Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод виявлення аномалій у журналах API для забезпечення безпеки та надійності програмних систем
(тема)

Виконав:
здобувач 2 року навчання,
групи СПм-23-4
Олександр АВДЄЄВ
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Віталій МАРТОВИЦЬКИЙ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Авдєєву Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод виявлення аномалій у журналах API для забезпечення безпеки та надійності програмних систем

затверджена наказом по університету від " 21 " квітня 2025 р. № 296 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи Набір даних для навчання

4. Перелік питань, що потрібно опрацювати у роботі

Актуальність та обґрунтування проблеми

Огляд сучасних підходів до виявлення аномалій

Аналіз набору даних CSIC HTTP 2010

Попередня обробка та побудова ознак

Вибір та опис моделей неконтрольованого навчання

Побудова та налаштування моделей

Оцінка якості моделей

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Огляд методів виявлення аномалій в	22.04.25-29.04.25	
	системах		
2	Вибір та обгрунтування методики дослідження	30.04.25-05.05.25	
3	Вибір інструментальних засобів	06.05.25-09.05.25	
4	Розробка моделей	10.05.25-20.05.25	
5	Проведення експериментів	21.05.25-02.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	03.06.25-05.06.25	
7	Подання кваліфікаційної роботи керівникові та	06.06.25-09.06.25	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	10.06.25-12.06.25	
	вання		

Дата видачі завдання " 21 " квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Віталій МАРТОВИЦЬКИЙ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 55 с., 9 рис., 6 табл., 1 дод., 7 джерел.

API, ЖУРНАЛИ ДОСТУПУ, ВИЯВЛЕННЯ АНОМАЛІЙ, МАШИННЕ НАВЧАННЯ, НЕКОНТРОЛЬОВАНЕ НАВЧАННЯ, КЛАСТЕРИЗАЦІЯ, БЕЗПЕКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, HTTP-ЗАПИТИ, GMM, ISOLATION FOREST, K-MEANS, OCSVM, АНАЛІЗ ПОВЕДІНКИ, ІНФОРМАЦІЙНА БЕЗПЕКА.

Метою кваліфікаційної роботи є розробка ефективного підходу до виявлення аномальної активності в журналах API-запитів з використанням методів неконтрольованого машинного навчання для підвищення рівня захищеності програмних систем.

У ході виконання кваліфікаційної роботи було проаналізовано типові загрози, пов’язані з API, а також вивчено можливості застосування алгоритмів кластеризації та виявлення аномалій (K-means, GMM, Isolation Forest, OCSVM) у контексті HTTP-журналів. У якості експериментальної бази використано датасет CSIC HTTP 2010, який було попередньо оброблено з урахуванням специфіки логів API. Було проведено налаштування моделей, оптимізовано гіперпараметри та оцінено якість класифікації за допомогою метрик Precision, Recall, F1-score, AUC. Результати показали, що моделі GMM та Isolation Forest демонструють найкращу ефективність у виявленні аномалій. Отримані результати можуть бути використані для впровадження систем моніторингу безпеки API в реальному часі без потреби у розмічених даних.

ABSTRACT

Master’s thesis: 55 pages, 9 figures, 6 tables, 1 appendices, 7 sources.

API, ACCESS LOGS, ANOMALY DETECTION, MACHINE LEARNING, UNSUPERVISED LEARNING, CLUSTERING, SOFTWARE SECURITY, HTTP REQUESTS, GMM, ISOLATION FOREST, K-MEANS, OCSVM, BEHAVIOR ANALYSIS, INFORMATION SECURITY.

The major goal of this thesis is to develop an effective approach for detecting anomalous activity in API request logs using unsupervised machine learning methods to enhance the security of software systems.
In the course of the work, typical API-related threats were analyzed, along with the applicability of clustering and anomaly detection algorithms (K-means, GMM, Isolation Forest, OCSVM) in the context of HTTP logs. The CSIC HTTP 2010 dataset was used as the experimental base and was preprocessed to reflect API log characteristics. The selected models were tuned, hyperparameters optimized, and classification performance evaluated using Precision, Recall, F1-score, and AUC metrics.
The results showed that GMM and Isolation Forest models achieved the highest effectiveness in anomaly detection. These findings can be applied to implement real-time API security monitoring systems without the need for labeled data.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

GMM – гаусівська змішана модель (англ., Gaussian Mixture Model)

HTTP – протокол передавання гіпертексту (англ., HyperText Transfer Protocol)

IF – ізолюючий ліс (англ., Isolation Forest)

OCSVM – метод опорних векторів для одного класу (англ., One-Class Support Vector Machine)

WDS – бездротова розподільча система (англ., Wireless Distribution System)

ВСТУП

У сучасному цифровому світі програмні системи дедалі частіше взаємодіють між собою за допомогою прикладних програмних інтерфейсів (API). Такі взаємодії стають критично важливими для функціонування як окремих сервісів, так і складних розподілених систем. Однак широке використання API також відкриває нові вектори для атак, порушення безпеки та збоїв у роботі програмного забезпечення. У цьому контексті своєчасне виявлення аномалій у журналах API (API logs) відіграє ключову роль у забезпеченні надійності та захисту систем.

Журнали API містять важливу інформацію про запити, відповіді, параметри викликів і поведінку клієнтів. Аналіз цих даних дозволяє виявляти підозрілі дії, потенційні вторгнення або програмні помилки. Проте, через велику кількість логів та різноманітність даних, ручне опрацювання є неефективним, що зумовлює потребу в автоматизованих методах виявлення аномалій.

Метою цієї кваліфікаційної роботи є розробка та дослідження методів виявлення аномалій у журналах API з використанням сучасних підходів до аналізу даних та машинного навчання. Запропоноване рішення має сприяти підвищенню рівня безпеки та надійності програмних систем шляхом раннього виявлення відхилень від нормальної поведінки.

Актуальність теми зумовлена зростаючою кількістю кібератак, уразливостей у програмному забезпеченні та необхідністю оперативного реагування на інциденти. Робота має як теоретичне, так і практичне значення, оскільки результати можуть бути впроваджені в реальні системи моніторингу та захисту інформаційних ресурсів.

1 АНАЛІЗ ПЕРЕДМЕТНОЇ ОБЛАСТІ

1.1 Логи API: сутність, роль і значення

Логи API – це записи, що описують виклики API, здійснені користувачами. Зазвичай у логах API міститься важлива інформація про виконаний виклик API, як зазначено в таблиці 1.1 [1]. У цій кваліфікаційній роботі основна увага приділяється виявленню аномалій у точках доступу API. Типи даних у логах можуть надавати різноманітну інформацію про виконаний виклик API. Наприклад, мітки часу та коди стану можуть використовуватися для виявлення періодів невдалих запитів, які потенційно є аномальними.

Таблиця 1.1 – Інформація, яка зазвичай міститься в журналах API

Інформація про лог API	Опис
Позначка часу (Timestamp)	Вказує, коли був зроблений виклик API
Кінцева точка (Endpoint)	Яка кінцева точка API була доступна
Параметри запиту (Request Parameters)	Деталізація запитаної команди
Ідентифікація користувача (User Identification)	Зазвичай ID або токени, що використовуються для ідентифікації
Дані відповіді (Response data)	Що API повернув у відповідь
Коди стану (Status Codes)	Вказує на успішність або помилку запиту
IP-адреси та інформація про пристрої	Інформація про місце та пристрій, з якого надійшов запит

У сучасних програмних системах, зокрема в архітектурі мікросервісів, API (Application Programming Interface) виступає ключовим елементом взаємодії між окремими компонентами. У результаті кожна взаємодія через

API – будь то запит користувача, виклик внутрішнього сервісу чи зовнішній доступ до ресурсу – фіксується у вигляді запису в журналі подій, тобто у логах API.

Логи API є невід’ємною частиною забезпечення надійної та безпечної роботи програмного забезпечення. завдяки логам можливо:

- аналізувати продуктивність системи, виявляти "вузькі місця" або неефективні ділянки;
- відстежувати збої та помилки, аналізуючи частоту та типи відповідей із кодами 4xx або 5xx;
- реагувати на інциденти безпеки, наприклад, несанкціоновані доступи, аномальну активність, підозрілі шаблони запитів;
- виконувати аудит користувацької активності – хто, коли і що саме робив у системі;
- використовувати дані для навчання моделей машинного навчання, зокрема для задач класифікації, прогнозування або виявлення аномалій.

Саме останній аспект – автоматизований аналіз та виявлення аномалій – є основною тематикою цієї кваліфікаційної роботи.

1.2 Безпека

Теорія, пов’язана з безпекою, подається в цьому розділі для розуміння контексту, в якому буде застосовано машинне навчання. Крім того, знання про те, яку інформацію можуть надати журнали API з точки зору безпеки, є важливим для вирішення питань, порушених у цій магістерській роботі. Тому тут буде розглянуто такі теми, як OWASP, ризики безпеки API та система виявлення вторгнень (IDS).

OWASP – це некомерційна організація, яка підвищує безпеку програмного забезпечення. Це відкрита спільнота, яка надає розділи, документи, форуми, проєкти та інструменти для безкоштовного використання будь-ким, хто бажає покращити безпеку свого застосунку.

Кожні кілька років вони публікують інформаційний документ, що містить найважливіші загрози безпеці вебзастосунків. Наприклад, остання версія цього списку "топ-10" була опублікована у 2021 році і вказує, що найсерйознішою вразливістю є порушення контролю доступу. Ці топ-десять рекомендовано використовувати компаніям, які прагнуть покращити свою безпеку.

API є критично важливими компонентами сучасної розробки програмного забезпечення, оскільки забезпечують безперебійну взаємодію між різними системами та застосунками. Проте їх відкритість і доступність також роблять їх вразливими до різноманітних загроз безпеці. У цьому розділі розглядаються поширені ризики безпеки API, деякі з яких згадуються OWASP. Багато ризиків безпеки API мають схожість із ризиками веббезпеки, проте важливо розуміти, що між ними існують певні відмінності. API зазвичай надають доступ до клієнтів через контроль доступу. Під час надсилання запитів до API користувач отримує набагато більше інформації, ніж у звичайному вебзапиті, де зазвичай видно лише URL. У випадку з API також стають доступними заголовки HTTP, параметри запиту та інші дані. Через це площа атаки зростає, що може призвести до збільшення кількості спроб атак на API.

Як згадувалося раніше некомерційна організація OWASP відіграє важливу роль у підвищенні безпеки програмного забезпечення завдяки своїм комплексним ресурсам і рекомендаціям, включаючи регулярно оновлюваний інформаційний документ, що описує критичні загрози безпеці вебзастосунків. Вразливості, описані нижче, відповідають проблемам, ідентифікованим OWASP, що підкреслює актуальність їхньої діяльності в контексті викликів, з якими стикаються API. Вразливості, представлені в цьому розділі, відображають ризики безпеки, висвітлені OWASP, особливо у списку ризиків безпеки API. Зосередження уваги на цих вразливостях дозволяє узгодити цю роботу з рекомендаціями OWASP і зробити внесок у загальні зусилля зі зміцнення безпеки API проти сучасних загроз.

Ін'єкція скриптів – це вразливість, коли зловмисник надсилає шкідливий скрипт через запит до API, і система помилково виконує або обробляє його як допустимий параметр. Це може призвести до виконання несанкціонованих дій, таких як крадіжка даних, компрометація облікових записів або запуск шкідливого коду на стороні сервера чи клієнта.

SQL-ін’єкція є поширеною вразливістю, яка виникає, коли зловмисник змінює стандартний SQL-запит через експлуатацію введених користувачем даних. Формуючи вхідні дані, що змінюють структуру запиту, зловмисник може отримати несанкціонований доступ до бази даних, змінити або викрасти конфіденційну інформацію, а також здійснити інші шкідливі дії.

Переповнення буфера експлуатує вразливість, коли застосунок або API не перевіряє межі буфера (область пам’яті), дозволяючи зловмиснику перевантажити буфер надмірними даними. Це може спричинити збій системи, пошкодження даних або навіть надати зловмиснику можливість виконати довільний код і потенційно отримати контроль над системою.

Розподілені атаки типу відмова в обслуговуванні (DDoS) націлені на API, перевантажуючи сервер великою кількістю трафіку, що суттєво перевищує його здатність обробляти запити. Це може зробити API та пов’язані сервіси недоступними, порушуючи роботу та блокуючи доступ для легітимних користувачів.

Атаки на логін експлуатують вразливості в механізмах автентифікації та авторизації. Поширені техніки включають fuzzing – надсилання великої кількості випадкових даних для виявлення вразливостей, а також credential stuffing – використання викрадених комбінацій імен користувачів і паролів для отримання несанкціонованого доступу. Метою цих атак є обхід захисних механізмів і доступ до обмежених розділів або конфіденційної інформації.

Деякі з ризиків безпеки API, наведених OWASP, можна класифікувати як атаки на логін. Наприклад, порушення автентифікації (Broken Authentication) та порушення контролю на рівні функцій (Broken Function Level Authorisation) можуть вважатися такими, оскільки зловмисник може

використати викрадені токени або облікові дані для отримання несанкціонованого доступу.

Фальсифікація запитів на стороні сервера (SSRF) є критичною вразливістю, за якої зловмисник може змусити сервер виконувати несанкціоновані запити до внутрішніх або зовнішніх ресурсів. Це може призвести до розкриття конфіденційної інформації, конфігурацій сервера або отримання несанкціонованого доступу до внутрішніх сервісів. Маніпулюючи сервером для виконання запитів від їхнього імені, зловмисники експлуатують SSRF для обходу заходів безпеки та доступу до захищених даних.

Вторгнення – це несанкціонована дія, що порушує безпеку комп’ютера або мережі з точки зору конфіденційності, цілісності та доступності. IDS (система виявлення вторгнень) – це програмний інструмент, призначений для виявлення несанкціонованого доступу до мережі або комп’ютерної системи. Її можливості включають виявлення шкідливого трафіку, зокрема атак на сервіси, які вважаються вразливими. IDS також може виявляти атаки на хості, такі як підвищення привілеїв, несанкціоновані входи в систему та доступ до конфіденційних файлів. Це стосується й мережевих атак, таких як DDoS-атаки та підміна (spoofing).

Мотивація використання IDS полягає в тому, щоб доповнити статичну систему моніторингу, таку як міжмережевий екран (фаєрвол), динамічною системою виявлення. IDS аналізує пакети, отримані від користувача, перевіряючи наявність порушень правил, які виявляються за допомогою алгоритму розпізнавання шаблонів. Коли таке порушення виявляється, система надсилає сповіщення адміністраторам про те, що сталася підозріла подія, яка потребує уваги.

Існує модель загроз, яка передбачає поділ різних типів загроз на три окремі класи: зовнішні проникнення, внутрішні проникнення та зловживання (misfeasance). Зовнішні проникнення охоплюють вторгнення, здійснені користувачами, які не мають дозволу на доступ до комп’ютерної системи. Внутрішні проникнення – це ситуації, коли користувач має загальний доступ

до системи, але не має права доступу до конкретних даних, що були скомпрометовані. Зловживання охоплює випадки, коли уповноважений користувач неправомірно використовує свій доступ як до самої системи, так і до її даних.

1.3 Виявлення аномалій

Виявлення аномалій – це процес аналізу даних з метою виявлення підозрілих випадків, які зазвичай називають викидами через їхню відмінність від встановленого шаблону поведінки. Виявлення аномалій є базовим підходом у різних сферах, особливо в реальному часі, де одним із важливих напрямів є системи безпеки. Існує кілька технік, які застосовуються для виявлення аномалій, зокрема класифікація, регресія та кластеризація.

Найпоширенішими підходами до виявлення аномалій є контрольоване (supervised) і неконтрольоване (unsupervised) навчання, хоча напівконтрольоване (semi-supervised) навчання також є життєздатним варіантом. Згадані техніки можуть відповідати цим підходам: наприклад, кластеризація вважається методом неконтрольованого навчання, тоді як класифікація та регресія належать до контрольованих.

У логах можуть існувати різні типи аномалій, зокрема точкові (point anomalies) та контекстуальні аномалії (contextual anomalies).

Точкова аномалія – це окремий випадок даних, який значно відрізняється від нормального шаблону у вибірці. Інтуїтивним прикладом точкової аномалії є ситуація, коли користувач намагається отримати доступ до заборонених розділів комп’ютерної системи або коли хтось зазвичай використовує три ресурси одночасно, а потім раптово використовує дев’ять. Точкові аномалії є найпоширенішими типами аномалій і саме на них орієнтується більшість систем виявлення аномалій. Проте точкові аномалії є малоефективними, коли необхідно виявити аномалії в певному контексті.

На рисунку 1.1 ілюструється два невеликих скупчення та дві точкові аномалії, щоб показати, як вони можуть виглядати візуально.

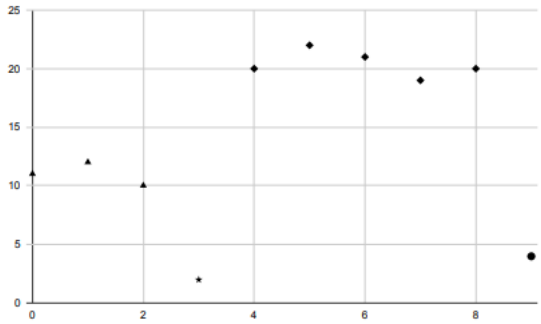

Рисунок 1.1 – Точкові аномалії при x = 3 та x = 9

У деяких ситуаціях окремого незалежного елемента даних недостатньо для виявлення аномалій. У таких випадках поведінка визначається в межах певного контексту. Контекстуальні дані можуть включати різні атрибути з наданого набору даних, наприклад, прив’язані до місця або часу.

Прикладом аномалії, пов’язаної з часом, може бути ситуація, коли користувач входить у бізнес-систему поза робочими годинами, хоча зазвичай здійснює вхід лише в робочий час. Вхід із іншого місця, наприклад, з іншого міста, також може вважатися контекстуальною аномалією у випадку контексту, пов’язаного з геолокацією.

На рисунку 1.2 подано візуальний приклад контекстуальної аномалії.

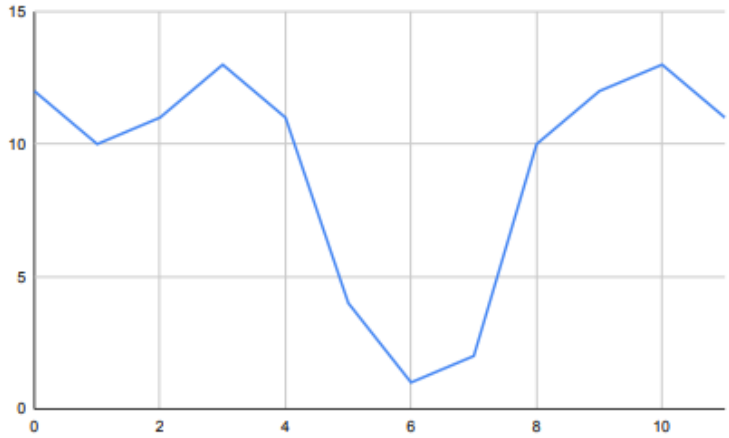

Рисунок 1.2 – Контекстна аномалія в діапазоні 5 <x< 7

Замість того, щоб один елемент даних був аномальним, колективна аномалія стосується групи елементів, які поводяться аномально відносно всього набору даних. При цьому окремий елемент з цієї групи не обов'язково є аномалією сам по собі.

Прикладом може бути ситуація, коли хтось вимірює свій пульс, і протягом певного часу спостерігається його підвищення, яке триває деякий період, а потім повертається до нормального рівня. Ця група підвищених значень пульсу може розглядатися як колективна аномалія.

Цей приклад представлений на рисунку 1.3, де процес 3 є колективною аномалією відносно інших процесів у наборі даних.

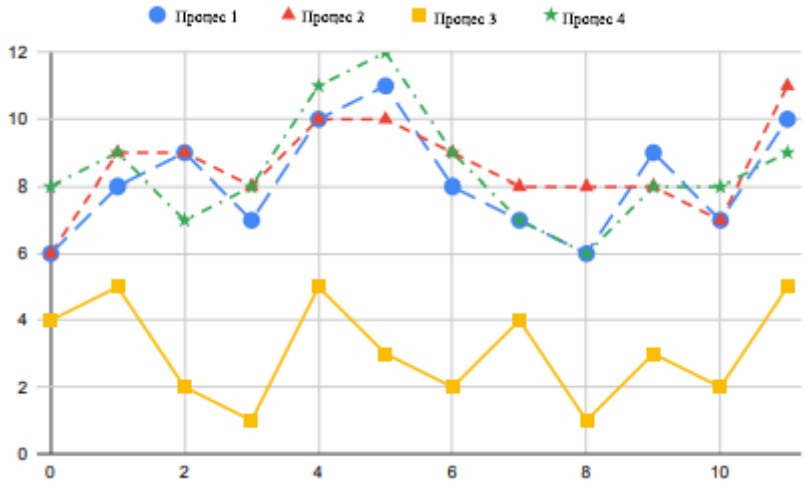

Рисунок 1.3 – Ілюстрація колективної аномалії

1.4 Неконтрольоване навчання

Неконтрольоване навчання – це категорія машинного навчання, яка працює з даними без участі людини або попередніх знань. Моделі неконтрольованого навчання не використовують розмічені дані для виявлення шаблонів – процес відбувається без втручання чи вказівок з боку людини. Існує три основні методи неконтрольованого навчання: кластеризація, асоціативні правила та зниження розмірності. У цій магістерській роботі будуть розглядатися кластеризація та зниження

розмірності, оскільки саме ці методи є релевантними до поставленої задачі.

Кластеризація – це техніка, яка використовується для дослідження та поділу нерозмічених даних на менші групи, що називаються кластерами, на основі їхніх подібностей та відмінностей. Алгоритми кластеризації розділяють дані на групи, виявляючи подібності у шаблонах некатегоризованих записів. Кластеризація є одним із найпоширеніших підходів у неконтрольованому навчанні, і для її реалізації існує багато різних алгоритмів. У таблиці 1.2 наведено підмножину доступних методів кластеризації.

Таблиця 1.2 – Типи кластеризації

Тип кластеризації	Опис
Ексклюзивна кластеризація (жорстка кластеризація)	Дані групуються так, що кожна точка може бути лише в одному кластері
Ієрархічна кластеризація	Дані поділяються на окремі кластери за подібністю, які об’єднуються на основі ієрархії
Перекривна кластеризація (м’яка кластеризація)	Дані можуть належати до кількох кластерів з певним ступенем належності
Імовірнісна кластеризація	Дані групуються в кластери за ймовірністю належності до певного кластеру

K-means є однією з найпоширеніших моделей, які використовуються в межах парадигми неконтрольованого навчання. Це метод жорсткої кластеризації, метою якого є визначення центрів кластерів (центроїдів), які можна розглядати як опорні точки для порівняння даних у наборі. Простішими словами, центроїд – це центр маси для кожного кластеру. У кожному кластері може існувати лише один центроїд.

Лістинг 1.1 демонструє послідовність дій під час використання методу K-means. Збіжність досягається тоді, коли результати двох послідовних ітерацій стають ідентичними.

Лістинг 1.1 – Код K-means

```
import numpy as np

def kmeans(X, k, max_iters=100):
    # Випадкова ініціалізація центроїдів
    centroids = X[np.random.choice(X.shape[0], k,
replace=False)]

    for _ in range(max_iters):
        # Обчислення відстаней до центроїдів
        distances = np.linalg.norm(X[:, np.newaxis] - centroids,
axis=2)
        # Призначення кожної точки до найближчого центроїда
        labels = np.argmin(distances, axis=1)

        # Оновлення центроїдів
        new_centroids = np.array([X[labels == i].mean(axis=0)
for i in range(k)])

        # Умова зупинки
        if np.all(centroids == new_centroids):
            break
        centroids = new_centroids

    return centroids, labels

# Приклад
X = np.array([[1, 2], [1, 4], [1, 0],
              [10, 2], [10, 4], [10, 0]])
centroids, labels = kmeans(X, k=2)

print("Centroids:", centroids)
print("Labels:", labels)
```

Для ймовірнісної кластеризації використовується модель змішування Гауса (Gaussian Mixture Model, GMM). GMM є різновидом моделі очікування-максимізації (Expectation-Maximisation, EM), тобто алгоритмом, який виконує оцінювання максимальної правдоподібності для прихованих змінних. Приховані змінні можуть бути або безпосередньо вимірюваними, або опосередковано виведеними за допомогою математичних моделей. Метою GMM є знаходження максимальної правдоподібності моделі шляхом оптимізації її параметрів.

Припускається, що всі точки даних є частинами суміші нормальних розподілів, параметри яких заздалегідь невідомі. GMM здійснює м’яку

класифікацію, тобто не прив’язує точку даних до єдиного кластеру з фіксованою ймовірністю, а дозволяє їй належати до кількох кластерів одночасно з певними ймовірностями.

Кластери, створені GMM, вважаються такими, що мають форму гаусового (нормального) розподілу, який визначається двома параметрами: середнім значенням (μ) та стандартним відхиленням (σ). Ці два параметри використовуються для максимізації GMM.

Розрахунок GMM за допомогою EM-алгоритму поділяється на два основні етапи: етап E та етап M. Етап E передбачає обчислення ймовірності того, що певна точка (x_i) належить до кожного кластеру (c_k). Етап M відповідає за оновлення значень μ, Σ і Π (де Π представляє щільність точок у розподілі). Обидва етапи виконуються ітеративно, з поступовою оптимізацією параметрів і максимізацією відповідної функції правдоподібності.

Лістинг 1.2 – Код GMM

```
import numpy as np

# Створимо синтетичні дані
np.random.seed(0)
X = np.concatenate([
    np.random.normal(0, 1, 100),
    np.random.normal(5, 1, 100)
])[:, np.newaxis]  # 1D → 2D

# Ініціалізація параметрів
def initialize_params(X, n_components):
    n_samples, _ = X.shape
    means = np.random.choice(X.flatten(), n_components)
    variances = np.ones(n_components)
    weights = np.ones(n_components) / n_components
    return means, variances, weights

# Функція гаусового розподілу
def gaussian(x, mean, var):
    coeff = 1.0 / np.sqrt(2 * np.pi * var)
    exponent = np.exp(-0.5 * ((x - mean) ** 2) / var)
    return coeff * exponent

# EM-алгоритм
def gmm_em(X, n_components=2, n_iters=100, tol=1e-6):
```

```
    n_samples = X.shape[0]
    means, variances, weights = initialize_params(X,
n_components)

    for iteration in range(n_iters):
        # E-step: обчислення ймовірностей
        responsibilities = np.zeros((n_samples, n_components))
        for k in range(n_components):
            responsibilities[:, k] = weights[k] * gaussian(X,
means[k], variances[k])
        responsibilities /= responsibilities.sum(axis=1,
keepdims=True)

        # M-step: оновлення параметрів
        Nk = responsibilities.sum(axis=0)
        weights = Nk / n_samples
        means = (responsibilities * X).sum(axis=0) / Nk
        variances = ((responsibilities * (X - means.reshape(1, -
1))**2).sum(axis=0)) / Nk

        # Лог-правдоподібність (необов’язково для перевірки
зупинки)
        log_likelihood =
np.sum(np.log((responsibilities.sum(axis=1))))
        if iteration > 1 and abs(log_likelihood - prev_ll) <
tol:
            break
        prev_ll = log_likelihood

    return means, variances, weights, responsibilities

# Запуск моделі
means, variances, weights, resp = gmm_em(X, n_components=2)

# Вивід результатів
for i in range(2):
    print(f"Гаус №{i+1}:")
    print(f"  Середнє     = {means[i]:.4f}")
    print(f"  Дисперсія   = {variances[i]:.4f}")
    print(f"  Вага        = {weights[i]:.4f}")
```

Зниження розмірності є ще одним підходом неконтрольованого навчання, при якому кількість ознак (характеристик) у наборі даних зменшується. Хоча в машинному навчанні більша кількість даних зазвичай вважається перевагою, надмірна кількість ознак може ускладнити візуалізацію або обробку даних через високу складність. Цей підхід дозволяє

виділити найважливіші ознаки, усуваючи зайві або випадкові характеристики з набору даних.

Для зменшення кількості вхідних даних без суттєвого впливу на зміст використовуються алгоритми, такі як метод головних компонент (PCA, Principal Component Analysis) та усічене сингулярне розкладання (Truncated SVD, Singular Value Decomposition) [20].

PCA є статистичним лінійним методом, який, зокрема, дозволяє візуалізувати дані шляхом побудови лінійних комбінацій з p випадкових змінних (A1, A2, ...Ap). Існує три ключові властивості, необхідні для коректного застосування PCA:

Властивість 1. Головні компоненти не мають кореляції між собою.

Властивість 2. Перша головна компонента має найвищу дисперсію (коваріацію), друга – другу за величиною дисперсію і т.д.

Властивість 3. Загальна варіація всіх головних компонент дорівнює загальній варіації початкових змінних (A1, A2, ..., Ap).

Коли ці умови виконуються, можна застосувати PCA для зменшення кількості вимірів у даних[2,3]. Важливо правильно визначити кількість головних компонент, оскільки метою є максимальне скорочення розмірності при мінімальних втратах інформації (варіації) у даних.

Лістинг 1.3 – Код PCA

```
import numpy as np
import matplotlib.pyplot as plt

# 1. Генеруємо штучні дані
np.random.seed(0)
X = np.dot(np.random.rand(2, 2), np.random.randn(2, 200)).T

# 2. Центруємо дані (віднімаємо середнє)
X_meaned = X - np.mean(X, axis=0)

# 3. Обчислюємо коваріаційну матрицю
cov_mat = np.cov(X_meaned, rowvar=False)

# 4. Обчислюємо власні значення та власні вектори
eigen_values, eigen_vectors = np.linalg.eigh(cov_mat)
```

```
# 5. Сортуємо власні вектори за спаданням власних значень
sorted_index = np.argsort(eigen_values)[::-1]
sorted_eigenvectors = eigen_vectors[:, sorted_index]
sorted_eigenvalues = eigen_values[sorted_index]

# 6. Вибираємо кількість головних компонент
n_components = 1
eigenvector_subset = sorted_eigenvectors[:, 0:n_components]

# 7. Перетворюємо дані
X_reduced = np.dot(X_meaned, eigenvector_subset)

# 8. Візуалізація
plt.figure(figsize=(8, 5))
plt.scatter(X_meaned[:, 0], X_meaned[:, 1], alpha=0.2,
label='Original Data')
plt.scatter(X_reduced, np.zeros_like(X_reduced), alpha=0.8,
label='PCA Projection')
plt.title("PCA зниження розмірності до 1 компоненти")
plt.legend()
plt.grid(True)
plt.show()
```

Окрім кластеризації та зниження розмірності, у сфері виявлення аномалій існують й інші підходи неконтрольованого навчання, зокрема Isolation Forest (IF) та One-Class Support Vector Machine (OCSVM), які будуть описані в цьому підрозділі.

IF є ансамблевим методом виявлення викидів, який використовує бінарні дерева ізоляції для виявлення аномалій у наборі даних. Ця техніка базується на принципі, що аномалії – це «рідкісні та відмінні» точки, які легше ізолювати порівняно з нормальними спостереженнями. У IF створюється ліс бінарних дерев, які формуються шляхом випадкового вибору підмножин із набору даних, причому кожне дерево вирощується до заданої межі висоти.

Ключ до ефективності IF полягає в особливому методі оцінки аномальності: для кожної точки даних розраховується довжина шляху від кореня до листа в кожному дереві. Коротший шлях вказує на вищу ймовірність того, що точка є аномальною.

Побудова дерев ізоляції в IF керується спеціальним алгоритмом. Тут X позначає вхідний набір даних, t – кількість дерев у лісі, а ψ – розмір

підвибірки, що використовується для побудови кожного дерева. Обмеження висоти дерева, яке обчислюється на основі ψ, забезпечує ефективність процесу ізоляції, дозволяючи ідентифікувати аномалії на основі рівнів ізоляції, що кількісно виражаються через довжини шляхів у деревах ізоляції [4].

OCSVM (One-Class Support Vector Machine) є спеціалізованою версією методу опорних векторів (SVM), призначеною для виявлення аномалій. Цей підхід особливо підходить для наборів даних, де більшість точок вважаються «нормальними», а метою є виявлення викидів або аномалій. На відміну від класичного SVM, який використовується для задач класифікації чи регресії між двома або більше класами, OCSVM зосереджений на відокремленні нормальних точок даних від відхилень шляхом побудови функції прийняття рішень, яка ізолює нормальні дані у просторі ознак.

OCSVM працює шляхом проєктування вхідних точок у простір ознак високої розмірності за допомогою ядрової функції (kernel function), після чого визначає гіперплощину з максимальною відстанню (margin), яка відокремлює початок координат від зображених нормальних точок. Точки, що опиняються ближче до початку координат, розглядаються як аномалії, тоді як ті, що знаходяться по інший бік гіперплощини, вважаються нормальними. Використання ядрової функції дозволяє OCSVM захоплювати складні залежності в даних та ефективно відокремлювати аномалії навіть у випадках, коли межа між нормальними даними та аномаліями не є лінійно відокремлюваною у вихідному просторі ознак.

Ефективність OCSVM залежить від вибору типу ядра (наприклад, RBF, поліноміальне) та його параметрів, а також від параметра ν (nu), який визначає компроміс між часткою аномалій і шириною межі функції рішень. Низьке значення ν створює вузьку межу навколо нормальних даних, підвищуючи чутливість до викидів. Вищі значення ν забезпечують більшу гнучкість, але можуть включати частину аномальних точок у зону, визначену як нормальна.

2 МЕТОДИ ВИЯВЛЕННЯ АНОМАЛІЙ У ЖУРНАЛАХ API ДЛЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ТА НАДІЙНОСТІ ПРОГРАМНИХ СИСТЕМ

2.1 Опис набору даних

Набір даних HTTP Dataset CSIC 2010 [5] був використаний у цій роботі та був виявлений завдяки дослідженню Chakir та ін. [6]. Цей набір містить автоматично згенеровані веб-запити до сайту електронної комерції та був створений в Інституті інформаційної безпеки CSIC (Іспанська національна рада з досліджень). Короткий опис основних характеристик набору даних наведено в таблиці 4.1. У вебзастосунку, використаному для створення даних, користувачі можуть, наприклад, купувати товари, додавати їх у кошик і вводити особисту інформацію.

Загалом набір містить 61 000 записів, з яких 36 000 – це нормальні запити, а 25 000 – аномальні. Кожен запит має мітку, яка вказує, чи є він нормальним, чи аномальним.

Дані були згенеровані Giménez та ін. [58] шляхом кількаетапного процесу. Спочатку було зібрано реальні дані для всіх параметрів вебзастосунку. Автентична інформація, включаючи імена, прізвища, адреси тощо, була отримана з реальних баз даних. Ці значення були розділені та збережені у двох окремих базах даних: одна містила нормальні значення, інша – аномальні, які моделювали шаблони атак. Додатково був складений список усіх публічно доступних сторінок вебзастосунку. Далі для кожної сторінки було згенеровано як нормальні, так і аномальні HTTP-запити. У нормальних запитах параметри заповнювалися випадковими значеннями з нормальної бази, тоді як у запитах, що імітували атаки, використовувалися дані з аномальної бази.

Аномальні запити були розділені на три категорії. Перша – це статичні атаки, які включають запити до прихованих або неіснуючих ресурсів, таких

як застарілі файли, ідентифікатори сесій у переписаних URL, конфігураційні файли, файли за замовчуванням тощо. Друга категорія – динамічні атаки, які передбачають зміну коректних параметрів запиту. До них належать SQL-ін'єкції, ін'єкції символів переносу рядка (CRLF), міжсайтове скриптування (XSS), переповнення буфера тощо. Нарешті, третя категорія – це ненавмисні некоректні запити. Хоча ці запити не є зловмисними за своєю природою, вони відхиляються від нормального функціонування вебзастосунку та мають іншу структуру параметрів, наприклад, телефонний номер, складений із букв.

Таблиця 2.1 – Короткий опис набору даних

Аспект	Значення
Розмір набору даних	61000
Нормальні записи	36000
Аномалії	25000
Розподіл класів	Норма: 59%, Аномалії: 41%
Джерело	Giménez та ін. [58]
Типи даних	Змішані (категоріальні, числові, текстові)

Стовпці, використані з версії набору даних у форматі CSVнаведені в таблиці 2.2. Вони були обрані з урахуванням специфіки завдання неконтрольованого виявлення аномалій у журналах API, за винятком стовпця класифікації, з особливою увагою до кінцевих точок API. Мотивація такого вибіркового включення полягає в характеристиках кожного стовпця, що дозволяють краще зрозуміти поведінку вебзастосунку та аспекти безпеки, які є подібними до характеристик логів API. Водночас у цьому наборі відсутні деякі цінні дані, такі як мітки часу, коди статусів і час відповіді.

Аналіз стовпців *method* та *url* формує основу для розуміння взаємодії між клієнтом і сервером в API-середовищі. URL, частково визначаючи кінцеву точку API, вказує на конкретний ресурс або дію, яка виконується у вебзастосунку. У поєднанні з HTTP-методами запиту, такими як GET або

POST, можна визначити, яку саме дію виконує клієнт на цій точці доступу. Це поєднання методу і кінцевої точки має вирішальне значення для виявлення аномалій. Наприклад, неочікувані POST-запити можуть свідчити про несанкціоновані зміни даних, а різке зростання кількості таких запитів – про витік даних або спробу ін’єкції. Такий цілісний підхід до аналізу взаємодій між методами та точками доступу дозволяє виявити потенційні вразливості безпеки, що підкреслює важливість врахування обох параметрів при пошуку ознак експлуатації.

Таблиця 2.2 – Опис використаних стовпців журналу CSIC

Стовпець	Опис
method	HTTP-метод, що використовується (наприклад, GET, POST).
host	Домен запиту.
accept	Типи контенту, які клієнт може обробити.
content_length	Довжина запиту.
content	Фактичний вміст запиту.
url	URL (кінцева точка API), до якого було зроблено запит.
classification	Мітки, що відповідають поняттю «клас», використовуються для оцінки.

Крім того, включення інформації про *host*, після відповідного кодування для аналізу, дозволяє виявляти запити до потенційно підозрілих хостів або до тих, що часто стають цілями кібератак. Це додає ще один контекстний рівень до аналізу. Аналогічно, аналіз заголовка *accept* (через кодування) дозволяє з’ясувати, які типи MIME запит може обробляти. Аномалії в цих типах можуть свідчити про спроби експлуатації.

Показник *content_length* надає пряму інформацію про розмір тіла запиту або відповіді, що дозволяє виявляти аномальні обсяги переданих даних, які можуть свідчити про спроби атаки, зокрема переповнення буфера. Крім того, аналіз вмісту запиту через похідні ознаки, такі як кількість

символів і шаблони, допомагає виявити як прямі, так і приховані ознаки шкідливих намірів, включаючи SQL-ін’єкції та атаки типу XSS (міжсайтове скриптування).

2.2 Процес виявлення аномалій

У цій роботі методологія базуватиметься на чотириетапній стратегії, яка зображена на рисунку 4.1. Чотири етапи включають: підготовку, виявлення аномалій, отримання результату та оцінку. Суцільні лінії на схемі вказують на кроки, що виконуються під час оцінювання, тоді як пунктирні лінії позначають кроки, які не входять до процедури,.

Передусім, необхідно підготувати відповідні дані, які будуть використані як вхідні для навчання та тестування. Як правило, це або набір даних, що дозволяє розрізняти легітимну та шкідливу активність, або сирі журнали подій. Вхідні дані часто потребують попередньої обробки з використанням методів, які враховують специфіку конкретного набору. Наприклад, IP-адреси мають ієрархічну структуру, портам відповідають числові значення, а протоколи можуть бути представлені як окремі категорії, наприклад, за допомогою енкодерів. Конкретні методи обробки обираються з урахуванням типу алгоритмів виявлення аномалій, які зазвичай класифікуються як контрольовані або неконтрольовані. У цій роботі, як було зазначено раніше, розглядається лише неконтрольоване виявлення аномалій.

Далі ці методи застосовуються до підготовлених даних, у результаті чого генеруються вихідні значення – оцінки або мітки. У межах цієї роботи будуть використовуватися лише оцінки, оскільки мітки переважно застосовуються у задачах класифікації, що тут не є релевантним. Отримані оцінки потім використовуються для оцінювання ефективності всієї процедури. Якщо результати оцінювання будуть незадовільними або з'явиться потенціал для покращення, можливе застосування ітеративного

підходу, який дозволяє налаштувати моделі з метою досягнення оптимальної продуктивності на вибраному наборі даних.

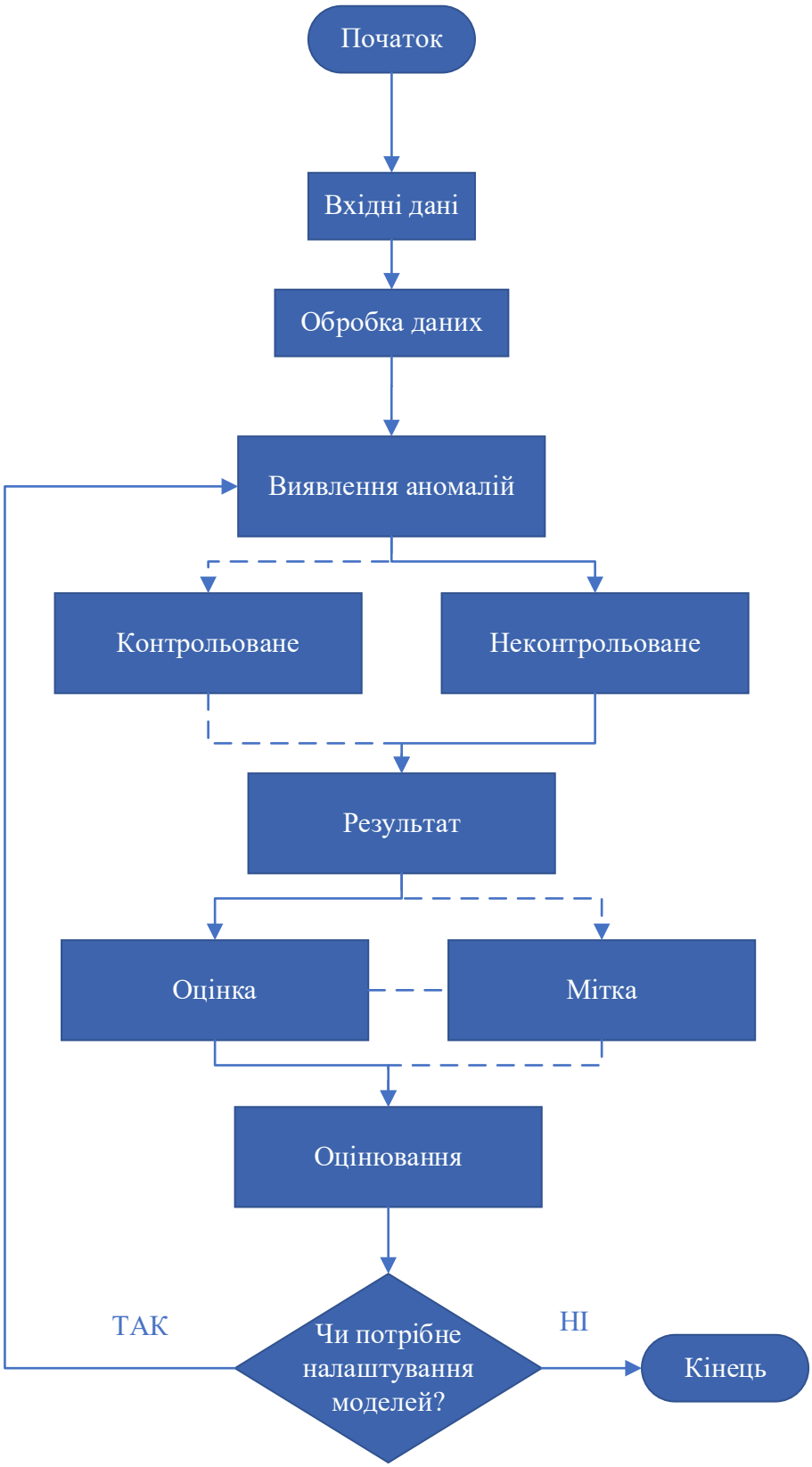

Рисунок 2.1 – Схема запропонованого підходу до виявлення аномалій

Етап підготовки може значно відрізнятися залежно від необхідної попередньої обробки даних. Підготовча фаза включає такі завдання, як збір даних, виділення ознак і іноді нормалізація даних або стандартизація. На рисунку 2.2 проілюстровано відповідні завдання підготовчого етапу, які складаються зі збору журналів, вилучення ознак, нормалізації даних і машинного навчання. Ці кроки було вжито під час неконтрольованого виявлення аномалії. Однак нормалізація і стандартизація не завжди потрібні. Але нормалізація була виконана в цій роботі, щоб потенційно підвищити точність передбачення та візуалізувати деякі моделі для подальшого дослідження.

Рисунок 2.2 – Етапи підготовки

Початкове збирання вхідних даних, що є ключовим завданням у зборі логів, зазвичай коливається між використанням реальних і синтетичних даних. Дефіцит публічно доступних реальних логів API пояснюється низкою факторів. Основними з них є політика конфіденційності компаній, великі обсяги наборів даних, велика кількість кінцевих точок API, характер незбалансованих даних і несумісність у форматах. Ці обмеження підсилюють важливість проєктів, таких як LogHub, який об’єднує 19 наборів логів загальним обсягом понад 77 ГБ, орієнтованих на потреби академічних досліджень і промисловості.

Однак у межах цієї роботи виявлені проблеми унеможливлюють використання логів із LogHub для аналізу логів API. Натомість обрано набір

даних CSIC HTTP 2010який було визнано придатним для інтерпретації як логи API. Незважаючи на його обмеження, саме цей набір був обраний як основний у цій магістерській роботі, що буде детальніше обговорено далі.

Після отримання структурованих логів із вибраного набору даних було виконано початкову обробку даних. Основна причина полягає в тому, що різні стовпці набору містять різні типи даних, зокрема числові та категоріальні у текстовому форматі. Щоб моделі для виявлення аномалій могли коректно працювати, всі стовпці повинні бути переведені в числовий формат. Тому перед тренуванням моделей набір даних було попередньо оброблено для забезпечення оптимального подання ознак і сумісності з алгоритмами машинного навчання для неконтрольованого виявлення аномалій.

На початковому етапі виконувалися операції з очищення даних, як показано в листингу 2.1. Зокрема, це стосувалося атрибута content_length, з якого видалялися нечислові символи, після чого ознаку було перетворено на числовий тип даних. Усі пропущені значення, виявлені під час цієї трансформації, заповнювалися нулями, виходячи з припущення, що відсутнє значення довжини вмісту означає відсутність самого вмісту.

Аналогічний підхід було застосовано до атрибута accept, де витягувалися числові частини, а нечислові або відсутні дані оброблялися шляхом присвоєння типового значення.

Лістинг 2.1 – Очищення даних

```
X['content_length'] = X['content_length'].astype(str)
X['content_length'] = X['content_length'].str.extract(r'(\d+)')
X['content_length'] = pd.to_numeric(X['content_length'],
errors='coerce').fillna(0)

X['accept'] = X['accept'].astype(str)
X['accept'] = X['accept'].str.extract(r'(\d+)')
X['accept'] = pd.to_numeric(X['accept'],
errors='coerce').fillna(1)
```

Після очищення даних було виконано побудову ознак (feature engineering) з метою створення інформативних атрибутів, які могли б відобразити специфіку HTTP-запитів в API. Частину побудованих ознак представлено у листингу 2.2. У цей процес входило створення різноманітних ознак, пов’язаних із URL, зокрема підрахунок кількості крапок, директорій, вбудованих доменних імен, а також визначення довжини URL і наявності підозрілих термінів, що можуть свідчити про зловмисні наміри. Підозрілі терміни, які можуть міститися в URL, оцінювались із урахуванням їхнього потенційного рівня загрози для API.

Аналогічні метрики були обчислені для атрибута content, включаючи підрахунок різних символів і перевірку статусу кодування вмісту. Крім того, категоріальний атрибут method був перетворений у числовий формат за допомогою кодування, що дозволило інтегрувати його в подальші етапи моделювання.

Лістинг 2.2 – Створення ознак

```
X['count_dot_url'] = X['url'].apply(count_dot)

...

X['special_count_url'] =
X['url'].apply(count_special_characters)
X['count_dot_content'] = X['content'].apply(apply_to_content,
function=count_dot)

...

X['is_encoded_content'] = X['content'].apply(apply_to_content,
function=is_encoded)
```

Було проведено розвідувальний аналіз даних з метою глибшого розуміння структури та характеристик набору даних. Цей етап включав виявлення найбільш часто вживаних URL-адрес та підрахунок унікальних значень у різних ознаках, що дозволило сформувати уявлення про різноманітність і розподіл даних.

Нечислові поля, такі як method, host та accept, було перетворено за допомогою кодування міток (label encoding), що означає присвоєння кожному унікальному значенню відповідного числового коду для подальшої обробки моделями машинного навчання, як показано в листингу 2.3.

Атрибут classification, який є бінарним представленням нормальних та аномальних запитів, було виділено як цільову змінну для тренування моделей.

Лістинг 2.3 – Кодування міток

```
lb_make = LabelEncoder()
X["method_enc"] = lb_make.fit_transform(X["method"])
X["host_enc"] = lb_make.fit_transform(X["host"])
X["accept_enc"] = lb_make.fit_transform(X["accept"])
```

У підсумку, на завершальному етапі попередньої обробки було сформовано вдосконалений набір ознак під назвою X_preprocessed, який містить мітки, що відображають виконану попередню обробку та створення ознак, як показано в листингу 2.4. Цей набір включає новостворені ознаки та числово закодовані категоріальні змінні, забезпечуючи насичений інформацією датасет, готовий до тренування моделі для виявлення аномалій.

Пайплайн попередньої обробки було ретельно спроєктовано з урахуванням збереження цілісності даних, що дозволило закласти надійну основу для наступних етапів процесу виявлення аномалій.

Лістинг 2.4 – Мітки, що використовуються після завдання обробки даних

```
labels = [
    'count_dot_url', 'count_dir_url', 'count_embed_domain_url',
'count-http_url',
    'count%_url', 'count2_url', 'count-_url', 'count=_url',
'url_length',
    'hostname_length_url', 'sus_url', 'count-digits_url',
'count-letters_url',
    'number_of_parameters_url', 'is_encoded_url',
'special_count_url',
    'unusual_character_ratio_url', 'method_enc', 'host_enc',
'accept_enc',
    'count_dot_content', 'count%_content', 'count-_content',
```

```
'count=_content',
    'sus_content', 'count_digits_content',
'count_letters_content',
    'content_length', 'is_encoded_content',
'special_count_content'
]
```

Для всіх моделей, окрім Isolation Forest (IF), оскільки вона не потребує використання PCA, дані також були масштабовані, як показано в листингах 2.5 та 2.6.

Лістинг 2.5 – Масштабування + PCA (без розділення на train/test)

```
scaler = StandardScaler()
X_preprocessed_scaled = scaler.fit_transform(X_preprocessed)

pca = PCA(n_components=2, random_state=42)
pca.fit(X_preprocessed_scaled)

X_train_pca = pca.transform(X_preprocessed_scaled)
```

Лістинг 2.6 – Масштабування + PCA (без розділення на train/test)

```
scaler = StandardScaler()
X_train, X_test, y_train, y_test = train_test_split(
    X_preprocessed, y, test_size=0.3, random_state=42
)

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

pca = PCA(n_components=2, random_state=42)
pca.fit(X_train_scaled)

X_train_pca = pca.transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

Фаза виявлення аномалій починається після завершення етапу підготовки, коли дані готові для подачі до моделей виявлення аномалій. Це найбільш прямолінійна частина процесу, оскільки єдиним ручним завданням тут є передача матриці ознак до обраних моделей для тренування та тестування [30, 49]. Основними завданнями цього етапу є вибір моделей та

їхнє застосування до попередньо оброблених даних для отримання результатів.

Після підготовки даних було обрано моделі для неконтрольованого виявлення аномалій. Вибір моделей залежав від кількох факторів: характеристик використовуваного набору даних, результатів, наведених у відповідній літературі, а також від інших релевантних досліджень, що викликають інтерес з точки зору подальшої оцінки.

Було обрано такі моделі: K-means, GMM, IF та OCSVM.

K-means і GMM є методами кластеризації, які вже були оцінені в попередніх дослідженнях із різними результатами. Використання моделей, які вже застосовувалися раніше, дозволяє зробити висновки щодо їхньої ефективності в контексті логів API. Зокрема, ці моделі були застосовані у задачах неконтрольованого виявлення аномалій, оскільки вони є одними з найвідоміших моделей для групування даних у кластери. Для того, щоб зробити результати K-means і GMM інтерпретованими, мати можливість їх візуалізувати, а також зменшити час обчислень для OCSVM через велику кількість ознак у наборі даних, було використано PCA як модель для зниження розмірності.

Крім загальних висновків, для оцінки були також використані моделі IF та OCSVM. Обидві є моделями неконтрольованого виявлення викидів: IF є ансамблевою моделлю, а OCSVM – представником сімейства алгоритмів SVM. IF був обраний через те, що є відомим підходом до неконтрольованого виявлення викидів і раніше демонстрував перспективні результати. OCSVM було обрано завдяки його позитивній оцінці в інших дослідженнях.

Під час першого застосування моделей проводиться мінімальне налаштування параметрів, щоб оцінити базову продуктивність обраних алгоритмів. Моделі та їхні параметри наведено в листингу 2.7. Для забезпечення надійності результатів базових моделей встановлюється фіксоване значення random state, що дозволяє досягти детермінованої поведінки моделей при повторному запуску. Виняток становить модель

OCSVM, оскільки для неї немає потреби у фіксованому random state для відтворення результатів.

Лістинг 2.7 – Базові моделі

```
# Базова модель K-середніх
kmeans = KMeans(n_clusters=2, random_state=42)

# Базова модель GMM (Гаусівська змішана модель)
gmm = GaussianMixture(n_components=2, random_state=42)

# Базова модель Isolation Forest (ліс ізоляції)
iso_forest = IsolationForest(n_estimators=100,
contamination=0.1, random_state=42)

# Базова модель OCSVM (однокласовий SVM)
ocsvm = OneClassSVM()
```

Після початкового запуску моделей було проведено низку аналітичних процедур для визначення оптимальних значень параметрів моделей з урахуванням особливостей набору даних. До таких процедур належали аналіз точки ліктя (elbow point analysis), аналіз інформаційного критерію Акаіке (AIC) і байєсівського інформаційного критерію (BIC), а також оцінка істинно позитивної частки (TPR) та ручний аналіз візуалізацій даних, де це було можливо – зокрема, за допомогою матриць плутанини (confusion matrices) та кривих ROC (Receiver Operating Characteristic) і AUC (Area Under Curve).

Після виконання кількох запусків із налаштуванням параметрів, оптимальні значення для моделей були визначені та наведені в листингу 2.8.

Лістинг 2.8 – Налаштовані моделі

```
# Налаштована модель K-середніх
kmeans = KMeans(n_clusters=5, random_state=42)

# Налаштована модель GMM (Гаусівська змішана модель)
gmm = GaussianMixture(n_components=18, random_state=42)

# Налаштована модель Isolation Forest (ліс ізоляції)
iso_forest = IsolationForest(n_estimators=300,
contamination=0.5, random_state=42)

# Налаштована модель OCSVM (однокласовий SVM)
ocsvm = OneClassSVM(kernel='rbf', nu=0.5, gamma='auto')
```

Коли моделі для неконтрольованого виявлення аномалій обрано і їхні параметри налаштовано, настає етап застосування моделей до попередньо обробленого набору даних і генерування результатів. У межах цієї роботи, під час застосування моделей до датасету, усі моделі, окрім Isolation Forest (IF), використовують масштабовану версію даних.

Причина цього полягає в тому, що масштаб (середнє значення та стандартне відхилення) набору даних не впливає на ефективність IF. Натомість для моделей K-means, GMM і OCSVM, а також для методу зниження розмірності PCA, масштабування суттєво впливає на результати, оскільки ці алгоритми чутливі до масштабів ознак.

3 РЕЗУЛЬТАТИ ЕКСПЕРЕМЕНТУ

3.1 Результати роботи базових моделей

Базові моделі були налаштовані з використанням параметрів за замовчуванням, як представлено в листингу 2.7. Показники ефективності цих моделей наведені в таблиці 3.1, а також порівнюються візуально за допомогою ROC-кривих на рисунку 3.1 та матриць плутанини на рисунку 3.2.

Починаючи з базової моделі GMM, можна відзначити, що вона показала збалансовані результати з коротким часом тренування. Точність (precision) на рівні 0.63 свідчить про здатність моделі достатньо добре виявляти справжні аномалії. Recall у 0.72 демонструє, що модель виявила більшість реальних аномалій. F1-оцінка 0.67 відображає хороший баланс між точністю й повнотою. Значення AUC на рівні 0.76 свідчить про хорошу здатність моделі розрізняти нормальні запити й аномалії. Точність класифікації (accuracy) в 0.71 підтверджує, що GMM може ефективно ідентифікувати різні типи запитів.

K-means продемонстрував більш змішані результати, хоча мав коротший час тренування порівняно з GMM. Його точність 0.52 вказує, що лише приблизно половина виявлених аномалій були дійсно такими. Recall становить 0.62, що означає виявлення значної частки реальних аномалій, але також 38% залишилися невиявленими. F1-оцінка 0.56 підтверджує необхідність подальшого налаштування моделі для досягнення кращого балансу. AUC у 0.58 наближається до випадкового вгадування, що свідчить про слабку роздільну здатність. Хоча точність досягла 0.61, K-means у своїй базовій конфігурації навряд чи є надійним варіантом для критичних завдань виявлення аномалій.

Isolation Forest (IF) швидко обробив вибраний набір даних – лише за 0.18 секунди, що свідчить про його ефективність та придатність для сценаріїв з високими вимогами до швидкості. Хоча модель продемонструвала високий AUC (висока здатність до розрізнення), її F1-оцінка була низькою – лише 0.40. Точність моделі (0.78) свідчить про високу кількість справжніх позитивів (TP) і низьку кількість хибних спрацьовувань (FP), але низький recall (0.27) вказує на проблеми з виявленням усіх справжніх аномалій. Це підтверджується високою кількістю FN (хибно негативних) у матриці плутанини. Таким чином, точність у 0.67 не повністю відображає реальний стан речей через дисбаланс класів у наборі даних.

OCSVM продемонстрував обнадійливі результати, хоч і мав довший час навчання порівняно з іншими моделями. Точність 0.55 означає, що більше половини виявлених аномалій були справжніми. Recall 0.66 демонструє, що модель виявила більшість реальних аномалій. F1-оцінка 0.60 свідчить про збалансованість моделі. AUC у 0.63 вказує на помірну здатність до розрізнення, а точність 0.63 підтверджує, що 63% запитів були класифіковані правильно. Водночас, є простір для вдосконалення моделі, зокрема для зменшення кількості хибно негативних випадків і збільшення TP.

Для K-means та OCSVM важливо враховувати поведінку ROC-кривих при високих порогах, особливо коли FPR (рівень хибнопозитивних) перевищує 0.6–0.7. У цих умовах обидві моделі працюють гірше за випадкове вгадування, що недопустимо для завдань виявлення аномалій.

Для GMM коефіцієнт силуету становив 0.41, що вказує на середню якість кластеризації. Проте значення ARI та NMI були низькими, що свідчить про слабку відповідність між кластеризацією та істинними мітками. Аналіз частки аномалій у кожному кластері показав, що моделі важко чітко розмежувати нормальні та аномальні дані.

K-means мав вищий коефіцієнт силуету – 0.64, що свідчить про кращу якість кластеризації порівняно з GMM. Втім, ARI та NMI залишаються

низькими, що знову ж вказує на недостатню відповідність справжній структурі даних. Незважаючи на кращий силует, K-means також не зміг точно ідентифікувати аномалії, що підтверджується високими частками аномальних точок у кластерах.

Загалом, результати таблиці базової продуктивності показують, що обидві кластеризаційні моделі – K-means і GMM – потребують подальшої оптимізації та налаштування для поліпшення точності кластеризації й виявлення аномалій.

Таблиця 3.1 – Показники ефективності для базових моделей

Модель	P	R	F1	AUC	A	Час (с)
GMM	0.63	0.72	0.67	0.76	0.71	0.35
K-середніх	0.52	0.62	0.56	0.58	0.61	0.12
Isolation Forest	0.78	0.27	0.40	0.84	0.67	0.18
OCSVM	0.55	0.66	0.60	0.63	0.63	32.35

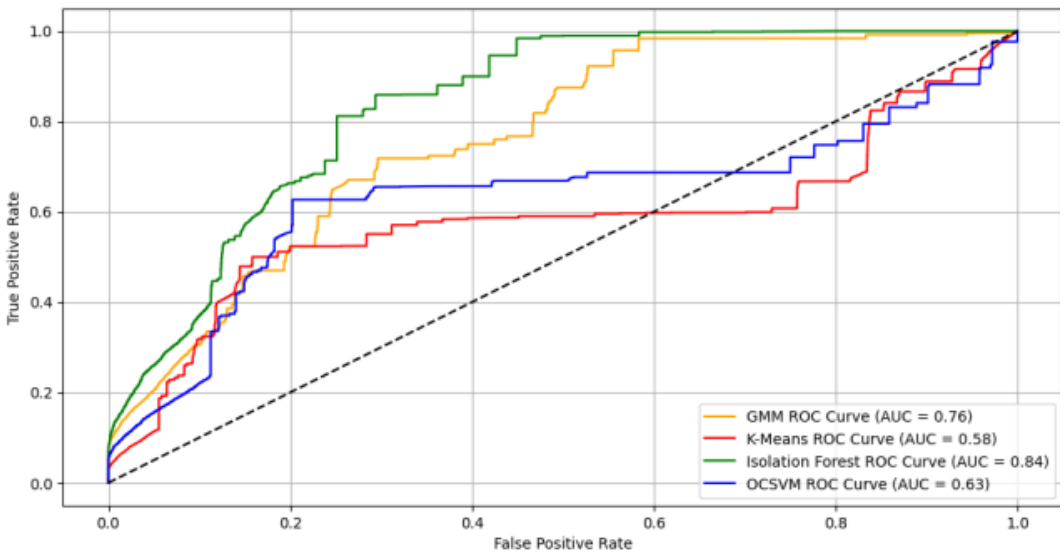

Рисунок 3.1 – Порівняння ROC-кривих для базових моделей

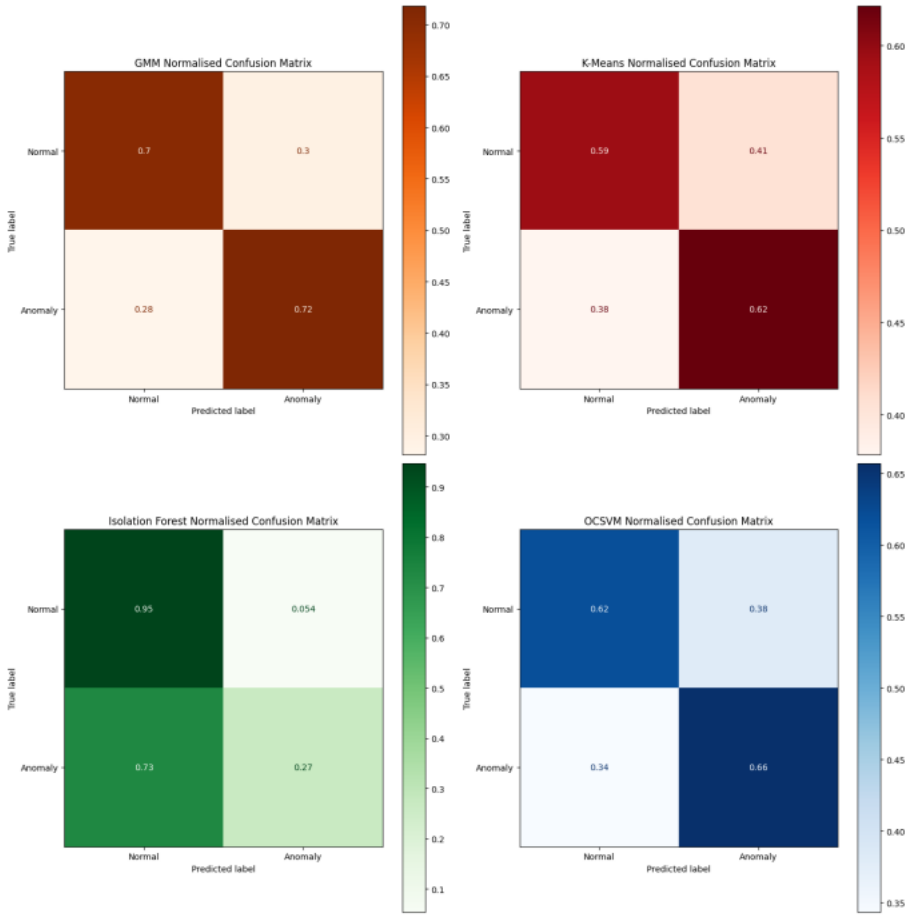

Рисунок 3.2 – Порівняння матриць плутанини для базових моделей

3.2 Результати роботи налаштованих моделей

Налаштовані моделі були сконфігуровані з параметрами, визначеними в процесі налаштування, як показано в листингу 2.8. Показники ефективності для цих моделей наведені в таблиці 3.2 і візуально порівнюються за допомогою ROC-кривих на рисунку 3.3 і матриць плутанини на рисунку 3.4. Час виконання трохи варіюється під час налаштування через варіативність запуску моделей, проте ці відмінності незначні.

Перш за все, налаштована модель GMM демонструє покращену ефективність у виявленні аномалій. Точність на рівні 0.66 свідчить про хорошу здатність моделі правильно класифікувати аномалії, а recall у 0.79 вказує на високу здатність виявляти справжні аномалії. F1-оцінка 0.71 відображає збалансованість між точністю й повнотою, що свідчить про

підвищену ефективність моделі. Значення AUC у 0.83 означає відмінну здатність моделі відрізняти нормальні запити від аномальних. Точність у 0.74 вказує, що більшість запитів були правильно класифіковані. Матриця плутанини підтверджує ці результати: 80% аномалій було ідентифіковано правильно, що свідчить про надійність моделі після налаштування.

Таблиця 3.2 – Показники ефективності для налаштованих моделей

Модель	P (Точність)	R (Повнота)	F1-міра	AUC	A (Точність класифікації)	Час (с)
GMM	0.66	0.79	0.71	0.83	0.74	0.75
K-середніх	0.56	0.65	0.60	0.77	0.65	0.13
Isolation Forest	0.67	0.80	0.73	0.83	0.75	0.70
OCSVM	0.59	0.71	0.64	0.71	0.67	30.84

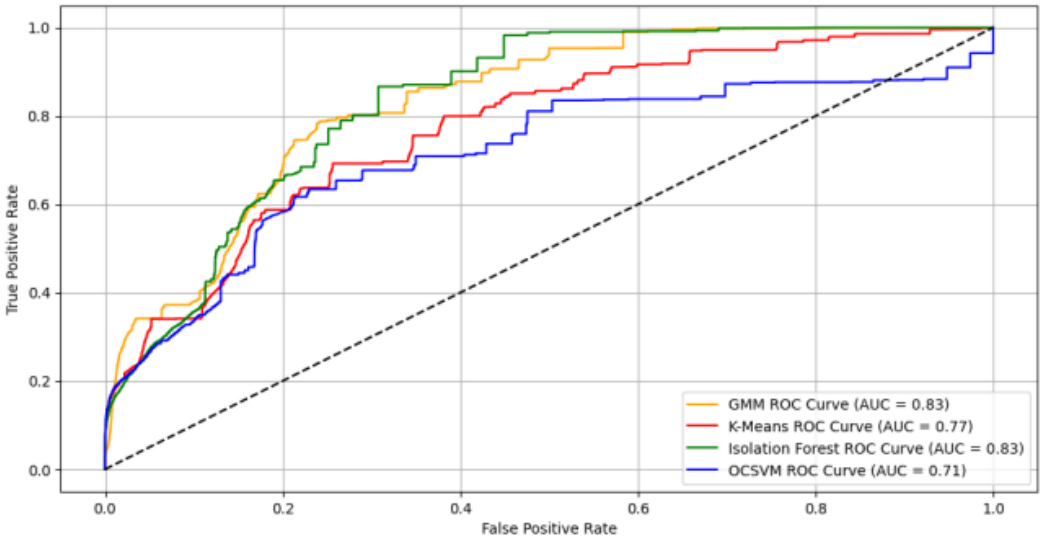

Рисунок 3.3 – Порівняння ROC-кривих для налаштованих моделей

K-means, після налаштування, демонструє помірне, але збалансоване покращення. Точність становить 0.56, що означає, що трохи більше половини передбачених аномалій були правильними. Recall на рівні 0.65 свідчить про

здатність виявити значну частину справжніх аномалій. F1-оцінка 0.60 вказує на середній баланс між точністю та повнотою, хоча результати не вражають. AUC підвищується до 0.77, що означає кращу здатність до розпізнавання. Точність 0.65 свідчить про правильну класифікацію приблизно двох третин запитів, що підтверджується більш яскравими відтінками в матриці плутанини.

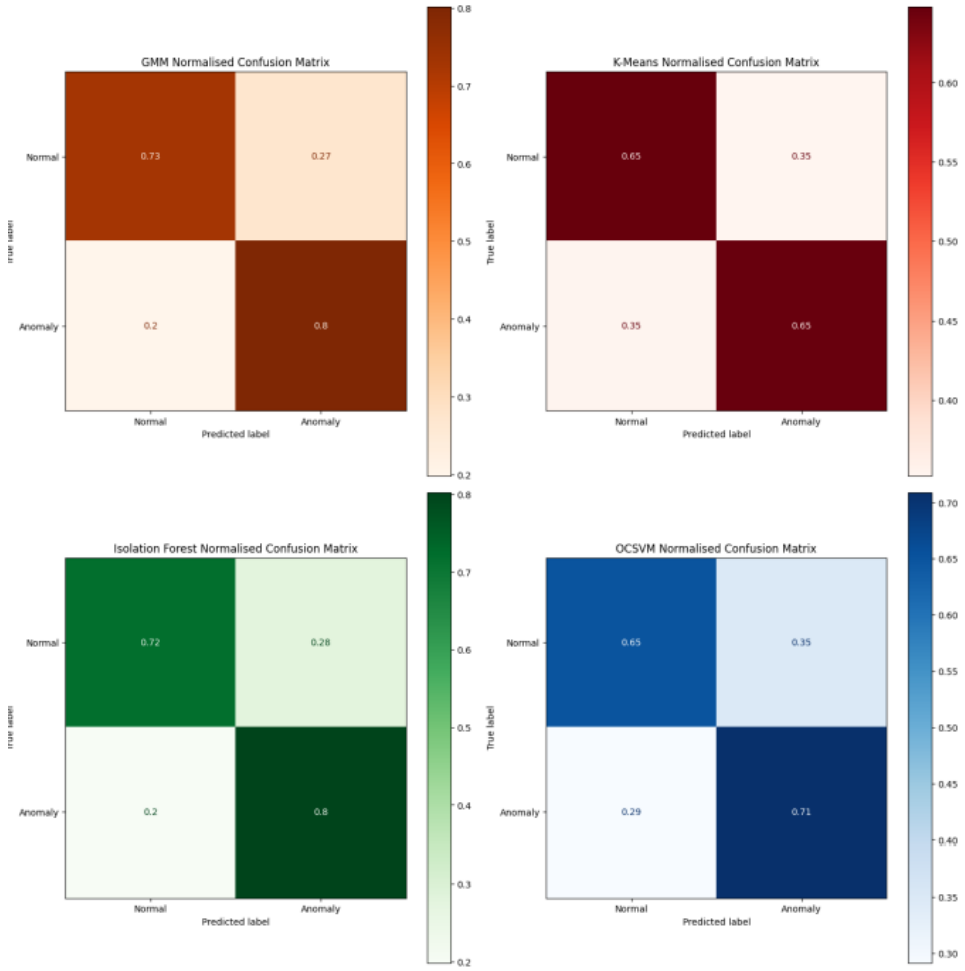

Рисунок 3.4 – Порівняння матриць плутанини для налаштованих моделей

Налаштована модель Isolation Forest (IF) показала значне покращення, особливо у показнику recall. Точність 0.67 і recall 0.80 свідчать про високу здатність виявляти справжні аномалії. Модель досягає F1-оцінки 0.73, що говорить про дуже хороший баланс. AUC у 0.83 демонструє сильну роздільну

здатність, підтверджену також загальною точністю у 0.75. Матриця плутанини показує, що велика частка аномалій була правильно ідентифікована, що підкреслює значне покращення порівняно з базовою моделлю.

Модель OCSVM зберігає стабільні результати. Точність 0.59 означає, що трохи більше половини виявлених аномалій були справжніми. Recall у 0.71 свідчить про здатність моделі виявляти більшість реальних аномалій. F1-оцінка 0.64 демонструє збалансованість між точністю й повнотою. AUC у 0.71 вказує на добру, але не виняткову, здатність моделі класифікувати нормальні та аномальні запити. Точність у 0.67 означає, що дві третини запитів були класифіковані правильно. Хоча OCSVM має найдовший час тренування серед усіх моделей, вона працює стабільно. Проте в порівнянні з іншими моделями, вона має потенціал для подальшої оптимізації, зокрема зменшення обчислювальної складності.

Загалом, продуктивність налаштованих моделей демонструє помітне покращення порівняно з базовими результатами. Згідно з ROC-кривими, усі моделі тепер мають стабільну й бажану поведінку на всьому діапазоні порогових значень — усі криві розташовані вище лінії випадкового вгадування, а AUC перевищує 0.5. Матриці плутанини для кожної моделі свідчать про загальне покращення якості класифікації порівняно з базовими моделями. Ці результати підкреслюють важливість налаштування параметрів для покращення ефективності моделей виявлення аномалій.

Варто також відзначити покращення якості кластеризації у GMM та K-means порівняно з базовими результатами. Для GMM значення коефіцієнта силуету зросло до 0.65, що вказує на краще розділення між кластерами. Значення ARI та NMI також покращилися, що свідчить про кращу відповідність справжній кластерній структурі. Крім того, аналіз виявлених аномалій показав більш збалансований розподіл аномалій по кластерах, деякі з яких досягли повного розділення між нормальними та аномальними точками (1.0).

Аналогічно, K-means також демонструє покращення: коефіцієнт силуету підвищився до 0.73, що означає покращену когерентність і розділення кластерів. Незважаючи на продовження труднощів з точною ідентифікацією аномалій (що видно з частки аномалій у кластерах), K-means суттєво покращив якість кластеризації порівняно з базовими результатами.

У підсумку, результати налаштування моделей демонструють ефективність оптимізації параметрів для покращення кластеризації в K-means та GMM. Покращення у якості кластерів вказує на тісніший зв’язок із реальною структурою даних, що є основою для більш точного виявлення аномалій і глибшого аналізу. Для повного розкриття потенціалу цих алгоритмів у практичних застосуваннях може знадобитися додаткова оптимізація й перевірка.

ВИСНОВКИ

У цій кваліфікаційній роботі було розроблено та досліджено підхід до виявлення аномалій у журналах API з використанням методів неконтрольованого машинного навчання. Основною метою дослідження було підвищення рівня безпеки та надійності програмних систем за допомогою автоматизованого аналізу HTTP-запитів, що надходять до вебзастосунку.

У зв’язку з обмеженою доступністю реальних логів API, як основний набір даних було обрано CSIC HTTP 2010. Цей набір даних було адаптовано до формату, схожого на логи API, із подальшою структуризацією, очищенням та побудовою ознак. Було реалізовано повноцінний процес попередньої обробки, включаючи кодування категоріальних ознак, нормалізацію числових значень та створення нових інформативних характеристик.

Для вирішення поставленого завдання було обрано чотири моделі неконтрольованого навчання: K-means, Gaussian Mixture Model (GMM), Isolation Forest (IF) і One-Class Support Vector Machine (OCSVM). Спочатку моделі було протестовано з базовими параметрами, після чого виконано їх налаштування з метою покращення продуктивності. Налаштування здійснювалося з використанням таких методів, як аналіз точки ліктя, AIC, BIC, а також оцінювання через метрики точності, повноти, F1-міри та площі під ROC-кривою.

У результаті було встановлено, що налаштовані моделі, особливо GMM та IF, значно покращили показники виявлення аномалій. GMM досягла AUC 0.83, точності 0.66 та повноти 0.79, що свідчить про її здатність ефективно ідентифікувати аномальні запити. IF також продемонструвала високу ефективність із F1-мірою 0.73. Інші моделі (K-means, OCSVM) також покращили свої результати, хоча в меншій мірі.

Крім оцінювання класифікаційних метрик, було проведено аналіз якості кластеризації. Значення коефіцієнта силуету, ARI та NMI свідчать про покращену структуру кластерів після налаштування моделей. Це підтверджує доцільність оптимізації параметрів у задачах, пов’язаних з аналізом поведінкових патернів у логах API.

Отже, проведене дослідження підтверджує, що методи неконтрольованого навчання є перспективним і ефективним інструментом для виявлення аномалій у логах API. Вони дозволяють виявляти підозрілу активність без необхідності в попередньому маркуванні даних, що є надзвичайно важливим у реальних умовах, де наявність розмічених даних обмежена або відсутня.

Напрями подальших досліджень можуть включати використання реальних API-журналів з виробничих систем, застосування напівконтрольованого навчання, а також впровадження цих моделей у системи моніторингу в реальному часі.

Загалом, результати роботи демонструють, що оптимізовані моделі машинного навчання можуть бути ефективним засобом забезпечення інформаційної безпеки та виявлення нетипової поведінки у програмних системах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. APItoolkit. API Logs: Your First Line of Defense in Cybersecurity. 2023. URL: https:// apitoolkit.io/blog/api-logs-in-cybersecurity/

2. Mohiuddin Ahmed, Abdun Naser, and Jiankun Hu. “A survey of network anomaly detection techniques”. In: Journal of Network and Computer Applications 60 (2016), pp. 19– 31. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2015.11.016.

3. Carmen Sánchez-Zas, Xavier Larriva-Novo, Víctor A Villagrá, Mario Sanz Rodrigo, and José Ignacio Moreno. “Design and Evaluation of Unsupervised Machine Learning Models for Anomaly Detection in Streaming Cybersecurity Logs”. In: Mathematics 10.21 (2022), p. 4043.

4. Zhangyu Cheng, Chengming Zou, and Jianwei Dong. “Outlier detection using isolation forest and local outlier factor”. In: Proceedings of the conference on research in adaptive and convergent systems. 2019, pp. 161–168.

5. Carmen Torrano Giménez, Alejandro Pérez Villegas, and Gonzalo Álvarez Marañón. HTTP DATASET CSIC 2010. 2012. URL: https : / / www . tic . itefi . csic . es / dataset/

6. Oumaima Chakir, Abdeslam Rehaimi, Yassine Sadqi, El Arbi Abdellaoui Alaoui, Moez Krichen, Gurjot Singh Gaba, and Andrei Gurtov. “An empirical assessment of ensemble methods and traditional machine learning techniques for web-based attack detection in industry 5.0”. In: Journal of King Saud University - Computer and Information Sciences 35.3 (2023), pp. 103–119. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2023.02.009.

7. МАРТОВИЦЬКИЙ, В., СВИРИДОВ, А., АВДЄЄВ, О., ГУДЗИНСЬКИЙ, І., & КОРОТЕЦЬКИЙ, О. (2025). ДОСЛІДЖЕННЯ МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ У API ЖУРНАЛАХ ДЛЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ТА НАДІЙНОСТІ ПРОГРАМНИХ СИСТЕМ. *Вісник Херсонського національного технічного університету*, 2(1 (92)), 142-148.