж передати дані до алгоритмів машинного навчання, потрібно провести низку попередніх етапів обробки:

- Видалити атрибути, специфічні для конкретного прикладу атаки, як-от IP-адреси чи MAC-ідентифікатори.
- Провести оцінку інформативності кожного поля, залишивши лише ті стовпці, що дійсно сприяють визначенню атак.
- Забезпечити однорідність у кожному стовпці: виявити і виправити помилкові або порожні значення, символи NULL, +infinity, -infinity, а також неприпустимі текстові значення у числових колонках.
- Здійснити кодування категоріальних змінних: наприклад, значення “DNS”, “SYN”, “UDP”, “NORMAL” можуть бути перетворені у числові значення – 1, 2, 3, 4 відповідно.

Для оцінки ефективності паралельної та послідовної реалізацій моделі класифікації планується зібрати наступні метрики:

- Час побудови моделі.
- Час класифікації вхідних даних.
- Розміри навчального набору.
- Кількість ознак (стовпців) і об’єктів (рядків), що використовуються під час тренування та тестування моделі.

Таблиця 3.1 – Перелік усіх файлів, що були використані у дослідженні, а також інформацію про їх розміри.

Назва файлу	Розмір
Portmap.csv	76 769 КБ
Small-Syn.csv	286 751 КБ
Syn.csv	1 833 372 КБ
AppDDos.pcap_Flow.csv	144015 КБ

3.2 Аналіз побудови класифікаційної моделі для обробки файлів

Розроблене програмне забезпечення, призначене для вимірювання ключових метрик продуктивності моделі, буде застосоване по черзі до кожного

з файлів, що входять до вибраного набору даних. Аналіз розпочнемо з файлу Portmap.csv.

У ході обробки цього файлу буде здійснено послідовне вимірювання наступних параметрів:

- час, витрачений на побудову моделі;
- час, необхідний для класифікації даних;
- обсяг даних у наборі (кількість рядків і стовпців);
- продуктивність моделі при послідовній та багатопоточній обробці.

На рисунках 3.1 та 3.2 подано відповідні візуалізації отриманих статистичних характеристик, що дозволяють наочно оцінити ефективність запропонованих підходів.

```
(env) C:\University\4course\Diploma\code>python paralel_forest.py 03-11\Portmap.csv
INFO:root:Using 03-11\Portmap.csv
INFO:root:Column 'Unnamed: 0' dropped successfully
INFO:root:Column 'Flow ID' dropped successfully
INFO:root:Column 'Source IP' dropped successfully
INFO:root:Column 'Destination IP' dropped successfully
INFO:root:Column 'Timestamp' dropped successfully
INFO:root:Column 'SimillarHTTP' dropped successfully
WARNING:root:column 'Dst IP' was not deleted.
WARNING:root:column 'Src IP' was not deleted.
INFO:root:Column 'Fwd Header Length.1' dropped successfully
INFO:root:Column 'Inbound' dropped successfully
WARNING:root:'Label' Failed fill 'None' value with median
Training Data:
(153355, 79)
(153355, 1)
Testing Data:
(38339, 79)
(38339, 1)
INFO:root:---Creating Forest Synchronycally---
INFO:root:Forest is ready. Trees:100
INFO:root:Score: 0.9997913351939278
INFO:root:time measurment: 4.679509700003109
INFO:root:---Creating Forest in Paralel---
INFO:root:Forest is ready. Trees:100
INFO:root:Score: 0.9997913351939278
INFO:root:time measurment: 2.687598699998489
INFO:root:---Predicting Forest Synchronycally---
INFO:root:time measurment: 0.16387070000200765
INFO:root:---Predicting Forest in Paralel---
Threads: 2. INFO:root:time measurment: 0.09658160000253702
Threads: 3. INFO:root:time measurment: 0.0863482999993721
Threads: 4. INFO:root:time measurment: 0.09712289999879431
Threads: 5. INFO:root:time measurment: 0.10727729999780422
Threads: 6. INFO:root:time measurment: 0.12418249999973341
Threads: 7. INFO:root:time measurment: 0.149307100000442
Threads: 8. INFO:root:time measurment: 0.163321600000927
Threads: 9. INFO:root:time measurment: 0.1742164999996021
Threads: 10. INFO:root:time measurment: 0.18539559999771882
Threads: 11. INFO:root:time measurment: 0.19602260000101523
Threads: 12. INFO:root:time measurment: 0.20802630000252975
Threads: 13. INFO:root:time measurment: 0.22149889999855077
Threads: 14. INFO:root:time measurment: 0.23020259999975679
Threads: 15. INFO:root:time measurment: 0.2536050000016985
Threads: 16. INFO:root:time measurment: 0.26074979999975767
Threads: 17. INFO:root:time measurment: 0.27403469999990193
Threads: 18. INFO:root:time measurment: 0.28499290000036126
Threads: 19. INFO:root:time measurment: 0.30726030000005267
Threads: 20. INFO:root:time measurment: 0.3100025999992795
```

Рисунок 3.1 – Візуалізація статистичних характеристик для файлу Portmap.csv

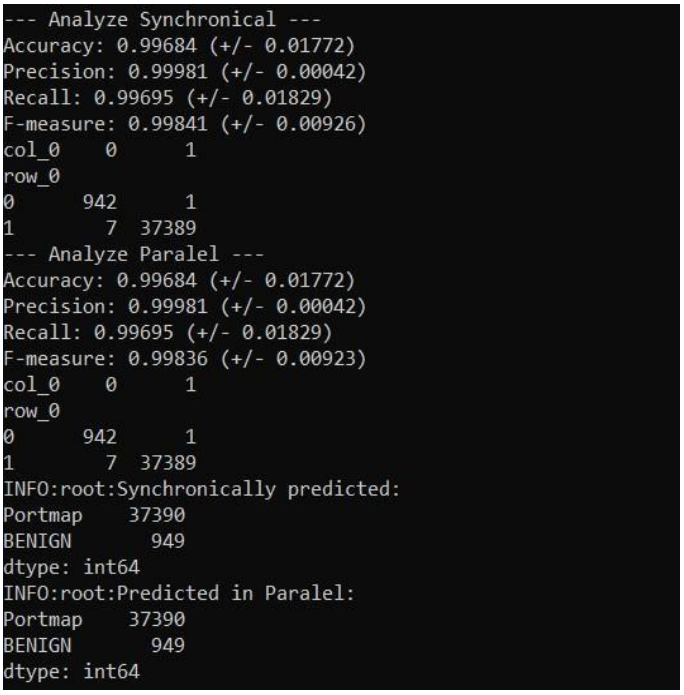


Рисунок 3.2 – Аналіз обробки даних *Portmap.csv* та оцінка класифікації

На даному етапі демонструється процес попередньої обробки вхідного файлу Portmap.csv, що виконується автоматизованою програмою згідно з методиками, описаними в першому розділі. Зокрема, було реалізовано:

- Видалення нерелевантних стовпців, які не мають впливу на класифікацію;
- Заповнення або уніфікація пропущених та некоректних значень, що могли спричинити помилки під час навчання моделі.

Система надає зворотний зв’язок щодо успішності виконаних дій. У деяких випадках певні стовпці не підлягали видаленню – це обумовлено тим, що програмне забезпечення адаптоване до обробки як наборів CIC-DDoS2019, так і новіших версій із можливими змінами у структурі даних.

Наступним етапом є розподіл набору даних:

- 80% записів призначено для навчання моделі (train);
- 20% – для валідації результатів (test).

У процесі класифікації формується дві матриці:

- X – вхідні атрибути;
- Y – мітки класів (цільова змінна).

Режим виконання фіксується програмою:

- У послідовному режимі задіяно один потік.
- У паралельному варіанті кількість потоків варіюється з 2 до 20, що дозволяє оцінити вплив масштабування.

Для оцінки якості класифікації обчислюються наступні метрики:

- Accuracy (точність).
- Recall (повнота).
- Precision (влучність).
- F1-score (збалансований показник точності та повноти).

На рисунку наведено також матрицю невідповідностей (confusion matrix), яка візуалізує помилки класифікації та дозволяє глибше зрозуміти ефективність роботи моделі.

Рисунок 3.3 містить чисельні значення, на підставі яких розраховано вищевказані метрики.

		Actual values	
		Positive	Negative
Predicted values	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Рисунок 3.3 – Матриця помилок

Фінальним етапом аналізу кожного окремого файлу є виведення передбачених значень класифікатора. У випадку з *Portmap.csv* – це мітки Portmap (атака) та BENIGN (звичайний трафік). Такий підхід дозволяє чітко ідентифікувати результат класифікації та впевнитись у коректності роботи моделі.

Отримавши повне уявлення про структуру вихідних даних, які генерує розроблене програмне забезпечення, доцільно послідовно провести аналогічний аналіз решти файлів із набору.

На рисунках 3.3 та 3.4 представлено результати запуску розробленої програми для обробки файлу *Syn.csv*, який містить ознаки мережевої атаки типу SYN – різновиду DDoS-загроз. Проведений експеримент охоплює повний цикл обробки даних: від етапу попереднього аналізу та очищення, через поділ вибірки на тренувальну й тестову частини, до навчання моделі класифікації в обох режимах – послідовному та паралельному.

Особливу увагу приділено оцінці основних метрик точності класифікації, які дозволяють порівняти продуктивність і надійність обох підходів, а також підтвердити доцільність використання багатопотокової реалізації в умовах підвищеного навантаження.

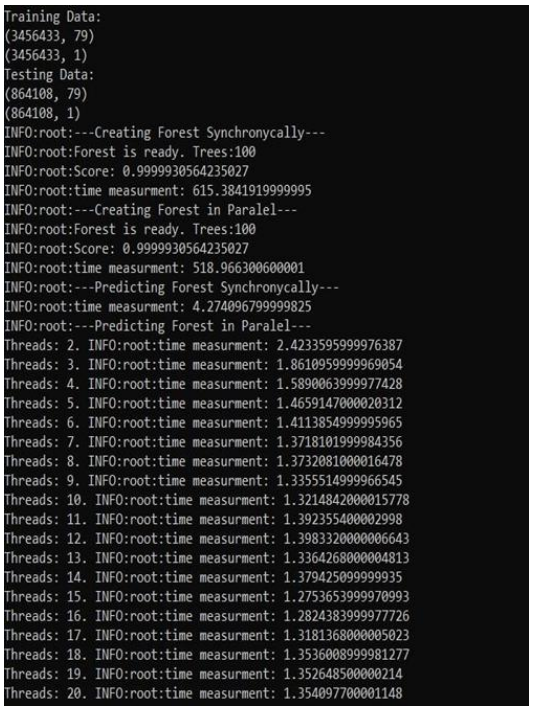


Рисунок 3.3 – Результати обробки набору Syn.csv: послідовна побудова моделі та оцінка ефективності

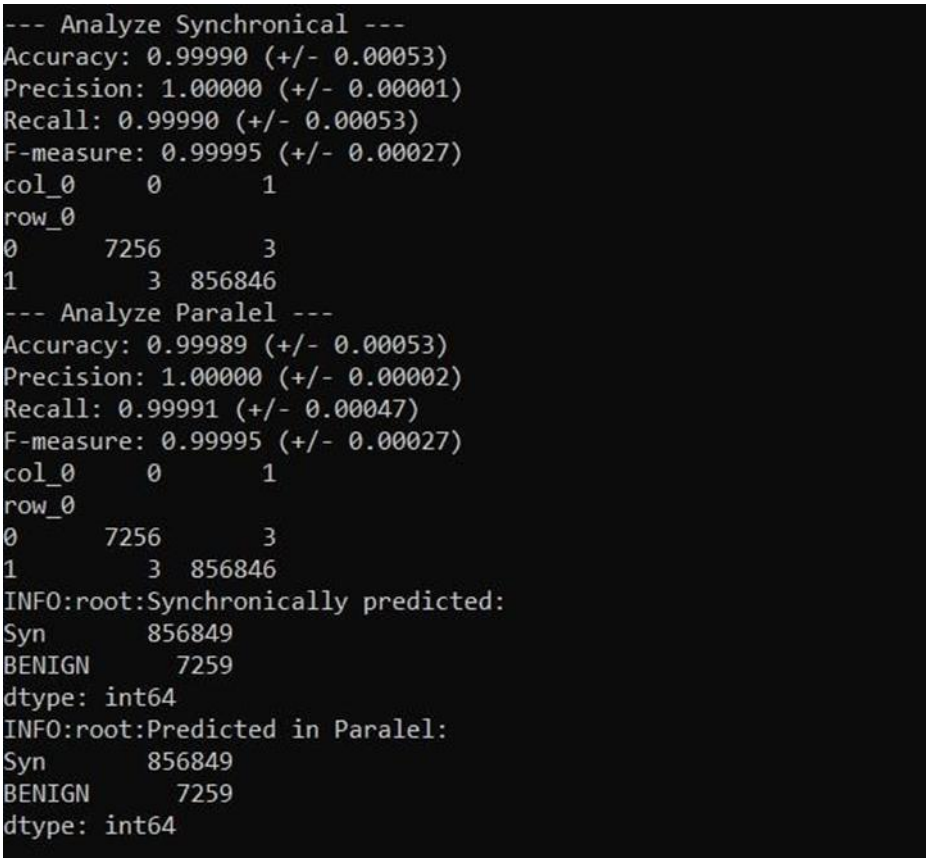


Рисунок 3.4 – Початковий аналіз файлу *Syn.csv*

У процесі проведення дослідження було отримано результати, наведені в таблиці 3.3, які безпосередньо стосуються етапу побудови моделей класифікації та відображають параметри їх навчання в різних режимах реалізації.

Таблиця 3.3 – Отримані результати

		Паралельне виконання			
Назва файлу	Кількість рядків	Час побудови	Повнота	Влучність	f-міра
Syn.csv	3456433	380с	0,99991	1,00000	0,99995
Small-Syn.csv	691286	35,4с	0,99952	0,99999	0,99977
Portmap.csv	153355	2,81с	0,99695	0,99981	0,99836
		Послідовне виконання			

Продовження таблиці 3.3

Назва файлу	Кількість рядків	Побудова моделі:	Повнота	Влучність	f-міра
Syn.csv	3456433	624c	0,99990	1,00000	0,99995
Small-Syn.csv	691286	82c	0,99950	0,99684	0,99975
Portmap.csv	153355	4,679c	0,99695	0,99981	0,99841

З таблиці 3.3 видно, що побудова моделі в один потік, тобто послідовно, займає більше часу ніж побудова моделі паралельно. Залежність часу побудови до кількості даних відображено на рисунку 3.5

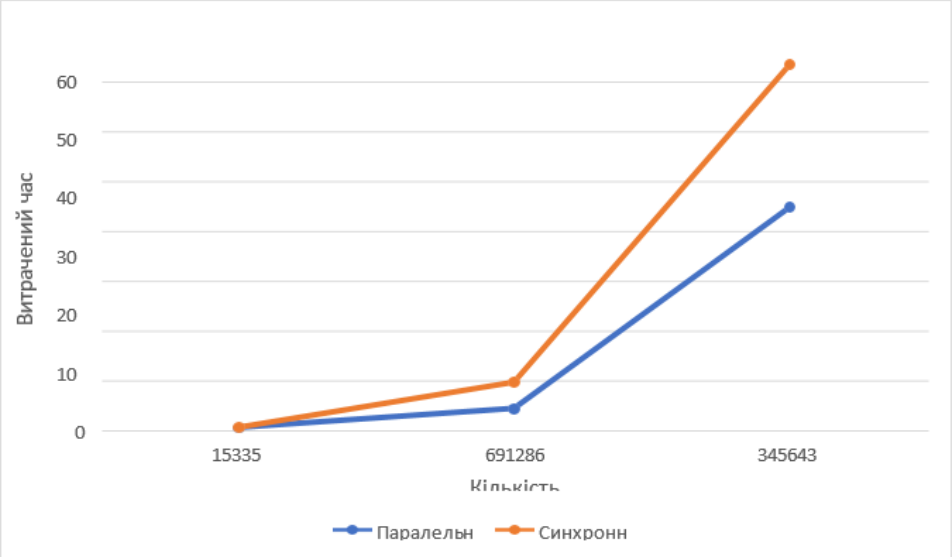


Рисунок 3.5 – Залежність часу побудови моделі від обсягу вхідних даних

На графіку, наведеному на рисунку 3.5, відображено, як змінюється час, необхідний для побудови моделі, відносно обсягу записів у датасеті. Спостерігається майже лінійне зростання витрат часу із збільшенням обсягу вхідних даних, що є очікуваним результатом з огляду на обчислювальну складність алгоритму RandomForest.

Для глибшого аналізу взаємозв’язку між розміром вибірки та часом побудови моделі було побудовано аналогічний графік у логарифмічному

масштабі (рисунок 3.6). Така візуалізація дає змогу виявити навіть незначні відхилення у швидкості зростання часу при різних розмірах вибірки.

Окрім того, графік також демонструє порівняння послідовного та паралельного виконання класифікації (із 10 активними потоками). Помітно, що використання багатопоточності забезпечує в середньому приблизно у 1.8 раза швидшу класифікацію, ніж при використанні одного потоку. Це свідчить про суттєву перевагу паралельної реалізації при роботі з великими масивами даних.

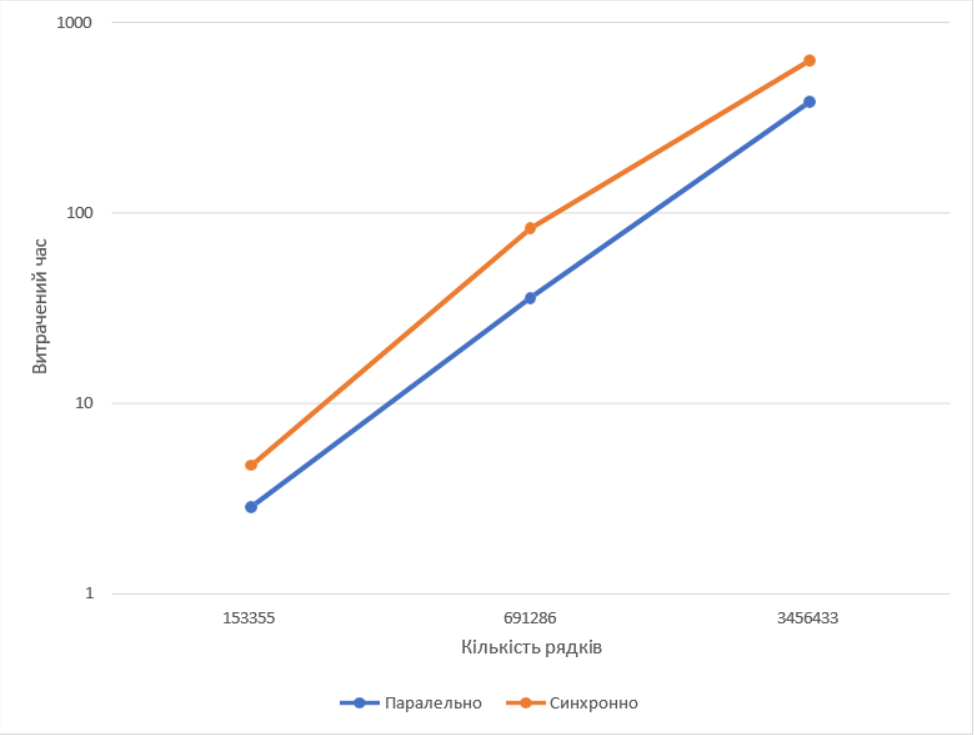


Рисунок 3.6 – Та сама залежність у логарифмічній шкалі

3.3 Оцінка результатів класифікації за допомогою створеної моделі

Оскільки ключовим завданням є оперативна класифікація даних за допомогою вже збудованої моделі, доцільно надати цій частині детальніший аналіз. На основі результатів, зображених на рисунках 3.1–3.4, можна зробити висновок, що кількість потоків безпосередньо впливає на швидкість класифікації: із зростанням кількості паралельних потоків час обробки змінюється не лінійно. Для зручності інтерпретації отримані значення зібрано у таблиці 3.4, яка відображає зміну часу виконання при різних режимах

багатопотоковості. Ці дані є основою для оцінки оптимальної конфігурації в умовах реального часу.

Попри те, що послідовна обробка (1 потік) демонструє загалом нижчі витрати часу, у деяких випадках багатопотокова реалізація (наприклад, із 4 або 10 потоками) перевершує її за ефективністю. Такі винятки слід враховувати при проектуванні системи для певних типів навантаження та обсягів даних.

Для наочного відображення цієї залежності було побудовано графік для файлу *Portmap.csv*, поданий на рисунку 3.7. Він ілюструє зміну часу класифікації в залежності від кількості потоків і підкреслює ефективність паралельного виконання на певних етапах.

Таблиця 3.4 – Отримані результати

Кількість потоків	Час		
	Portmap	Small-Syn	Syn
1	0,167113	1,0370759	4,274097
2	0,095513	0,6407790	2,423360
3	0,081043	0,4217110	1,861096
4	0,087132	0,3941999	1,589006
5	0,103936	0,3111512	1,465915
6	0,119567	0,3048949	1,411385
7	0,150357	0,3484710	1,371810
8	0,156191	0,3308401	1,373208
9	0,166884	0,3339860	1,335551
10	0,185908	0,3689111	1,321484
11	0,197642	0,3421322	1,392355
12	0,211195	0,3592047	1,398332
13	0,230281	0,3687290	1,336427
14	0,246230	0,3787009	1,379425
15	0,250359	0,4078883	1,275365
16	0,259985	0,4069255	1,282438
17	0,272923	0,4367646	1,318137
18	0,286298	0,4522401	1,353601
19	0,299016	0,4853544	1,352649
20	0,311642	0,7501998	1,354098

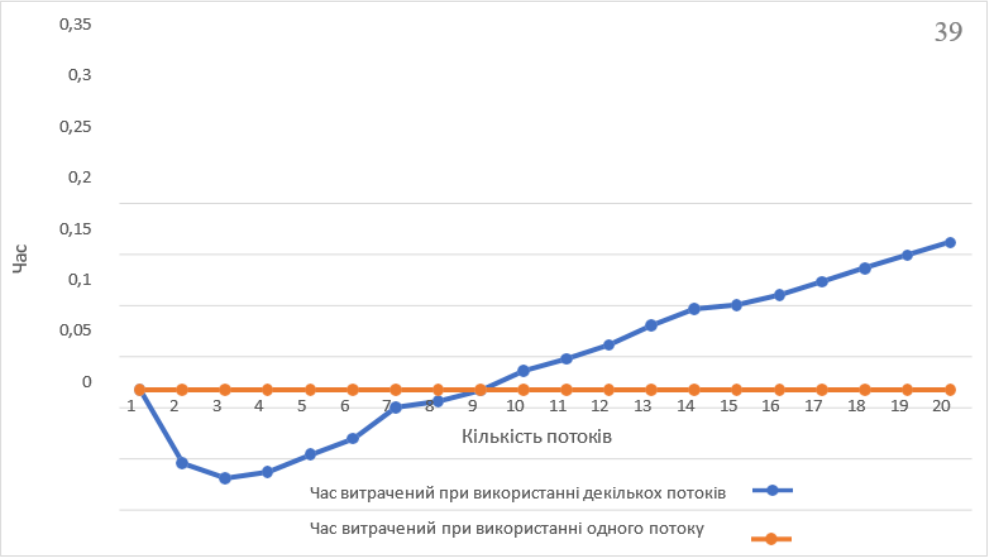


Рисунок 3.7 – Зміна часу класифікації в залежності від кількості потоків і підкреслює ефективність паралельного виконання на певних етапах.

Аналіз показує, що при зростанні кількості потоків час обробки спершу зменшується, але з певного моменту стабілізується, а подальше збільшення потоків не дає суттєвого покращення швидкодії. Як і у попередніх експериментах, можна відзначити, що після досягнення певного порогу продуктивності кількість додаткових обчислювальних операцій компенсує приріст ефективності класифікації. Це означає, що оптимальна кількість потоків для роботи з даними цього файлу знаходиться в певному діапазоні, після якого зростання кількості потоків призводить до збільшення часу класифікації.

Portmap – це найменший файл у вибірці, тому його аналіз дає змогу оцінити ефективність класифікації на невеликих обсягах даних. Як видно з графіка, найшвидше класифікація виконується при використанні 3 потоків, що вказує на їхню оптимальність для обробки малих наборів даних. При збільшенні кількості потоків до 9, швидкість класифікації зрівнюється з додатковими операціями, необхідними для обробки отриманих результатів. Це означає, що паралельне виконання при такій кількості потоків не дає переваги, адже загальний час виконання залишається незмінним. При подальшому збільшенні потоків час класифікації поступово зростає, що свідчить про досягнення межі ефективності багатопотокової обробки для цього конкретного

файлу. Отже, використання 3–4 потоків є доцільним при обробці малих обсягів даних. Наступний файл, використаний для тестування,–Small-Syn.csv. Його результати графічно представлені на рисунку 3.8.

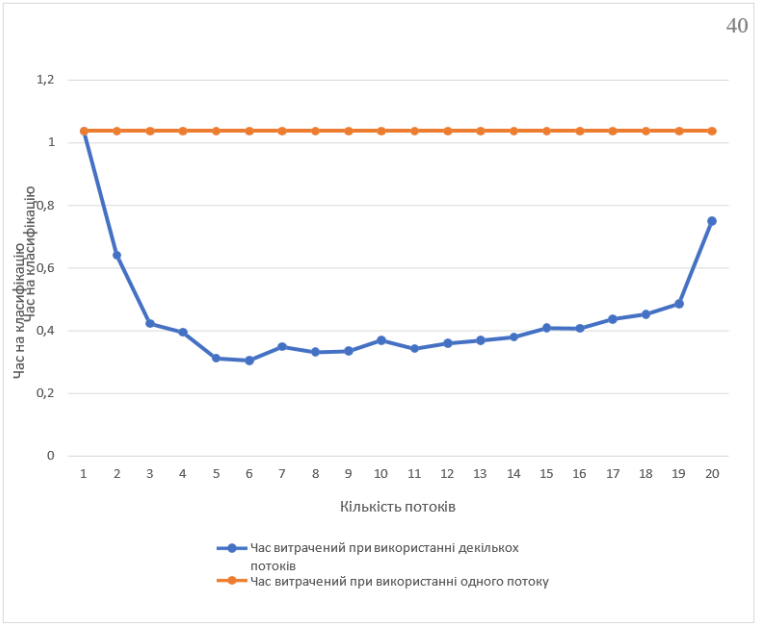


Рисунок 3.8 – Взаємозв’язок між часом, витраченим на класифікацію, та кількістю задіяних потоків при обробці файлу Small-Syn.csv.

Як свідчать результати, оптимальна продуктивність спостерігається при використанні шести потоків – саме за цієї конфігурації досягнуто найменшої тривалості обробки. Подальше збільшення кількості потоків не дає суттєвої переваги за швидкістю, що може вказувати на наближення до межі ефективного розпаралелювання в умовах даного обсягу вхідних даних.

Отримані результати можуть слугувати орієнтиром під час вибору параметрів системи для роботи з аналогічними за структурою і розміром наборами даних.

Рисунок 3.9 – Взаємозалежність між часом класифікації та кількістю потоків при обробці датасету Syn.csv.

Дослідження, проведене на найбільшому файлі, дозволяє виявити низку цікавих закономірностей. Зокрема, у початковій фазі збільшення кількості потоків супроводжується помітним скороченням часу обробки, проте надалі графік стабілізується – подальше нарощування потоків не призводить до істотного приросту швидкодії.

У межах діапазону від 8 до 20 потоків спостерігається сталий рівень часу виконання, незалежно від збільшення паралельності. Водночас, беручи до уваги результати попередніх експериментів, можна припустити, що надмірне зростання кількості потоків призведе до зворотного ефекту – тобто зростання загального часу виконання через перевантаження системи, синхронізаційні затримки або конфлікти доступу до ресурсів. Це дозволяє зробити висновок про існування межі доцільності паралельного масштабування, яку варто враховувати при налаштуванні системи для обробки великих наборів даних.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Охорона праці

Професійна діяльність фахівця з кібербезпеки в рамках даної кваліфікаційної роботи пов'язана з тривалою, інтенсивною роботою за персональним комп'ютером, що вимагає високої концентрації та глибоких знань. Ця спеціальність передбачає систематичний аналіз вразливостей вебсайтів, включаючи проведення комплексних пентестів, використання спеціалізованих інструментів, таких як Burp Suite для перехоплення та модифікації запитів, sqlmap для виявлення та експлуатації SQL-ін'єкцій, Nmap для сканування мережі та виявлення відкритих портів, а також Hydra для атак методом перебору паролів. Необхідність розробки власних скриптів та адаптації існуючих інструментів до специфічних завдань також є невід'ємною частиною роботи. Окрім практичного тестування, фахівець з кібербезпеки повинен постійно вивчати нормативно-правові акти, міжнародні та національні стандарти у сфері інформаційної безпеки, такі як ISO/IEC 27001, NIST, ДСТУ, щоб забезпечувати відповідність розроблених рішень чинному законодавству. На основі цих знань відбувається розробка детальних рекомендацій щодо захисту інформації, політик безпеки, інструкцій для користувачів та процедур реагування на кіберінциденти. Також важливим аспектом є налаштування та тестування різноманітних програмних і апаратних засобів захисту, включаючи фаєрволи, системи виявлення та запобігання вторгненням (IDS/IPS), антивірусне програмне забезпечення, системи керування подіями безпеки (SIEM), а також засоби криптографічного захисту[26].

Не менш значущими є психофізіологічні фактори, які безпосередньо впливають на нервову систему та загальний стан працівника. Значне зорове напруження виникає внаслідок тривалої роботи з монітором, що включає читання дрібного тексту, аналіз складних структур даних, багатогодинне кодування та переключення уваги між кількома екранами. Це часто призводить

до синдрому комп'ютерного зору, що проявляється сухістю, печінням, втомою очей, а також головними болями. Розумове перенапруження є наслідком високої концентрації уваги, необхідності постійного аналізу великих обсягів інформації, вирішення нестандартних логічних задач та відповідальності за кібербезпеку систем. Це може призвести до зниження когнітивних функцій, дратівливості та підвищеного рівня стресу. Статичне навантаження на опорно-руховий апарат є неминучим наслідком тривалого сидіння в одній позі. М'язи спини, шиї, плечей та кінцівок перебувають у постійному напруженні, що може спричинити болі в спині, шийний остеохондроз, синдром зап'ястного каналу та порушення кровообігу. Попри складність і різноманітність завдань, деякі аспекти роботи фахівця з кібербезпеки, такі як рутинний моніторинг журналів або багаторазові перевірки, можуть мати елементи монотонності, що також сприяє втомі та зниженню уваги.

Для створення безпечних та комфортних умов праці, що сприятимуть збереженню здоров'я та підвищенню продуктивності фахівця з кібербезпеки, необхідно суворо дотримуватися вимог нормативних документів, зокрема Державних санітарних правил і норм роботи з візуальними дисплейними терміналами ЕОМ (ДСанПіН 3.3.2.007-98). Це включає організацію належної ергономіки робочого місця, де стіл та крісло мають бути регульованими по висоті, що дозволяє індивідуально налаштувати їх під фізіологічні особливості працівника та забезпечити правильну поставу. Монітор повинен бути розташований на оптимальній відстані 60-70 см від очей, а його верхній край має знаходитися на рівні очей або трохи нижче, щоб зменшити навантаження на шийний відділ хребта. Щоб уникнути статичного напруження та зорової втоми, рекомендовано робити регулярні, короткі перерви кожні 45-60 хвилин роботи за комп'ютером, під час яких слід виконувати гімнастику для очей, легкі розминки для шиї та спини, а також змінювати положення тіла[8]. Крім того, надзвичайно важливо підтримувати оптимальний мікроклімат у приміщенні: температуру повітря в межах 22-24 °С, що є комфортною для більшості людей, та відносну вологість 40-60%, що запобігає пересиханню слизових оболонок та дихальних шляхів. Регулярне та ефективне провітрювання робочої зони є

обов'язковим для забезпечення притоку свіжого повітря та видалення забруднень. Освітлення на робочому місці має бути достатнім, у межах 300-500 лк, забезпечуючи рівномірне розсіювання світла без створення відблисків на екрані монітора, що може викликати зорову втому. Електробезпека гарантується винятковим використанням заземленого обладнання, що знижує ризик ураження електричним струмом, а також регулярним контролем за справністю кабелів живлення, розеток та електричних приладів. Крім технічних аспектів, важливо проводити інструктажі з охорони праці, навчаючи працівників правилам безпечної експлуатації обладнання та діям у надзвичайних ситуаціях. Комплексний підхід до цих питань є запорукою збереження здоров'я фахівців та забезпечення високої ефективності їхньої праці у довгостроковій перспективі.

4.2 Безпека життєдіяльності

Найбільш імовірною надзвичайною ситуацією техногенного характеру, яка може загрожувати безпеці в офісі, є пожежа. Причини її виникнення можуть бути різноманітними і часто взаємопов'язаними, включаючи, наприклад, коротке замикання в електромережі, яке може статися через перевантаження, старіння проводки, пошкодження ізоляції або неякісне обладнання[26]. Несправність офісного чи серверного обладнання, такого як перегрів комп'ютерів, принтерів, джерел безперебійного живлення або зарядних пристроїв, також є поширеною причиною загорянь. Крім того, необережне поводження з вогнем, таке як куріння в недозволених місцях, неправильне використання електронагрівальних приладів або залишені без нагляду джерела тепла, може стати спусковим гачком для початку пожежі. Розуміння цих першопричин дозволяє зосередити профілактичні заходи на найбільш уразливих ланках.

Забезпечення пожежної безпеки вимагає комплексного, багатопланового підходу, який охоплює як технічні засоби, так і організаційні заходи. Насамперед, приміщення повинно бути обладнане сучасною автоматичною

пожежною сигналізацією, яка здатна оперативно виявляти ознаки загоряння (дим, підвищення температури) та сповіщати про небезпеку звуковими та світловими сигналами. Ця система має регулярно перевірятися та обслуговуватися для забезпечення її бездоганної роботи. Окрім сигналізації, необхідно наявність первинних засобів пожежогасіння. Йдеться про вогнегасники – зокрема, вуглекислотні або порошкові – які є ефективними для гасіння більшості типів пожеж, включаючи ті, що виникають через електрообладнання. Ці вогнегасники повинні бути розміщені на видних, легкодоступних місцях, без жодних перешкод, а персонал має бути обізнаний з їхнім розташуванням та правилами використання. Важливо також, щоб вогнегасники регулярно перевірялися та перезаряджалися відповідно до встановлених термінів. Окрім технічних засобів, ключову роль відіграють організаційні заходи. Кожне офісне приміщення повинно мати чітко розроблений та вивішений план евакуації, що містить схеми виходу, розташування пожежних виходів, вогнегасників та місць збору. Цей план має бути зрозумілим та доступним для всіх працівників. Надзвичайно важливо постійно підтримувати проходи, коридори та евакуаційні виходи вільними від будь-якого захаращення – жодні меблі, обладнання чи сміття не повинні перешкоджати швидкій та безпечній евакуації. І, безумовно, невід'ємною частиною профілактики є проведення з персоналом регулярних інструктажів та тренувань з пожежної безпеки, які включають як теоретичні знання (правила, причини пожеж, класи вогнегасників), так і практичні навички (використання вогнегасника, дії під час евакуації).

Порядок дій у разі виникнення пожежі є чітко регламентованим і вимагає послідовності та спокою. Першочерговим кроком є негайне повідомлення про пожежу пожежно-рятувальної служби. Це слід зробити за єдиним телефоном служби порятунку – 101, максимально точно вказавши адресу об'єкта, поверх, місце загоряння та інші відомі деталі, що допоможуть рятувальникам швидше зорієнтуватися. Одночасно з викликом пожежної служби необхідно сповістити керівництво офісу та всіх інших працівників, що перебувають у приміщенні, про небезпеку. Це можна зробити за допомогою внутрішньої системи

оповіщення, гучним голосом або будь-яким іншим доступним способом. Після оповіщення слід негайно розпочати безпечну евакуацію людей з приміщення, неухильно дотримуючись розробленого плану евакуації. Важливо пам'ятати, що паніка є головним ворогом під час евакуації, тому слід діяти спокійно та допомагати іншим, особливо тим, хто може потребувати допомоги. Якщо це можливо і, найголовніше, безпечно, необхідно знеструмити електрообладнання, щоб запобігти поширенню пожежі та ураженню електричним струмом. Тільки після цього, якщо загоряння незначне і не загрожує життю, можна розпочати гасіння пожежі за допомогою наявних первинних вогнегасників. Проте слід пам'ятати, що власні дії з гасіння пожежі допустимі лише до прибуття пожежних підрозділів і лише у випадку, якщо це не створює загрози для власного життя. Після прибуття пожежних підрозділів всі працівники повинні негайно припинити будь-які самостійні дії та неухильно виконувати всі вказівки та розпорядження керівника гасіння пожежі[26].

Дотримання вищезазначених вимог з охорони праці та безпеки життєдіяльності є фундаментальним принципом для будь-якого робочого середовища. Лише завдяки комплексній реалізації профілактичних заходів, регулярним інструктажам та чітко відпрацьованим алгоритмам дій у надзвичайних ситуаціях, можна мінімізувати потенційні ризики для здоров'я та життя працівника. Це забезпечує не тільки відповідність нормативним вимогам, а й створює умови для високої продуктивності та безпеки виконання завдань кваліфікаційної роботи, гарантуючи захист як людського капіталу, так і матеріальних цінностей компанії.

ВИСНОВОК

У сучасних умовах атаки типу DDoS стали однією з найрозповсюдженіших та найнебезпечніших загроз інформаційній безпеці. Це зумовлює необхідність постійного вдосконалення засобів їх виявлення та протидії. Метою цієї дипломної роботи стало дослідження підходів до своєчасного виявлення подібних атак із використанням сучасних методів обробки даних та машинного навчання. У процесі дослідження було розглянуто основні класифікації та характеристики DDoS-атак, оцінено недоліки традиційних підходів до їх виявлення, зокрема статистичних методів, що не завжди здатні адаптуватися до змін у поведінці загроз. У цьому контексті особливу увагу приділено застосуванню алгоритму Random Forest, який зарекомендував себе як ефективний інструмент класифікації мережевого трафіку.

Розроблено та реалізовано модель паралельної класифікації, що дозволяє скоротити час обробки вхідних даних і підвищити точність виявлення DDoS-атак. Проведений аналіз ефективності показав доцільність застосування багатопотокового підходу. За результатами дослідження сформовано заявку на авторське право, що перебуває на стадії розгляду. Окрім того, встановлено перспективність використання розподілених обчислень для обробки масштабних обсягів мережевих даних, що відкриває напрям для подальших досліджень у магістерському рівні підготовки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Метод виявлення мережевих атак в комп'ютеризованих системах управління. URL: http://konkurs.khnu.km.ua/wp-content/uploads/sites/25/2019/04/DP3_Eugen.pdf

2. Network Attacks and Their Detection Mechanisms. URL: https://www.researchgate.net/publication/263052997_Network_Attacks_and

3. _Their_Detection_Mechanisms_A_Review

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

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

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

7. Intrusion-Detection-using-Machine-Learning-on-NSL--KDD-dataset. URL: https://github.com/Deepthi10/Intrusion-Detection-using-Machine-Learning- on-NSL--KDD-dataset/blob/master/IDS.ipynb

8. Distributed Data Processing 101 – The Only Guide You'll Ever Need. URL: https://www.scaleyourapp.com/distributed-data-processing-101-the-only- guide-youll-ever-need/

9. Методи боротьби з DoS або DDoS атаками. URL: https://wiki.tntu.edu.ua/Методи_боротьби_з_Dos_або_DDos_атаками

10. Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets. URL: https://arxiv.org/pdf/1702.03681.pdf

11. Statistical Analysis Driven Optimized Deep Learning System for Intrusion Detection.URL:

12. https://www.researchgate.net/publication/327110465_Statistical_Analysis_ Driven_Optimized_Deep_Learning_System_for_Intrusion_Detection

13. L. Galchynsky and M. Grayvoronsky, "Evaluation of machine learning methods to detect DoS / DDoS attacks in IoT," Theoretical And Applied Cybersecurity Vol. 5 No. 1 (2022)

14. When Accuracy Isn't Enough, Use Precision and Recall to Evaluate Your

Classification Model. Will Koehrsen. URL: https://builtin.com/data- science/precision-and-recall

15. Deep learning approaches for detecting DDoS attacks: a systematic review. Meenakshi Mittal, Krishan Kumar, Sunny Behal. 2022. URL: https://link.springer.com/article/10.1007/s00500-021-06608-1

16. ZAGORODNA, N., STADNYK, M., LYPA, B., GAVRYLOV, M., & KOZAK, R. (2022). Network Attack Detection Using Machine Learning Methods. Challenges to national defence in contemporary geopolitical situation, 2022(1), 55-61.

17. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.

18. Tymoshchuk, D., & Yatskiv, V. (2024). Slowloris ddos detection and prevention in real-time. Collection of scientific papers «ΛΌГОΣ», (August 16, 2024; Oxford, UK), 171-176.

19. Augmented Reality Enhanced Learning Tools Development for Cybersecurity Major Zagorodna N., Skorenkyy Y., Kunanets N., Baran I., Stadnyk M (2022) CEUR Workshop Proceedings, 3309 , pp. 25-32.

20. Boltov, Y., Skarga-Bandurova, I., & Derkach, M. (2023, September). A Comparative Analysis of Deep Learning-Based Object Detectors for Embedded Systems. In 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 1156-1160). IEEE.

21. Lupenko, S., Orobchuk, O., & Xu, M. (2019, September). Logical-structural models of verbal, formal and machine-interpreted knowledge representation in Integrative scientific medicine. In Conference on Computer Science and Information Technologies (pp. 139-153).

22. Muzh, V., & Lechachenko, T. (2024). Computer technologies as an object and source of forensic knowledge: challenges and prospects of development. Вісник Тернопільського національного технічного університету, 115(3), 17-22.

23. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник. Комп'ютерні мережі. Книга 1 [навчальний посібник] - Львів, "Магнолія 2006", 2013. - 256 с.

24. Микитишин, А. Г., Митник, М. М., Голотенко, О. С., & Карташов, В. В. (2023). Комплексна безпека інформаційних мережевих систем. Навчальний посібник для студентів спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка».

25. Nedzelskyi, D., Derkach, M., Tatarchenko, Y., Safonova, S., Shumova, L., & Kardashuk, V. (2019, August). Research of efficiency of multi-core computers with shared memory. In 2019 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW) (pp. 111-114). IEEE.

26. Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., Lechachenko T., (2024). Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings, 3896, pp. 1-11.

27. Derkach, M., Lysak, V., Skarga-Bandurova, I., & Kotsiuba, I. (2019, September). Parking Guide Service for Large Urban Areas. In *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* (Vol. 1, pp. 567-571). IEEE.

28. Nedzelky, D., Derkach, M., Skarga-Bandurova, I., Shumova, L., Safonova, S., & Kardashuk, V. (2021, September). A Load Factor and its Impact on the Performance of a Multicore System with Shared Memory. In *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* (Vol. 1, pp. 499-503). IEEE.

29. Babakov, R. M., et al. "Internet of Things for Industry and Human Application. Vol. 3." (2019): 1-917.

30. Sachenko, A. O., et al. "Internet of Things for intelligent transport systems."

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()
```