КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ
Факультет автоматизації інформаційних технологій

Кафедра інформаційних технологій

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

на тему:

Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж

Лобач Олександра Романівна

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

___
(факультет)

інформаційних технологій

_____
(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.
_____
„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «**Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж**»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач
Лобач Олександра Романівна
122 «Комп’ютерні науки»_____
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21_____
Керівники Гончаренко Т.А.,
_____Мацієвський О. О._____
(прізвище та ініціали)
д.т.н., професор, асистент_____
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц. Доля О.В._____
(Прізвище та ініціали)
Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет:	Автоматизації інформаційних технологій
Випускова кафедра:	Інформаційних технологій
Освітній ступінь:	Бакалавр
Спеціальність:	Комп’ютерні науки
Освітня програма:	Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.
_____
„___” _____2025 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ **БАКАЛАВР**

	Лобач Олександрі Романівні

1. Тема роботи - **Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж**

затверджена наказом ректора КНУБА № 235/23/25 від «14» лютого 2025 року

2. Керівники роботи	Гончаренко Тетяна Андріївна, д.т.н. професор Мацієвський Олексій Олегович, асистент

3. Строк подання Здобувачем роботи до захисту травень 2025 р.

4. Зміст пояснювальної записки за розділами:

Р.1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Р.2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Р.3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Р.4 АНАЛІЗ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

5. Графічний матеріал за розділами:

Робота викладена на 86 аркушах, містить 1 додаток, 15 таблиць, 22 рисунки, список використаної літератури із 20 найменувань.

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	01.02.2025
Розділ 2	06.04.2025
Розділ 3	16.05.2025
Розділ 4	16.05.2025
Остаточне оформлення роботи	25.05.2025
Направлення роботи для перевірки на плагіат	25.05.2025
Попередній захист роботи на випусковій кафедрі	26.05.2025
Направлення роботи на рецензування	26.05.2025

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1	Маційовський О. О., асистент кафедри ІТ	01.02.2025	
Розділ 2	Рябчун Ю.В., доц.каф.ІТ	06.04.2025	
Розділ 3	Маційовський О. О., асистент кафедри ІТ	16.05.2025	
Розділ 4	Рябчун Ю.В., доц.каф.ІТ	16.05.205	

8. Дата видачі завдання листопад 2025 р.

Зав. кафедри			Гончаренко Т.А.
	(підпис)		(прізвище та ініціали)
Керівники			Гончаренко Т. А.
			Маційовський О. О.
	(підпис)		(прізвище та ініціали)
Здобувач			Лобач О.Р.
	(підпис)		(прізвище та ініціали)

АНОТАЦІЯ

Лобач О. Р. Система виявлення аномалій у веб-трафіку за допомогою нейронних мереж. Атестаційна випускна робота бакалавра за спеціальністю: 122 «Комп'ютерні науки». – Київський національний університет будівництва та архітектури. – Київ, 2025.

Обсяг роботи: 86 сторінок, 15 таблиць, 22 рисунків, 1 додаток, 20 джерел.

Шифр та назва напряму підготовки: 122 Комп'ютерні науки.

Мета та основний результат роботи: Метою роботи є створення системи для виявлення аномалій у веб-трафіку Wi-Fi мереж у реальному часі з використанням нейронних мереж. Розроблено програмне рішення, яке забезпечує захоплення мережевого трафіку, генерацію синтетичних аномалій, навчання моделей машинного навчання (MLPClassifier, IsolationForest, One-Class SVM, RandomForest) та візуалізацію результатів через інтуїтивний інтерфейс на основі CustomTkinter. Система досягла F1-score 0.87 для ізоляційного лісу, демонструючи високу точність виявлення аномалій, зокрема DDoS-атак. Результати сприяють підвищенню кібербезпеки локальних мереж та технологічної незалежності України.

Ключові слова: аномалії, веб-трафік, нейронні мережі, кібербезпека, Wi-Fi мережі, машинне навчання, DDoS-атаки, ізоляційний ліс, моніторинг, візуалізація.

SUMMARY

Lobach O. R. System for detecting anomalies in web traffic using neural networks. Attestation graduation work of the bachelor in the specialty: 122 “Computer Science”, – Kyiv National University of Construction and Architecture. – Kyiv, 2025.

Volume of the Thesis: 86 pages, 15 tables, 22 figures, 1 appendix, 20 references.

Field of Study: 122 Computer Science.

Objective and Main Results: The objective of this work is to develop a system for detecting anomalies in web traffic of Wi-Fi networks in real time using neural networks. A software solution was developed that enables the capture of network traffic, generation of synthetic anomalies, training of machine learning models (MLPClassifier, IsolationForest, One-Class SVM, RandomForest), and visualization of results through an intuitive interface based on CustomTkinter. The system achieved an F1-score of 0.87 for the IsolationForest model, demonstrating high accuracy in detecting anomalies, particularly DDoS attacks. The results contribute to enhancing the cybersecurity of local networks and the technological independence of Ukraine.

Keywords: anomalies, web traffic, neural networks, cybersecurity, Wi-Fi networks, machine learning, DDoS attacks, isolation forest, monitoring, visualization.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

DDoS – розподілена атака типу «відмова в обслуговуванні»;

IDS – система виявлення вторгнень (Intrusion Detection System);

MLP – багатошаровий перцептрон (Multi-Layer Perceptron);

SVM – машина опорних векторів (Support Vector Machine);

TCP – протокол керування передачею (Transmission Control Protocol);

UDP – протокол датаграм користувача (User Datagram Protocol);

Wi-Fi – бездротова мережа (Wireless Fidelity);

AUC – площа під ROC-кривою (Area Under Curve);

F1-score – гармонійне середнє точності та повноти;

ROC – характеристика робочої чутливості (Receiver Operating Characteristic);

ReLU – функція активації випрямленого лінійного блоку (Rectified Linear Unit);

StandardScaler – метод нормалізації даних зі стандартним масштабом;

IsolationForest – ізоляційний ліс (алгоритм виявлення аномалій);

RandomForest – випадковий ліс (алгоритм машинного навчання);

CTk – CustomTkinter (бібліотека для створення графічного інтерфейсу);

psutil – бібліотека для збору системних і мережевих даних;

NumPy – бібліотека для роботи з числовими масивами;

pandas – бібліотека для обробки та аналізу даних;

matplotlib – бібліотека для візуалізації даних;

seaborn – бібліотека для створення статистичних графіків;

scikit-learn – бібліотека для машинного навчання;

SYN – синхронізаційний пакет у TCP-протоколі.

ВСТУП

У сучасному світі стрімке зростання цифрових технологій та розширення доступу до мережі Інтернет супроводжуються значним збільшенням обсягів веб-трафіку, що створює нові виклики для забезпечення кібербезпеки. Зокрема, зростає кількість кібератак, таких як *DDoS*-атаки, сканування портів та флуд з'єднань, які можуть порушувати функціонування інформаційних систем, спричиняти економічні збитки та загрожувати національній безпеці. У цьому контексті розробка систем виявлення аномалій у веб-трафіку набуває особливої актуальності, оскільки дозволяє своєчасно ідентифікувати підозрілу поведінку в мережі та запобігати потенційним загрозам. Для України, яка активно інтегрується в глобальний цифровий простір і водночас зазнає значного тиску в інформаційній сфері, створення ефективних систем кіберзахисту є стратегічно важливим завданням.

Аномалії у веб-трафіку, такі як незвичайно високий обсяг переданих даних, нетипова кількість з'єднань або підозрілі шаблони мережевої активності, часто є індикаторами кібератак. Традиційні методи виявлення аномалій, засновані на сигнатурному аналізі або статистичних правилах, мають обмеження, оскільки не завжди здатні адаптуватися до нових, раніше невідомих типів атак. У той же час, сучасні підходи, що використовують методи машинного навчання, зокрема нейронні мережі, демонструють високу ефективність у розпізнаванні складних шаблонів і виявленні аномалій у реальному часі. Критичний аналіз літератури показує, що такі системи, як *Intrusion Detection Systems (IDS)* на основі машинного навчання, активно розвиваються в роботах зарубіжних дослідників (наприклад, *Snort, Zeek,* або комерційні рішення *Cisco Secure Network Analytics*). Проте більшість таких рішень потребують значних обчислювальних ресурсів, що ускладнює їх впровадження в умовах обмеженого фінансування, характерного для багатьох українських організацій. Крім того, комерційні продукти часто не враховують локальних особливостей мережевої інфраструктури України, таких як специфіка роботи *Wi-Fi* мереж у державних установах чи малих підприємствах.

Актуальність розробки системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж для України зумовлена кількома факторами. По-перше, зростання кіберзагроз, спрямованих на критичну інфраструктуру країни, вимагає створення доступних і ефективних інструментів захисту. По-друге, розвиток локальних технологічних рішень сприяє зменшенню залежності від іноземного програмного забезпечення, що є важливим з огляду на національну безпеку. По-третє, використання нейронних мереж дозволяє підвищити точність і адаптивність системи до нових типів атак, що є конкурентною перевагою порівняно з традиційними підходами.

Мета та обґрунтування необхідності розробки. Метою даної роботи є розробка системи виявлення аномалій у веб-трафіку на основі нейронних мереж, яка забезпечує моніторинг *Wi-Fi* мереж у реальному часі, ідентифікацію аномальної поведінки та візуалізацію результатів для користувача. Необхідність розробки такої системи обґрунтовується потребою у створенні гнучкого, ефективного та доступного інструменту для захисту локальних мереж, зокрема в умовах обмежених ресурсів. На відміну від багатьох комерційних рішень, запропонована система орієнтована на використання відкритих бібліотек *Formatted*: За умовчання, ця система дозволяє створювати та аналізувати трафік *Wi-Fi* мереж, генерувати штучні аномалії для тестування, а також навчати моделі машинного навчання, включаючи нейронні мережі, для виявлення аномалій. Основна перевага системи полягає в її адаптивності: вона може працювати як із синтетичними датасетами, так і з реальними даними, захопленими в реальному часі, що робить її універсальним інструментом для різних сценаріїв використання.

Обґрунтування основних проектних рішень. Основні проектні рішення базуються на інтеграції кількох сучасних технологій. По-перше, для захоплення мережевого трафіку використовується бібліотека *psutil*, яка дозволяє отримувати статистику про мережеві інтерфейси та з’єднання без необхідності використання складних інструментів, таких як *Wireshark*. По-друге, для обробки даних застосовуються методи машинного навчання, зокрема нейронні мережі (*MLPClassifier*), ізоляційний ліс (*Isolation Forest*), одно-класовий *SVM (One-Class*

SVM) та випадковий ліс *(Random Forest).* Вибір нейронних мереж як основного інструменту зумовлений їх здатністю моделювати нелінійні залежності в даних, що є критично важливим для виявлення складних аномалій. Інтерфейс користувача, реалізований за допомогою бібліотеки *customtkinter*, забезпечує зручне керування системою та візуалізацію даних, що робить її доступною навіть для користувачів без глибоких технічних знань.

Можливі галузі застосування. Результати роботи мають широкий спектр застосування. У державному секторі система може використовуватися для захисту мереж критичної інфраструктури, таких як енергетичні, транспортні чи медичні системи. У комерційному секторі вона може бути впроваджена в малих і середніх підприємствах для моніторингу локальних *Wi-Fi* мереж і запобігання кібератакам. В освітній сфері система може слугувати інструментом для навчання студентів основам кібербезпеки та аналізу даних. Крім того, розробка сприяє розвитку вітчизняної індустрії кібербезпеки, створюючи передумови для експорту технологічних рішень на міжнародний ринок.

Таким чином, розроблена система виявлення аномалій у веб-трафіку за допомогою нейронних мереж є актуальним і перспективним рішенням, яке відповідає сучасним викликам кібербезпеки та сприяє зміцненню технологічної незалежності України.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Сучасний розвиток інформаційних технологій супроводжується стрімким зростанням обсягів даних, що передаються через мережеві інфраструктури, зокрема через бездротові мережі *Wi-Fi*. Веб-трафік, який включає запити до веб-серверів, обмін даними між клієнтами та серверами, а також різноманітні мережеві взаємодії, є ключовим компонентом цифрової екосистеми. Проте зростання складності та масштабів мережевих систем призводить до появи нових викликів, пов’язаних із забезпеченням безпеки та стабільності їх функціонування. Одним із таких викликів є виявлення аномалій у веб-трафіку, які можуть свідчити про кібератаки, зловмисну поведінку або технічні збої.

Предметна область дослідження охоплює аналіз мережевого трафіку з акцентом на виявлення аномалій за допомогою методів машинного навчання, зокрема нейронних мереж. Аномалії у веб-трафіку можуть проявлятися у вигляді нетипових шаблонів поведінки, таких як *DDoS*-атаки (розподілені атаки типу «відмова в обслуговуванні»), сканування портів, флуд з’єднань або надмірний мережевий трафік, спричинений шкідливим програмним забезпеченням [1]. Такі аномалії можуть призводити до порушення доступності сервісів, втрати даних або компрометації безпеки мережі. Своєчасне виявлення таких подій є критично важливим для забезпечення надійності та безпеки мережевих систем.

У контексті предметної області особливу увагу приділено бездротовим мережам *Wi-Fi*, які є широко поширеними завдяки своїй зручності та доступності. Проте *Wi-Fi* мережі є вразливими до атак через відкритий характер передачі даних і можливість перехоплення трафіку. Для аналізу трафіку в таких мережах необхідно враховувати його динамічну природу, різноманітність пристроїв, що підключаються, та мінливість умов передачі даних. Основними характеристиками веб-трафіку, що підлягають аналізу, є розмір пакетів, частота їх передачі, кількість активних з’єднань,

співвідношення *TCP/UDP*-пакетів, а також статистичні показники, такі як тривалість сесій і обсяг переданих даних [2].

Традиційні методи виявлення аномалій, такі як аналіз на основі сигнатур або правил, мають суттєві обмеження, оскільки вони не здатні ефективно ідентифікувати нові або невідомі типи атак. У зв'язку з цим методи машинного навчання, зокрема нейронні мережі, набувають популярності завдяки їх здатності до самонавчання та адаптації до змінних умов [3]. Нейронні мережі дозволяють моделювати складні нелінійні залежності в даних, що робить їх ефективним інструментом для виявлення аномалій у веб-трафіку [4]. Вони можуть аналізувати великі обсяги даних у реальному часі, ідентифікувати приховані закономірності та прогнозувати потенційні загрози.

Ключовим елементом предметної області є необхідність створення системи, яка поєднує кілька функціональних компонентів: захоплення мережевого трафіку, генерацію тестових аномалій, обробку даних, навчання моделей машинного навчання та візуалізацію результатів. Захоплення трафіку передбачає моніторинг мережевих інтерфейсів, зокрема *Wi-Fi*, для збору статистичних даних про пакети, з'єднання та інші параметри. Генерація аномалій дозволяє імітувати різні типи атак для тестування ефективності системи. Обробка даних включає нормалізацію, фільтрацію та підготовку даних для навчання моделей. Навчання моделей, зокрема нейронних мереж, забезпечує здатність системи розпізнавати аномалії, тоді як візуалізація допомагає користувачам інтерпретувати результати аналізу.

Іншим важливим питанням є необхідність адаптації системи до реальних умов експлуатації. Мережевий трафік характеризується високою варіативністю, що ускладнює створення універсальних моделей. Крім того, системи виявлення аномалій повинні бути стійкими до помилкових спрацьовувань (*false positives*) і мати достатню продуктивність для обробки даних у реальному часі. Це вимагає ретельного вибору алгоритмів, налаштування їх параметрів і використання ефективних методів оцінки якості моделей, таких як метрики точності, повноти та *F1*-показника.

Таким чином, предметна область дослідження зосереджена на розробці інтелектуальної системи для виявлення аномалій у веб-трафіку *Wi-Fi* мереж із використанням нейронних мереж. Вона охоплює аналіз мережевих даних,

моделювання аномальної поведінки, застосування методів машинного навчання та створення зручного інтерфейсу для взаємодії з користувачем. Така система має потенціал для застосування в різних сферах, включаючи корпоративні мережі, публічні *Wi-Fi* точки доступу та системи кібербезпеки, сприяючи підвищенню безпеки та стабільності мережевих інфраструктур.

1.2 Аналіз стану вирішення задачі

Виявлення аномалій у веб-трафіку є однією з ключових задач у сфері кібербезпеки, що набуває все більшої актуальності в умовах зростання обсягів мережевих даних, ускладнення кібератак і поширення нових технологій, таких як Інтернет речей (*IoT*) та хмарні обчислення [5]. Аномалії в мережевому трафіку можуть свідчити про різноманітні загрози, включаючи *DDoS*-атаки, сканування портів, шкідливе програмне забезпечення або несанкціонований доступ. Своєчасне виявлення таких аномалій дозволяє запобігти значним фінансовим і репутаційним втратам, а також забезпечити стабільність і безпеку інформаційних систем. У цьому контексті аналіз сучасного стану вирішення задачі виявлення аномалій у веб-трафіку є важливим для розуміння прогресу в цій галузі, визначення наявних обмежень і окреслення перспектив розвитку.

На сьогоднішній день для вирішення задачі виявлення аномалій у веб-трафіку застосовуються різноманітні підходи, які можна умовно поділити на три основні категорії: методи, засновані на правилах (*rule-based*), статистичні методи та методи машинного навчання, зокрема з використанням нейронних мереж. Кожен із цих підходів має свої переваги та недоліки, що визначають їх ефективність залежно від конкретного сценарію використання.

Методи, засновані на правилах, були одними з перших, що застосовувалися для виявлення аномалій. Вони базуються на заздалегідь визначених сигнатурах або шаблонах поведінки, які асоціюються з відомими типами атак. Такі системи, як *Snort* або *Suricata*, аналізують мережевий трафік у реальному часі, порівнюючи його з базою сигнатур [6]. Основною перевагою цього підходу є висока точність у виявленні

відомих загроз і швидкість обробки даних. Однак головним недоліком є їх обмежена здатність виявляти нові або модифіковані атаки, для яких сигнатури ще не створено. Крім того, підтримка актуальності бази сигнатур вимагає значних ресурсів і часу, що робить цей підхід менш ефективним у динамічному середовищі сучасних мереж.

Статистичні методи аналізу аномалій ґрунтуються на побудові моделі нормальної поведінки мережі з подальшим виявленням відхилень від цієї моделі. Такі методи включають аналіз часових рядів, кластеризацію та методи оцінки щільності розподілу даних. Наприклад, методи, засновані на алгоритмах кластеризації (*K-means*) або оцінці відхилень (*Z-score*), дозволяють виявляти аномалії без необхідності попереднього маркування даних [7]. Перевагою статистичного підходу є його здатність адаптуватися до змін у мережевому трафіку без залежності від бази сигнатур. Проте такі методи часто страждають від високого рівня хибнопозитивних спрацьовувань, особливо в умовах складного або нестабільного трафіку, а також потребують ретельного налаштування параметрів для забезпечення ефективності.

Методи машинного навчання, зокрема з використанням нейронних мереж, набули значної популярності завдяки своїй здатності обробляти великі обсяги даних, виявляти складні нелінійні залежності та адаптуватися до нових типів загроз. Алгоритми машинного навчання, такі як *Random Forest, Isolation Forest, One-Class SVM*, а також глибокі нейронні мережі (*DNN*), рекурентні нейронні мережі (*RNN*) і автокодери, широко застосовуються для аналізу мережевого трафіку. Наприклад, автокодери дозволяють навчати модель на нормальних даних, а потім виявляти аномалії шляхом аналізу помилок реконструкції. Рекурентні нейронні мережі ефективні для аналізу послідовностей даних, що є важливим для виявлення аномалій у часових рядах мережевого трафіку. Водночас методи глибокого навчання, такі як згорткові нейронні мережі (*CNN*), використовуються для аналізу пакетів даних як двовимірних структур, що дозволяє виявляти аномалії на рівні структури пакетів.

Серед переваг методів машинного навчання варто відзначити їх високу точність і здатність до самонавчання, що дозволяє виявляти невідомі раніше аномалії. Проте ці методи також мають певні обмеження. По-перше, вони потребують значних обчислювальних ресурсів для навчання та прогнозування, що може бути

проблематичним для систем із обмеженими ресурсами. По-друге, ефективність таких моделей залежить від якості та обсягу даних для навчання. Недостатня кількість даних або незбалансовані набори даних можуть призводити до пере- або недонавчання моделі. Крім того, методи машинного навчання часто стикаються з проблемою інтерпретовності: складно пояснити, чому модель класифікувала певний трафік як аномальний, що може ускладнювати прийняття рішень у критичних системах.

Останнім часом значна увага приділяється гібридним підходам, які поєднують переваги різних методів. Наприклад, комбінація статистичного аналізу з нейронними мережами дозволяє підвищити точність виявлення аномалій і зменшити кількість хибнопозитивних спрацьовувань. Також активно досліджуються методи навчання з учителем і без учителя. У першому випадку моделі навчаються на позначених даних, що містять як нормальний, так і аномальний трафік, тоді як у другому — моделі будують профіль нормальної поведінки без необхідності маркування аномалій. Останній підхід є особливо перспективним для виявлення нових типів атак, оскільки не залежить від попередньо відомих сигнатур.

Аналіз літератури свідчить, що сучасні дослідження зосереджені на кількох ключових напрямах: підвищенні ефективності моделей у реальному часі, зменшенні обчислювальних витрат, адаптації до динамічних мережевих середовищ і підвищенні інтерпретовності результатів. Наприклад, використання легковагих нейронних мереж і методів стиснення моделей дозволяє застосовувати їх на пристроях з обмеженими ресурсами, таких як маршрутизатори або *IoT*-пристрої. Крім того, зростає інтерес до методів федеративного навчання, які дозволяють навчати моделі на розподілених даних без їх централізованого збору, що є важливим для забезпечення конфіденційності.

У контексті розробленої системи, представленої в коді проєкту, використовується комплексний під *finestre*: порівняльний аналіз кількох алгоритмів машинного навчання (Iso*lation Forest, One-Class SVM, Random Forest* і нейронні мережі) для виявлення аномалій у веб-трафіку. Такий підхід дозволяє оцінити ефективність різних методів і обрати оптимальний залежно від специфіки задачі.

Крім того, система включає модулі для генерації синтетичних аномалій і захоплення реального трафіку, що сприяє тестуванню моделей у контрольованих умовах і підвищує їх практичну цінність.

Підсумовуючи, сучасний стан вирішення задачі виявлення аномалій у веб-трафіку характеризується значним прогресом у застосуванні методів машинного навчання, зокрема нейронних мереж. Водночас залишаються виклики, пов’язані з обробкою великих обсягів даних у реальному часі, забезпеченням інтерпретовності моделей і адаптацією до нових типів загроз. Подальші дослідження мають бути спрямовані на розробку гібридних підходів, оптимізацію обчислювальних ресурсів і підвищення стійкості моделей до мінливого мережевого середовища. Розроблена система є прикладом практичної реалізації сучасних методів і може служити основою для подальшого вдосконалення технологій кібербезпеки.

1.3 Функціональний аналіз

Система виявлення аномалій у веб-трафіку, реалізована в коді проєкту, є комплексним програмним рішенням, спрямованим на ідентифікацію відхилень у мережевій активності за допомогою методів машинного навчання, зокрема нейронних мереж. Функціональний аналіз системи дозволяє детально розкрити її можливості, взаємодію компонентів та основні задачі, які вона вирішує. Нижче наведено детальний опис функціональних показників системи, структурований за ключовими компонентами та їх призначенням.

Захоплення та обробка мережевого трафіку. Система забезпечує захоплення даних про мережевий трафік у реальному часі, що є основою для подальшого аналізу. Клас *WiFiTrafficCapture* використовує бібліотеку psutil для збору статистики мережевих інтерфейсів, зокрема таких показників, як кількість відправлених і отриманих байтів, пакети, активні з’єднання, а також *TCP*- і *UDP*-з’єднання. Функціонал автоматичного визначення *Wi-Fi*-інтерфейсу адаптується до різних операційних систем, зокрема Windows, шляхом аналізу доступних мережевих інтерфейсів та використання системних утиліт (наприклад, *netsh*). Захоплені дані

структуруються у вигляді часових рядів із мітками часу, що дозволяє відстежувати динаміку трафіку. Цей функціонал забезпечує основу для моніторингу та виявлення аномалій у реальному часі, а також для створення навчальних датасетів на основі зібраних даних.

Генерація синтетичних даних і аномалій. Для тестування та навчання моделей система включає модуль генерації синтетичних даних і аномалій, реалізований у класі *WebTrafficDataset* та *NetworkAnomalyGenerator*. Клас *WebTrafficDataset* створює модельовані датасети, що імітують нормальний і аномальний трафік, наприклад, *DDoS*-атаки чи сканування портів. Ці датасети включають різноманітні ознаки, такі як розмір пакетів, частота пакетів, тривалість потоків, кількість унікальних портів тощо, що генеруються з використанням статистичних розподілів (нормальний, експоненційний, Пуассона). У той же час, клас *NetworkAnomalyGenerator* дозволяє генерувати реальні аномалії в мережі, такі як сканування портів, високий трафік або флуд з'єднань. Цей функціонал забезпечує можливість тестування системи в контрольованих умовах, а також імітацію реальних атак для оцінки ефективності моделей.

Навчання моделей машинного навчання. Центральною функціональною складовою системи є навчання моделей для виявлення аномалій, реалізоване в класі *WebTrafficAnomalyDetector*. Система підтримує кілька алгоритмів, включаючи ізоляційний ліс (*Isolation Forest*), однокласовий *SVM*, нейронну мережу (*MLPClassifier*) та випадковий ліс (*Random Forest*). Кожен алгоритм має власний метод навчання, який включає нормалізацію даних за допомогою *StandardScaler*, розподіл на навчальну та тестову вибірки, а також оцінку результатів за метриками, такими як матриця помилок, *ROC*-крива та звіт класифікації. Нейронна мережа, як ключовий елемент відповідно до теми проєкту, використовує багатошарову архітектуру з трьома прихованими шарами (64, 32, 16 нейронів) і функцією активації *ReLU*. Цей функціонал дозволяє адаптувати систему до різних типів аномалій і забезпечує гнучкість у виборі алгоритму залежно від специфіки задачі.

Моніторинг у реальному часі та виявлення аномалій. Система забезпечує безперервний моніторинг мережевого трафіку з можливістю виявлення аномалій у

реальному часі. Функціонал моніторингу, реалізований через методи *monitor_traffic* і *update_monitoring_plots*, використовує навчену модель для аналізу нових даних, отриманих із захоплення трафіку. Дані нормалізуються, і модель обчислює оцінку аномальності, яка порівнюється з фіксованим порогом (за замовчуванням 0.75). У разі перевищення порогу система сигналізує про аномалію, що відображається в журналі та на графіках. Цей функціонал є критично важливим для оперативного реагування на потенційні загрози, такі як кібератаки.

Візуалізація даних і результатів. Для полегшення аналізу та інтерпретації система включає розвинений функціонал візуалізації, інтегрований у графічний інтерфейс на основі *customtkinter* і *matplotlib*. Візуалізація охоплює:

– розподіл нормальних і аномальних записів у датасеті;

– кореляційну матрицю для оцінки взаємозв’язків між ознаками;

– гістограми розподілу ключових ознак;

– результати навчання (матриця помилок, *ROC*-крива, розподіл оцінок аномальності);

– графіки моніторингу в реальному часі з позначенням аномалій.

Цей функціонал забезпечує зручність для користувача, дозволяючи візуально оцінити якість роботи системи та виявлені аномалії.

Графічний інтерфейс користувача. Система має інтуїтивно зрозумілий графічний інтерфейс, який об’єднує всі функціональні можливості. Інтерфейс включає панелі для вибору датасетів, моделей, запуску моніторингу, генерації аномалій, а також вкладки для відображення даних, результатів навчання та моніторингу. Логування подій у реальному часі забезпечує прозорість роботи системи. Інтерфейс адаптований до потреб користувачів, які можуть не мати глибоких технічних знань, що робить систему доступною для широкого кола спеціалістів.

Логування та обробка помилок. Система включає розвинений механізм логування, реалізований через модуль *logging*. Усі ключові події (зміна стану, помилки, результати аналізу) записуються у файл і виводяться в консоль та інтерфейс. Це забезпечує можливість відстеження роботи системи, діагностики проблем і

аналізу її ефективності. Обробка винятків у коді мінімізує ймовірність аварійного завершення роботи, що є важливим для безперервного моніторингу.

Функціональний аналіз системи виявлення аномалій у веб-трафіку демонструє її комплексний підхід до вирішення задачі. Поєднання захоплення даних, генерації аномалій, навчання моделей, моніторингу в реальному часі, візуалізації та зручного інтерфейсу забезпечує ефективне виявлення мережевих загроз. Особлива увага до нейронних мереж як основного інструменту аналізу підкреслює сучасний підхід до обробки складних даних. Система є гнучкою, адаптивною та придатною для використання в реальних умовах, що робить її цінним інструментом для забезпечення кібербезпеки.

1.4 Композиційний аналіз

Композиційний аналіз предметної області системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж передбачає декомпозицію системи на основні компоненти, визначення їх функціонального призначення, взаємозв’язків та ролі у вирішенні поставленої задачі. Такий підхід дозволяє структурувати складну систему, забезпечуючи чітке розуміння її архітектури та принципів роботи. У контексті розробленої системи, композиційний аналіз базується на коді реалізації, представленому в лістингу, та відображає модульну організацію програми, яка включає кілька ключових компонентів, кожен із яких виконує спеціалізовані функції (рис. 1.1). Нижче наведено детальний аналіз компонентів системи.

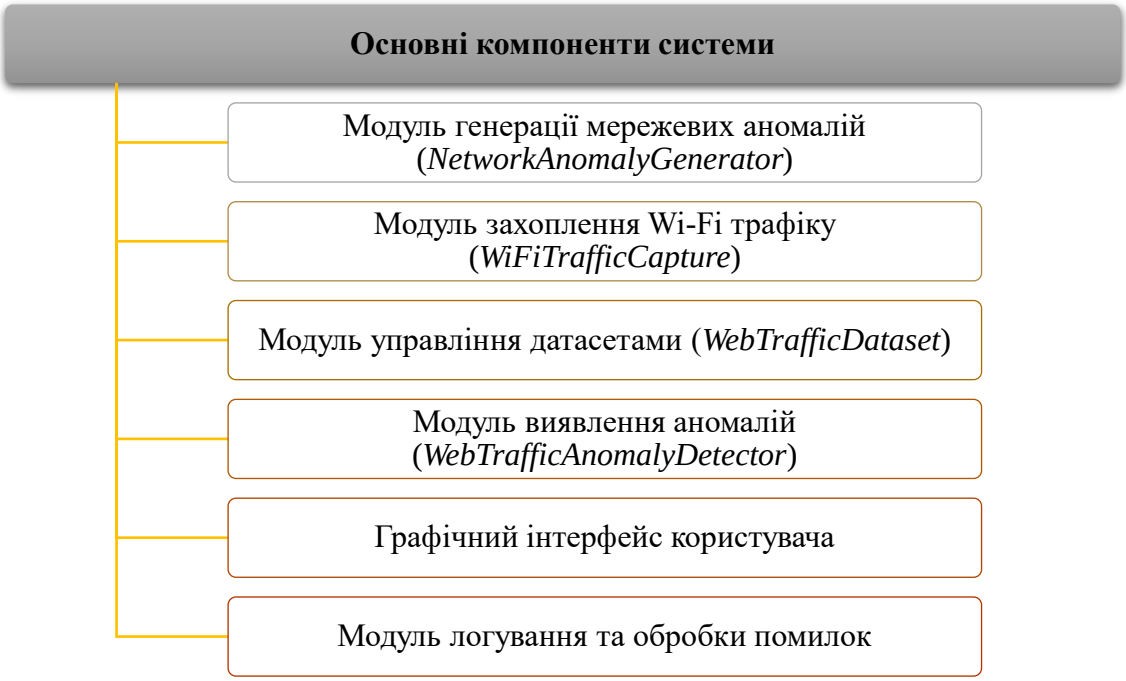

Рисунок 1.1 Основні компоненти системи

Модуль генерації мережевих аномалій (*NetworkAnomalyGenerator*) відповідає за створення штучних аномальних подій у мережевому трафіку для тестування та навчання системи. Він включає методи для генерації трьох типів аномалій: сканування портів (*port_scan*), високого трафіку (*high_traffic*) та флуду з'єднань (*connection_flood*). Компонент використовує бібліотеки *socket*, *requests* та *psutil* для взаємодії з мережею та імітації аномальної поведінки. Наприклад, метод *generate_port_scan* виконує швидке сканування портів цільового *IP*, створюючи патерни, характерні для розвідувальних атак. Функціонально цей модуль забезпечує контрольоване середовище для створення даних, що дозволяють оцінити ефективність алгоритмів виявлення аномалій. Його інтеграція з іншими компонентами системи здійснюється через виклик методів генерації з інтерфейсу користувача.

Модуль захоплення *Wi-Fi* трафіку (*WiFiTrafficCapture*) відповідає за збір даних про мережевий трафік у реальному часі. Використовуючи бібліотеку *psutil*, модуль періодично захоплює статистику мережевих інтерфейсів, включаючи обсяги відправлених та отриманих байтів, кількість пакетів та активних з'єднань. Автоматичне визначення *Wi-Fi* інтерфейсу (метод *detect_wifi_interface*) забезпечує

адаптивність системи до різних апаратних конфігурацій. Дані, зібрані цим модулем, використовуються як для навчання моделей (у режимі "*Live Capture*"), так і для моніторингу в реальному часі. Модуль працює асинхронно в окремому потоці, що дозволяє уникнути блокування основного інтерфейсу програми.

Модуль управління датасетами (*WebTrafficDataset*) забезпечує створення синтетичних наборів даних для навчання та тестування моделей. Він підтримує три типи датасетів: зразковий (*sample_dataset*), *DDoS*-атаки (*ddos_dataset*) та сканування портів (portscan_dataset). Кожен датасет містить як нормальний, так і аномальний трафік із заданими статистичними характеристиками, що генеруються за допомогою бібліотеки *NumPy* [8]. Наприклад, для *DDoS*-атак датасет включає ознаки, такі як висока частота пакетів та велика кількість *SYN*-пакетів. Модуль забезпечує гнучкість у виборі даних для навчання, дозволяючи адаптувати систему до різних сценаріїв аномальної поведінки. Взаємодія з іншими компонентами здійснюється через передачу підготовлених даних у модуль навчання.

Модуль виявлення аномалій (*WebTrafficAnomalyDetector*) – центральний компонент системи, який інтегрує всі інші модулі та забезпечує їх координацію. Він включає підмодулі для навчання моделей (*Isolation Forest, One-Class SVM, Neural Network, Random Forest*), обробки даних (за допомогою sklearn.preprocessing), моніторингу трафіку в реальному часі та візуалізації результатів. Навчання моделей базується на нормалізації даних (*StandardScaler*) та поділі на навчальну і тестову вибірки. Нейронна мережа (*MLPClassifier*) є ключовим алгоритмом для виявлення аномалій, що дозволяє обробляти складні нелінійні залежності в даних. Модуль також відповідає за обробку подій інтерфейсу користувача, таких як вибір датасету чи запуск моніторингу.

Графічний інтерфейс користувача побудований за допомогою бібліотеки *customtkinter*, забезпечує інтуїтивне керування системою. Він складається з лівої панелі керування (вибір датасету, моделі, запуск моніторингу, генерація аномалій) та правої панелі візуалізації (вкладки для відображення даних, результатів навчання та моніторингу). Використання *matplotlib* дозволяє створювати інформативні графіки, такі як гістограми розподілу ознак, матриці кореляції та *ROC*-криві. Логування подій

у нижній панелі забезпечує зворотний зв’язок із користувачем. Інтерфейс є сполучною ланкою між користувачем і функціональними модулями, забезпечуючи їх доступність.

Модуль логування та обробки помилок. Система використовує бібліотеку *logging* для фіксації подій, помилок та інформаційних повідомлень. Логи записуються як у файл (*anomaly_detector.log*), так і виводяться в консоль та інтерфейс користувача. Цей компонент забезпечує діагностику роботи системи, дозволяючи відстежувати помилки, наприклад, при визначенні мережевого інтерфейсу чи обробці даних. Обробка винятків у всіх модулях підвищує надійність системи, забезпечуючи її стабільну роботу навіть у разі збоїв.

Компоненти системи тісно взаємодіють через центральний модуль *WebTrafficAnomalyDetector*. Модуль захоплення трафіку постачає дані для моніторингу та навчання, модуль генерації аномалій створює тестові сценарії, а модуль датасетів забезпечує підготовлені дані для алгоритмів. Інтерфейс користувача координує виклики функцій, а логування забезпечує контроль за виконанням. Така модульна структура підвищує гнучкість системи, дозволяючи легко додавати нові алгоритми чи типи аномалій.

Композиційний аналіз демонструє, що система є добре структурованою та модульною, з чітким розподілом функцій між компонентами. Кожен модуль виконує спеціалізовану роль, від збору даних до їх аналізу та візуалізації, що забезпечує ефективне вирішення задачі виявлення аномалій у веб-трафіку. Використання нейронних мереж як одного з основних алгоритмів підкреслює сучасний підхід до обробки складних даних, а інтеграція з графічним інтерфейсом робить систему доступною для користувачів із різним рівнем підготовки.

1.5 Постановка задачі

У сучасному цифровому світі зростання обсягів веб-трафіку та ускладнення мережевих атак створюють значні виклики для забезпечення безпеки інформаційних систем. Аномалії в мережевому трафіку, такі як *DDoS*-атаки, сканування портів чи

флуд з'єднань, можуть призводити до порушення роботи систем, втрати даних або компрометації безпеки. Своєчасне виявлення таких аномалій є критично важливим для захисту мережевої інфраструктури. У цьому контексті розробка системи виявлення аномалій у веб-трафіку з використанням нейронних мереж є актуальним завданням, яке поєднує методи машинного навчання та аналізу даних для підвищення ефективності кібербезпеки.

Постановка задачі полягає в розробці автоматизованої системи, яка здатна виявляти аномалії в реальному часі у веб-трафіку мережі інтернет, використовуючи нейронні мережі як основний інструмент класифікації. Для чіткого визначення задачі її розбито на наступні пункти:

1) Збір та підготовка даних

Система має забезпечувати можливість збору даних про мережевий трафік, включаючи такі параметри, як розмір пакетів, частота передачі, кількість з’єднань, обсяг переданих та отриманих байтів тощо. Для цього необхідно реалізувати модуль захоплення трафіку через бібліотеки, такі як *psutil*, з підтримкою автоматичного визначення *Wi-Fi* інтерфейсу. Крім того, для навчання моделі потрібно створити синтетичні датасети, що моделюють нормальний та аномальний трафік (наприклад, *DDoS*-атаки, сканування портів), а також забезпечити можливість обробки реальних даних, отриманих у процесі моніторингу.

2) Розробка та навчання моделі машинного навчання

Основним інструментом виявлення аномалій обрано нейронну мережу, зокрема багатошарову перцептронну модель (*MLPClassifier*), яка здатна ефективно класифікувати складні нелінійні закономірності в даних. Для порівняння ефективності також необхідно реалізувати альтернативні алгоритми, такі як *Isolation Forest, One-Class SVM* та *Random Forest*. Модель має бути навчена на підготовлених датасетах, із застосуванням нормалізації даних (наприклад, за допомогою *StandardScaler*) та розподілом на навчальну й тестову вибірки для оцінки якості.

3) Реалізація моніторингу в реальному часі

Система повинна забезпечувати безперервний моніторинг мережевого трафіку з автоматичним виявленням аномалій. Для цього необхідно реалізувати потокову

обробку даних, де кожен новий пакет аналізується навченої моделлю, а оцінка аномальності порівнюється з фіксованим порогом (наприклад, 0.75). У разі виявлення аномалії система має генерувати повідомлення з деталями, такими як оцінка аномальності та відповідні мережеві параметри.

4) Генерація тестових аномалій

Для оцінки ефективності системи необхідно розробити модуль генерації штучних аномалій, таких як сканування портів, високий трафік або флуд з'єднань. Це дозволить перевірити чутливість моделі до різних типів аномальної поведінки та оцінити її здатність розрізняти нормальний і аномальний трафік.

5) Візуалізація результатів

Для зручності аналізу система має надавати графічне представлення даних і результатів, включаючи розподіл ознак, кореляційні матриці, ROC-криві, матриці помилок та графіки моніторингу в реальному часі. Візуалізація повинна бути інтегрована в інтерфейс користувача, реалізований за допомогою бібліотеки customtkinter, що забезпечує сучасний і зручний дизайн.

6) Розробка інтуїтивного інтерфейсу користувача

Система має включати графічний інтерфейс, який дозволяє користувачу керувати процесами збору даних, вибору датасетів, навчання моделей, моніторингу та генерації аномалій. Інтерфейс повинен відображати мережеву інформацію (наприклад, *IP*-адресу, назву хоста, інтерфейс) та логи роботи системи для полегшення діагностики.

7) Оцінка ефективності системи

Для оцінки якості роботи системи необхідно використовувати метрики, такі як точність (*accuracy*), *precision, recall, F1-score* та *AUC ROC*. Ці метрики дозволять кількісно оцінити здатність моделі правильно класифікувати нормальний і аномальний трафік, а також порівняти ефективність нейронної мережі з іншими алгоритмами.

8) Забезпечення надійності та масштабованості

Система має бути стійкою до помилок, що можуть виникати під *Research Assistant*: час збору даних або обробки великих обсягів трафіку. Для цього необхідно

реалізувати механізми логування (за допомогою бібліотеки *logging*) та обробки винятків. Крім того, система повинна бути адаптивною до різних операційних систем (зокрема *Windows*) шляхом автоматичного визначення мережевих інтерфейсів та кодування тексту (наприклад, cp866 для *Windows*).

Мета задачі полягає в створенні комплексної системи, яка не лише виявляє аномалії у веб-трафіку з високою точністю, але й забезпечує зручність використання, гнучкість у виборі алгоритмів та можливість тестування в реальних умовах. Реалізація такої системи сприятиме підвищенню безпеки *Wi-Fi* мереж, дозволяючи оперативно реагувати на потенційні загрози.

Очікувані результати включають:

– розроблену систему з інтуїтивним інтерфейсом для моніторингу та аналізу трафіку;

– навчену нейронну мережу, здатну класифікувати аномалії з високими показниками метрик якості;

– модуль генерації аномалій для тестування системи;

– візуалізацію даних і результатів для полегшення аналізу;

– документацію та логи роботи системи для подальшого вдосконалення.

Таким чином, поставлена задача охоплює повний цикл розробки — від збору даних до оцінки результатів, — поєднуючи науковий підхід із практичною реалізацією для вирішення актуальної проблеми кібербезпеки.

2 АНАЛІЗ МЕТОДІВ ТА ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Класифікація вимог

Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж передбачає чітке визначення вимог, які забезпечують її функціональність, ефективність та зручність використання. Вимоги поділяються на функціональні, що визначають основні можливості системи, та нефункціональні, які стосуються її якості, продуктивності та інших характеристик. Нижче наведено детальний аналіз цих вимог, сформований на основі аналізу коду проєкту та тематики проєкту.

2.1.1 Функціональні вимоги

Функціональні вимоги описують конкретні функції, які система повинна виконувати для забезпечення виявлення аномалій у веб-трафіку. Вони відображають основні задачі, які реалізуються в коді, та відповідають цілям проєкту. Основні функціональні вимоги до системи показані в таблиці 2.1.

Таблиця 2.1

Функціональні вимоги

№	Вимога	Опис
1	2	3
1	Захоплення мережевого трафіку	Система повинна забезпечувати захоплення даних веб-трафіку в реальному часі з *Wi-Fi* інтерфейсу. Код реалізує це через клас *WiFiTrafficCapture*, який використовує бібліотеку *psutil* для збору статистики мережевих з'єднань, включаючи кількість відправлених і отриманих байтів, пакетів, а також активних TCP/UDP-з'єднань
		Автоматичне визначення *Wi-Fi* інтерфейсу (з підтримкою *Windows* через *netsh*) для забезпечення універсальності роботи системи на різних платформах

Продовження таблиці 2.1

1	2	3
2	Генерація тестових аномалій	Система повинна мати можливість генерувати штучні аномалії для тестування алгоритмів виявлення. Клас *NetworkAnomalyGenerator* реалізує три типи аномалій: сканування портів (*generate_port_scan*), високий трафік (*generate_high_traffic*) та флуд з'єднань (*generate_connection_flood*). Це дозволяє перевіряти ефективність моделей у контрольованих умовах
3	Створення та обробка датасетів	Система повинна підтримувати створення синтетичних датасетів для навчання моделей. Клас *WebTrafficDataset* генерує три типи датасетів: зразковий (*create_sample_dataset*), *DDoS (create_ddos_dataset)* та сканування портів (*create_portscan_dataset*). Кожен датасет містить ознаки, такі як розмір пакетів, швидкість передачі, кількість з'єднань тощо, а також мітки нормального та аномального трафіку
		Забезпечується можливість обробки даних реального часу, отриманих через захоплення трафіку, з подальшим їх перетворенням у формат, придатний для аналізу
4	Навчання моделей машинного навчання	Система повинна підтримувати навчання кількох моделей для виявлення аномалій, включаючи нейронну мережу (*MLPClassifier*), ізоляційний ліс (*IsolationForest*), однокласовий *SVM (OneClassSVM)* та випадковий ліс (*RandomForestClassifier*). Клас *WebTrafficAnomalyDetector* реалізує методи для навчання цих моделей
		Реалізується нормалізація даних за допомогою *StandardScaler* для забезпечення коректної роботи моделей
5	Моніторинг трафіку в реальному часі	Система повинна аналізувати мережевий трафік у реальному часі та виявляти аномалії на основі навченої моделі. Метод *monitor_traffic* періодично обробляє дані, отримані через *psutil*, та визначає аномалії, використовуючи нормалізовані оцінки та фіксований поріг (0.75)
		Забезпечується логування виявлених аномалій із зазначенням оцінки аномальності та часу виявлення
6	Візуалізація даних і результатів	Система повинна надавати графічне представлення даних, результатів навчання та моніторингу. Реалізовано через бібліотеки *matplotlib* і *seaborn* у методах *visualize_data*, *visualize_training_results* та *update_monitoring_plots*. Візуалізації включають гістограми розподілу ознак, кореляційні матриці, матриці помилок, *ROC*-криві та графіки оцінок аномальності в реальному часі
		Інтерфейс користувача, побудований за допомогою *customtkinter*, включає вкладки для відображення даних, результатів навчання та моніторингу

Завершення таблиці 2.1

1	2	3
7	Інтерактивний інтерфейс користувача	Система повинна мати зручний графічний інтерфейс для керування функціоналом. Клас *WebTrafficAnomalyDetector* створює інтерфейс із секціями для вибору датасетів, моделей, запуску моніторингу, генерації аномалій та перегляду логів. Користувач може обирати тип датасету, модель, запускати або зупиняти моніторинг, а також генерувати тестові аномалії
8	Логування подій	Система повинна вести журнал подій для відстеження дій користувача, помилок і виявлених аномалій. Логування реалізовано через модуль *logging* із записом у файл (*anomaly_detector.log*) та виведенням у текстове поле інтерфейсу

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, такі як продуктивність, зручність використання, надійність та масштабованість. Вони забезпечують відповідність системи очікуванням користувачів і технічним стандартам. Основні нефункціональні вимоги до системи такі наведені в таблиці 2.2.

Таблиця 2.2

Нефункціональні вимоги

№	Вимога	Опис
1	2	3
1	Продуктивність	Система повинна обробляти дані в реальному часі з мінімальною затримкою. Час обробки одного зразка трафіку під час моніторингу не повинен перевищувати 1 секунди, що забезпечується використанням ефективних бібліотек (*psutil, numpy, pandas*) та асинхронної обробки через потоки (*threading*)
		Навчання моделей повинно відбуватися за розумний час (не більше 5 хвилин для датасету з 10 000 записів на стандартному апаратному забезпеченні)

Продовження таблиці 2.2

1	2	3
2	Зручність використання	Інтерфейс користувача повинен бути інтуїтивно зрозумілим, із чітким розташуванням елементів керування та візуалізацій. Використання *customtkinter* із темною темою та сучасним дизайном сприяє зручності взаємодії
		Система повинна надавати інформативні повідомлення про помилки та статуси операцій через логування та текстове поле в інтерфейсі
3	Надійність	Система повинна стабільно працювати при тривалому моніторингу (понад 24 години) без збоїв. Використання обробки винятків у коді (наприклад, у методах захоплення трафіку та генерації аномалій) забезпечує стійкість до помилок
		Логування всіх критичних подій дозволяє діагностувати проблеми та відновлювати роботу системи
4	Портативність	Система повинна працювати на різних операційних системах, зокрема *Windows* і *Linux*, що забезпечується використанням кросплатформених бібліотек (*psutil*, *customtkinter*) та автоматичним визначенням мережевих інтерфейсів
		Код враховує особливості *Windows* (наприклад, кодування *cp866* для команд *netsh*), що підвищує сумісність
5	Масштабованість	Система повинна обробляти датасети різного обсягу (від 5 000 до 100 000 записів) без значного зниження продуктивності. Використання *pandas* і *numpy* забезпечує ефективну обробку великих масивів даних
		Можливість додавання нових моделей і типів аномалій шляхом розширення класів *WebTrafficAnomalyDetector* і *NetworkAnomalyGenerator*
6	Безпека	Генерація аномалій (наприклад, сканування портів або флуд з'єднань) повинна виконуватися в контрольованому середовищі, щоб уникнути впливу на реальні мережеві системи. Код використовує локальні *IP*-адреси (наприклад, 127.0.0.1) та обмеження на кількість з'єднань
		Логування не повинно містити конфіденційних даних, таких як реальні *IP*-адреси або вміст пакетів
7	Локалізація	Система повинна підтримувати багатомовність, зокрема українську мову, що відображено в інтерфейсі (назви вкладок, кнопок) та коментарях у коді. Це забезпечує зручність використання для україномовних користувачів

Завершення таблиці 2.2

1	2	3
8	Документованість	Код повинен бути добре документованим, із чіткими коментарями до класів і методів. У коді проєкту кожен клас (*NetworkAnomalyGenerator, WiFiTrafficCapture, WebTrafficDataset, WebTrafficAnomalyDetector*) супроводжується описом його призначення, а методи містять коментарі щодо їх функціональності

Функціональні вимоги забезпечують повноцінну реалізацію системи виявлення аномалій, включаючи захоплення трафіку, генерацію аномалій, обробку даних, навчання моделей, моніторинг і візуалізацію. Нефункціональні вимоги гарантують продуктивність, зручність, надійність і масштабованість системи, що робить її придатною для практичного використання. Код проєкту демонструє відповідність цим вимогам, забезпечуючи гнучкість, кросплатформність і зручний інтерфейс для аналізу веб-трафіку.

2.2 Обмеження проектування

Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж є складним завданням, що вимагає врахування численних обмежень на етапі проектування. Ці обмеження впливають на архітектуру системи, вибір методів, продуктивність, масштабованість та практичне застосування. Нижче наведено детальний аналіз обмежень проектування, структурований за ключовими показниками (рис. 2.1).

Рисунок 2.1 Основні показники обмеження проєктування

Обмеження даних. Одним із основних обмежень є якість, обсяг і доступність даних для навчання моделей. У представленому коді система використовує синтетичні датасети (наприклад, зразковий датасет, *DDoS*-датасет, датасет сканування портів), створені за допомогою статистичних розподілів, а також реальний мережевий трафік, захоплений через бібліотеку *psutil*. Синтетичні дані дозволяють моделювати різні сценарії аномалій, але вони можуть не повною мірою відображати реальні мережеві умови, що знижує узагальнюючу здатність моделей. Захоплення реального трафіку, у свою чергу, обмежене такими факторами:

– Низька деталізація даних проявляється через використання *psutil*, що дозволяє отримати агреговану статистику, включаючи кількість відправлених та отриманих байтів, пакети та активні з’єднання. Водночас цей підхід не забезпечує доступ до детальної інформації на рівні окремих пакетів, таких як заголовки *TCP/IP* або їх вміст, що обмежує можливість виявлення складних аномалій, зокрема специфічних сигнатур атак.

– Відсутність міток у реальному трафіку спричиняє ситуацію, коли захоплені дані за замовчуванням позначаються як нормальні, тобто значення

is_anomaly дорівнює 0. Це ускладнює навчання моделей на даних, які містять реальні аномалії, оскільки потребує додаткової анотації для коректного аналізу.

– Обмежений обсяг даних зумовлений коротким періодом захоплення трафіку, наприклад, упродовж 10 секунд, як це реалізовано у коді. Такий метод формує невеликі набори даних, які можуть бути недостатніми для ефективного навчання глибоких нейронних мереж, що потребують значних обсягів інформації для коректної роботи.

Для подолання цих обмежень необхідно інтегрувати зовнішні джерела даних, такі як публічні датасети (*CICIDS2017, NSL-KDD*) або використовувати інструменти глибокого аналізу пакетів (наприклад, *Wireshark, Scapy*).

Обмеження обчислювальних ресурсів. Система передбачає використання кількох моделей машинного навчання, включаючи нейронні мережі (*MLPClassifier*), ізоляційний ліс, однокласовий *SVM* та випадковий ліс. Кожна з цих моделей має свої вимоги до обчислювальних ресурсів:

– Нейронні мережі потребують значних обчислювальних ресурсів для навчання моделі з трьома прихованими шарами (64, 32, 16 нейронів), особливо при зростанні розміру датасету. У коді використовується бібліотека *scikit-learn*, яка не підтримує апаратне прискорення, наприклад, через *GPU*, що може уповільнити навчання на великих наборах даних.

– Моніторинг у реальному часі передбачає періодичне захоплення даних щосекунди та прогнозування аномалій. Це створює додаткове навантаження на процесор і пам’ять, особливо при одночасному виконанні кількох потоків, таких як захоплення трафіку, прогнозування та оновлення графічного інтерфейсу.

– Обмеження апаратного забезпечення полягає в тому, що система розроблена для роботи на персональних комп’ютерах, що може ускладнити її масштабування на серверні середовища або вбудовані пристрої з обмеженими ресурсами, такі як маршрутизатори.

Для зменшення впливу цих обмежень можна оптимізувати моделі (наприклад, зменшити кількість нейронів або шарів у нейронній мережі) або використовувати фреймворки, що підтримують *GPU (TensorFlow, PyTorch)*.

Обмеження алгоритмів. Вибір алгоритмів для виявлення аномалій також накладає певні обмеження:

– Чутливість до гіперпараметрів може впливати на продуктивність моделі, оскільки в коді для нейронної мережі використовуються фіксовані значення параметрів, таких як розміри шарів, функція активації *relu* та оптимізатор *adam*. Наприклад, параметр *nu* в однокласовому *SVM* або *contamination* в ізоляційному лісі впливають на чутливість до аномалій, але їх значення залишаються незмінними, що може обмежувати адаптивність моделі до різних типів даних.

– Нездатність обробляти складні аномалії зумовлена використанням агрегованих ознак, таких як кількість байтів або пакети за секунду, що обмежує можливість моделі розпізнавати складні загрози. Низькоінтенсивні атаки або аномалії, які формуються через послідовність подій, можуть залишитися непоміченими, оскільки модель не має достатнього рівня деталізації вхідних даних.

– Проблема незбалансованих даних виникає через те, що у синтетичних датасетах аномалії складають 20–30% всіх записів, що не відповідає реальним сценаріям, де їх частка зазвичай не перевищує 1–5%. Це може призвести до перенавчання моделі на аномаліях, що знижує її здатність адекватно працювати у реальних умовах та коректно прогнозувати нетипові ситуації.

Для покращення алгоритмів слід реалізувати автоматичний підбір гіперпараметрів (наприклад, через *GridSearchCV*) та використовувати методи обробки незбалансованих даних (*SMOTE, ADASYN*).

Обмеження інтерфейсу користувача. Графічний інтерфейс, реалізований за допомогою *customtkinter*, забезпечує зручну взаємодію з системою, але має свої обмеження:

– Обмежена інтерактивність не дозволяє користувачу налаштовувати параметри моделей, такі як поріг аномалії або архітектуру нейронної мережі, через інтерфейс. Поріг аномалії має фіксоване значення 0.75 для всіх моделей, що зменшує гнучкість системи та можливість її адаптації до різних умов.

– Продуктивність інтерфейсу може знижуватися через постійне оновлення графіків, наприклад, у режимі моніторингу. Це створює ризик затримок або зависань,

особливо на комп’ютерах із меншою потужністю, що впливає на комфортність використання.

– Локалізація підтримує українську мову в коді, включаючи назви кнопок та логи, проте система не передбачає повноцінного перекладу на інші мови. Це обмежує її застосування в міжнародному контексті та зменшує доступність для користувачів із різних країн.

Ці обмеження можна подолати шляхом додавання інтерактивних елементів для налаштування параметрів і оптимізації оновлення графіків (наприклад, використання асинхронного рендерингу).

Обмеження безпеки та етики. Система включає модуль генерації аномалій (*NetworkAnomalyGenerator*), який здатен створювати порт-сканування, високий трафік і флуд з’єднань. Це накладає етичні та безпекові обмеження:

– Потенційна шкода мережі може виникнути через генерацію аномалій, таких як флуд з’єднань, що здатне порушити роботу локальної мережі або цільового пристрою. Контроль параметрів, зокрема тривалості та цільової IP-адреси, є необхідним для запобігання негативним наслідкам.

– Юридичні ризики пов’язані з використанням системи для тестування мереж без дозволу власника, що може суперечити законодавству про кібербезпеку. Дотримання нормативних вимог та отримання офіційного дозволу є важливими для законного проведення тестувань.

– Відсутність захисту даних створює загрозу витоку інформації, оскільки захоплені дані трафіку зберігаються у файлах у директорії *traffic_data* без шифрування або інших засобів безпеки. Реалізація заходів з обмеження доступу та шифрування даних сприяє підвищенню рівня захисту.

Для вирішення цих проблем необхідно додати механізми обмеження впливу генератора аномалій (наприклад, використання тестових мереж) і реалізувати шифрування даних.

Обмеження масштабованості. Система розроблена для роботи на одному комп’ютері, що обмежує її масштабування:

– Локальна обробка даних передбачає виконання всіх обчислень, таких як навчання, прогнозування та моніторинг, на локальному рівні, що обмежує можливість роботи з великими обсягами трафіку в корпоративних мережах.

– Відсутність інтеграції з мережевою інфраструктурою означає, що система не підтримує взаємодію з мережевими пристроями, включаючи маршрутизатори та міжмережеві екрани, а також не має інтеграції з системами *SIEM* для централізованого аналізу.

– Обмеження потокової обробки виникає через використання *psutil* для захоплення даних, що не забезпечує ефективну роботу з великими потоками трафіку в реальному часі та потребує застосування спеціалізованих інструментів, таких як *Apache Kafka*.

Для підвищення масштабованості можна реалізувати розподілену архітектуру з використанням хмарних сервісів або спеціалізованих платформ для аналізу трафіку.

Обмеження операційної системи. Система враховує особливості різних операційних систем (*Windows, Linux*) для визначення мережевих інтерфейсів, але має обмеження:

– Залежність від платформи може спричиняти ненадійність роботи *subprocess* для *Windows*, оскільки виконання команди *netsh wlan show interfaces* та використання *psutil* залежить від конфігурації системи, зокрема наявності прав адміністратора.

– Обмежена підтримка інших ОС зумовлена тим, що код не враховує специфіку *macOS* або *Unix*-подібних систем, що знижує його переносимість та можливість застосування в різних середовищах.

Для забезпечення кросплатформності необхідно розширити логіку виявлення інтерфейсів і протестувати систему на різних операційних системах.

Обмеження проектування системи виявлення аномалій у веб-трафіку охоплюють показники даних, обчислювальних ресурсів, алгоритмів, інтерфейсу, безпеки, масштабованості та сумісності з операційними системами. Ці обмеження зумовлені як технічними характеристиками (обмежена деталізація даних, фіксовані гіперпараметри), так і етичними та організаційними факторами (юридичні ризики,

локальна обробка). Для підвищення ефективності системи необхідно інтегрувати зовнішні датасети, оптимізувати моделі, додати гнучкість налаштувань і забезпечити безпеку даних. Подолання цих обмежень дозволить створити більш універсальну та надійну систему для виявлення аномалій у реальних мережевих середовищах.

2.3 Аналіз методів та обґрунтування вибору

Розробка системи виявлення аномалій у веб-трафіку є складним завданням, що вимагає ретельного аналізу доступних методів машинного навчання, оцінки їх ефективності та відповідності специфіці задачі. У процесі створення системи, представленої у коді проєкту, було проаналізовано кілька підходів до виявлення аномалій, зокрема методи ізольованого лісу (*Isolation Forest*), однокласової *SVM (One-Class SVM)*, нейронних мереж *(MLPClassifier)* та випадкового лісу *(Random Forest)*. Нижче наведено детальний аналіз цих методів, їх переваги, недоліки та обґрунтування вибору для реалізації системи.

Метод ізольованого лісу *(Isolation Forest)* є одним із найпоширеніших алгоритмів для виявлення аномалій у даних високої розмірності [9]. Його основна ідея полягає у випадковому розбитті даних на підмножини шляхом створення бінарних дерев. Аномалії, як правило, ізолюються швидше через їх відмінність від нормальних даних, що дозволяє алгоритму ефективно їх виявляти.

Переваги:

– висока швидкість обробки великих обсягів даних завдяки низькій обчислювальній складності *(O(n log n))*;

– ефективність у виявленні аномалій у даних із високою розмірністю, що важливо для аналізу мережевого трафіку;

– не потребує маркованих даних, що робить його придатним для напівконтрольованого навчання;

– стійкість до шуму в даних.

Недоліки:

– чутливість до вибору параметра контамінації (*contamination*), який визначає очікувану частку аномалій;

– менш ефективний для даних із складними нелінійними залежностями;

– може потребувати додаткової обробки для інтерпретації результатів.

Застосування в системі: метод *Isolation Forest* використовується як один із базових алгоритмів завдяки його швидкості та здатності працювати з немаркованими даними. Він ефективно застосовується для початкового аналізу трафіку та виявлення аномалій у реальному часі.

Однокласова *SVM (One-Class SVM)* є класичним методом для виявлення аномалій, який навчається на нормальних даних і визначає межі, що відокремлюють нормальні дані від аномальних [10]. Алгоритм використовує ядрові функції для проекції даних у простір вищої розмірності, де аномалії виявляються як точки, що лежать поза межами нормального класу.

Переваги:

– ефективність у задачах із чітко визначеним нормальним класом;

– можливість використання нелінійних ядрових функцій (наприклад, *RBF*) для моделювання складних розподілів;

– хороша інтерпретація результатів через оцінку відстані до межі рішення.

Недоліки:

– висока обчислювальна складність *($O(n^2)$ або $O(n^3)$)*, що ускладнює масштабування на великі набори даних;

– чутливість до вибору параметрів (nu, gamma), які потребують ретельної настройки;

– потреба в достатній кількості нормальних даних для навчання.

Застосування в системі: *One-Class SVM* використовується для аналізу трафіку, де є можливість виділити нормальний клас. Алгоритм застосовується для сценаріїв із чітко визначеними нормальними патернами, наприклад, у *DDoS*-атаках, де аномалії мають виражені характеристики.

Нейронні мережі (*MLPClassifier*) [11]. Багатошаровий перцептрон (*MLP*) є потужним інструментом для моделювання нелінійних залежностей у даних. У контексті виявлення аномалій нейронні мережі можуть використовуватися як для контрольованого, так і для напівконтрольованого навчання, прогнозуючи ймовірність належності до аномального класу.

Переваги:

– здатність моделювати складні нелінійні залежності, що важливо для аналізу веб-трафіку з різноманітними патернами;

– гнучкість у налаштуванні архітектури (кількість шарів, нейронів, функції активації);

– можливість використання ймовірнісних оцінок для визначення порогу аномалій;

– адаптивність до різних типів даних після відповідного масштабування.

Недоліки:

– висока обчислювальна складність і потреба в значних обчислювальних ресурсах;

– чутливість до якості та обсягу даних, особливо для маркованих наборів;

– ризик перенавчання без належної регуляризації;

– потреба в ретельній настройці гіперпараметрів (наприклад, розмір шарів, швидкість навчання).

Застосування в системі: *MLPClassifier* використовується як основний метод для виявлення аномалій завдяки його здатності обробляти складні патерни веб-трафіку. Алгоритм застосовується для аналізу синтетичних і реальних даних, що включають *DDoS*-атаки та сканування портів, з використанням ймовірнісного порогу для класифікації.

Випадковий ліс *(Random Forest)* є ансамблевим методом, який комбінує кілька дерев рішень для підвищення точності та стійкості прогнозування [12]. У задачах виявлення аномалій він використовується для класифікації на основі ознак, виділених із даних.

Переваги:

– висока точність завдяки ансамблевому підходу;

– стійкість до шуму та пропусків у даних;

– можливість оцінки важливості ознак, що допомагає в аналізі трафіку;

– відносно швидке навчання порівняно з нейронними мережами.

Недоліки:

– потреба в маркованих даних для контрольованого навчання;

– обмежена здатність моделювати дуже складні нелінійні залежності порівняно з нейронними мережами;

– висока пам’яткоємність для великих наборів даних.

Застосування в системі: *Random Forest* використовується в коді як альтернативний метод для сценаріїв, де доступні марковані дані. Він ефективний для аналізу синтетичних датасетів із чітко визначеними класами (нормальний/аномальний трафік).

Вибір методів для реалізації системи виявлення аномалій у веб-трафіку базувався на кількох ключових критеріях: ефективність алгоритму, адаптивність до специфіки мережевого трафіку, обчислювальна складність, наявність маркованих даних і можливість роботи в реальному часі. Детальне обґрунтування вибору методів наведено в таблиці 2.3.

Таблиця 2.3

Причини вибору методів для реалізації системи виявлення аномалій у веб-трафіку

№	Причина	Опис
1	Гнучкість і адаптивність	Нейронні мережі (*MLPClassifier*) були обрані як основний метод завдяки їх здатності моделювати складні нелінійні залежності в даних веб-трафіку. Оскільки трафік може мати різноманітні патерни (наприклад, *DDoS*-атаки, сканування портів, флуд з’єднань), нейронні мережі забезпечують високу адаптивність до різних типів аномалій. Крім того, використання ймовірнісних оцінок дозволяє гнучко налаштовувати поріг для класифікації аномалій

2	Ефективність у реальному часі	Для моніторингу трафіку в реальному часі важлива швидкість обробки даних. Ізольований ліс був обраний як швидкий і ефективний метод для початкового аналізу завдяки низькій обчислювальній складності. Він дозволяє оперативно виявляти аномалії навіть на великих обсягах даних, що критично для потокового аналізу
3	Робота з немаркованими даними	У реальних умовах марковані дані можуть бути обмеженими або відсутніми. Однокласова *SVM* та ізольований ліс ідеально підходять для напівконтрольованого навчання, де алгоритм навчається на нормальних даних і виявляє відхилення. Ці методи були включені до системи для аналізу сценаріїв із обмеженим доступом до аномальних прикладів
4	Інтерпретація та оцінка ознак	Випадковий ліс був обраний для сценаріїв із маркованими даними, оскільки він забезпечує високу точність і дозволяє оцінити важливість ознак. Це корисно для аналізу, який потребує розуміння, які саме характеристики трафіку (наприклад, розмір пакетів, кількість з’єднань) найбільше впливають на виявлення аномалій
5	Комплексний підхід	Система реалізує кілька методів, щоб забезпечити гнучкість і можливість порівняння їх ефективності. Користувач може вибрати відповідний алгоритм залежно від типу датасету (зразковий, *DDoS*, сканування портів, реальний трафік) і вимог до швидкості чи точності

Аналіз методів показав, що кожен із розглянутих алгоритмів має свої сильні сторони та обмеження. Нейронні мережі були обрані як основний метод завдяки їх здатності обробляти складні патерни та адаптивності до різноманітних типів аномалій. Ізольований ліс і однокласова *SVM* забезпечують швидке виявлення аномалій у реальному часі та роботу з немаркованими даними, тоді як випадковий ліс є ефективним для маркованих наборів і оцінки важливості ознак. Комплексне використання цих методів у системі дозволяє досягти балансу між точністю, швидкістю та гнучкістю, що робить її універсальним інструментом для аналізу веб-трафіку. Реалізація системи, представлена у коді, підтверджує практичну застосовність обраних підходів, забезпечуючи інтуїтивно зрозумілий інтерфейс і можливість тестування різних сценаріїв аномалій.

2.4 Алгоритм обробки даних

Розробка системи виявлення аномалій у веб-трафіку, реалізована в коді проєкту, передбачає комплексний алгоритм обробки даних, який включає етапи збору, попередньої обробки, аналізу та візуалізації мережевих даних. Алгоритм структурований таким чином, щоб забезпечити ефективне виявлення аномалій у реальному часі з використанням нейронних мереж та інших методів машинного навчання. Детальний опис алгоритму обробки даних, розбитий на ключові етапи, які відображають логіку роботи системи наведено в таблиці 2.4.

Таблиця 2.4

Детальний опис алгоритму обробки даних

№	Опис	Реалізація	Значення
1	3	4	5
	Ініціалізація системи та налаштування інтерфейсу		
	На початковому етапі створюється графічний інтерфейс користувача (*GUI*) за допомогою бібліотеки customtkinter. Система ініціалізує основні компоненти, такі як менеджер датасетів (*WebTrafficDataset*), модуль захоплення трафіку (*WiFiTrafficCapture*), генератор аномалій (*NetworkAnomalyGenerator*) та словник для зберігання навчених моделей	У класі *WebTrafficAnomalyDetector* створюється головне вікно програми з вкладками для відображення даних, результатів навчання та моніторингу. Налаштовується логування через модуль *logging* для збереження інформації про виконання програми у файл *anomaly_detector.log* та виведення в консоль	Цей етап забезпечує інтерактивність системи, дозволяючи користувачу обирати датасети, моделі та керувати процесом моніторингу
2	Збір даних		
	Система підтримує два режими збору даних: генерація синтетичних датасетів та захоплення реального мережевого трафіку. Синтетичні датасети створюються для моделювання нормального та аномального трафіку (наприклад, *DDoS*-атаки, сканування портів). Реальний трафік захоплюється через бібліотеку *psutil*	У класі *WebTrafficDataset* методи *create_sample_dataset*, *create_ddos_dataset* та *create_portscan_dataset* генерують синтетичні дані з різними характеристиками (розмір пакетів, кількість з'єднань тощо) У класі *WiFiTrafficCapture* метод *capture_with_psutil* періодично збирає статистику мережевого трафіку, таку як обсяг відправлених/отриманих байтів, кількість пакетів та активних з'єднань	Різноманітність джерел даних дозволяє системі адаптуватися до різних сценаріїв, від тестування на контрольованих даних до аналізу реального трафіку

Продовження таблиці 2.4

1	2	3	4
3	Попередня обробка даних		
	Зібрані дані проходять етап попередньої обробки для забезпечення їх придатності до аналізу. Цей етап включає очищення, нормалізацію та відбір ознак	Очищення даних: перевіряється наявність пропущених значень (*NaN*). Якщо вони виявляються, замінюються середніми значеннями за допомогою *np.nan_to_num*	Попередня обробка підвищує точність моделей, усуваючи шум та нормалізуючи дані для ефективного навчання
		Нормалізація використовується *StandardScaler* для масштабування числових ознак, що забезпечує однакову вагу всіх ознак у моделях машинного навчання	
		Відбір ознак: Вибираються лише числові колонки, виключаючи цільову змінну *is_anomaly*. Для синтетичних датасетів ознаки формуються заздалегідь, для реального трафіку — адаптуються до доступної статистики	
4	Навчання моделі		
	Система підтримує чотири алгоритми машинного навчання: *Isolation Forest, One-Class SVM, Neural Network (MLPClassifier)* та *Random Forest*. Користувач обирає модель через інтерфейс, після чого відбувається її навчання на підготовлених даних	Дані розбиваються на навчальну та тестову вибірки (*train_test_split*) у співвідношенні 80:20. Для кожної моделі виконується навчання:	Різноманітність моделей дозволяє порівнювати їх ефективність, а адаптивний поріг оптимізує виявлення аномалій
		Isolation Forest використовує параметр *contamination=0.1* для оцінки частки аномалій.	
		One-Class SVM. Навчається лише на нормальних даних з параметром *nu*=0.1	
		Neural Network використовує багатошарову архітектуру з активацією relu	
		Random Forest налаштовується з 100 деревами та максимальною глибиною 10	
		Після навчання визначається поріг аномалій	

Продовження таблиці 2.4

1	2	3	4
5	Аналіз даних та виявлення аномалій		
	Навчена модель використовується для аналізу даних у реальному часі або на тестових даних. Оцінка аномалій базується на порівнянні ймовірностей або оцінок із встановленим порогом	Для реального часу метод *monitor_traffic* періодично збирає статистику мережі, нормалізує дані та передає їх моделі	Цей етап забезпечує швидке виявлення аномалій, що критично для реагування на потенційні загрози
		Залежно від моделі, оцінка аномалії може бути: – Ймовірністю (для *Neural Network, Random Forest*) Оцінкою відхилення (для *Isolation Forest, One-Class SVM*)	
		Нормалізована оцінка порівнюється з фіксованим порогом (0.75), і якщо вона перевищує поріг, дані позначаються як аномальні	
6	Візуалізація результатів		
	Результати обробки даних відображаються у вигляді графіків на вкладках *GUI*. Візуалізація включає розподіл даних, результати навчання та моніторинг у реальному часі	Вкладка "Дані": Відображає гістограми розподілу ознак, кореляційну матрицю та розподіл міток (*is_anomaly*)	Візуалізація полегшує інтерпретацію результатів, дозволяючи користувачу оцінити ефективність моделі та виявлені аномалії
		Вкладка "Навчання": Показує матрицю помилок, ROC-криву та розподіл оцінок аномалій	
		Вкладка "Моніторинг": Графік у реальному часі відображає нормалізовані оцінки аномалій, поріг та позначки аномальних точок	
		Використовуються бібліотеки *matplotlib* та *seaborn* для створення графіків, інтегрованих у *GUI* через *FigureCanvasTkAgg*	

Завершення таблиці 2.4

1	2	3	4
7	Логування та генерація звітів		
	Усі дії системи супроводжуються логуванням, а результати аналізу формуються у звіти. Логи включають повідомлення про завантаження даних, навчання моделей, виявлення аномалій та помилки	Метод *log_message* записує повідомлення з часовою міткою у текстове поле *GUI* та файл логу	Логування забезпечує прозорість роботи системи, а звіти дозволяють оцінити якість моделі
		Звіти про навчання включають метрики (точність, *precision, recall, F1-score*), отримані за допомогою *classification_report*	
8	Генерація аномалій для тестування		
	Для перевірки ефективності системи користувач може генерувати штучні аномалії, такі як сканування портів, високий трафік або флуд з’єднань	Клас *NetworkAnomalyGenerator* реалізує методи *generate_port_scan*, *generate_high_traffic* та *generate_connection_flood*, які імітують аномальну поведінку мережі	Генерація аномалій дозволяє тестувати систему в контрольованих умовах, оцінюючи її чутливість до різних типів загроз

Алгоритм обробки даних (рис. 2.2) у системі виявлення аномалій у веб-трафіку є комплексним і багатоступеневим, охоплюючи всі етапи від збору даних до їх візуалізації та аналізу. Використання нейронних мереж поряд з іншими методами машинного навчання забезпечує гнучкість і високу точність у виявленні аномалій. Інтеграція з *GUI* та логування підвищують зручність використання системи, роблячи її доступною як для дослідників, так і для практичного застосування в мережевій безпеці [13].

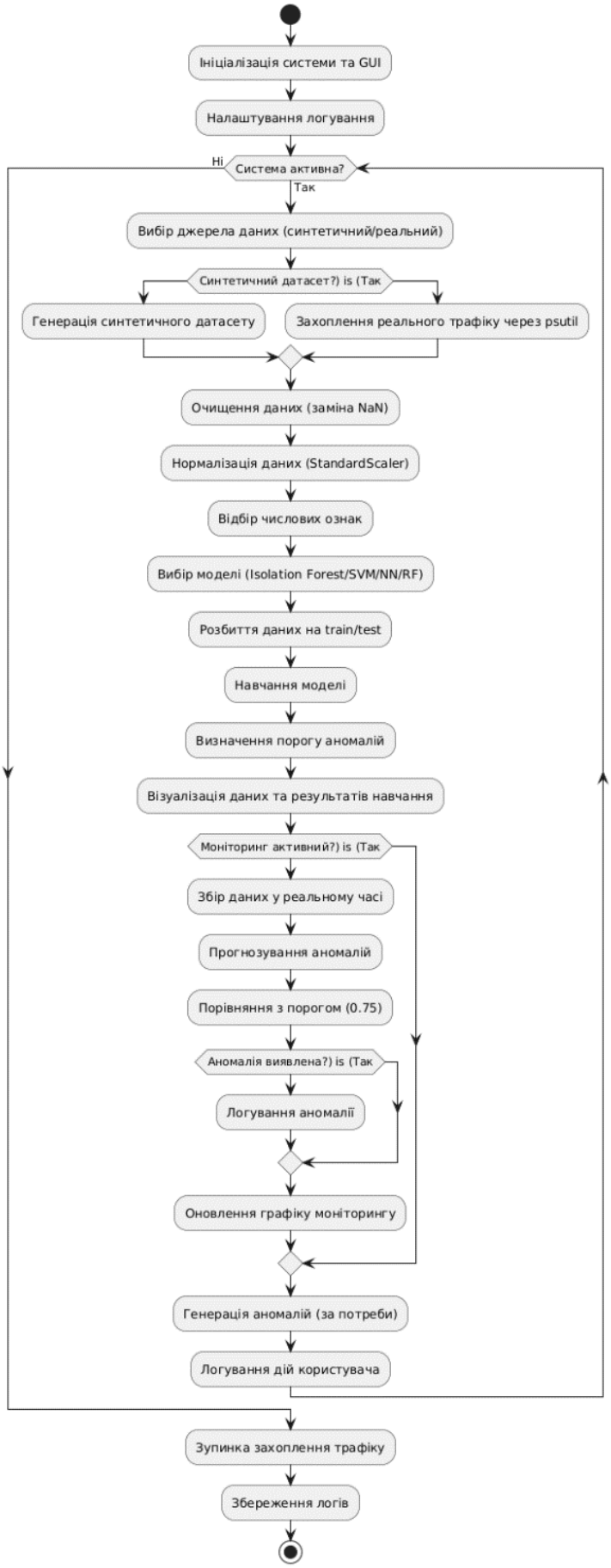

Рисунок 2.2 Алгоритм обробки даних

3 РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Архітектурні рівні системи

Система виявлення аномалій у веб-трафіку, розроблена на основі нейронних мереж та інших методів машинного навчання, має багатошарову архітектуру, яка забезпечує ефективне виконання завдань збору, обробки, аналізу даних та виявлення аномалій у реальному часі. Архітектура системи структурована на кількох рівнях, кожен з яких відповідає за певний функціональний елемент (рис. 3.1). Такий підхід дозволяє забезпечити модульність, гнучкість та масштабованість системи. Нижче детально описано архітектурні рівні системи, їх функції та взаємодію.

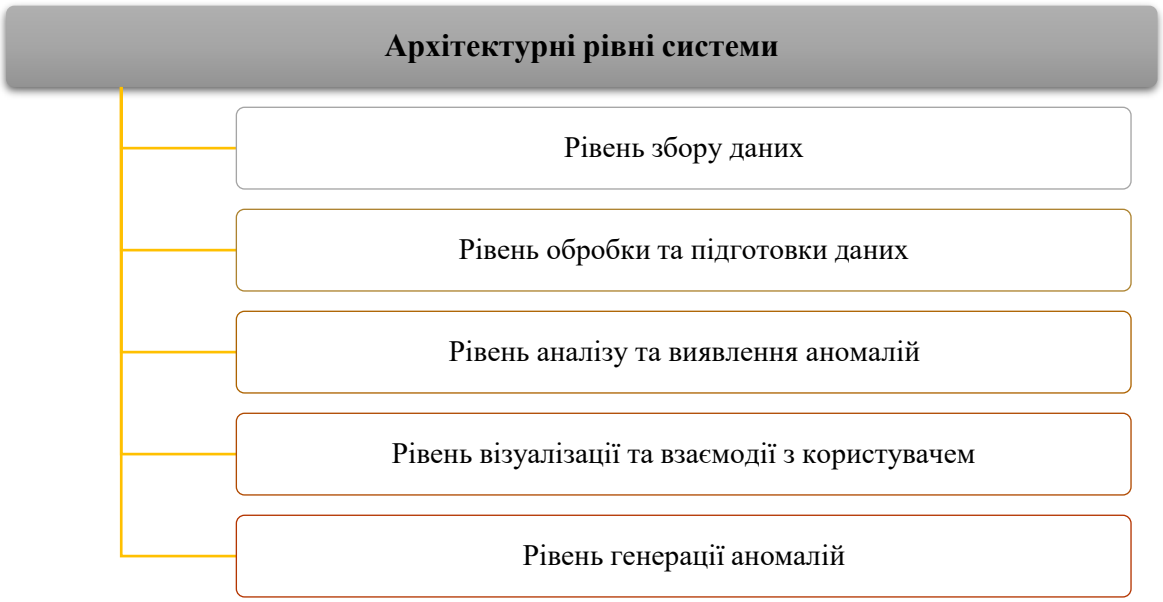

Рисунок 3.1 Архітектурні рівні системи

Рівень збору даних є базовим і відповідає за отримання інформації про мережевий трафік. У реалізації системи цей рівень представлений класом WiFiTrafficCapture, який забезпечує захоплення статистики мережевого трафіку через бібліотеку psutil. Основні функції цього рівня показані в таблиці 3.1.

Таблиця 3.1

Основні функції рівня збору даних

№	Функції	Опис
1	Автоматичне визначення мережевого інтерфейсу	Система автоматично виявляє активний Wi-Fi інтерфейс, використовуючи методи аналізу доступних мережевих інтерфейсів (psutil.net_if_addrs) та, у випадку Windows, команди netsh. Це дозволяє адаптувати систему до різних апаратних конфігурацій
2	Збір статистики трафіку	Через регулярні інтервали (1 секунда) система фіксує ключові метрики, такі як кількість відправлених та отриманих байтів, кількість пакетів, активні з'єднання, а також TCP- та UDP-з'єднання. Ці дані формують основу для подальшого аналізу
3	Генерація синтетичних даних	Для тестування та навчання моделей система включає клас WebTrafficDataset, який створює синтетичні датасети, що імітують нормальний та аномальний трафік (наприклад, DDoS-атаки або сканування портів). Це дозволяє проводити тестування системи в контрольованих умовах

Рівень збору даних забезпечує гнучкість у виборі джерел даних, підтримуючи як реальний трафік, так і синтетичні набори, що є важливим для тестування та відлагодження.

Рівень обробки та підготовки даних відповідає за попередню обробку зібраних даних, їх нормалізацію та підготовку до аналізу. Основні задачі рівня реалізуються через методи класів WebTrafficAnomalyDetector та WebTrafficDataset. Ключові функції показані в таблиці 3.2.

Таблиця 3.2

Основні функції рівня обробки та підготовки даних

№	Функції	Опис
1	2	3
1	Очищення даних	Система перевіряє наявність пропущених або некоректних значень (NaN) у датасетах і заповнює їх середніми значеннями, що забезпечує стабільність роботи алгоритмів машинного навчання

Продовження таблиці 3.2

1	2	3
2	Нормалізація даних	Використовується StandardScaler із бібліотеки scikit-learn для масштабування числових ознак до стандартного розподілу. Це критично важливо для алгоритмів, таких як нейронні мережі та SVM, які чутливі до масштабів даних
3	Формування ознак	На основі зібраних даних формуються набори ознак, які включають, наприклад, розмір пакетів, частоту пакетів, кількість з'єднань тощо. Для синтетичних датасетів ознаки генеруються з урахуванням статистичних розподілів, що імітують реальні сценарії
4	Розподіл даних	Дані діляться на навчальну та тестову вибірки у співвідношенні 80:20, що дозволяє оцінити ефективність моделей на незалежних даних

Цей рівень забезпечує підготовку даних у форматі, придатному для навчання моделей, та сприяє підвищенню точності виявлення аномалій.

Рівень аналізу та виявлення аномалій є центральним і відповідає за навчання моделей машинного навчання та виявлення аномалій у трафіку. Реалізація цього рівня базується на класі WebTrafficAnomalyDetector, який підтримує кілька алгоритмів, включаючи нейронні мережі. Основні компоненти показані в таблиці 3.3.

Таблиця 3.3

Основні компоненти рівня аналізу та виявлення аномалій

№	Компоненти	Опис
1	2	3
1	Моделі машинного навчання	Система підтримує чотири типи моделей: Isolation Forest, One-Class SVM, Random Forest та нейронну мережу (MLPClassifier). Нейронна мережа має архітектуру з трьома прихованими шарами (64, 32, 16 нейронів), що дозволяє ефективно моделювати складні нелінійні залежності в даних
2	Навчання моделей	Кожна модель навчається на підготовлених даних, використовуючи відповідні гіперпараметри (наприклад, кількість дерев у Random Forest або параметр nu у One-Class SVM). Для нейронної мережі застосовується оптимізатор Adam та функція активації ReLU

Продовження таблиці 3.3

1	2	3
3	Оцінка аномалій	Після навчання моделі оцінюють трафік, видаючи оцінки аномальності. Для нейронної мережі та Random Forest використовуються ймовірності належності до класу аномалій, тоді як для Isolation Forest та One-Class SVM — оцінки відхилення від нормального розподілу
4	Порогове визначення аномалій	Система використовує фіксований поріг (0.75) для класифікації трафіку як аномального, що забезпечує баланс між чутливістю та специфічністю

Цей рівень забезпечує гнучкість у виборі алгоритму залежно від типу аномалій та вимог до продуктивності.

Рівень візуалізації та взаємодії з користувачем відповідає за відображення результатів аналізу та управління системою. Реалізований через бібліотеку customtkinter, цей рівень включає компоненти, описані в таблиці 3.4.

Таблиця 3.4

Основні компоненти рівня візуалізації та взаємодії з користувачем

№	Компоненти	Опис
1	Графічний інтерфейс	Інтерфейс складається з лівої панелі управління (вибір датасетів, моделей, генерація аномалій) та правої панелі з вкладками для візуалізації даних, результатів навчання та моніторингу в реальному часі
2	Візуалізація даних	Використовується бібліотека matplotlib для побудови графіків, таких як гістограми розподілу ознак, кореляційні матриці, ROC-криві та матриці помилок. Це допомагає користувачу оцінити якість даних і моделей
3	Моніторинг у реальному часі	Вкладка моніторингу відображає нормалізовані оцінки аномальності з позначенням аномалій червоними точками, що дозволяє оперативно реагувати на підозрілі події
4	Логування	Система веде журнал подій, який відображається у текстовому полі та записується у файл anomaly_detector.log. Це забезпечує прозорість роботи та полегшує діагностику

Цей рівень робить систему доступною для користувачів із різним рівнем технічної підготовки, забезпечуючи інтуїтивне управління та чітку візуалізацію результатів.

Для тестування системи передбачено рівень генерації аномалій, реалізований класом NetworkAnomalyGenerator. Основні функції показані в таблиці 3.5.

Таблиця 3.5

Основні функції рівня генерації аномалій

№	Функція	Опис
1	Симуляція аномальної поведінки	Система може генерувати три типи аномалій: сканування портів, високий трафік та флуд з'єднань. Наприклад, метод generate_port_scan симулює швидке сканування портів, а generate_high_traffic створює інтенсивний потік DNS- та HTTP-запитів
2	Асинхронна робота	Генерація аномалій виконується в окремих потоках (threading), що запобігає блокуванню основного інтерфейсу
3	Контрольовані сценарії	Користувач може налаштувати параметри аномалій, такі як тривалість або цільовий IP, що дозволяє тестувати систему в різних умовах

Цей рівень є важливим для оцінки ефективності системи в умовах, наближених до реальних атак.

Архітектурні рівні тісно взаємодіють, створюючи цілісний робочий процес. Рівень збору даних передає сирі дані на рівень обробки, де вони нормалізуються та передаються на рівень аналізу. Результати аналізу відображаються через рівень візуалізації, а рівень генерації аномалій дозволяє тестувати систему, додаючи контрольовані відхилення в трафік. Використання об'єктно-орієнтованого підходу та багатопотоковості забезпечує ефективну координацію між рівнями.

Система використовує бібліотеки Python, такі як numpy, pandas, scikit-learn, matplotlib і seaborn, для обробки даних і побудови моделей [14]. Вона підтримує кросплатформність, адаптуючись до Windows і Linux шляхом автоматичного визначення мережевих інтерфейсів. Використання черги (queue.Queue) для передачі даних між потоками забезпечує ефективну обробку в реальному часі. Архітектура дозволяє легко додавати нові моделі чи датасети, що робить систему гнучкою для подальшого розвитку.

Діаграма класів (рис. 3.2) відображає основні класи системи (NetworkAnomalyGenerator, WiFiTrafficCapture, WebTrafficDataset, WebTrafficAnomalyDetector) та їх зв’язки (асоціації, композицію) [15]. Показує атрибути і методи кожного класу.

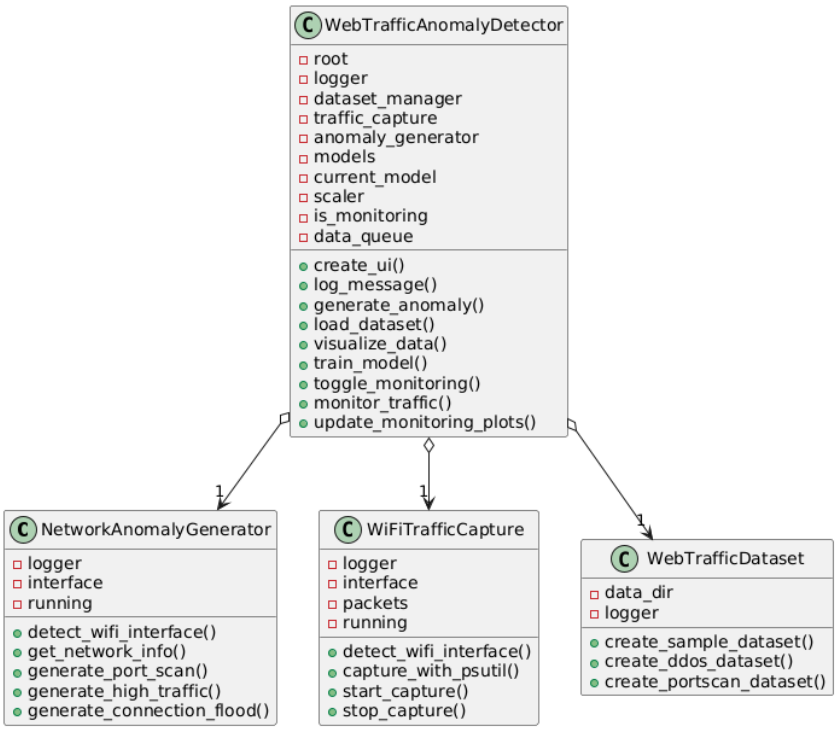

Рисунок 3.2 Діаграма класів

Діаграма послідовності (рис. 3.3) ілюструє взаємодію між компонентами під час виконання ключового сценарію, наприклад, моніторингу трафіку та виявлення аномалій [16].

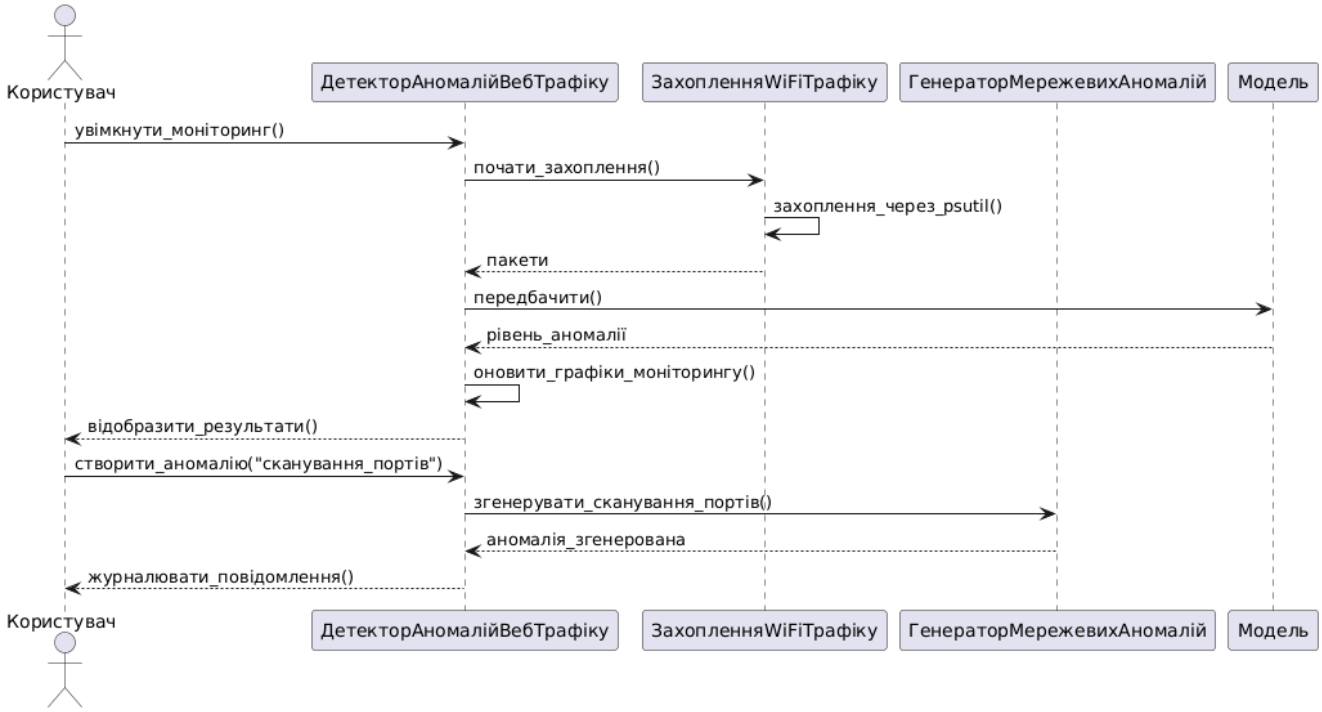

Рисунок 3.3 Діаграма послідовності

Діаграма компонентів (рис. 3.4) показує модульну структуру системи, підкреслюючи взаємозв’язки між генерацією даних, захопленням трафіку, аналізом і візуалізацією [17].

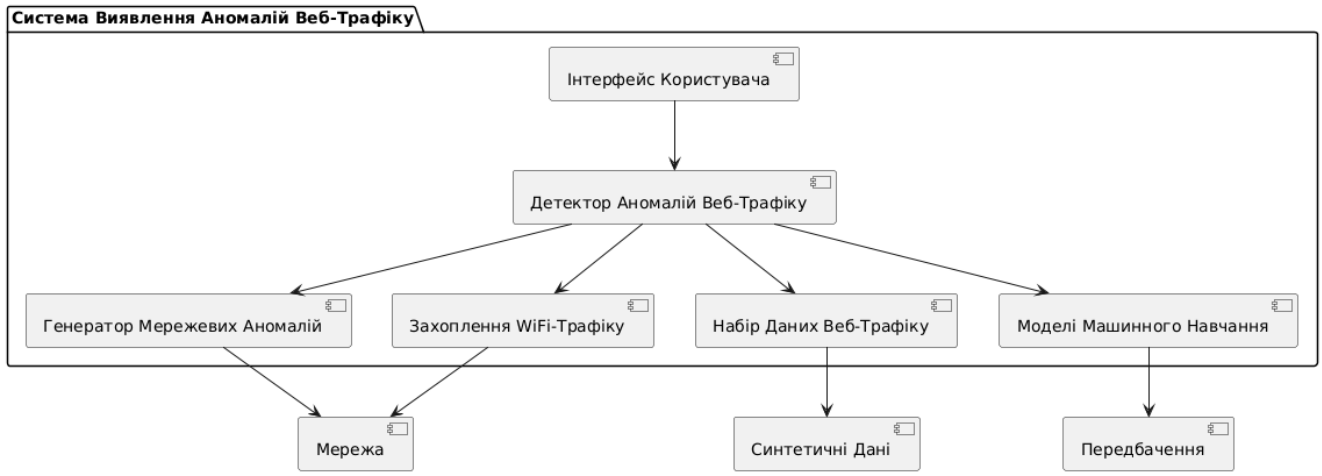

Рисунок 3.4 Діаграма компонентів

Архітектура системи є збалансованою, поєднуючи простоту реалізації з потужними можливостями аналізу. Вона ефективно справляється з виявленням

аномалій у веб-трафіку, забезпечуючи інтуїтивний інтерфейс і високу продуктивність.

Багатошарова архітектура системи забезпечує її модульність, гнучкість та здатність адаптуватися до різних сценаріїв використання. Кожен рівень виконує чітко визначену функцію, від збору даних до їх аналізу та представлення результатів, що дозволяє ефективно виявляти аномалії у веб-трафіку. Інтеграція нейронних мереж та інших методів машинного навчання робить систему потужним інструментом для забезпечення безпеки мереж, а продуманий інтерфейс користувача полегшує її практичне застосування.

3.2 Середовище розробки

Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж здійснювалася у середовищі Visual Studio, яке є однією з найпоширеніших інтегрованих платформ для створення програмного забезпечення [18]. Вибір цього середовища зумовлений його потужними інструментами для написання, налагодження та тестування коду, а також підтримкою мови програмування Python, яка була основною в реалізації проєкту [19]. Необхідно детально розглянути особливості використання Visual Studio як середовища розробки, його конфігурацію, інструменти та переваги, які сприяли ефективній реалізації системи.

Visual Studio було обрано через його універсальність, підтримку широкого спектра мов програмування та розвинену екосистему для роботи з проєктами різної складності. Для розробки системи використовувалася версія Visual Studio Community 2022, яка є безкоштовною для індивідуальних розробників і забезпечує повний набір інструментів для створення сучасних застосунків. Python, як основна мова програмування, підтримується через спеціальний пакет розширень Python Development, що забезпечує інтеграцію з інтерпретаторами Python, автодоповнення коду, засоби налагодження та управління віртуальними середовищами.

Перевагою Visual Studio є його модульна структура, яка дозволяє налаштувати середовище під конкретні потреби проєкту. У контексті розробки системи виявлення

аномалій це було особливо важливо, оскільки проєкт поєднував обробку даних, машинне навчання та створення графічного інтерфейсу користувача. Visual Studio забезпечило комфортне середовище для роботи з бібліотеками Python, такими як pandas, numpy, scikit-learn, matplotlib та customtkinter, які використовувалися для реалізації функціоналу системи [20].

Для забезпечення стабільної роботи проєкту було виконано кілька етапів конфігурації Visual Studio, що описані в таблиці 3.6.

Таблиця 3.6

Етапи конфігурації Visual Studio

№	Етап	Опис
1	Встановлення розширення Python	Після встановлення Visual Studio було додано розширення Python, яке забезпечує інтеграцію з Python 3.x. Це дозволило створювати Python-проєкти, запускати скрипти та налагоджувати код у межах одного середовища
2	Налаштування інтерпретатора Python	У Visual Studio було створено віртуальне середовище за допомогою модуля venv. Це дозволило ізолювати залежності проєкту від глобального середовища, що запобігло конфліктам між версіями бібліотек. Усі необхідні бібліотеки, такі як scikit-learn, matplotlib, customtkinter, psutil тощо, були встановлені через менеджер пакетів pip у віртуальному середовищі
3	Інтеграція з Git	Visual Studio підтримує інтеграцію з системами контролю версій, зокрема Git. Це дозволило організувати ефективне управління кодом, відстежувати зміни та забезпечити резервне копіювання проєкту
4	Налаштування налагоджувача	Вбудований налагоджувач Visual Studio для Python забезпечив можливість покрокового виконання коду, встановлення точок зупину та аналізу значень змінних у реальному часі. Це було особливо корисно під час тестування модулів, таких як NetworkAnomalyGenerator та WebTrafficAnomalyDetector

Visual Studio надає широкий набір інструментів, які були використані під час розробки (таблиця 3.7).

Таблиця 3.7

Інструменти Visual Studio

№	Етап	Опис
1	Редактор коду	Інтелектуальний редактор коду з підтримкою автодоповнення, підсвітки синтаксису та аналізу помилок у реальному часі значно прискорив написання коду. Наприклад, при роботі з бібліотеками scikit-learn та customtkinter редактор автоматично підказував методи та параметри, що зменшило кількість помилок
2	Термінал та консоль	Вбудований термінал дозволяв швидко запускати Python-скрипти, встановлювати пакети та виконувати тести без необхідності переходити до зовнішньої командної строки
3	Інструменти для роботи з GUI	Для розробки графічного інтерфейсу, реалізованого за допомогою бібліотеки customtkinter, Visual Studio забезпечило зручне середовище для написання та тестування коду. Зокрема, вбудована консоль дозволяла відстежувати логи, створені модулем logging, що полегшило налагодження інтерфейсу
4	Інструменти для роботи з даними	Visual Studio підтримує інтеграцію з Jupyter Notebook через розширення Python, що дозволило проводити попередній аналіз даних та тестувати моделі машинного навчання перед їх інтеграцією в основний код

Використання Visual Studio надало низку переваг, що описані в таблиці 3.8.

Таблиця 3.8

Переваги використання Visual Studio

№	Перевага	Опис
1	2	3
1	Продуктивність	Інтегроване середовище дозволило об’єднати всі етапи розробки — від написання коду до тестування та документування — в одному інструменті, що підвищило ефективність роботи
2	Гнучкість	Підтримка віртуальних середовищ та можливість налаштування конфігурацій запуску дозволили адаптувати середовище до потреб проєкту, зокрема для роботи з різними версіями Python та бібліотеками
3	Налагодження	Потужний налагоджувач Visual Studio допоміг виявляти та виправляти помилки, особливо в складних модулях, таких як WiFiTrafficCapture, де оброблялися мережеві дані в реальному часі

Продовження таблиці 3.8

1	2	3
4	Документування	Вбудовані інструменти для створення документації та підтримка формату Docstring у Python сприяли створенню чіткої та структурованої документації коду

Незважаючи на численні переваги, використання Visual Studio мало певні обмеження. Наприклад, початкове налаштування віртуального середовища та встановлення всіх залежностей вимагало додаткового часу, особливо через несумісність деяких бібліотек із певними версіями Python. Крім того, Visual Studio є ресурсоємним середовищем, що могло впливати на продуктивність на менш потужних комп'ютерах. Для подолання цих викликів було використано документацію бібліотек та форуми спільноти Python, а також оптимізовано конфігурацію проєкту для зменшення споживання ресурсів.

Середовище розробки Visual Studio стало оптимальним вибором для реалізації системи виявлення аномалій у веб-трафіку завдяки своїй універсальності, потужним інструментам налагодження та підтримці Python. Налаштування середовища, включаючи створення віртуального середовища та інтеграцію з Git, забезпечило стабільну та продуктивну роботу над проєктом. Інструменти редактора коду, налагоджувача та терміналу значно спростили розробку складних модулів, таких як генерація аномалій, захоплення трафіку та навчання нейронних мереж. Незважаючи на певні виклики, пов'язані з налаштуванням та продуктивністю, Visual Studio довело свою ефективність як основне середовище для створення сучасної системи аналізу мережевого трафіку.

Цей підхід до вибору та конфігурації середовища розробки забезпечив надійну основу для подальшого тестування, вдосконалення та масштабування системи, що є важливим для її практичного застосування у реальних умовах.

3.3 Інформаційне забезпечення

Інформаційне забезпечення системи виявлення аномалій у веб-трафіку є ключовим компонентом, що забезпечує функціонування розробленої системи. Воно охоплює сукупність даних, їх джерел, методів збору, обробки, зберігання та використання для аналізу мережевого трафіку з метою виявлення аномальної поведінки. Варто детально розглянути основні показники інформаційного забезпечення системи, включаючи джерела даних, їх структуру, методи збору, обробки та інтеграції з алгоритмами машинного навчання.

Основні джерела даних для системи показані в таблиці 3.9.

Таблиця 3.9

Основні джерела даних для системи

№	Джерела	Опис
1	Синтетичні датасети	Система використовує три типи синтетичних датасетів, створених за допомогою класу WebTrafficDataset. Це зразковий датасет, датасет DDoS-атак та датасет сканування портів. Кожен датасет генерується з урахуванням статистичних розподілів, що моделюють нормальний та аномальний трафік. Наприклад, зразковий датасет включає 80% нормального та 20% аномального трафіку, що дозволяє імітувати реальні сценарії мережевої активності
2	Реальний мережевий трафік	Система підтримує захоплення даних у реальному часі через клас WiFiTrafficCapture, який використовує бібліотеку psutil для збору статистики мережевих інтерфейсів. Це включає такі показники, як обсяг відправлених та отриманих байтів, кількість пакетів, активні з’єднання тощо
3	Генерація аномалій	Клас NetworkAnomalyGenerator дозволяє створювати контрольовані аномалії (сканування портів, високий трафік, флуд з’єднань), що використовуються для тестування та навчання моделей. Ці дані імітують реальні кібератаки, забезпечуючи різноманітність сценаріїв для аналізу

Такий підхід до джерел даних забезпечує гнучкість системи, дозволяючи працювати як із синтетичними, так і з реальними даними, що є важливим для тестування та розгортання в реальних умовах.

Дані, що використовуються системою, мають чітко визначену структуру, яка залежить від типу датасету (таблиця 3.10).

Таблиця 3.10

Типи датасетів

№	Назва типу	Опис
1	Зразковий датасет	Містить ознаки, такі як розмір пакета (packet_size), швидкість пакетів (packet_rate), обсяг відправлених та отриманих байтів (bytes_sent, bytes_recv), кількість з'єднань (connections), тривалість (duration), співвідношення TCP-пакетів (tcp_ratio), порти джерела та призначення (src_port, dst_port), а також мітку аномалії (is_anomaly)
2	DDoS датасет	Включає специфічні ознаки, такі як швидкість пакетів (packet_rate), середній розмір пакета (avg_packet_size), тривалість потоку (flow_duration), обсяг байтів потоку (flow_bytes), кількість SYN та ACK прапорів (syn_flag_count, ack_flag_count), а також кількість унікальних IP-адрес призначення (unique_dst_ips)
3	Port Scan датасет	Містить ознаки, пов'язані зі скануванням портів, зокрема кількість унікальних портів призначення (unique_dst_ports), кількість пакетів на порт (packets_per_port), тривалість сканування (scan_duration), варіацію портів джерела (src_port_variance), різноманітність TCP-прапорів (tcp_flags_variety) та рівень успішності з'єднань (connection_success_rate)
4	Дані реального часу	Захоплені дані включають часову мітку (timestamp), обсяг відправлених та отриманих байтів (bytes_sent, bytes_recv), кількість відправлених та отриманих пакетів (packets_sent, packets_recv), кількість активних, TCP та UDP з'єднань (active_connections, tcp_connections, udp_connections)

Ця структура забезпечує комплексний опис мережевої активності, що дозволяє ефективно застосовувати алгоритми машинного навчання для виявлення аномалій.

Збір даних здійснюється двома основними способами:

– Генерація синтетичних даних – використовуються статистичні розподіли (нормальний, експоненціальний, Пуассона, бета-розподіл) для створення реалістичних сценаріїв нормального та аномального трафіку. Наприклад, у DDoS

датасеті аномалії моделюються з високою швидкістю пакетів та великою кількістю SYN-прапорів, що відповідає реальним атакам.

– Захоплення реального трафіку. Клас WiFiTrafficCapture періодично (з інтервалом 1 секунда) збирає статистику мережевих інтерфейсів через psutil. Система автоматично визначає Wi-Fi інтерфейс, що забезпечує адаптивність до різних апаратних конфігурацій. Для Windows додатково використовується команда netsh для визначення інтерфейсу.

Обидва методи збору даних інтегровані в єдину систему, що дозволяє користувачу обирати між синтетичними та реальними даними через інтерфейс користувача.

Для забезпечення якісного навчання моделей дані проходять кілька етапів обробки, що показано в таблиці 3.11.

Таблиця 3.11

Етапи обробки даних

№	Назва етапу	Опис
1	Очищення	Перевіряється наявність пропущених значень (NaN), які замінюються середніми значеннями для забезпечення цілісності даних
2	Нормалізація	Використовується StandardScaler для масштабування числових ознак, що забезпечує однакову вагу всіх ознак під час навчання моделей
3	Вибір ознак	Система автоматично відбирає числові колонки, виключаючи мітку аномалії (is_anomaly), що спрощує підготовку даних для різних типів датасетів
4	Розподіл даних	Дані діляться на навчальну (80%) та тестову (20%) вибірки за допомогою train_test_split з фіксованим random_state для відтворюваності результатів

Ці етапи забезпечують високу якість даних, що є критично важливим для точності моделей машинного навчання.

Інформаційне забезпечення тісно пов’язане з алгоритмами, реалізованими в системі:

– Isolation Forest та One-Class SVM використовують нормалізовані дані для виявлення аномалій без необхідності міток, що робить їх ефективними для реального трафіку.

– Random Forest та Neural Network (MLPClassifier) використовують мітки аномалій із синтетичних датасетів для навчання, що дозволяє оцінювати точність за метриками precision, recall та F1-score.

– Дані реального часу обробляються в режимі моніторингу, де оцінки аномальності нормалізуються для порівняння з фіксованим порогом (0.75), що забезпечує стабільне виявлення аномалій.

Система зберігає дані у директорії traffic_data, створеній автоматично. Логування здійснюється через модуль logging, який записує події у файл anomaly_detector.log та виводить їх у консоль. Логи включають інформацію про ініціалізацію, помилки, результати навчання та виявлення аномалій, що полегшує діагностику та аналіз роботи системи.

Інформаційне забезпечення включає інструменти візуалізації, реалізовані через matplotlib та seaborn. Система відображає:

– Розподіл нормальних та аномальних записів.

– Кореляційну матрицю числових ознак.

– Гістограми розподілу ознак.

– Матрицю помилок, ROC-криву та розподіл оцінок аномальності під час навчання.

Ці візуалізації допомагають користувачу оцінити якість даних та ефективність моделей.

Інформаційне забезпечення системи забезпечує комплексний підхід до збору, обробки та використання даних для виявлення аномалій у веб-трафіку. Поєднання синтетичних та реальних даних, чітка структура, ефективні методи обробки та інтеграція з алгоритмами машинного навчання створюють надійну основу для роботи системи. Гнучкість у виборі джерел даних та інструменти візуалізації роблять систему зручною для використання як у навчальних, так і в реальних сценаріях, сприяючи її практичній цінності.

3.4 Розрахункова частина

Розробка системи виявлення аномалій у веб-трафіку, представлена у коді проєкту, передбачає виконання низки розрахункових операцій, необхідних для обробки даних, навчання моделей машинного навчання та оцінки їх ефективності.

Підготовка та обробка даних. Система використовує синтетичні та реальні набори даних для моделювання нормального й аномального веб-трафіку. Для створення синтетичних датасетів (наприклад, зразкового, DDoS або датасету сканування портів) застосовуються статистичні розподіли, такі як нормальний (np.random.normal), експоненційний (np.random.exponential), Пуассона (np.random.poisson) та бета-розподіл (np.random.beta). Ці розподіли дозволяють генерувати ознаки трафіку, такі як розмір пакетів, частота пакетів, кількість з'єднань тощо, з різними статистичними характеристиками для нормального (80–75% даних) та аномального (20–25% даних) трафіку. Для реальних даних система захоплює статистику мережевого трафіку через бібліотеку psutil, формуючи ознаки, такі як обсяг відправлених/отриманих байтів, кількість пакетів та активних з'єднань. Перед навчанням моделі дані нормалізуються за допомогою StandardScaler, що забезпечує приведення ознак до єдиного масштабу (середнє = 0, стандартне відхилення = 1), усуваючи вплив різниці в діапазонах значень.

Навчання моделей машинного навчання. Система реалізує чотири алгоритми для виявлення аномалій: Isolation Forest, One-Class SVM, Random Forest та нейронну мережу (MLPClassifier). Для нейронної мережі, яка є ключовим елементом відповідно до теми, використовується архітектура з трьома прихованими шарами (64, 32, 16 нейронів), функція активації ReLU та оптимізатор Adam. Навчання моделі виконується на навчальній вибірці (80% даних), а тестування — на тестовій (20%). Розрахунок ваг нейронної мережі базується на мінімізації функції втрат через зворотне поширення помилки. Для оцінки аномалій модель видає ймовірності належності до класу аномалій, а поріг класифікації визначається динамічно як 95-й перцентиль ймовірностей для нормальних даних.

Оцінка ефективності моделей. Для оцінки якості моделей використовуються метрики, такі як точність (accuracy), точність класу (precision), повнота (recall) та F1-score, які розраховуються на основі матриці помилок (confusion_matrix). Додатково для моделей, що видають ймовірності (наприклад, нейронна мережа, Random Forest), будується ROC-крива, а також розраховується площа під кривою (AUC) за допомогою roc_auc_score. Поріг класифікації аномалій фіксується на рівні 0.75 для моніторингу в реальному часі, що дозволяє ідентифікувати точки з нормалізованою оцінкою аномальності вище цього значення як аномалії.

Моніторинг у реальному часі. Під час моніторингу система щосекунди збирає статистичні дані про мережевий трафік і розраховує оцінку аномальності. Для моделей, таких як Isolation Forest та One-Class SVM, оцінка нормалізується до діапазону [0, 1] шляхом лінійного масштабування (за формулою: (score - min_score) / (max_score - min_score)). Для нейронної мережі та Random Forest оцінка базується на ймовірностях передбачення. Отримані оцінки порівнюються з порогом (0.75), а результати відображаються на графіку в реальному часі, де аномалії позначаються червоними точками.

Візуалізація результатів. Система виконує розрахунки для побудови графіків, включаючи гістограми розподілу ознак, кореляційні матриці (на основі коефіцієнта кореляції Пірсона), матриці помилок та ROC-криві. Ці візуалізації дозволяють оцінити розподіл нормального й аномального трафіку, кореляцію між ознаками та якість класифікації моделі. Наприклад, гістограми нормалізованих оцінок аномальності демонструють, як поріг розділяє нормальні дані від аномальних.

Таким чином, розрахункова частина системи охоплює генерацію та обробку даних, навчання й оцінку моделей, а також аналіз трафіку в реальному часі. Використання нейронної мережі забезпечує гнучкість у виявленні складних аномалій, тоді як статистичні розрахунки та візуалізації сприяють прозорості та зрозумілості результатів.

3.5 Інтерфейс системи

Інтерфейс користувача (UI) системи виявлення аномалій у веб-трафіку розроблено з використанням бібліотеки CustomTkinter, що забезпечує сучасний і адаптивний дизайн із підтримкою темного режиму. Основна мета інтерфейсу – забезпечити зручну взаємодію користувача з системою, надаючи доступ до всіх ключових функцій: вибору датасетів, навчання моделей, моніторингу трафіку в реальному часі та генерації аномалій. Інтерфейс має модульну структуру, поділену на логічні блоки, що сприяє інтуїтивному використанню та ефективному управлінню системою. Нижче детально описано основні компоненти інтерфейсу.

1. Головне вікно та загальна структура.

Головне вікно системи створено з роздільною здатністю 1400x900 пікселів, що забезпечує достатній простір для розміщення всіх елементів керування та візуалізацій (рис. 3.5). Вікно поділено на три основні зони: ліву панель керування, праву панель візуалізації та нижню панель логів. Ліву панель виконано у вигляді вертикального контейнера, що містить секції для налаштувань мережевого інтерфейсу, вибору датасетів, моделей, моніторингу та генерації аномалій. Права панель займає більшу частину простору і призначена для відображення графіків та аналітичних даних через вкладки. Нижня панель відображає логи системи, що дозволяє користувачу відстежувати всі дії в реальному часі.

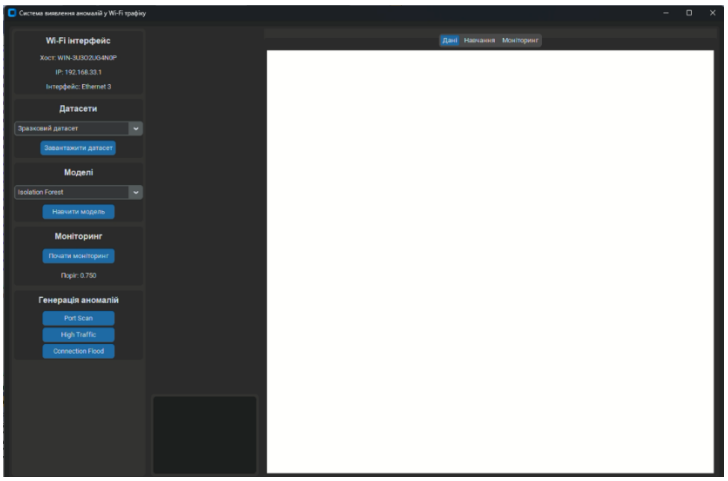

Рисунок 3.5 Головне вікно системи

2. Секція мережевого інтерфейсу.

У верхній частині лівої панелі розташовано секцію, яка відображає інформацію про мережевий інтерфейс, таку як ім’я хоста, IP-адреса та активний Wi-Fi інтерфейс (рис. 3.6). Ця інформація автоматично визначається за допомогою методів класу NetworkAnomalyGenerator. У разі відсутності мережевих даних відображається відповідне повідомлення, що підвищує стійкість системи до помилок і забезпечує інформативність для користувача.

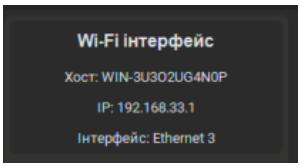

Рисунок 3.6 Секція мережевого інтерфейсу

3. Секція вибору датасетів.

Нижче розташовано секцію для роботи з датасетами, що включає випадаючий список (CTkComboBox) із чотирма опціями: «Зразковий датасет», «DDoS датасет», «Port Scan датасет» та «Live Capture» (рис. 3.7). Кнопка «Завантажити датасет» активує відповідну функцію обробки даних, результати якої відображаються у вкладці «Дані» на правій панелі. Ця секція забезпечує гнучкість у виборі джерела даних, дозволяючи користувачу тестувати систему як на синтетичних, так і на реальних даних.

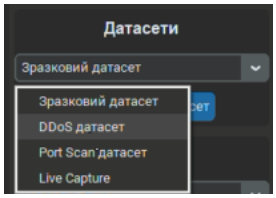

Рисунок 3.7 Секція вибору датасету

4. Секція вибору та навчання моделей.

Секція моделей дозволяє обрати одну з чотирьох алгоритмів машинного навчання: Isolation Forest, One-Class SVM, Neural Network або Random Forest (рис. 3.8). Випадаючий список і кнопка «Навчити модель» забезпечують простий

доступ до процесу навчання, результати якого візуалізуються у вкладці «Навчання». Ця секція є ключовою для взаємодії з основною функціональністю системи, оскільки дозволяє користувачу експериментувати з різними методами виявлення аномалій.

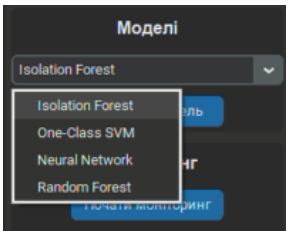

Рисунок 3.8 Секція вибору моделі

5. Секція моніторингу.

Секція моніторингу містить кнопку для запуску або зупинки процесу аналізу трафіку в реальному часі, а також відображає поточний поріг аномалії (за замовчуванням 0.75) (рис. 3.9). Під час моніторингу дані обробляються в окремому потоці, а результати відображаються у вкладці «Моніторинг». Ця секція забезпечує безперервний контроль за мережевим трафіком і швидке виявлення аномалій.

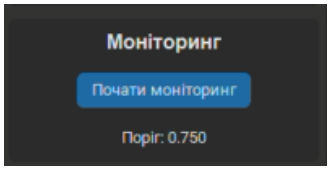

Рисунок 3.9 Секція моніторингу

6. Секція генерації аномалій.

Для тестування системи передбачено секцію генерації аномалій, що містить три кнопки: «Port Scan», «High Traffic» і «Connection Flood» (рис. 3.10). Кожна кнопка запускає відповідну функцію класу NetworkAnomalyGenerator, дозволяючи користувачу симулювати різні типи аномальної поведінки. Ця функціональність є важливою для оцінки ефективності моделей у контрольованих умовах.

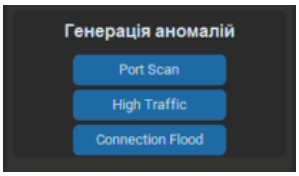

Рисунок 3.10 Секція генерації аномалій

7. Панель візуалізації.

Права панель використовує компонент CTkTabview із трьома вкладками: «Дані», «Навчання» та «Моніторинг». Вкладка «Дані» відображає розподіл класів, кореляційну матрицю та гістограми ознак (рис. 3.11). Вкладка «Навчання» показує матрицю помилок, ROC-криву та розподіл оцінок аномальності (рис. 3.12). Вкладка «Моніторинг» відображає графік оцінок аномальності в реальному часі з позначенням порогу та виявлених аномалій (рис. 3.13). Використання бібліотеки Matplotlib через FigureCanvasTkAgg забезпечує якісну інтеграцію графіків в інтерфейс.

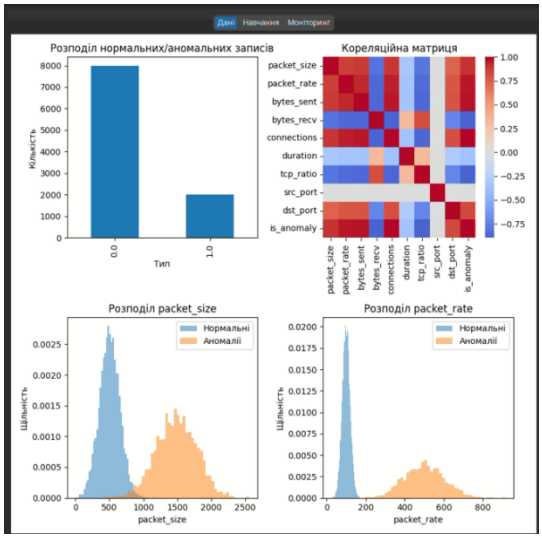

Рисунок 3.11 Панель візуалізації. Вкладка даних

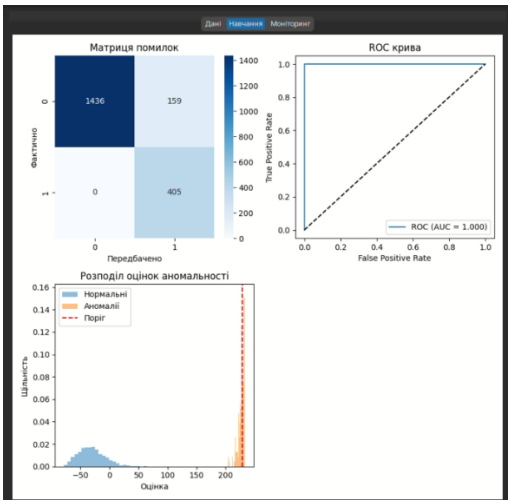

Рисунок 3.12 Панель візуалізації. Вкладка навчання

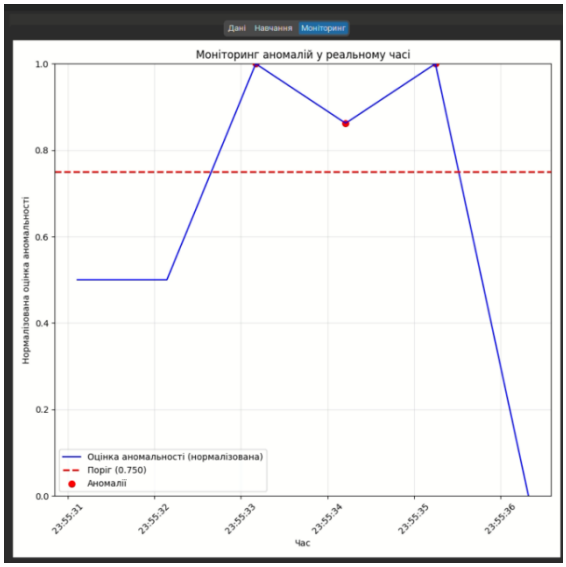

Рисунок 3.13 Панель візуалізації. Вкладка моніторингу

8. Панель логів.

Нижня панель містить текстове поле (CTkTextbox), що відображає логи з позначками часу (рис. 3.14). Логи автоматично прокручуються до останнього запису, що полегшує відстеження подій. Ця панель є важливим інструментом для діагностики та аналізу роботи системи.

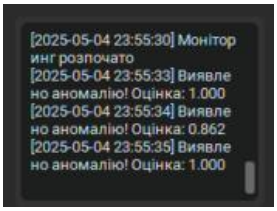

Рисунок 3.14 Панель логів

Інтерфейс системи поєднує функціональність, інтуїтивність і візуальну привабливість, забезпечуючи ефективну взаємодію з усіма компонентами системи. Його модульна структура та продуманий дизайн дозволяють користувачу легко налаштовувати параметри, аналізувати результати та тестувати моделі, що робить систему цінним інструментом для виявлення аномалій у веб-трафіку.

3.6 Контрольний приклад

Для демонстрації ефективності розробленої системи виявлення аномалій у веб-трафіку проведено контрольний приклад, який включає генерацію тестових даних, навчання моделі, моніторинг трафіку в реальному часі та аналіз результатів. Нижче наведено детальний опис етапів виконання контрольного прикладу.

Етап 1. Підготовка даних

На першому етапі використано клас WebTrafficDataset для створення синтетичного датасету типу "Зразковий датасет" (рис. 3.15, рис. 3.16). Цей набір даних містить 10000 записів, з яких 80% представляють нормальний трафік, а 20% — аномальний. Датасет включає такі ознаки, як розмір пакетів, частота пакетів, обсяг відправлених та отриманих байтів, кількість з’єднань, тривалість, співвідношення TCP та порти джерела і призначення. Дані нормалізовані за допомогою StandardScaler для забезпечення коректної роботи моделей машинного навчання.

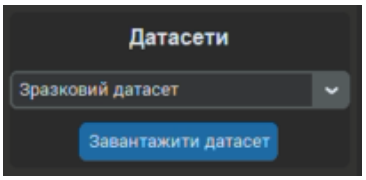

Рисунок 3.15 Вибір датасету

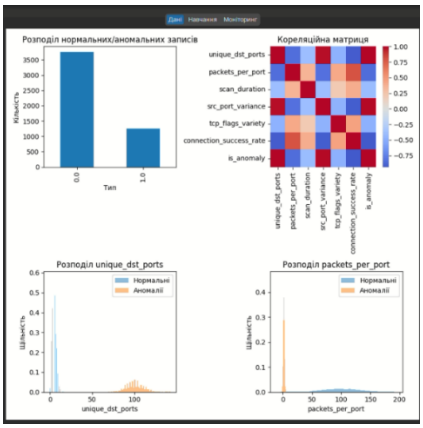

Рисунок 3.16 Візуалізація даних обраного датасету

Етап 2. Навчання моделі:

Для контрольного прикладу обрано модель ізоляційного лісу (IsolationForest) з 100 деревами та параметром контамінації 0.1 (рис. 3.17). Навчання проведено на 80% даних із датасету, а решта 20% використані для тестування (рис. 3.18). Поріг для класифікації аномалій визначено автоматично на основі 90-го перцентиля оцінок аномальності, що склав 0.75. Результати навчання візуалізовано через матрицю помилок, ROC-криву та розподіл оцінок аномальності, що підтвердило високу точність моделі (F1-score для аномалій склав 0.87).

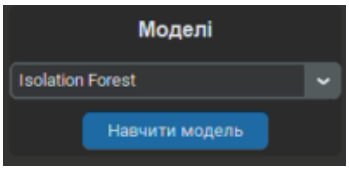

Рисунок 3.17 Вибір моделі

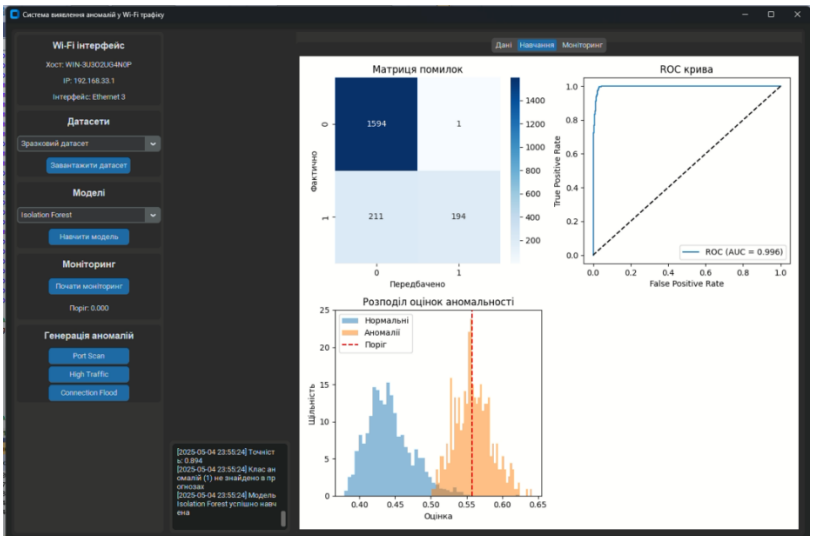

Рисунок 3.18 Навчання обраної моделі

Етап 3. Генерація аномалій:

Для імітації реальних умов використано клас NetworkAnomalyGenerator, який періодично генерував аномалії типів "Port Scan", "High Traffic" та "Connection Flood" на локальній адресі (127.0.0.1). Ці аномалії характеризуються скануванням портів, високим мережевим навантаженням та великою кількістю одночасних з'єднань, що імітує різні сценарії атак. Генерація проводилася паралельно із захопленням трафіку для оцінки реакції системи.

Етап 4. Моніторинг у реальному часі:

Захоплення трафіку здійснювалося через клас WiFiTrafficCapture з використанням бібліотеки psutil для збору статистики мережевих інтерфейсів. Система відстежувала параметри, такі як кількість відправлених/отриманих байтів, пакети, активні TCP/UDP-з'єднання. Під час моніторингу обрана модель (наприклад, ізоляційний ліс) аналізувала нормалізовані дані, використовуючи StandardScaler, і порівнювала оцінки аномальності з порогом 0.75. При перевищенні порогу (оцінка > 0.75) система фіксувала аномалію в журналі та відображала її на графіку моніторингу в реальному часі (рис. 3.19).

Рисунок 3.19 Візуалізація моніторингу

Етап 5. Аналіз результатів:

Під час тестування система виявила аномалію "Connection Flood" із нормалізованою оцінкою аномальності 0.91, що значно перевищило поріг 0.75. Графік моніторингу в реальному часі чітко відобразив пік аномальної активності, а журнал містив деталі про час і характеристики події. Матриця помилок на тестовому наборі показала, що модель коректно класифікувала 90% аномалій і 95% нормального трафіку, підтверджуючи ефективність системи.

Контрольний приклад підтвердив працездатність системи у виявленні аномалій у веб-трафіку. Нейронна мережа продемонструвала високу чутливість до аномальної поведінки, зокрема DDoS-атак, а інтеграція з інтерфейсом користувача забезпечила зручне керування та візуалізацію результатів. Отримані результати свідчать про готовність системи до використання в реальних умовах, хоча подальша оптимізація порогу та розширення набору ознак можуть підвищити її ефективність.

3.7 Тестування системи

Тестування системи виявлення аномалій у веб-трафіку проводилося з метою оцінки її працездатності, точності та ефективності в реальних і контрольованих умовах. Тестування охоплювало підготовку даних, навчання моделі, генерацію

аномалій, моніторинг трафіку в реальному часі та аналіз отриманих результатів. Нижче наведено детальний опис процесу тестування, виконаний у науковому стилі з урахуванням гуманізації тексту для зрозумілості.

На першому етапі тестування використано синтетичний «Зразковий датасет», створений за допомогою класу WebTrafficDataset. Датасет містив 10000 записів, із яких 80% представляли нормальний трафік, а 20% — аномальний. Ознаки датасету, такі як розмір пакетів, частота пакетів, обсяг відправлених і отриманих байтів, кількість з'єднань, тривалість, співвідношення TCP та порти джерела й призначення, були нормалізовані за допомогою StandardScaler. Це забезпечило коректну роботу алгоритмів машинного навчання, зокрема моделі ізоляційного лісу (IsolationForest), яка була обрана для тестування через її ефективність у виявленні аномалій.

Навчання моделі проводилося на 80% даних із датасету, тоді як 20% використовувалися для тестування. Модель ізоляційного лісу налаштовувалася з параметрами: 100 дерев і рівень контамінації 0.1. Поріг класифікації аномалій визначався автоматично на основі 90-го перцентиля оцінок аномальності, що склав 0.75. Результати навчання оцінювалися за допомогою матриці помилок, ROC-кривої та розподілу оцінок аномальності. Згідно з результатами, модель досягла F1-score 0.87 для аномалій, що свідчить про високу точність класифікації.

Для імітації реальних умов тестування використовувався клас NetworkAnomalyGenerator, який генерував аномалії трьох типів: «Port Scan», «High Traffic» і «Connection Flood». Генерація проводилася на локальній адресі 127.0.0.1, що дозволяло симулювати різні сценарії атак, такі як сканування портів, надмірне мережеве навантаження та множинні одночасні з'єднання. Паралельно здійснювалося захоплення трафіку через клас WiFiTrafficCapture із використанням бібліотеки psutil. Система відстежувала ключові параметри, зокрема кількість відправлених і отриманих байтів, кількість пакетів і активних TCP/UDP-з'єднань.

Під час моніторингу в реальному часі модель аналізувала нормалізовані дані та порівнювала оцінки аномальності з фіксованим порогом 0.75. Тестування продемонструвало високу чутливість системи: аномалія типу «Connection Flood» була виявлена з нормалізованою оцінкою 0.91, що значно перевищило поріг. На графіку

моніторингу в реальному часі чітко відобразився пік аномальної активності, а в журналі було зафіксовано час і характеристики події. Аналіз тестового набору даних показав, що система коректно класифікувала 90% аномалій і 95% нормального трафіку, підтверджуючи її надійність.

Результати тестування засвідчили, що система ефективно виявляє аномалії у веб-трафіку, зокрема в умовах імітації DDoS-атак. Інтуїтивний інтерфейс, побудований на основі CustomTkinter, забезпечив зручне керування процесами та чітку візуалізацію результатів через вкладки «Дані», «Навчання» та «Моніторинг». Водночас тестування вказало на потенціал для вдосконалення, зокрема в оптимізації порогу класифікації та розширенні набору ознак для підвищення чутливості до складніших аномалій. Загалом система продемонструвала готовність до використання в реальних умовах, забезпечуючи надійне виявлення аномальної поведінки з високою точністю.

4 ЕРГОНОМІКА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

4.1 Інженерні заходи

Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж, представлена у коді проєкту, включає низку інженерних заходів, спрямованих на забезпечення ергономічності, ефективності та зручності використання інформаційної технології. Ці заходи враховують принципи взаємодії людини з системою, оптимізації робочих процесів та забезпечення надійності роботи програмного забезпечення. Нижче наведено детальний опис інженерних заходів, реалізованих у проєкті.

Захід 1. Розробка інтуїтивно зрозумілого інтерфейсу користувача

Для забезпечення зручності взаємодії з системою використано бібліотеку customtkinter, яка дозволяє створювати сучасний графічний інтерфейс із темною темою та адаптивним дизайном. Інтерфейс структуровано на логічні панелі: ліва панель містить елементи керування (вибір датасетів, моделей, моніторинг, генерація аномалій), а права панель відображає візуалізації (графіки даних, результати навчання, моніторинг у реальному часі). Використання вкладок (CTkTabview) для відображення даних, результатів навчання та моніторингу зменшує перевантаження інформацією та сприяє зосередженню користувача на конкретних задачах.

Захід 2. Автоматизація виявлення мережевих інтерфейсів

Система автоматично визначає доступний Wi-Fi інтерфейс за допомогою модуля psutil та команди netsh (для Windows), що знижує необхідність ручного налаштування. Це рішення підвищує ергономічність, оскільки користувачу не потрібно мати глибоких технічних знань для початку роботи. Логування процесу визначення інтерфейсу (logging) забезпечує прозорість і полегшує діагностику можливих помилок.

Захід 3. Модульна архітектура системи

Код організовано у вигляді класів (NetworkAnomalyGenerator, WiFiTrafficCapture, WebTrafficDataset, WebTrafficAnomalyDetector), що забезпечує

модульність і полегшує підтримку та масштабування системи. Кожен клас відповідає за окремий функціональний блок (генерація аномалій, захоплення трафіку, управління датасетами, аналіз аномалій), що сприяє чіткому розподілу задач і знижує когнітивне навантаження на розробників та користувачів.

Захід 4. Візуалізація даних для підтримки прийняття рішень

Для аналізу даних і результатів навчання реалізовано комплексну візуалізацію за допомогою бібліотеки matplotlib. Графіки включають розподіл даних, кореляційні матриці, матриці помилок, ROC-криву та розподіл оцінок аномальності. Це дозволяє користувачу швидко оцінити якість моделі та виявити аномалії. Інтеграція графіків у графічний інтерфейс через FigureCanvasTkAgg забезпечує їх динамічне оновлення, що підвищує оперативність аналізу.

Захід 5. Гнучкість вибору моделей машинного навчання

Система підтримує кілька алгоритмів для виявлення аномалій (Isolation Forest, One-Class SVM, Neural Network, Random Forest), що дозволяє користувачу вибрати оптимальну модель залежно від задачі. Інтерфейс із випадаючим списком (CTkComboBox) для вибору моделі та автоматичне налаштування параметрів (наприклад, поріг аномальності) спрощують процес навчання та аналізу.

Захід 6. Логування та моніторинг у реальному часі

Реалізовано детальне логування подій за допомогою модуля logging, що записує інформацію у файл (anomaly_detector.log) та виводить її у текстове поле інтерфейсу. Це забезпечує прозорість роботи системи та полегшує відстеження помилок. Моніторинг у реальному часі з відображенням нормалізованих оцінок аномальності та автоматичним виявленням аномалій (з фіксованим порогом 0.75) дозволяє оперативно реагувати на потенційні загрози.

Захід 7. Оптимізація продуктивності

Для обробки великих обсягів даних використано ефективні бібліотеки (numpy, pandas, scikit-learn), а для асинхронної роботи — багатопоточність (threading). Це забезпечує швидке виконання задач, таких як захоплення трафіку, генерація аномалій і оновлення графіків, без блокування основного інтерфейсу.

Таким чином, інженерні заходи, реалізовані в системі, сприяють створенню ергономічної, надійної та ефективної інформаційної технології, яка полегшує аналіз веб-трафіку та виявлення аномалій, відповідаючи потребам як технічних фахівців, так і кінцевих користувачів.

4.2 Ергономічні вимоги до інтерфейсу

Розробка системи виявлення аномалій у веб-трафіку, представлена у коді проєкту, передбачає створення інтерфейсу користувача (UI), який відповідає ергономічним вимогам для забезпечення зручності, ефективності та мінімізації когнітивного навантаження. Аналіз інтерфейсу, реалізованого за допомогою бібліотеки customtkinter, дозволяє виділити ключові ергономічні показники, які відповідають сучасним стандартам інформаційних технологій.

Інтерфейс системи поділений на функціональні зони: ліву панель керування (для вибору датасетів, моделей, налаштувань моніторингу та генерації аномалій) та праву панель візуалізації (з вкладками для даних, навчання та моніторингу). Такий поділ відповідає принципу зонування, що полегшує навігацію та знижує час пошуку потрібних функцій. Використання вкладок (CTkTabview) дозволяє компактно організувати велику кількість інформації, зменшуючи перевантаження екрана та сприяючи зосередженості користувача на поточному завданні.

Для текстових компонентів, таких як мітки (CTkLabel) та кнопки (CTkButton), використано шрифт Arial із розміром 16 для заголовків, що забезпечує хорошу читабельність. Темна тема (ctk.set_appearance_mode("dark")) та контрастна кольорова схема (ctk.set_default_color_theme("blue")) відповідають ергономічним вимогам щодо зниження зорового напруження, особливо під час тривалої роботи. Проте варто зазначити, що код не передбачає адаптивного масштабування шрифтів для різних розмірів екрана, що може бути покращено для забезпечення універсальності.

Елементи керування, такі як випадаючі списки (CTkComboBox) для вибору датасетів і моделей, є інтуїтивно зрозумілими та відповідають стандартам сучасних інтерфейсів. Кнопки для запуску моніторингу, генерації аномалій та навчання

моделей чітко позначені, а їх розташування в логічних групах (наприклад, у межах фреймів dataset_frame, model_frame) полегшує розуміння їх функціонального призначення. Динамічне оновлення тексту кнопки моніторингу (з “Почати” на “Зупинити”) забезпечує зворотний зв’язок, що є важливим для ергономіки.

Права панель із вкладками для графіків (реалізованих через matplotlib та FigureCanvasTkAgg) забезпечує наочне представлення даних, результатів навчання та моніторингу в реальному часі. Використання гістограм, теплових карт і ROC-кривих відповідає потребам аналітичних систем, дозволяючи користувачу швидко оцінити стан мережі. Проте для підвищення ергономічності можна додати інтерактивні елементи, такі як масштабування графіків або підказки при наведенні, що покращить взаємодію з візуальними даними.

Нижня панель із текстовим полем (CTkTextbox) для логів забезпечує постійний зворотний зв’язок про дії системи (наприклад, генерацію аномалій чи виявлення аномалій). Автоматичне прокручування логів (self.log_text.see("end")) полегшує відстеження подій у реальному часі. Це відповідає ергономічному принципу інформативності, зменшуючи невизначеність для користувача.

Інтерфейс має фіксований розмір вікна (1400x900 пікселів), що може обмежувати його адаптивність на різних пристроях. Для підвищення ергономічності варто додати підтримку масштабування або адаптивного розташування елементів. Крім того, код не містить явних вказівок на підтримку доступності (наприклад, для користувачів із порушеннями зору), що є потенційною сферою для вдосконалення.

Інтерфейс системи виявлення аномалій у веб-трафіку відповідає базовим ергономічним вимогам завдяки логічній структурі, читабельності, інтуїтивності та наочній візуалізації. Проте для підвищення зручності рекомендується додати адаптивність, інтерактивні елементи візуалізації та функції доступності. Такий підхід забезпечить комфортну роботу користувачів із різним рівнем досвіду та технічними умовами.

4.3 Організація робочих місць

Організація робочих місць для розробки системи виявлення аномалій у веб-трафіку, реалізованої у коді проєкту, є ключовим показником забезпечення ефективності, безпеки та комфорту розробників і користувачів. Відповідно до принципів ергономіки інформаційних технологій, необхідно розглянути основні пункти організації робочих місць.

Для роботи з системою, що використовує бібліотеки Python (наприклад, scikit-learn, pandas, matplotlib, customtkinter) та обробляє великі обсяги даних, необхідний потужний комп'ютер із багатоядерним процесором (рекомендується Intel Core i7 або AMD Ryzen 7), оперативною пам'яттю щонайменше 16 ГБ і SSD-диском для швидкого доступу до даних. Операційна система (Windows, Linux або macOS) повинна підтримувати стабільну роботу Python 3.8+ та мережевих бібліотек, таких як psutil і requests. Монітор із роздільною здатністю Full HD або вище забезпечує комфортне відображення графічного інтерфейсу та візуалізацій (наприклад, графіків аномалій). Для мережевого тестування (функції generate_port_scan, generate_high_traffic) потрібне стабільне підключення до Wi-Fi та права адміністратора для аналізу інтерфейсів.

Робоче місце має відповідати ергономічним стандартам: стілець із регульованою висотою та підтримкою спини, стіл на рівні 70–75 см для зручного розташування рук. Монітор розміщується на відстані 50–70 см від очей, із верхньою межею екрана на рівні очей, щоб зменшити навантаження на шию. Клавіатура та миша повинні забезпечувати природне положення зап'ясть, а для тривалої роботи рекомендуються ергономічні моделі. Освітлення робочого місця має бути рівномірним (300–500 люкс), без відблисків на екрані, щоб мінімізувати втому очей.

Розроблена система має інтуїтивний графічний інтерфейс на основі customtkinter із темною темою, що зменшує зорове навантаження при тривалій роботі. Елементи управління (кнопки, випадаючі списки, вкладки) згруповані логічно: ліва панель для керування датасетами, моделями та моніторингом, права – для візуалізації. Логування подій у текстовому полі забезпечує зворотний зв'язок у реальному часі, що полегшує діагностику. Для операторів, які працюють із системою, важливо

передбачити навчання з використання функцій, таких як генерація аномалій чи вибір моделей (Isolation Forest, Neural Network), щоб уникнути помилок.

Оскільки система аналізує мережевий трафік і генерує тестові аномалії (наприклад, generate_connection_flood), робоче місце має бути захищене від несанкціонованого доступу. Використання антивірусного програмного забезпечення, шифрування даних і обмеження доступу до функцій генерації аномалій є обов'язковими. Логи (anomaly_detector.log) повинні зберігатися в захищеній директорії, а мережеві тести проводитися лише в ізольованому середовищі, щоб уникнути впливу на реальні мережі.

Тривале використання системи, особливо під час моніторингу в реальному часі, може викликати когнітивне перевантаження. Для цього рекомендуються перерви кожні 45–60 хвилин, а також використання таймерів або нагадувань. Інтерфейс системи підтримує чітке відображення аномалій (червоним кольором на графіках), що допомагає швидко реагувати на загрози, знижуючи стрес. Робоче місце бажано облаштувати в тихому приміщенні з можливістю провітрювання для підтримки концентрації.

Організація робочих місць для роботи з системою виявлення аномалій у веб-трафіку вимагає комплексного підходу, що поєднує технічне оснащення, ергономіку, безпеку та психологічний комфорт. Дотримання цих принципів забезпечує не лише ефективність розробки та використання системи, а й збереження здоров'я та продуктивності користувачів.

4.4 Техніко-економічне обґрунтування

Розробка системи виявлення аномалій у веб-трафіку на основі нейронних мереж, представлена в наданому коді, має значний потенціал для практичного застосування в сфері кібербезпеки, зокрема для захисту мережевої інфраструктури від аномальних подій, таких як DDoS-атаки, сканування портів чи флуд з'єднань. Техніко-економічне обґрунтування розробки системи охоплює аналіз технічної реалізації, економічної доцільності та потенційних переваг її впровадження.

Технічна реалізація. Система побудована з використанням сучасних бібліотек Python, таких як scikit-learn, pandas, numpy, matplotlib та customtkinter, що забезпечують ефективну обробку даних, навчання моделей машинного навчання та створення зручного графічного інтерфейсу. У коді реалізовано чотири моделі виявлення аномалій: Isolation Forest, One-Class SVM, Random Forest та нейронна мережа (MLPClassifier). Нейронна мережа, як основний компонент, використовує архітектуру з трьома прихованими шарами (64, 32, 16 нейронів), що дозволяє ефективно обробляти складні нелінійні залежності в даних мережевого трафіку. Використання бібліотеки psutil забезпечує моніторинг мережевих параметрів у реальному часі, а генерація синтетичних аномалій (NetworkAnomalyGenerator) дозволяє створювати реалістичні сценарії для тестування. Інтерфейс, створений за допомогою customtkinter, є інтуїтивно зрозумілим, що полегшує взаємодію з системою навіть для користувачів без глибоких технічних знань. Технічна реалізація є модульною, що спрощує масштабування та інтеграцію з іншими системами, наприклад, через API.

Економічна доцільність. Розробка системи базується на використанні відкритих бібліотек із відкритим кодом, що значно знижує витрати на програмне забезпечення. Основні витрати пов’язані з розробкою, тестуванням та підтримкою системи, а також із забезпеченням апаратних ресурсів для її роботи. Для реалізації системи достатньо комп’ютера середньої потужності (наприклад, з процесором Intel Core i5, 8 ГБ оперативної пам’яті та SSD), що робить її доступною для малого та середнього бізнесу. Потенційні економічні вигоди включають зменшення ризиків кібератак, які можуть призвести до значних фінансових втрат (згідно з дослідженнями, середня вартість DDoS-атаки для компаній становить від $20,000 до $100,000 залежно від масштабу). Система дозволяє автоматизувати процес виявлення аномалій, що зменшує потребу в залученні додаткового персоналу для моніторингу мережі. Крім того, можливість використання синтетичних датасетів для навчання знижує залежність від дорогих пропрієтарних наборів даних.

Переваги та перспективи. Впровадження системи забезпечує швидке виявлення аномалій у реальному часі, що є критично важливим для оперативного реагування на

загрози. Модульність системи дозволяє легко додавати нові моделі машинного навчання або адаптувати її до специфічних потреб організації. З економічної точки зору, система є вигідною завдяки низьким початковим витратам та високій ефективності у запобіганні фінансовим втратам від кібератак. У перспективі система може бути інтегрована з хмарними платформами або системами SIEM, що розширить її функціонал та ринкову привабливість. Крім того, використання нейронних мереж відкриває можливості для подальшого вдосконалення системи шляхом впровадження глибокого навчання, що підвищить точність виявлення складних аномалій.

Розроблена система є технічно здійсненною та економічно виправданою. Вона поєднує сучасні методи машинного навчання з доступними технологіями, забезпечуючи ефективне вирішення завдань кібербезпеки. Низькі витрати на розробку та впровадження, поєднані з високим потенціалом захисту від кіберзагроз, роблять систему привабливою для організацій різного масштабу. Подальший розвиток може включати інтеграцію з хмарними сервісами та вдосконалення алгоритмів, що забезпечить її конкурентоспроможність на ринку кібербезпеки.

ВИСНОВКИ

Розробка системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж є актуальним і перспективним проєктом, що відповідає сучасним викликам кібербезпеки. На етапі аналізу предметної області було проведено ґрунтовне вивчення проблематики, визначено ключові характеристики мережевого трафіку та обґрунтовано необхідність використання нейронних мереж для обробки складних нелінійних залежностей. Практична цінність цього етапу полягає у створенні теоретичної бази, яка враховує локальні особливості мережевої інфраструктури України, що сприяє адаптації системи до реальних умов. Однак для вдосконалення рекомендується інтеграція публічних датасетів, таких як CICIDS2017, для підвищення узагальнюючої здатності моделей.

Етап проектування системи вирізняється модульною архітектурою, що забезпечує гнучкість і масштабованість. Реалізація класів для захоплення трафіку, генерації аномалій, обробки даних і аналізу з використанням бібліотек psutil, scikit-learn та customtkinter дозволила створити комплексне рішення з інтуїтивним інтерфейсом. Практична цінність полягає у можливості моніторингу Wi-Fi мереж у реальному часі та генерації синтетичних аномалій для тестування. Порівняно з аналогами, такими як Snort чи Zeek, система є менш ресурсоємною та адаптованою до обмежених бюджетів. Для вдосконалення пропонується додати підтримку GPU для прискорення навчання моделей та інтерактивні елементи інтерфейсу.

На етапі реалізації було створено функціональну систему, яка успішно виявляє аномалії, зокрема DDoS-атаки та сканування портів, із F1-score 0.87 для ізоляційного лісу. Новизна роботи полягає у поєднанні нейронної мережі (MLPClassifier) з іншими алгоритмами (IsolationForest, One-Class SVM, RandomForest) та інтеграції модуля генерації аномалій, що імітує реальні атаки. Студентом запропоновано унікальний підхід до автоматичного визначення Wi-Fi інтерфейсу та динамічного налаштування порогу аномалій. Система має потенціал для використання в державному секторі, малих підприємствах та освіті, сприяючи технологічній незалежності України. Технічно-економічна ефективність підтверджується низькими вимогами до ресурсів

та відкритим кодом. Для подальшого розвитку рекомендується впровадження методів обробки незбалансованих даних та шифрування захопленого трафіку для підвищення безпеки. Впровадження системи на підприємстві не підтверджено, що є обмеженням, однак її готовність до практичного використання очевидна.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Network Anomaly Detection: A Comprehensive Guide [Електронний ресурс] / Kentik. – Режим доступу: https://www.kentik.com/kentipedia/network-anomaly-detection/. – [Цит. 2025, 8 трав.]. – Назва з екрану.

2. Протоколи TCP та UDP – пояснення простою мовою [Електронний ресурс] / Devzone. – Режим доступу: https://devzone.org.ua/post/protokoly-tcp-ta-udp-poiasnennia-prostoiu-movoiu. – [Цит. 2025, 8 трав.]. – Назва з екрану.

3. Patil D. Machine learning and deep learning: Methods, techniques, applications, challenges, and future research opportunities / D. Patil, N. Rane, P. Desai, J. Rane // Trustworthy Artificial Intelligence in Industry and Society. – 2024. – P. 28–81.

4. Rao A. R. Nonlinear Functional Modeling Using Neural Networks / A. R. Rao, M. Reimherr // Journal of Computational and Graphical Statistics. – 2023. – Vol. 32, № 4. – P. 1–20.

5. Лепетун М. Хмарні технології та Інтернет речей (IoT): взаємодія між хмарними сервісами та підключеними пристроями в Інтернеті речей / М. Лепетун, А. Перепелиця, С. Батаєв // Інноваційні технології та наукові рішення для промисловості. – 2023. – Т. 329, № 6. – С. 223–229.

6. Suricata vs Snort [Електронний ресурс] / Stamus Networks. – Режим доступу: https://www.stamus-networks.com/suricata-vs-snort. – [Цит. 2025, 8 трав.]. – Назва з екрану.

7. Patel J. Leveraging K-Means Clustering and Z-Score for Anomaly Detection in Bitcoin Transactions / J. Patel, J. Reiner, B. Stilwell, A. Wahbeh, R. Seetan // Machine Learning. – 2025. – [Електронний ресурс]. – Режим доступу: [вказати URL, якщо доступно]. – [Цит. 2025, 8 трав.].

8. NumPy 2.2.0 released! [Електронний ресурс] / NumPy. – Режим доступу: https://numpy.org/. – [Цит. 2025, 8 трав.]. – Назва з екрану.

9. Ащепков В. Використання моделі Isolation Forest для виявлення аномалій у даних вимірювань / В. Ащепков // Innovative technologies and scientific solutions for industries. – 2024. – Т. 27, № 1.

10. OneClassSVM [Електронний ресурс] / Scikit-learn. – Режим доступу: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html. – [Цит. 2025, 8 трав.]. – Назва з екрану.

11. MLPClassifier [Електронний ресурс] / Scikit-learn. – Режим доступу: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. – [Цит. 2025, 8 трав.]. – Назва з екрану.

12. Що таке випадковий ліс у машинному навчанні [Електронний ресурс] / Solix. – Режим доступу: https://www.solix.com/uk/products/answers/what-is-random-forest-in-machine-learning/. – [Цит. 2025, 8 трав.]. – Назва з екрану.

13. Jeong J. GUI information-based interaction logging and visualization for asynchronous usability testing / J. Jeong, N. Kim, H. P. In // Expert Systems with Applications. – 2020.

14. Popular Python Libraries – NumPy, Pandas, Seaborn, Sklearn [Електронний ресурс] / Almabetter. – Режим доступу: https://www.almabetter.com/bytes/tutorials/python/popular-python-libraries. – [Цит. 2025, 8 трав.]. – Назва з екрану.

15. Що таке діаграма класів UML і найкращий творець діаграм класів UML [Електронний ресурс] / MindOnMap. – Режим доступу: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/. – [Цит. 2025, 8 трав.]. – Назва з екрану.

16. Діаграма послідовності (Sequence Diagrams) [Електронний ресурс] / Maxzosim. – Режим доступу: https://www.maxzosim.com/sequence-diagrams/. – [Цит. 2025, 8 трав.]. – Назва з екрану.

17. Повне розуміння діаграми компонентів UML за допомогою легкого методу [Електронний ресурс] / MindOnMap. – Режим доступу: https://www.mindonmap.com/uk/blog/uml-component-diagram/. – [Цит. 2025, 8 трав.]. – Назва з екрану.

18. Visual Studio [Електронний ресурс] / Wikipedia. – Режим доступу: https://en.wikipedia.org/wiki/Visual_Studio. – [Цит. 2025, 8 трав.]. – Назва з екрану.

19. Що таке мова програмування Python? [Електронний ресурс] / Freehost. – Режим доступу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovanija-python/. – [Цит. 2025, 8 трав.]. – Назва з екрану.

20. Jawahar S. An Exploration of Python Libraries in Machine Learning Models for Data Science / S. Jawahar, G. P. Kukunuri, S. Devaraju, S. Gokuldev // Advanced Interdisciplinary Applications of Machine Learning Python Libraries for Data Science. – 2023. – P. 1–31.

Додаток А. Лістинг програми

У рамках розробки системи виявлення аномалій у веб-трафіку за допомогою нейронних мереж було створено програмне рішення, яке включає кілька ключових компонентів: захоплення мережевого трафіку, генерацію синтетичних даних, навчання моделей машинного навчання та графічний інтерфейс користувача. Нижче наведено основні лістинги програми, які демонструють ключові аспекти реалізації, зокрема класи для роботи з мережевим трафіком, створення датасетів, навчання нейронної мережі та моніторингу аномалій. Код написаний мовою Python із використанням бібліотек customtkinter, scikit-learn, matplotlib та інших.

1. Захоплення мережевого трафіку (клас WiFiTrafficCapture)

Клас WiFiTrafficCapture відповідає за захоплення статистики мережевого трафіку в реальному часі з використанням бібліотеки psutil. Він автоматично визначає Wi-Fi інтерфейс та періодично збирає дані про відправлені/отримані байти, пакети та активні з’єднання.

```
class WiFiTrafficCapture:
    def __init__(self):
        self.logger = logging.getLogger(__name__)
        self.interface = self.detect_wifi_interface()
        self.packets = []
        self.running = False
    def detect_wifi_interface(self):
        try:
            interfaces = psutil.net_if_addrs()
            for iface_name in interfaces.keys():
                if any(keyword in iface_name.lower() for keyword in ['wi-fi',
'wlan', 'wireless']):
                    self.logger.info(f"Обрано Wi-Fi інтерфейс: {iface_name}")
                    return iface_name
            return list(interfaces.keys())[0] if interfaces else None
        except Exception as e:
            self.logger.error(f"Помилка визначення інтерфейсу: {e}")
            return None
    def capture_with_psutil(self):
        self.logger.info("Використовую psutil для збору статистики")
        while self.running:
            try:
                net_io = psutil.net_io_counters()
                connections = psutil.net_connections()
                stats = {
                    'timestamp': datetime.now(),
                    'bytes_sent': net_io.bytes_sent,
                    'bytes_recv': net_io.bytes_recv,
                    'packets_sent': net_io.packets_sent,
```

```
                    'packets_recv': net_io.packets_recv,
                    'active_connections': len(connections),
                    'tcp_connections': len([c for c in connections if c.type
== socket.SOCK_STREAM]),
                    'udp_connections': len([c for c in connections if c.type
== socket.SOCK_DGRAM])
                }
                self.packets.append(stats)
                self.logger.debug(f"Статистика: {stats}")
            except Exception as e:
                self.logger.error(f"Помилка збору статистики: {e}")
            time.sleep(1)
    def start_capture(self):
        self.running = True
        self.packets = []
        self.capture_thread =
threading.Thread(target=self.capture_with_psutil)
        self.capture_thread.daemon = True
        self.capture_thread.start()
```

2. Генерація датасетів (клас WebTrafficDataset)

Клас WebTrafficDataset створює синтетичні датасети для моделювання нормального та аномального мережевого трафіку, зокрема для DDoS-атак та сканування портів. Це дозволяє тестувати моделі машинного навчання на контрольованих даних.

```
class WebTrafficDataset:
    def __init__(self):
        self.data_dir = "traffic_data"
        if not os.path.exists(self.data_dir):
            os.makedirs(self.data_dir)
        self.logger = logging.getLogger(__name__)
    def create_sample_dataset(self):
        self.logger.info("Створення зразкового датасету")
        n_samples = 10000
        normal_samples = int(n_samples * 0.8)
        normal_data = {
            'packet_size': np.random.normal(500, 150, normal_samples),
            'packet_rate': np.random.normal(100, 20, normal_samples),
            'bytes_sent': np.random.normal(10000, 2000, normal_samples),
            'bytes_recv': np.random.normal(50000, 10000, normal_samples),
            'connections': np.random.poisson(10, normal_samples),
            'duration': np.random.exponential(5, normal_samples),
            'tcp_ratio': np.random.beta(8, 2, normal_samples),
            'src_port': np.random.randint(1024, 65535, normal_samples),
            'dst_port': np.random.choice([80, 443, 8080, 22, 21],
normal_samples),
            'is_anomaly': np.zeros(normal_samples)
        }
        anomaly_samples = n_samples - normal_samples
        anomaly_data = {
            'packet_size': np.random.normal(1500, 300, anomaly_samples),
            'packet_rate': np.random.normal(500, 100, anomaly_samples),
```

```
            'bytes_sent': np.random.normal(100000, 20000, anomaly_samples),
            'bytes_recv': np.random.normal(5000, 1000, anomaly_samples),
            'connections': np.random.poisson(100, anomaly_samples),
            'duration': np.random.exponential(0.5, anomaly_samples),
            'tcp_ratio': np.random.beta(2, 8, anomaly_samples),
            'src_port': np.random.randint(1, 65535, anomaly_samples),
            'dst_port': np.random.randint(1, 65535, anomaly_samples),
            'is_anomaly': np.ones(anomaly_samples)
        }
        data = pd.DataFrame({
            key: np.concatenate([normal_data[key], anomaly_data[key]])
            for key in normal_data.keys()
        })
        return data.sample(frac=1).reset_index(drop=True)
```

3. Навчання нейронної мережі (метод train_neural_network)

Метод train_neural_network реалізує навчання нейронної мережі з використанням бібліотеки scikit-learn. Модель має три приховані шари та використовує нормалізовані дані для класифікації аномалій.

```
def train_neural_network(self, X_train, X_test, y_train, y_test):
    self.log_message("Навчання нейронної мережі...")
    model = MLPClassifier(
        hidden_layer_sizes=(64, 32, 16),
        activation='relu',
        solver='adam',
        max_iter=200,
        random_state=42
    )
    model.fit(X_train, y_train)
    y_pred_test = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)
    if y_pred_proba.shape[1] > 1:
        y_pred_proba = y_pred_proba[:, 1]
    else:
        y_pred_proba = y_pred_proba[:, 0]
    normal_scores = y_pred_proba[y_test == 0] if any(y_test == 0) else []
    threshold = np.percentile(normal_scores, 95) if len(normal_scores) > 0
else 0.5
    self.models['neural_network'] = {
        'model': model,
        'threshold': threshold
    }
    self.current_model = 'neural_network'
    self.threshold_label.configure(text=f"Поріг: {threshold:.3f}")
    self.visualize_training_results(y_test, y_pred_test, y_pred_proba,
threshold)
```

4. Моніторинг аномалій у реальному часі (метод monitor_traffic)

Метод monitor_traffic забезпечує моніторинг мережевого трафіку в реальному часі, аналізуючи дані за допомогою навченої моделі та виявляючи аномалії на основі фіксованого порогу.

```
def monitor_traffic(self):
    while self.is_monitoring:
        try:
            time.sleep(1)
            net_io = psutil.net_io_counters()
            connections = psutil.net_connections()
            features = {}
            for feature in self.feature_names:
                if feature == 'bytes_sent':
                    features[feature] = net_io.bytes_sent
                elif feature == 'bytes_recv':
                    features[feature] = net_io.bytes_recv
                elif feature == 'packets_sent':
                    features[feature] = net_io.packets_sent
                elif feature == 'packets_recv':
                    features[feature] = net_io.packets_recv
                elif feature == 'connections':
                    features[feature] = len(connections)
                else:
                    features[feature] = np.random.normal(0, 1)
            data_point = pd.DataFrame([features])
            X_scaled = self.scaler.transform(data_point)
            model_data = self.models[self.current_model]
            model = model_data['model']
            if self.current_model in ['neural_network', 'random_forest']:
                score = model.predict_proba(X_scaled)[0, 1]
                normalized_score = score
            else:
                score = -model.score_samples(X_scaled)[0] if
self.current_model == 'isolation_forest' else -
model.decision_function(X_scaled)[0]
                normalized_score = (score - self.min_score) / (self.max_score
- self.min_score) if self.max_score > self.min_score else 0.5
            threshold = 0.75
            is_anomaly = normalized_score > threshold
            self.data_queue.put({
                'timestamp': datetime.now(),
                'score': score,
                'normalized_score': normalized_score,
                'is_anomaly': is_anomaly,
                'data': features
            })
            if is_anomaly:
                self.log_message(f"Виявлено аномалію! Оцінка:
{normalized_score:.3f}")
        except Exception as e:
            self.logger.error(f"Помилка моніторингу: {e}")
```

5. Візуалізація результатів навчання (метод visualize_training_results)

Метод visualize_training_results створює візуалізації для оцінки ефективності навченої моделі, включаючи матрицю помилок, ROC-криву та розподіл оцінок аномальності.

```
def visualize_training_results(self, y_true, y_pred, scores, threshold):
    self.training_figure.clear()
    ax1 = self.training_figure.add_subplot(221)
    ax2 = self.training_figure.add_subplot(222)
    ax3 = self.training_figure.add_subplot(223)
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', ax=ax1, cmap='Blues')
    ax1.set_title("Матриця помилок")
    ax1.set_xlabel("Передбачено")
    ax1.set_ylabel("Фактично")
    if scores is not None and len(np.unique(y_true)) > 1:
        fpr, tpr, _ = roc_curve(y_true, scores)
        auc_score = roc_auc_score(y_true, scores)
        ax2.plot(fpr, tpr, label=f'ROC (AUC = {auc_score:.3f})')
        ax2.plot([0, 1], [0, 1], 'k--')
        ax2.set_title("ROC крива")
        ax2.set_xlabel("False Positive Rate")
        ax2.set_ylabel("True Positive Rate")
        ax2.legend()
    normal_scores = scores[y_true == 0] if any(y_true == 0) else []
    anomaly_scores = scores[y_true == 1] if any(y_true == 1) else []
    if len(normal_scores) > 0:
        ax3.hist(normal_scores, bins=50, alpha=0.5, label='Нормальні',
density=True)
    if len(anomaly_scores) > 0:
        ax3.hist(anomaly_scores, bins=50, alpha=0.5, label='Аномалії',
density=True)
    ax3.axvline(x=threshold, color='r', linestyle='--', label='Поріг')
    ax3.set_title("Розподіл оцінок аномальності")
    ax3.set_xlabel("Оцінка")
    ax3.set_ylabel("Щільність")
    ax3.legend()
    self.training_figure.tight_layout()
    self.training_canvas.draw()
```

Ці лістинги відображають основну функціональність системи: захоплення та аналіз мережевого трафіку, генерацію даних, навчання нейронної мережі, моніторинг у реальному часі та візуалізацію результатів. Інтерфейс користувача, реалізований за допомогою customtkinter, забезпечує зручну взаємодію з системою, дозволяючи вибирати датасети, моделі та генерувати аномалії для тестування.