МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ШОРІН Владислав Сергійович

Програмний модуль виявлення аномалій у смарт-системі обліку на основі ансамблевих методів машинного навчання / Software Module for Anomaly Detection in a Smart Metering System Based on Ensemble Machine Learning Methods

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
В.С. Шорін

Науковий керівник:
д.т.н., професор М.П. Комар

Кваліфікаційну роботу допущено до захисту
«___»_____ 2026 р.

Завідувач кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ШОРІН Владислав Сергійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Програмний модуль виявлення аномалій у смарт-системі обліку на основі ансамблевих методів машинного навчання / Software Module for Anomaly Detection in a Smart Metering System Based on Ensemble Machine Learning Methods
керівник роботи д.т.н., професор М. П. Комар

затверджені наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити
- описати предметну область смарт-обліку та типові джерела спотворень/аномалій у вимірюваннях;
- проаналізувати підходи до виявлення аномалій і обґрунтувати доцільність використання ансамблевих методів;
- визначити вимоги до програмного модуля: вхідні дані, формат результату, критерії якості, вимоги до швидкодії та надійності;
- побудувати конвеєр підготовки даних і формування ознак, придатний до обробки великих обсягів/потоків;
- реалізувати та порівняти ансамблеві моделі, налаштувати процедуру навчання й тестування;
- провести експериментальне оцінювання і сформулювати висновки щодо придатності модуля до практичного використання.
5. Перелік графічного матеріалу в роботі
- архітектура підходу до виявлення аномалій у смарт-обліку.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ В. С. Шорін
підпис

Керівник роботи_____ д.т.н., професор М. П. Комар
підпис

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль виявлення аномалій у смарт-системі обліку на основі ансамблевих методів машинного навчання» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 70 сторінокі містить 8 ілюстрацій, 11 таблиць, 2 додатки та 41 використане джерело.

Метою кваліфікаційної роботи є розроблення програмного модуля виявлення аномалій у смарт-системі обліку на основі ансамблевих методів машинного навчання та оцінювання його ефективності на даних обліку.

Методи досліджень включають методи системного аналізу, статистичної обробки часових рядів, попередньої обробки даних, формування ознак, машинного навчання, ансамблевої класифікації, а також методи моделювання та експериментального оцінювання якості програмного модуля.

Розроблено програмний модуль виявлення аномалій у смарт-системі обліку, який забезпечує приймання даних смарт-лічильників, первинний контроль якості, очищення та підготовку вимірювань, формування ознак на основі часової динаміки споживання, виявлення підозрілих або нетипових записів за допомогою ансамблевих методів машинного навчання, а також фіксацію результатів детектування у вигляді мітки аномалії, оцінки ризику та журналу спрацювань.

Результати дослідження можуть бути використані для створення та вдосконалення програмних засобів моніторингу смарт-систем обліку, автоматизації контролю якості даних, своєчасного виявлення технічних збоїв, нетипових профілів споживання та можливих маніпуляцій із показниками.

Ключові слова: СМАРТ-СИСТЕМА ОБЛІКУ, АНОМАЛІЇ, МАШИННЕ НАВЧАННЯ, АНСАМБЛЕВІ МЕТОДИ, ВЕЛИКІ ДАНІ, ЧАСОВІ РЯДИ, КОНТРОЛЬ ЯКОСТІ ДАНИХ.

ANNOTATION

Qualification work on the topic «Software Module for Anomaly Detection in a Smart Metering System Based on Ensemble Machine Learning Methods» for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 70 pages and it contains 8 figures, 11 tables, 2 annexes and 41 sources.

The purpose of the qualification work is to develop a software module for detecting anomalies in a smart metering system based on ensemble machine learning methods and to evaluate its effectiveness using metering data.

The research methods include system analysis, statistical processing of time series, data preprocessing, feature engineering, machine learning, ensemble classification, as well as methods of modelling and experimental evaluation of the software module quality.

A software module for detecting anomalies in a smart metering system has been developed. It provides the reception of smart meter data, primary quality control, cleaning and preparation of measurements, feature generation based on the temporal dynamics of consumption, detection of suspicious or atypical records using ensemble machine learning methods, as well as recording detection results in the form of an anomaly label, risk score, and alert log.

The research results can be used to create and improve software tools for monitoring smart metering systems, automating data quality control, timely detection of technical failures, atypical consumption profiles, and possible manipulations with readings.

Keywords: SMART METERING SYSTEM, ANOMALIES, MACHINE LEARNING, ENSEMBLE METHODS, BIG DATA, TIME SERIES, DATA QUALITY CONTROL.

ЗМІСТ

Вступ 7
1 Аналіз предметної області та постановка задачі 10
1.1 Смарт-системи обліку та особливості даних 10
1.2 Види аномалій і загрози в даних смарт-обліку 13
1.3 Аналіз існуючих підходів та обґрунтування вибору ансамблевих методів 16
1.4 Постановка задачі та вимоги до програмного модуля 19
2 Методи та проєктування підходу до виявлення аномалій 23
2.1 Архітектура підходу та потік даних 23
2.2 Підготовка даних і формування ознак для виявлення аномалій 26
2.3 Ансамблеві методи машинного навчання для детектування аномалій 29
2.4 Інтерпретованість результатів і пояснення спрацювань моделі 32
3 Реалізація прототипу та експериментальне оцінювання 36
3.1 Програмна реалізація модуля 36
3.2 Дані та методика експериментів 38
3.3 Результати порівняння моделей 41
3.4 Аналіз помилок, обмеження та рекомендації щодо впровадження 45
Висновки 47
Список використаних джерел 49
Додаток А Лістинги програмного забезпечення 53
Додаток Б Апробація результатів роботи 63

ВСТУП

Смарт-системи обліку застосовуються для автоматизованого збору показників споживання електроенергії, газу чи води та подальшого використання цих даних у розрахунках і керуванні інфраструктурою [15]. З технічного погляду такі рішення поєднують лічильники, канали зв’язку, серверні компоненти та програмні підсистеми, що забезпечують приймання, збереження й аналіз вимірювань [36]. Практика впровадження подібних систем демонструє, що ключовим ресурсом стають саме дані: вони надходять потоково, накопичуються у великих обсягах, потребують очищення та контролю якості [14, 33].

Разом із перевагами смарт-облік створює нові ризики. Дані можуть спотворюватися через збої вимірювання, помилки передавання, некоректні налаштування або зловмисні дії. У контексті енергетики окремою проблемою залишається крадіжка ресурсу та інші несанкціоновані втручання, які проявляються у вигляді нетипових профілів споживання [9, 37]. Додатково актуалізуються питання кібербезпеки та приватності в мережах смарт-лічильників, оскільки компрометація компонентів може впливати і на достовірність обліку, і на стійкість інфраструктури [19, 26]. Саме тому потрібні засоби, здатні оперативно виявляти підозрілі відхилення та підтримувати контроль за коректністю показників.

У цій роботі аномалія розуміється як значення або поведінковий шаблон у даних смарт-обліку, що істотно відрізняється від типової “норми” для конкретного споживача, групи споживачів або вузла вимірювання. До практично значущих проявів належать різкі стрибки й провали споживання, аномально тривалі нульові значення, нетипова періодичність, а також ознаки можливих маніпуляцій, що імітують “правдоподібні” показники [37]. Виявлення таких ситуацій доцільно реалізовувати у вигляді окремого програмного модуля, який отримує потік вимірювань, формує ознаки, застосовує модель та видає результат у форматі повідомлення/мітки ризику.

Перспективним напрямом для цієї задачі є застосування методів

машинного навчання, зокрема ансамблевих. Ансамблі на основі дерев рішень (Random Forest, градієнтний бустинг тощо) є практичними для табличних даних і часто демонструють кращу стійкість до шуму та випадкових відхилень, ніж одиничні моделі [2, 5, 9]. Водночас для смарт-обліку важливими є не лише показники точності, а й придатність до впровадження: стабільність роботи на великих потоках даних, прийнятна затримка обробки та зрозуміле пояснення спрацювань [14, 33]. Тому інженерна частина (підготовка даних, потокова обробка, контроль якості, журналювання) має розглядатися разом із побудовою моделей [16, 30].

Метою кваліфікаційної роботи є розроблення програмного модуля виявлення аномалій у смарт-системі обліку на основі ансамблевих методів машинного навчання та оцінювання його ефективності на даних обліку.

Для досягнення мети необхідно розв’язати такі завдання:

– описати предметну область смарт-обліку та типові джерела спотворень/аномалій у вимірюваннях;

– проаналізувати підходи до виявлення аномалій і обґрунтувати доцільність використання ансамблевих методів;

– визначити вимоги до програмного модуля: вхідні дані, формат результату, критерії якості, вимоги до швидкодії та надійності;

– побудувати конвеєр підготовки даних і формування ознак, придатний до обробки великих обсягів/потоків;

– реалізувати та порівняти ансамблеві моделі, налаштувати процедуру навчання й тестування;

– провести експериментальне оцінювання і сформулювати висновки щодо придатності модуля до практичного використання.

Об’єктом дослідження є процеси збору та аналізу даних у смарт-системах обліку.

Предметом дослідження є методи та програмні засоби виявлення аномалій у даних смарт-обліку з використанням ансамблевих моделей машинного навчання.

Практичне значення роботи полягає у створенні прототипу програмного

модуля, який може підвищити достовірність даних обліку та підтримати оперативне реагування на підозрілі відхилення, зокрема пов’язані з можливими маніпуляціями або технічними збоями.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах IV Всеукраїнської науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі» (ІКСМ), м. Тернопіль, ЗУНУ, 20 травня 2026 р. (додаток Б).

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Смарт-системи обліку та особливості даних

Смарт-система обліку (smart metering) – це комплекс технічних і програмних засобів, який забезпечує автоматичне зчитування показників лічильників, передавання даних до центру обробки та подальше використання цих даних для розрахунків і аналітики [15, 19]. На відміну від традиційного обліку, де показники збираються періодично вручну, у смарт-обліку вимірювання надходять регулярно (наприклад, щогодини або частіше), що формує потік даних та підвищує вимоги до їхнього зберігання і контролю якості [33].

У типових архітектурах смарт-обліку виділяються такі елементи: вузол вимірювання (лічильник), канал зв’язку, серверна частина (збирання та зберігання), а також підсистема аналізу (перевірка коректності, виявлення аномалій, формування звітів) [15, 36]. У промислових та енергетичних сценаріях додаються вимоги до кібербезпеки, оскільки підміна або викривлення даних прямо впливають на точність нарахувань та керування мережевими режимами [19, 26].

На рисунку 1.1 подано узагальнену схему руху даних у смарт-системі обліку – від лічильника до модуля аналітики.

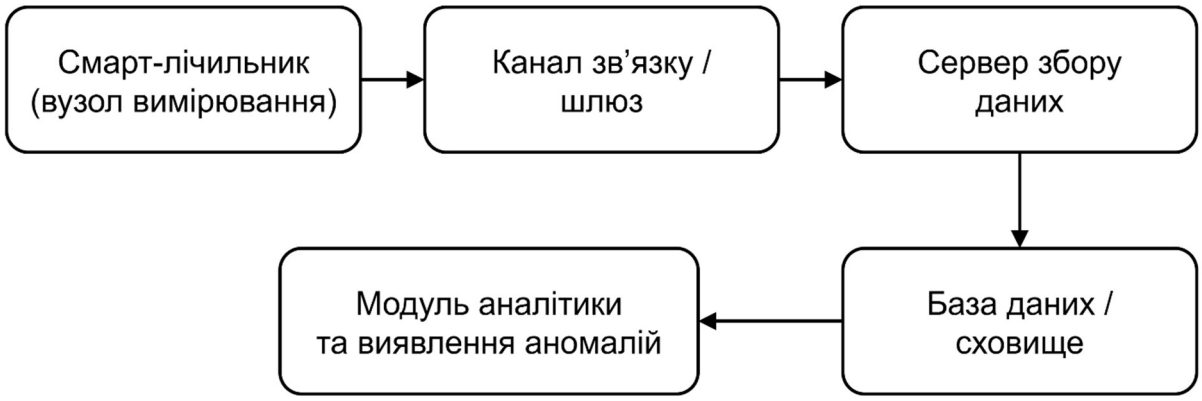

Рисунок 1.1 – Узагальнена схема смарт-системи обліку [15, 36]

Практика проєктування систем вимірювання показує, що навіть за коректної апаратної частини значна частина проблем виникає на рівні даних:

вони можуть надходити із затримками, містити пропуски, дублювання або нетипові значення [33]. Для систем, що працюють у великих масштабах, ці проблеми ускладнюються обсягами та швидкістю надходження інформації, тобто фактично набувають рис “big data” (великий обсяг, різноманітність, швидкість) [14, 33]. Окремі дослідження також підкреслюють роль аналітики великих даних у побудові сучасних вимірювальних систем та підвищенні ефективності їх роботи [1, 29].

Особливість даних смарт-обліку полягає в тому, що вони мають часову природу. Для кожного споживача (або точки обліку) формується часовий ряд, де показники залежать від режиму дня, дня тижня, сезону, а також від індивідуальних звичок споживання [15]. Через це “норма” для різних користувачів може істотно відрізнятися, а порівняння лише за абсолютними значеннями часто є некоректним. У реальних системах доречно аналізувати не тільки “скільки спожито”, а й як саме змінюється споживання: темп зростання/спаду, повторюваність, стабільність, частку нульових значень тощо [33].

Джерела викривлень у даних доцільно поділити на дві групи. Перша група пов’язана з технічними причинами: несправності датчика, нестабільний канал зв’язку, некоректна синхронізація часу, помилки інтеграції між підсистемами [33]. Друга група стосується навмисних впливів, коли дані можуть бути змінені з метою приховати реальне споживання або створити “правдоподібну” картину для обходу контролю [19]. У літературі розглядаються підходи до протидії крадіжкам та маніпуляціям, зокрема фільтраційні та поведінкові методи, які намагаються виявити підозрілу динаміку показників [37]. Це підкреслює актуальність задачі автоматичного виявлення аномалій як елемента захисту та контролю якості даних.

У таблиці 1.1 подано приклади типових полів у записах смарт-обліку та поширені проблеми, що виникають під час збору й передавання.

Таблиця 1.1 – Типові дані смарт-обліку та поширені проблеми якості

Категорія даних	Приклад поля	Короткий зміст	Типові проблеми
Ідентифікація	meter_id, customer_id	ідентифікатор лічильника/споживача	дублікати, помилки відповідності
Час	timestamp	час вимірювання	зсув часу, пропуски інтервалів
Показник споживання	consumption (kWh)	величина за інтервал	нетипові стрибки/провали, нульові значення
Службові параметри	status_code, voltage (за наявності)	технічні стани/якість вимірювання	некоректні коди, шум у вимірюваннях
Канал/маршрут	gateway_id, rssi (за наявності)	інформація про передавання	затримки, втрати пакетів, повторні надсилання

Зауважені особливості безпосередньо впливають на вимоги до програмного модуля виявлення аномалій. По-перше, модуль має працювати з часовими рядами та враховувати індивідуальність профілів споживання. По-друге, необхідна стійкість до шуму, пропусків і нерівномірності надходження даних, що є типовими для реальних інформаційних систем. По-третє, з огляду на кіберризики в смарт-мережах, результати детектування бажано інтегрувати з процедурами моніторингу та реагування, орієнтованими на безпеку. Саме тому в подальших підрозділах доцільно розглянути види аномалій детальніше та обґрунтувати вибір ансамблевих методів як практичного інструменту їх виявлення.

1.2 Види аномалій і загрози в даних смарт-обліку

Дані смарт-обліку мають високу практичну цінність, оскільки на їх основі виконуються нарахування, аналіз споживання та приймаються управлінські рішення [15, 19]. Водночас саме ці дані часто є “вразливою ланкою”: навіть невеликі спотворення здатні накопичуватися і призводити до помітних похибок у підсумках. З цієї причини у смарт-системах обліку важливо відрізняти технічні збої від потенційно небезпечних аномалій, пов’язаних із втручанням або шахрайськими діями [19, 26].

У цій роботі аномалією вважається відхилення в даних, яке суперечить типовій поведінці споживання або очікуваним технічним умовам роботи лічильника. Важливий момент полягає в тому, що “норма” не є сталою: на неї впливають сезонність, графік роботи споживача, аварійні ситуації, зміни обладнання та інші фактори [15]. Тому коректне виявлення аномалій потребує аналізу не лише окремого значення, а й динаміки (контексту у часі) та властивостей потоку даних (повнота, затримки, повтори) [33].

З практичної точки зору доцільно виділити щонайменше три групи аномалій: дані-аномалії, поведінкові аномалії та атаки/маніпуляції. Такий поділ зручний, оскільки кожна група має різні причини виникнення і потребує різного реагування.

1) Аномалії якості даних. Це відхилення, що виникають через помилки вимірювання або передавання: пропуски вимірювань, дублікати, “стрибки” часу, зсув часових міток, некоректні коди статусу, поява неможливих значень (від’ємне споживання, неприродні піки) [33]. Такі ситуації часто не є ознакою атаки, однак вони спотворюють аналіз і можуть маскувати реальні проблеми. У великих системах, де дані надходять потоково, навіть невеликий відсоток помилок здатний дати значний абсолютний обсяг “поганих” записів, що потребує автоматизованої фільтрації та контролю [14, 33].

2) Поведінкові аномалії. Ця група пов’язана із нетиповою динамікою споживання: різкі піки або провали без очевидних причин, нетипова періодичність, тривале “плато” на одному рівні, надмірна регулярність, яка

виглядає штучно, або незвично часті переходи між нульовим і високим споживанням [37]. Такі відхилення можуть бути викликані реальними подіями (поломка обладнання, переїзд, зміна режиму роботи), але також можуть бути індикаторами маніпуляцій. Саме тому виявлення поведінкових аномалій слід розглядати як задачу з імовірнісним висновком: результат має відображати рівень підозрілості, а не “абсолютний вирок” [9, 19].

3) Аномалії, пов’язані з втручанням і крадіжками. Для енергетичних мереж типовим прикладом є заниження показників через несанкціоновані зміни (фізичні або програмні) та інші сценарії шахрайства [9]. У літературі пропонуються підходи, які намагаються розпізнавати такі випадки через аналіз динаміки, зокрема фільтраційні методи та поведінкову ідентифікацію “підозрілих” профілів [37]. На відміну від звичайних збоїв, такі аномалії часто намагаються виглядати правдоподібно, тому ключовим стає використання більш стійких методів аналізу, зокрема ансамблевих моделей, здатних уловлювати складні поєднання ознак [2, 9].

Загрози для смарт-обліку умовно можна поділити на два рівні: рівень даних і рівень інфраструктури.

На рівні даних основний ризик полягає у втраті достовірності вимірювань. Це проявляється у фінансових збитках, некоректній аналітиці попиту, хибних прогнозах навантаження та, як наслідок, у зниженні ефективності керування мережею [19]. На рівні інфраструктури важливими стають питання кібербезпеки: компрометація елементів мережі лічильників, підміна повідомлень, несанкціонований доступ до вузлів, що може впливати не лише на облік, а й на стійкість енергетичної інфраструктури загалом [19, 26]. У рекомендаціях з кібербезпеки для smart grid підкреслюється необхідність постійного моніторингу, сегментації доступів і виявлення нетипових подій як частини системного захисту [26].

Варто зазначити, що практична система детектування аномалій має враховувати компроміс між чутливістю та кількістю хибних спрацювань. Надто “чутливий” детектор створює перевантаження повідомленнями, що знижує довіру до системи й ускладнює реагування. Натомість надто “обережний”

детектор пропускає підозрілі випадки, які й становлять найбільшу загрозу [33]. Саме тому в подальших розділах доцільно опиратися на ансамблеві методи, які зазвичай дають більш стабільну якість на шумних табличних даних, а також дозволяють налаштовувати поріг прийняття рішення залежно від вимог експлуатації [2, 5, 9].

Для узагальнення типів аномалій та їх можливих причин у таблиці 1.2 подано класифікацію, зручну для подальшого проєктування ознак та сценаріїв експериментального оцінювання.

Таблиця 1.2 – Типи аномалій у смарт-обліку та можливі причини

Тип аномалії	Типові прояви в даних	Ймовірні причини	Потенційні наслідки
Пропуски / затримки	відсутні інтервали, “дірки” у часовому ряді	проблеми каналу, збої збору	неповні нарахування, перекоси аналітики
Дублікати	повтори одного вимірювання	повторна передача, помилки інтеграції	завищення підсумків, помилки статистики
Неможливі значення	від’ємні, надто великі піки	збій датчика, помилки перетворення	хибні тривоги, некоректні рішення
Нетипова динаміка	різкі стрибки/провали, “плато”	зміна режиму, поломка, аварія	потреба перевірки, ремонт/діагностика
Підозрілі шаблони	“надто рівно”, дивні повтори	маніпуляції, крадіжка	фінансові втрати, ризики безпеки

Таким чином, задача виявлення аномалій у смарт-обліку має як технічний, так і безпековий зміст. З одного боку, вона спрямована на підвищення якості даних і надійності вимірювань. З іншого боку, виявлення підозрілих відхилень є складовою ширшої системи кіберзахисту та управління ризиками для smart grid.

У наступному підрозділі буде розглянуто підходи до детектування аномалій і обґрунтовано вибір ансамблевих методів машинного навчання як практичного інструменту реалізації програмного модуля.

1.3 Аналіз існуючих підходів та обґрунтування вибору ансамблевих методів

Виявлення аномалій у смарт-системах обліку належить до задач, де “ідеального” універсального методу не існує. Причина полягає у поєднанні кількох факторів: дані надходять у вигляді часових рядів, мають сезонність і різну “норму” для різних споживачів, містять пропуски та шум, а також можуть включати навмисні викривлення [15, 19, 33]. Тому на практиці використовують набір підходів – від простих правил до моделей машинного навчання, які здатні узагальнювати закономірності та виявляти нетипові патерни [9].

1) Правила, порогові перевірки та фільтрація.

Найпростішим підходом є правила “якщо–то”: обмеження на допустимі значення, перевірка різких стрибків, контроль нульового споживання понад певний час, контроль дублювань і часових зсувів [33]. Подібні методи є зрозумілими, швидкими та придатними для первинної санітарної обробки потоку. У прикладних роботах також зустрічаються фільтраційні рішення, спрямовані на зменшення впливу крадіжок або маніпуляцій через аналіз динаміки та “поведінкових” ознак [37].

Разом із тим правила мають суттєве обмеження: вони погано масштабуються на різні типи споживачів і різні умови. Те, що є “аномалією” для одного профілю, може бути нормальною ситуацією для іншого. Крім того, зловмисні сценарії часто намагаються виглядати правдоподібно, тобто “вписуватися” у прості пороги [19]. Тому правила доцільно розглядати як базовий шар, але не як самодостатній механізм детектування.

2) Статистичні методи та моделі часових рядів.

Другий клас підходів базується на статистиці: контроль середнього та дисперсії, z-score, контроль відхилення від ковзного середнього, аналіз

сезонності та залишків. Такі методи корисні як пояснювані й прості у реалізації, особливо для сигналів з відносно стабільною структурою [33]. Проте у смарт-обліку часто виникають складні випадки: комбінація сезонності, різних режимів (робочий/вихідний), “дрейфу” поведінки користувача (зміна звичок, зміна обладнання). У таких ситуаціях статистичні пороги або дають надто багато хибних спрацювань, або, навпаки, стають занадто “поблажливими” [15, 33].

Окремо слід враховувати явище concept drift – поступову або різку зміну розподілу даних у часі. Для смарт-обліку це типовий випадок: профіль споживання може змінюватися сезонно, після ремонту, через зміну тарифів або поведінки користувача [8]. Отже, методи, що не адаптуються до змін, з часом деградують за якістю.

3) Підходи машинного навчання: класифікація, кластеризація, детектування “викидів”.

Методи машинного навчання використовують для двох сценаріїв:

– навчання з учителем, коли є мітки “норма/аномалія” або типи аномалій;

– навчання без учителя, коли шукаються нетипові точки або кластери без явної розмітки.

У задачах, пов’язаних з імовірною крадіжкою ресурсу, часто застосовують класифікаційні моделі та порівнюють різні ML-підходи, зокрема ансамблі [9]. Для задач без розмітки популярним baseline є Isolation Forest, який спеціально орієнтований на пошук “ізольованих” спостережень у просторі ознак [20]. Практична перевага ML підходів полягає в тому, що вони можуть враховувати багато ознак одночасно (наприклад, агрегати за різними вікнами часу, індикатори стабільності, частку нульових значень), тобто “вчаться” на структурі даних, а не на одному правилі [33].

Водночас ML-рішення потребують уважної організації експериментів і контролю якості: коректного поділу даних на навчальну та тестову вибірки, добору метрик, налаштування порога рішення, а також контролю деградації моделі у часі через drift [8, 33]. Саме тому доцільність вибору конкретного сімейства моделей визначається не лише точністю, а й стабільністю, інженерною

простотою та здатністю працювати у виробничих умовах.

4) Обґрунтування вибору ансамблевих методів для програмного модуля.

Для поставленої теми найбільш придатним є клас ансамблевих методів на основі дерев рішень, оскільки він поєднує достатню якість і практичність реалізації. Ансамбль у загальному розумінні – це поєднання багатьох простіших моделей, рішення яких агрегується у фінальний результат. Такий підхід зменшує ризик "випадкових" помилок окремого алгоритму і підвищує стабільність на шумних даних [2].

Random Forest є одним із базових ансамблевих методів, який будує багато дерев рішень на випадкових підвибірках даних і ознак, а потім усереднює їхні рішення [2]. Для смарт-обліку це важливо з двох причин. По-перше, дані нерідко містять шум і пропуски, а ансамбль зменшує чутливість до окремих "поганих" вимірювань. По-друге, RF добре працює з табличними ознаками, які природно формуються з часових рядів (агрегати за вікнами, прирости, частки, індикатори стабільності) [33].

Другий практично значущий напрям – градієнтний бустинг дерев рішень. Його ідея полягає в послідовному додаванні дерев, які виправляють помилки попередніх, що часто дає високу точність на структурованих даних [7]. У прикладних системах широко використовуються реалізації XGBoost та LightGBM, які орієнтовані на ефективність і масштабованість [5, 18]. Для задач детектування аномалій це корисно тоді, коли різниця між "нормою" і "аномалією" визначається складною комбінацією ознак, а не одним показником [9].

Важливим аргументом на користь ансамблів є також можливість інтерпретації. Навіть коли модель є складною, для деревових ансамблів існують методи пояснення (наприклад, SHAP), що дозволяють показати, які ознаки найбільше вплинули на конкретне спрацювання [22]. Для прикладної системи обліку це суттєво: оператору або аналітику потрібне не тільки "попередження", а й зрозумілий сигнал, чому випадок вважається підозрілим.

Окремо слід зазначити інженерний аспект. Смарт-облік часто має ознаки "big data" через кількість лічильників і регулярність вимірювань [14]. Це означає,

що метод має бути відносно економним обчислювально та придатним до пакетної або потокової обробки. У порівнянні з багатьма глибокими нейромережевими моделями, деревові ансамблі зазвичай простіші у навчанні та впровадженні, добре працюють з обмеженим обсягом ознак і можуть інтегруватися у стандартні конвеєри обробки даних [28, 33]. У разі необхідності потокового режиму дані можуть надходити через брокер повідомлень та оброблятися стримінговими компонентами, що відповідає типовим архітектурам big data-сервісів [16, 30, 33].

Таким чином, аналіз підходів дозволяє сформулювати практичний висновок: правила й статистика є корисними як базова перевірка якості даних, але для стійкого виявлення поведінкових і потенційно шахрайських аномалій доцільно застосувати ансамблеві методи машинного навчання. Вони забезпечують компроміс між якістю, стійкістю до шуму та можливістю пояснення результатів, що є критично важливим для програмного модуля, орієнтованого на реальні умови смарт-обліку.

1.4 Постановка задачі та вимоги до програмного модуля

Результати аналізу предметної області (підрозділ 1.1) та класифікації аномалій і загроз (підрозділ 1.2) свідчать, що дані смарт-обліку є потоковими, часовими та схильними до помилок як технічного, так і безпекового характеру [15, 19, 26, 33]. За таких умов завдання виявлення аномалій доцільно формулювати не як одноразову “перевірку файлу”, а як програмний модуль, який може бути вбудований у контур збору та аналізу вимірювань і працювати з регулярними надходженнями показників [33]. Оскільки виявлення аномалій часто використовується як елемент моніторингу та реагування, важливо заздалегідь визначити, що саме вважати “аномалією”, які дані є на вході, який формат результату потрібен на виході, а також якими метриками оцінюватиметься ефективність рішення [26, 33].

Задача кваліфікаційної роботи полягає в тому, щоб розробити прототип програмного модуля, який на основі даних смарт-лічильників автоматично

визначає підозрілі або нетипові вимірювання/інтервали та формує рішення про наявність аномалії. Враховуючи поширені сценарії викривлення даних, модуль має бути орієнтований як на виявлення збоїв якості даних (пропуски, дублікати, неможливі значення), так і на виявлення поведінкових відхилень, що можуть бути пов’язані з маніпуляціями або крадіжками ресурсу [9, 19, 37].

У прикладному формулюванні задача може бути подана у вигляді відображення: вхідні дані → ознаки → модель → вихідне рішення, де вихідне рішення – це мітка “аномалія/норма” або числова оцінка ризику, за якою встановлюється поріг спрацювання [2, 33]. З огляду на практику застосування ML в енергетиці, як базове ядро детектування пропонується використовувати ансамблеві методи (зокрема Random Forest або градієнтний бустинг), які демонструють стійкість на шумних табличних даних і добре поєднуються з інженерним конвеєром підготовки ознак [2, 5, 7, 9, 18].

Вхідними даними модуля вважаються вимірювання смарт-обліку, що надходять у вигляді записів з мінімальним набором полів:

– ідентифікатор лічильника/точки обліку (meter_id);

– час вимірювання (timestamp);

– значення споживання за інтервал (наприклад, kWh);

– за наявності – службові поля (коди стану, технічні параметри, ознаки якості каналу) [33, 36].

Дані можуть надходити пакетно (файл/таблиця) або потоково (через брокер повідомлень). Оскільки смарт-облік у великих системах має риси “big data” (швидкість надходження і обсяг), модуль має допускати обробку серій вимірювань без суттєвих затримок [14, 33]. Архітектурно джерела даних у таких системах часто інтегруються через сервісні компоненти, що відповідає типовим підходам до побудови сервісної архітектури для великих даних [33]. За потреби потокового режиму допускається використання інструментів стримінгу та брокерів повідомлень, що є стандартними рішеннями у практиці обробки потоків [16, 30].

До основних функцій програмного модуля виявлення аномалій доцільно віднести:

1. Приймання та первинна перевірка даних. Модуль має зчитувати записи, перевіряти їхню структуру, обробляти пропуски та очевидні помилки (наприклад, некоректний формат часу) [33].

2. Підготовка даних і формування ознак. Для часових рядів споживання мають формуватися ознаки, що відображають динаміку та контекст (агрегати за вікнами часу, прирости, стабільність, частка нульових значень тощо) [33]. Саме цей етап визначає “якість інформації”, яку отримує модель, і часто критично впливає на результат.

3. Детектування аномалій на основі ансамблевої моделі. Модуль має виконувати інференс моделі та повертати:

– мітку (аномалія/норма) або клас аномалії (за наявності такої постановки),

– числову оцінку (ймовірність/скор) для налаштування порога [2, 5, 7]. У якості базових моделей доцільно розглядати Random Forest та бустингові дерева як основні конкуренти для порівняння [2, 5, 18].

4. Формування вихідних повідомлень та журналювання. Модуль має зберігати інформацію про спрацювання (час, meter_id, скор, ключові ознаки) та забезпечувати можливість інтеграції з системою сповіщень [33]. Для прикладної експлуатації це пов’язано з вимогами безпеки та аудиту подій [26].

5. Пояснення спрацювань (опційно, але бажано). Щоб знизити кількість “незрозумілих” тривог, корисно надавати пояснення: які ознаки найбільше вплинули на рішення моделі. Для ансамблевих моделей це може бути реалізовано через підходи інтерпретації, зокрема SHAP [22].

Нефункціональні вимоги визначають придатність модуля для реального використання:

1. Точність та стабільність. Модуль має забезпечувати прийнятний баланс між виявленням аномалій та кількістю хибних спрацювань, оскільки перевантаження оператора тривогами знижує практичну цінність системи [33].

2. Швидкодія та масштабованість. Обробка повинна виконуватися з прийнятною затримкою для заданого обсягу даних. Для великих потоків важливі пакетна обробка та можливість горизонтального масштабування [14, 33].

3. Надійність і відтворюваність. Рішення має бути відтворюваним: фіксована конфігурація ознак, версійність моделі, контроль параметрів навчання, коректний поділ вибірок [33].

4. Безпека обробки та контроль доступу. Дані обліку є чутливими, тому обробка має враховувати вимоги кібербезпеки, принаймні на рівні організації доступів, журналювання та контролю підозрілих подій [26].

5. Адаптація до змін у даних. Оскільки поведінка споживання змінюється, в перспективі необхідно враховувати drift і можливість оновлення моделі або переналаштування порогів [8].

Оцінювання ефективності модуля доцільно проводити за двома групами показників:

1. Якість детектування. Для задачі “аномалія/норма” стандартно використовуються Precision, Recall та F1-міра, а також оцінка кількості хибних спрацювань (False Alarm Rate) [33]. Для порогових рішень корисним є аналіз залежності метрик від вибраного порога, оскільки різні сценарії експлуатації можуть вимагати різного рівня чутливості.

2. Обчислювальні показники. Оцінюються затримка обробки запису/пакета та пропускна здатність (кількість записів за одиницю часу) [14, 33]. Для потокових сценаріїв це є критичним, оскільки модуль має працювати у режимі постійного надходження даних.

Таким чином, постановка задачі в межах кваліфікаційної роботи зводиться до розроблення та обґрунтування програмного модуля, який здатний виявляти аномалії в даних смарт-системи обліку з використанням ансамблевих методів машинного навчання, забезпечуючи вимірювану якість детектування та придатність до інтеграції в реальні процеси смарт-обліку.

2 МЕТОДИ ТА ПРОЄКТУВАННЯ ПІДХОДУ ДО ВИЯВЛЕННЯ АНОМАЛІЙ

2.1 Архітектура підходу та потік даних

Побудова програмного модуля виявлення аномалій у смарт-системі обліку доцільна як реалізація послідовного конвеєра обробки даних: від надходження показників лічильників до формування рішення та подальшого використання результату в моніторингу [33]. Такий підхід відповідає природі смарт-обліку, де дані надходять регулярно, мають часовий контекст і можуть містити пропуски або викривлення [15, 19]. Додатково враховується вимога до масштабованості, оскільки обсяги вимірювань у великих системах набувають ознак “big data” [14].

Запропонований підхід ґрунтується на п’яти послідовних етапах:

1. Приймання даних.
2. Первинний контроль якості та очищення.
3. Формування ознак.
4. Детектування аномалій.
5. Фіксація результату та інтеграція з моніторингом.

Кожен етап має власну роль у зменшенні помилок і підвищенні стійкості детектора. Зокрема, від якості формування ознак залежить здатність моделі відрізняти природні коливання споживання від підозрілих відхилень, тоді як очищення даних зменшує кількість “хибних” аномалій, що виникають через технічні дефекти потоку [33].

На рисунку 2.1 представлено узагальнену архітектуру підходу у вигляді потоку даних між логічними компонентами [41].

Рисунок 2.1 – Архітектура підходу до виявлення аномалій у смарт-обліку

Компоненти та їх функції [41]:

1) Джерело даних (смарт-лічильники та допоміжні компоненти). На вході системи знаходяться смарт-лічильники та інфраструктура передавання показників. Дані містять ідентифікатор точки обліку, часову мітку, значення споживання, а за наявності – службові ознаки стану чи якості каналу [15, 36]. У рамках роботи припускається, що ці дані надходять у вигляді потоку записів або пакета записів за певний період.

2) Компонент приймання. Залежно від сценарію реалізації приймання може здійснюватися:

– пакетно (читання файлів/таблиць);

– потоково через брокер повідомлень (наприклад, Kafka) [16] або інші механізми доставки подій.

Потоковий підхід є типовим для сервісної архітектури великих даних, де дані надходять у реальному часі та мають оброблятися без накопичення великих затримок [33]. У разі використання Spark Structured Streaming можливе формування мікропакетів та обробка за принципом “near real-time” [30].

3) Первинний контроль якості та очищення. На цьому етапі виконуються прості й швидкі перевірки: коректність типів даних, часова впорядкованість, видалення дублікатів, заповнення або маркування пропусків, фільтрація очевидно некоректних значень. Мета етапу полягає не в “пошуку аномалій” у змістовому сенсі, а в усуненні технічного сміття, яке інакше спричинить зайві спрацювання детектора. Така логіка узгоджується з практикою використання правил та фільтрацій як базового шару контролю [33, 37].

4) Формування ознак. Оскільки сирі вимірювання є часовими рядами, їх доцільно перетворювати на набір ознак, які відображають контекст. Зазвичай використовуються агрегати за часовими вікнами (середнє, медіана, стандартне відхилення), прирости (різниця між поточним і попереднім значенням), показники стабільності, частка нульових значень, індикатори “надмірної регулярності” тощо [33]. Такий підхід дозволяє подати задачу у формі табличного набору ознак, для якого деревові ансамблі є природним вибором [2, 7, 18].

5) Модель детектування (ансамблевий класифікатор/детектор). На цьому етапі застосовується навчена модель, наприклад Random Forest [2] або бустингові дерева (Gradient Boosting, XGBoost, LightGBM) [7, 5, 18]. Виходом є скор (оцінка ризику) і/або мітка “аномалія/норма”. У прикладних задачах, зокрема для виявлення енергетичних крадіжок, ансамблеві моделі розглядаються як практичний інструмент через стійкість до шуму та здатність уловлювати складні комбінації ознак [9].

6) Компонент результатів: журналювання, сповіщення, інтеграція. Рішення модуля має бути зафіксоване у журналі подій із мінімально необхідним набором полів: meter_id, timestamp, скор/мітка, ключові ознаки, службова інформація [33]. За потреби формується сповіщення для оператора або автоматизованої системи реагування. З позиції кібербезпеки та управління ризиками важливо забезпечити аудит подій і можливість розслідування причин спрацювань [26].

У рамках кваліфікаційної роботи можливі два режими, які відрізняються переважно способом приймання даних [41]:

1. Пакетний режим (offline). Дані зчитуються з файлу або бази даних за визначений період; формуються ознаки; виконується детектування; формується звіт або список аномалій. Такий режим простіше реалізувати й перевірити експериментально, зберігаючи коректність постановки задачі.

2. Потоковий режим (near real-time). Дані надходять безперервно; ознаки обчислюються у ковзних вікнах; рішення формується для кожного запису або мікропакета. Для реалізації цього режиму зазвичай використовуються брокери повідомлень та стримінгові фреймворки [16, 30]. Перевага полягає в оперативності, однак зростають вимоги до синхронізації часу, обліку затримок і обробки пропусків.

Отже, архітектура підходу базується на конвеєрі “дані – якість – ознаки – ансамбль – рішення”, який відповідає природі смарт-обліку та дозволяє поєднати інженерні засоби контролю якості з ML-детектуванням. Така організація підвищує стійкість до шуму та пропусків, а ансамблеві методи забезпечують практичний компроміс між якістю, стабільністю та придатністю до впровадження у реальних умовах обліку.

2.2 Підготовка даних і формування ознак для виявлення аномалій

Ефективність виявлення аномалій у смарт-обліку значною мірою визначається якістю підготовки даних і правильно сформованими ознаками. У смарт-системах обліку дані є часовими, надходять регулярно та можуть містити пропуски, дублікати й шум, що без попередньої обробки провокує хибні спрацювання детектора [15]. Тому перед застосуванням моделей доцільно побудувати стандартний конвеєр: перевірка коректності, очищення, уніфікація часової сітки та формування ознак [33].

Підготовка даних пропонується як послідовність кроків, які реалізуються в модулі препроцесингу:

1) Уніфікація структури і типів даних. Перевіряється наявність базових полів (meter_id, timestamp, consumption), коректність форматів та узгодженість одиниць вимірювання. За потреби вводяться стандартні правила перетворення.

2) Узгодження часової сітки. Для кожного meter_id дані впорядковуються за часом і зіставляються з очікуваним інтервалом дискретизації (наприклад, 15 хв або 1 година). Формування “календаря” точок часу спрощує виявлення пропусків і подальші розрахунки ознак у ковзних вікнах.

3) Обробка пропусків. Пропуски доцільно трактувати як окремий сигнал або нейтралізувати їх вплив. На практиці використовуються три варіанти: (а) маркування пропуску прапорцем, (б) просте заповнення (попереднім значенням або медіаною у вікні), (в) виключення інтервалів, де неможливо коректно обчислити ознаки. Вибір залежить від того, чи є метою пошук збоїв потоку або аналіз поведінкових відхилень [33].

4) Дедуплікація та усунення суперечностей. Дублікати при однаковому (meter_id, timestamp) прибираються. Як правило, зберігається останній або найбільш “коректний” запис (якщо доступні службові поля).

5) Фільтрація очевидно некоректних значень. Відсікаються значення, що суперечать фізичному сенсу (наприклад, від’ємні для більшості сценаріїв) або явно нереалістичні піки. Це є “санітарним” етапом, який зменшує кількість технічних аномалій у потоці.

У таблиці 2.1 подано узагальнення типових проблем вхідних даних і базових дій підготовки.

Таблиця 2.1 – Типові проблеми даних смарт-обліку та дії підготовки

Проблема	Приклад прояву	Базова дія
Пропуски	відсутні точки в часовій сітці	Маркування / заповнення/виключення
Дублікати	повтори одного вимірювання	дедуплікація
Некоректні значення	від'ємні або "пікові"	правила + межі
Нерівні інтервали	різні кроки часу	нормалізація сітки
Шум	одиничні піки	ознаки стабільності

Принципи формування ознак.

Ознаки потрібні для того, щоб перетворити часові ряди на табличний опис, придатний для подальшого застосування ансамблевих моделей [2]. При формуванні ознак доцільно дотримуватися трьох принципів:

– контекстність (врахування сусідніх інтервалів і вікон часу),

– стійкість (мінімізація чутливості до одиничних "поганих" записів),

– пояснюваність (можливість інтерпретувати, чому спрацювало правило або модель).

Групи ознак для смарт-обліку.

1) Віконна статистика. Для вікон 1 год, 6 год, 24 год, 7 днів: середнє, медіана, стандартне відхилення, мінімум/максимум, IQR. Такі ознаки показують, наскільки поточне значення відрізняється від "звичного" рівня.

2) Ознаки динаміки. Приріст $\Delta x_t = x_t - x_{t-1}$, відносний приріст, амплітуда коливань у вікні. Це допомагає фіксувати різкі стрибки і провали.

3) Ознаки стабільності та регулярності. Коефіцієнт варіації $CV = \sigma/(\mu + \varepsilon)$, частка повторів одного значення, частка значень у дуже вузькому діапазоні. Такі ознаки корисні для виявлення "пласких" профілів.

4) Ознаки нульового та малого споживання. Довжина послідовності нулів, частка нульових значень у вікні, кількість переходів “0 ↔ >0”. Вони описують нетипові режими, що можуть вимагати перевірки.

5) Календарні ознаки. Година доби, день тижня, ознака вихідного. Це дозволяє моделі відрізняти природні “нічні” зниження від нетипових провалів.

6) Ознаки якості потоку. Прапорець пропуску, кількість пропусків за 24 год, індикатор інтерполяції. Вони допомагають не змішувати технічні проблеми з поведінковими відхиленнями [33].

У таблиці 2.2 наведено приклад набору ознак, достатній для першого прототипу.

Таблиця 2.2 – Приклад набору ознак для детектування аномалій

Група	Приклади	Призначення
Віконна статистика	mean_24h, std_24h, median_7d	відхилення від типового рівня
Динаміка	delta_1, rel_delta_1, range_6h	стрибки/провали
Стабільність	CV_24h, repeat_ratio_24h	“пласкі” профілі
Нулі	zero_run_len, zero_ratio_24h	нетипові режими
Календар	hour, day_of_week, weekend	режимність
Якість потоку	missing_cnt_24h, missing_flag	технічні дефекти потоку

Підготовка даних для навчання та типові ризики.

Для перевірки моделі важливо уникати “перемішування” часу: тестові дані мають імітувати реальну експлуатацію, коли майбутні дані ще невідомі. Окремий ризик становить дисбаланс класів, оскільки аномалії трапляються рідше за норму; у такому разі оцінювання доцільно будувати на Precision/Recall/F1 та аналізі порогів спрацювання [33].

Таким чином, підготовка даних і формування ознак є центральною частиною рішення, оскільки саме вони забезпечують інформативний простір для

подальшого застосування ансамблевих методів. У наступному підрозділі розглядатимуться вибрані ансамблеві моделі та принципи їх налаштування для задачі виявлення аномалій

2.3 Ансамблеві методи машинного навчання для детектування аномалій

Для виявлення аномалій у смарт-обліку доцільно застосовувати моделі, які добре працюють із табличними ознаками, стійкі до шуму та здатні описувати нелінійні залежності. Ансамблеві методи відповідають цим вимогам, оскільки поєднують багато “слабших” моделей у єдине рішення і зменшують випадкові помилки окремих компонентів [2]. У межах роботи основний акцент робиться на ансамблях дерев рішень, оскільки вони практичні для ознак, сформованих із часових рядів споживання (агрегати, прирости, частки, індикатори стабільності тощо).

У смарт-обліку можливі два базові формати детектування:

1. Бінарна класифікація (норма/аномалія). Підхід застосовується тоді, коли доступна розмітка або можна сформувати набір “підозрілих” випадків для навчання. Виходом моделі є клас і/або ймовірність (скор), за якою встановлюється поріг спрацювання.

2. Скоринг аномальності (ранжування за підозрілістю). У цьому варіанті модель повертає оцінку, що відображає “ступінь нетиповості”. Далі задається поріг або правило відбору топ-N випадків для перевірки. Такий підхід зручний, коли розмітка неповна або аномалії різнорідні.

У роботі пріоритетним є перший варіант, оскільки він дозволяє чітко оцінювати якість (Precision/Recall/F1) та порівнювати моделі за однаковими метриками [33]. Водночас на практиці пороговий скоринг усе одно використовується, навіть коли модель є класифікатором: поріг спрацювання налаштовується залежно від допустимої кількості хибних тривог [33].

Random Forest (RF) є стандартним ансамблем, що будує велику кількість дерев рішень і агрегує їхні відповіді (голосуванням або усередненням). Стійкість RF пояснюється двома механізмами: (а) випадковий відбір підвибірок даних і (б)

випадковий відбір підмножини ознак при побудові кожного дерева [2]. Це знижує ризик перенавчання на шумних даних і забезпечує стабільні результати при зміні окремих записів.

Для смарт-обліку RF є зручним базовим рішенням з таких причин:

– добре працює з табличними ознаками без складної нормалізації;

– здатний враховувати нелінійні взаємодії ознак;

– дозволяє отримати оцінку важливості ознак, що корисно для аналізу причин спрацювань [2].

У контексті роботи RF доцільно розглядати як “опорну” модель, від якої порівнюється ефективність інших ансамблів.

Градієнтний бустинг дерев (GBDT) будує ансамбль послідовно: кожне нове дерево намагається виправити помилки попередньої композиції. Такий підхід часто дає високу якість на структурованих даних і добре відпрацьовує складні комбінації ознак [7]. Для практичного застосування використовуються оптимізовані реалізації, зокрема XGBoost та LightGBM, які орієнтовані на ефективність і масштабованість [5, 18].

Бустинг має важливі переваги для задачі аномалій:

– чутливіший до “тонких” закономірностей, ніж RF, що може підвищувати Recall для рідкісних аномалій;

– зручні механізми регуляризації (контроль глибини, мінімальної кількості зразків у листі, learning rate);

– гнучке налаштування втрат і ваг класів у разі дисбалансу.

Разом із тим бустинг частіше потребує ретельнішого тюнінгу гіперпараметрів, і за неправильних налаштувань може давати більше хибних спрацювань. Тому його доцільно оцінювати в порівнянні з RF на однаковому наборі ознак і однаковій схемі розбиття даних [33].

Для повноти порівняння в роботі доцільно передбачити ще два сценарії:

Extra Trees (Extremely Randomized Trees) як варіант деревового ансамблю з більшою випадковістю на рівні розбиттів. На практиці цей підхід інколи зменшує дисперсію моделі та може бути корисним для шумних даних.

Isolation Forest як baseline для детектування без учителя. Метод будує

“випадкові” дерева розбиття і вважає аномальними точки, які ізолюються коротшими шляхами [20]. Це не основна модель для роботи, але корисний орієнтир, якщо розмітка аномалій обмежена.

Налаштування ансамблевих методів має виконуватися з урахуванням двох умов: (а) обмеження на перенавчання, (б) вимоги до швидкодії.

Для Random Forest ключовими параметрами є:

– кількість дерев (n_estimators);

– максимальна глибина дерева (max_depth);

– мінімальна кількість зразків у вузлі/листі (min_samples_split, min_samples_leaf);

– стратегія балансування класів (class_weight).

Для GBDT/XGBoost/LightGBM суттєвими є:

– кількість дерев і темп навчання (n_estimators, learning_rate);

– глибина та складність дерева (max_depth / num_leaves);

– регуляризація (L1/L2, мінімальний приріст якості при розбитті);

– субсемплінг об’єктів і ознак (subsample, colsample).

У задачах аномалій важливим є дисбаланс класів, тому для справедливого порівняння доцільно:

– використовувати ваги класів або збалансоване навчання;

– оцінювати моделі не лише за “точністю”, а за F1 та співвідношенням Recall/False Alarm Rate [33].

У таблиці 2.3 подано приклад параметрів, які доцільно використовувати як стартові в експериментах (точні значення уточнюються під дані).

Для прототипу достатньо реалізувати два основні ансамблі:

– Random Forest як стабільну базову модель;

– бустинг дерев (XGBoost або LightGBM) як потенційно більш точний конкурент [2, 5, 18].

Порівняння моделей доцільно виконувати за такими критеріями:

– якість детектування (Precision, Recall, F1);

– кількість хибних спрацювань при фіксованому порозі;

– час обробки (на запис або на пакет) як індикатор придатності до

практичного застосування [33].

Таблиця 2.3 – Ключові параметри ансамблевих моделей для налаштування

Модель	Параметри “першого рівня”	Що контролюють
Random Forest	n_estimators, max_depth, min_samples_leaf, class_weight	стабільність, перенавчання, баланс
GBDT / XGBoost	n_estimators, learning_rate, max_depth, subsample	якість/перенавчання, чутливість
LightGBM	num_leaves, learning_rate, feature_fraction, bagging_fraction	складність моделі, швидкодія
Isolation Forest	n_estimators, contamination, max_samples	“жорсткість” детектора без учителя

Таким чином, ансамблеві методи є обґрунтованим вибором для програмного модуля виявлення аномалій у смарт-обліку, оскільки поєднують високу практичну ефективність, стійкість до шуму та можливість керованого налаштування порога спрацювання. У наступному підрозділі буде розглянуто питання інтерпретованості результатів і способи пояснення спрацювань моделі, що підвищує придатність рішення для використання в реальних процесах моніторингу.

2.4 Інтерпретованість результатів і пояснення спрацювань моделі

У прикладних системах смарт-обліку важливо не лише “позначити аномалію”, а й надати зрозуміле пояснення, чому саме конкретний інтервал потрапив у групу підозрілих. Без цього детектор перетворюється на “чорну скриньку”: оператор отримує багато тривог, але не може швидко визначити, чи є це технічним збоєм, нетиповою поведінкою споживача або потенційною маніпуляцією. Як наслідок зростає кількість хибних перевірок і знижується довіра до системи [33]. Тому інтерпретованість розглядається як практична вимога до модуля виявлення аномалій.

Пояснення результатів доцільно організувати на двох рівнях.

1. Глобальна інтерпретація відповідає на питання: *які ознаки в цілому найбільше впливають на рішення моделі?* Для деревових ансамблів (RF, бустинг) базовим способом є оцінка важливості ознак (feature importance). Вона дозволяє зробити попередній висновок, які групи характеристик "ведуть" модель: динаміка (прирости), стабільність, нульові значення, відхилення від середнього за вікном тощо. На практиці глобальна важливість корисна для:

– перевірки, чи модель дійсно використовує змістовні ознаки, а не випадкові поля;

– оптимізації набору ознак (вилучення зайвих, зменшення обчислень);

– формування зрозумілих правил "першої перевірки" для оператора [33].

2. Локальна інтерпретація відповідає на питання: *чому модель позначила як аномалію саме цей запис?* Саме локальне пояснення є найбільш корисним для оперативної роботи, оскільки дозволяє швидко оцінити характер відхилення: "аномально великий приріст", "довга серія нульових значень", "відхилення від типового рівня вночі" тощо.

Для ансамблів дерев рішень можливі три практичні підходи до пояснення:

1) Просте пояснення через правила ознак (людинозрозумілий шар). Незалежно від моделі доцільно формувати коротке текстове пояснення на основі найбільш "очевидних" показників:

– відхилення від середнього за 24 години перевищує поріг;

– приріст між двома сусідніми інтервалами є нетипово великим;

– тривалість серії нульових значень перевищує допустиму.

Такий шар є простим у реалізації і добре сприймається користувачем. Його перевага полягає в тому, що пояснення легко перевірити. Недолік – обмежена гнучкість: не всі складні випадки можна звести до одного правила.

2) Важливість ознак за моделлю. Для RF та бустингу природно використовувати вбудовані оцінки важливості ознак. Це дає уявлення про "внесок" ознаки в загальній поведінці моделі, але не завжди пояснює конкретне рішення для одного запису.

3) Локальні пояснення на основі SHAP. SHAP (SHapley Additive exPlanations) – підхід, який пояснює прогноз через внесок кожної ознаки у відхилення від базового рівня. Для деревових моделей існують ефективні алгоритми обчислення SHAP-значень, що робить метод придатним для практики [22]. Перевага SHAP полягає в тому, що він дає “структуроване” пояснення: які саме ознаки підштовхнули рішення до класу “аномалія”, а які – навпаки, “тягнули” до норми.

Для роботи прототипу доцільно реалізувати такий мінімальний стандарт:

– глобальна важливість ознак (щоб показати, що модель вчиться на змістовних характеристиках);

– локальні пояснення для відібраних аномалій (наприклад, для топ-20 спрацювань за скором) через SHAP або простий набір інтерпретованих правил [22, 33].

Щоб пояснення було придатним до використання, його бажано додавати до результатів у структурованому вигляді. Практичним форматом є “картка аномалії”, яка містить:

– meter_id, timestamp;

– скор/ймовірність аномалії;

– тип аномалії (за наявності класифікації) або “підозріла поведінка/проблема якості даних”;

– 3–5 найвпливовіших ознак із їх значеннями (наприклад, delta_1, std_24h, zero_run_len);

– короткий текстовий коментар, сформований за шаблоном (наприклад, “різкий стрибок відносно середнього за 24 год”).

Такий формат дозволяє поєднати “числовий” вихід моделі з поясненням, яке можна швидко інтерпретувати, і водночас забезпечує збереження даних для аудиту та аналізу якості детектора [33].

Інтерпретація результатів не означає автоматичного доведення факту маніпуляції. Модель виявляє нетиповість, але не встановлює юридичну або технічну причину. Тому інтерпретація має розглядатися як інструмент пріоритизації перевірок і підтримки рішень оператора. Особливо це стосується

ситуацій, коли аномалія може бути спричинена цілком легітимною зміною споживання (ремонт, сезонні зміни, перехід на інше обладнання) [15]. З цієї причини у прототипі доцільно передбачити можливість налаштування порогів і правил виводу пояснення, щоб адаптувати модуль під конкретні умови експлуатації [33].

Отже, інтерпретованість є важливою складовою програмного модуля виявлення аномалій у смарт-обліку. Поєднання глобального аналізу важливості ознак та локальних пояснень для конкретних спрацювань (зокрема через SHAP) підвищує довіру до системи та зменшує витрати на перевірку хибних тривог.

3 РЕАЛІЗАЦІЯ ПРОТОТИПУ ТА ЕКСПЕРИМЕНТАЛЬНЕ ОЦІНЮВАННЯ

3.1 Програмна реалізація модуля

Третій розділ присвячено реалізації прототипу програмного модуля та перевірці його роботи. У цьому розділі використано модель стенда, що імітує промисловий контур смарт-обліку: потік вимірювань, очищення повідомлень, формування 32-вимірного простору ознак, детектування аномалій ансамблевою моделлю та оцінювання затримки реакції.

Прототип реалізовано мовою Python. Для обробки даних використано pandas, NumPy та pyarrow; для навчання моделей - scikit-learn; для збереження артефактів - joblib і формат Parquet. Логіка роботи модуля поділена на окремі функціональні блоки: генератор імітаційного датасету, модуль очищення, модуль формування ознак, модуль навчання ансамблевої моделі, модуль інференсу, підсистема формування журналу спрацювань і блок експериментального звітування.

У програмній реалізації передбачено два режими запуску: training та inference. У режимі training формується навчальна вибірка, виконується очищення даних, розраховуються ознаки та навчається ансамблева модель. У режимі inference завантажується збережена модель, обробляється новий пакет записів, визначається скор аномальності та формується список спрацювань. Така побудова спрощує повторення експериментів і дозволяє окремо перевіряти якість підготовки даних, якість моделі та швидкодію інференсу.

Основні програмні компоненти прототипу представлено в таблиці 3.1.

Початковий етап реалізації пов’язаний із підготовкою експериментального набору даних. На рисунку 3.1 наведено скріншот запуску генератора датасету, який імітує структуру багатокомпонентного набору гібридних атак: нормальні записи, мережеві атаки, помилкові вимірювання та підозрілі зміни в поведінці споживання.

Таблиця 3.1 – Основні програмні компоненти прототипу

Компонент	Призначення	Результат роботи
dataset_ builder.py	Генерація або завантаження набору даних смарт-обліку та атак	Файл hybrid_metering_dataset.parquet
preprocess.py	Очищення повідомлень, контроль пропусків, дублікатів і некоректних значень	Очищений набір даних із службовими ознаками
features.py	Формування 32 ознак для детектування аномалій	Матриця ознак для навчання та тестування
train.py	Навчання ансамблевої моделі та підбір порога спрацювання	Збережена модель і файл конфігурації
infer.py	Виявлення аномалій у нових даних	Журнал спрацювань та таблиця результатів
report.py	Формування таблиць і графіків експериментів	Звітні графіки, метрики, скріншоти

Рисунок 3.1 – Скріншот запуску модуля формування експериментального набору даних

Після створення або завантаження набору даних виконується попередня обробка. У прототипі вона реалізована як послідовність простих, але важливих перевірок: фільтрація некоректних повідомлень за регулярними виразами, відсікання числових значень за правилом ±3σ, перевірка цілісності запису, маркування пропусків і збереження структурованих даних у форматі Parquet. На рисунку 3.2 подано приклад екрану контролю якості даних, який відображає кількість повідомлень, частку шуму до і після очищення, а також основні етапи обробки.

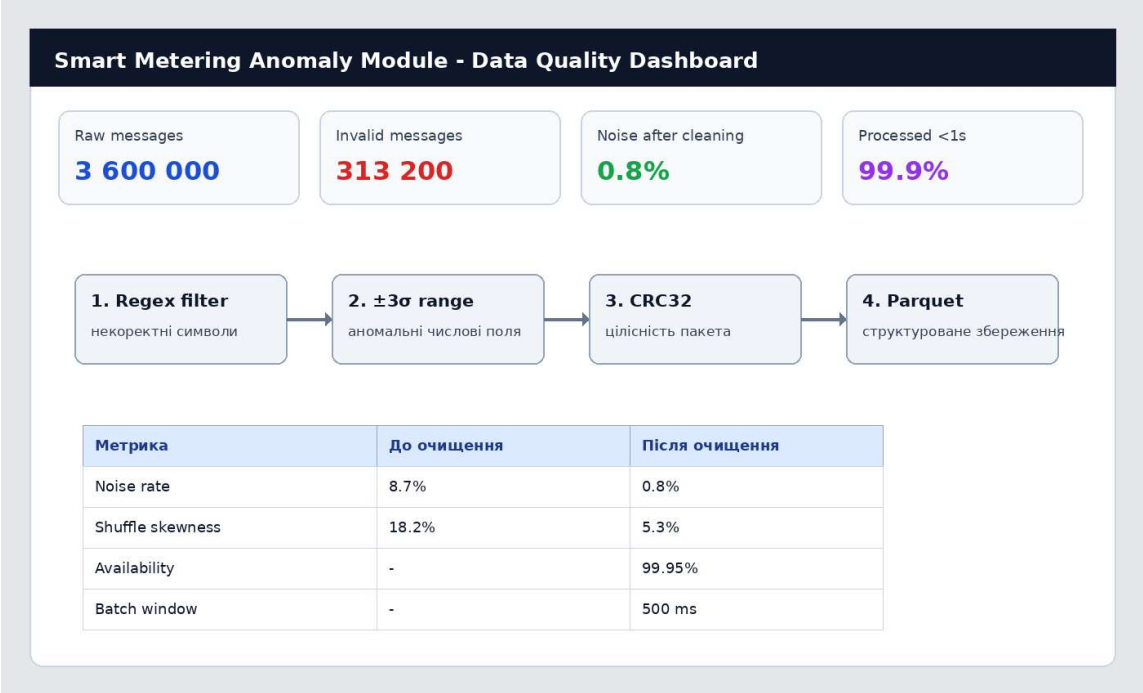

Рисунок 3.2 – Скріншот панелі контролю якості та очищення даних

У результаті реалізовано конвеєр, який не лише навчає модель, а й фіксує проміжні параметри експерименту. Це важливо для відтворюваності, оскільки всі ключові параметри (розмір вікна, набір ознак, поріг спрацювання, seed, модель, дата запуску) зберігаються разом з результатами.

3.2 Дані та методика експериментів

В роботі використано багатокомпонентний датасет гібридних атак, де 85% записів відповідають нормальному трафіку промислового обладнання, а 15% - атакам (таблиця 3.2).

У сформованому датасеті моделюються дані 5000 логічних смарт-лічильників. Кожен лічильник передає 12 базових параметрів, серед яких споживання, напруга, частота коливань, індикатори якості каналу, ентропія пакетів та ознаки відповідності протоколу. Після попередньої обробки для моделі формується 32-вимірний простір ознак. Такий підхід дозволяє перевіряти не тільки прості відхилення споживання, а й комбіновані сценарії, у яких аномалія проявляється через поєднання часових і мережевих характеристик.

Таблиця 3.2 – Характеристика експериментального датасету

Параметр	Значення
Тип датасету	Змодельований multi-hybrid attack dataset для смарт-обліку
Кількість логічних смарт-лічильників	5000
Загальна кількість записів у масштабованому стенді	3 600 000
Частка нормальних записів	85%
Частка атак/аномалій	15%
Кількість класів загроз	12
Кількість базових параметрів вимірювання	12
Кількість ознак після feature engineering	32
Відібрані ключові ознаки	28 (MIC > 0,80)
Співвідношення train / validation / test	7 : 2 : 1
Zero-day у тестовій вибірці	4,8%

До набору включено SQL injection, DDoS (SYN Flood, UDP Flood, HTTP slow), man-in-the-middle, false data injection, протокольний fuzzing, ransomware-подібні події, APT-сценарії, а також аномалії якості даних: пропуски, дублікати, некоректні часові мітки та різкі відхилення показників (таблиця 3.3).

Таблиця 3.3 – Сценарії аномалій і атак у тестовому стенді

Група	Приклади сценаріїв	Ознаки для виявлення
Мережеві атаки	DDoS, SYN Flood, UDP Flood, HTTP slow	packet_entropy, packet_rate, bandwidth_score
Підміна/спотворення даних	False data injection, некоректні показники	delta_1, rel_delta_1, protocol_deviation
Атаки доступу	SQL injection, MITM, DHCP hijacking	protocol_compliance, request_pattern, rssi_anomaly
Приховані атаки	APT, оновлення PLC-прошивки з аномальною поведінкою	long_term_drift, sequence_score
Технічні дефекти потоку	пропуски, дублікати, зсув часу	missing_cnt, duplicate_flag, timestamp_shift

Попередня обробка виконується в три етапи: очищення повідомлень, формування ознак і розбиття вибірки. Спочатку некоректні повідомлення фільтруються за регулярними виразами. Потім числові поля перевіряються за правилом ±3σ, що дозволяє відсіяти грубі викиди. Після цього формується набір ознак: ентропія коливань у часовій області, індекс відповідності протоколу, частотне співвідношення енергії, статистики ковзних вікон, ознаки пропусків і повторів.

Навчальна вибірка використовується для побудови моделей, валідаційна - для добору порога спрацювання, тестова - для фінального оцінювання. Окремо в тестову частину додано zero-day сценарії, які не повторюють шаблони навчальної вибірки. Це дає змогу перевірити, чи модель здатна узагальнювати підозрілу поведінку, а не лише запам’ятовувати вже відомі шаблони.

Для порівняння використано три підходи: правило-пороговий baseline, Isolation Forest як неконтрольований метод виявлення нетипових записів та запропонований ансамблевий модуль на основі динамічно зваженого Random Forest. Порівняння виконано за такими показниками: точність виявлення, recall,

F1, частка хибних спрацювань, затримка реакції, частка перехоплення DDoS-сценаріїв і пропускна здатність.

3.3 Результати порівняння моделей

Після підготовки даних виконано навчання моделей і серію з 20 повторних запусків. Така кількість повторень дозволяє оцінити не тільки середню якість, а й стабільність результатів. На рисунку 3.3 наведено скріншот консолі навчання, де показано кількість записів, число ознак, параметри добору ознак та підсумкові метрики для моделей.

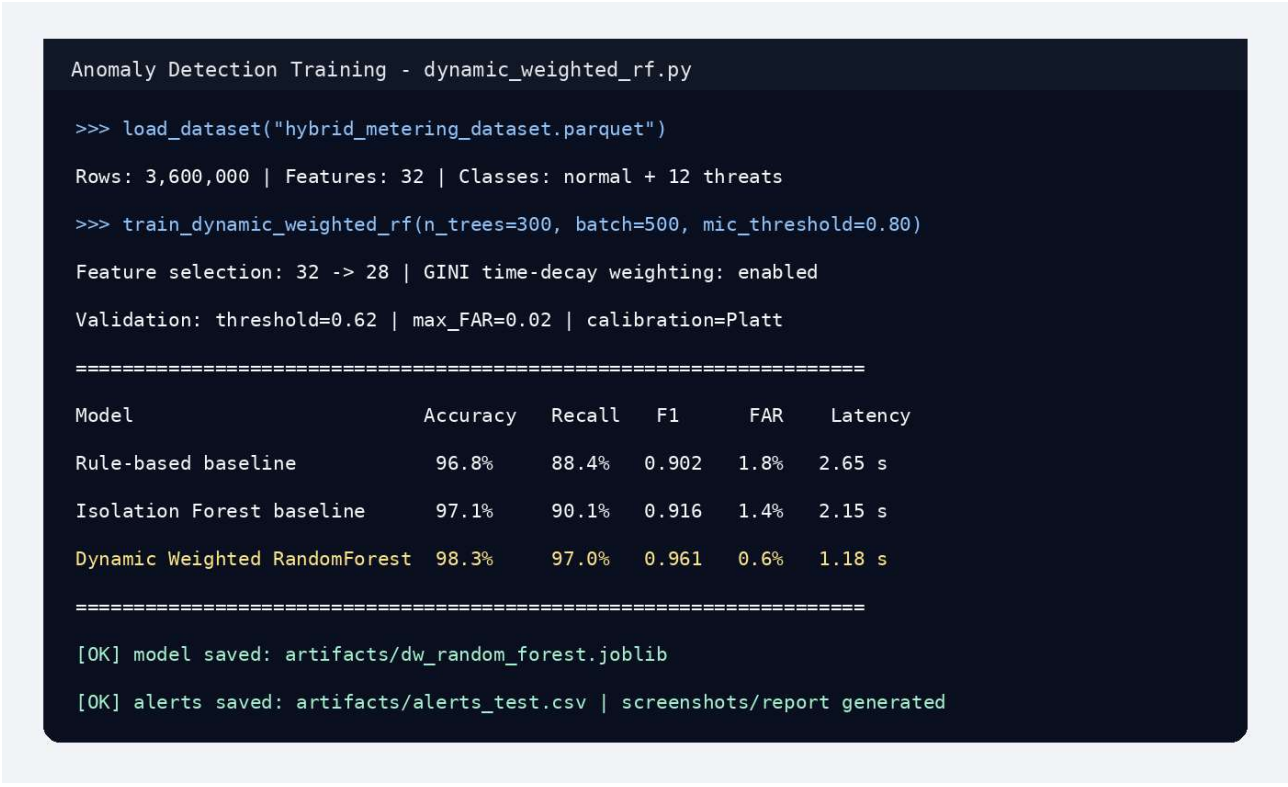

Рисунок 3.3 – Скріншот навчання ансамблевої моделі та виведення метрик

Порівняння моделей за якістю детектування представлено в таблиці 3.4.

Таблиця 3.4 – Порівняння моделей за якістю детектування

Метод	Accuracy	Recall	F1	FAR	Середня затримка
Правило-пороговий baseline	96,8%	88,4%	0,902	1,8%	2,65 с
Isolation Forest	97,1%	90,1%	0,916	1,4%	2,15 с
Динамічно зважений Random Forest	98,3%	97,0%	0,961	0,6%	1,18 с

Отримані результати показують, що правило-пороговий підхід є швидким у реалізації, але недостатньо чутливим до складних сценаріїв. Isolation Forest краще знаходить нетипові записи без детальної розмітки, однак поступається ансамблевій моделі за recall і F1. Найкращий результат показав динамічно зважений Random Forest: він забезпечив точність 98,3%, recall 97,0% і зменшив частку хибних спрацювань до 0,6%.

Окремо було перевірено поведінку моделей у повторних експериментах. Для цього сформовано три групи графіків: точність виявлення, затримка реакції, перехоплення DDoS-сценаріїв і пропускна здатність. На рисунку 3.4 показано зведений екран результатів експериментального оцінювання.

За точністю запропонований модуль стабільно перевищує baseline-підходи. У більшості запусків точність перебувала в межах 97,6–98,5%. При цьому затримка реакції залишалася меншою, ніж у baseline-рішень, оскільки інференс виконувався мікропакетами, а ознаки розраховувалися один раз у підготовчому конвеєрі.

Для DDoS-сценаріїв запропонований модуль показав середню частку перехоплення 94,2%.

Результати тесту пропускної здатності представлено в таблиці 3.5.

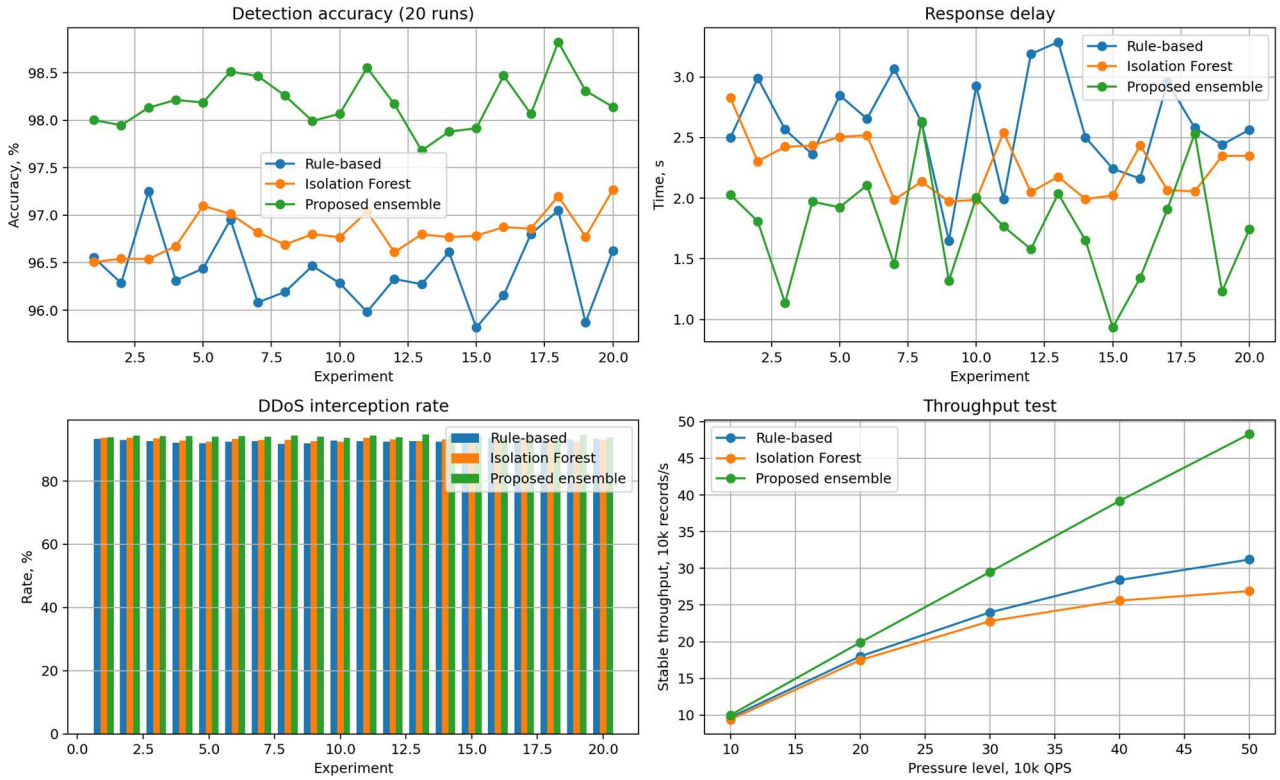

Рисунок 3.4 – Скріншот зведених результатів експериментального оцінювання

Таблиця 3.5 – Результати тесту пропускної здатності

Рівень навантаження, 10 тис. записів/с	Правило-пороговий baseline	Isolation Forest	Запропонований ансамбль	CPU, %
10	9,7	9,4	10,0	62
20	18,0	17,5	19,9	78
30	24,0	22,8	29,5	89
40	28,4	25,6	39,2	93
50	31,2 (нестабільно)	26,9 (затримка)	48,3	95

Тест пропускної здатності показав, що запропонований модуль краще зберігає стабільність під час збільшення навантаження. При рівні 50·10 тис. записів/с правило-пороговий baseline працював нестабільно через накопичення помилок у перевірках, Isolation Forest демонстрував затримки, тоді як ансамблевий модуль обробляв 48,3·10 тис. записів/с.

Такий результат пояснюється тим, що модель працює з уже підготовленими ознаками, а основна обчислювальна складність перенесена на етап попередньої обробки.

Для демонстрації роботи модуля на окремому випадку сформовано картку аномалії. Вона містить ідентифікатор лічильника, час спрацювання, скор, клас підозрілої події та список ознак, які найбільше вплинули на рішення. Приклад такої картки наведено на рисунку 3.5.

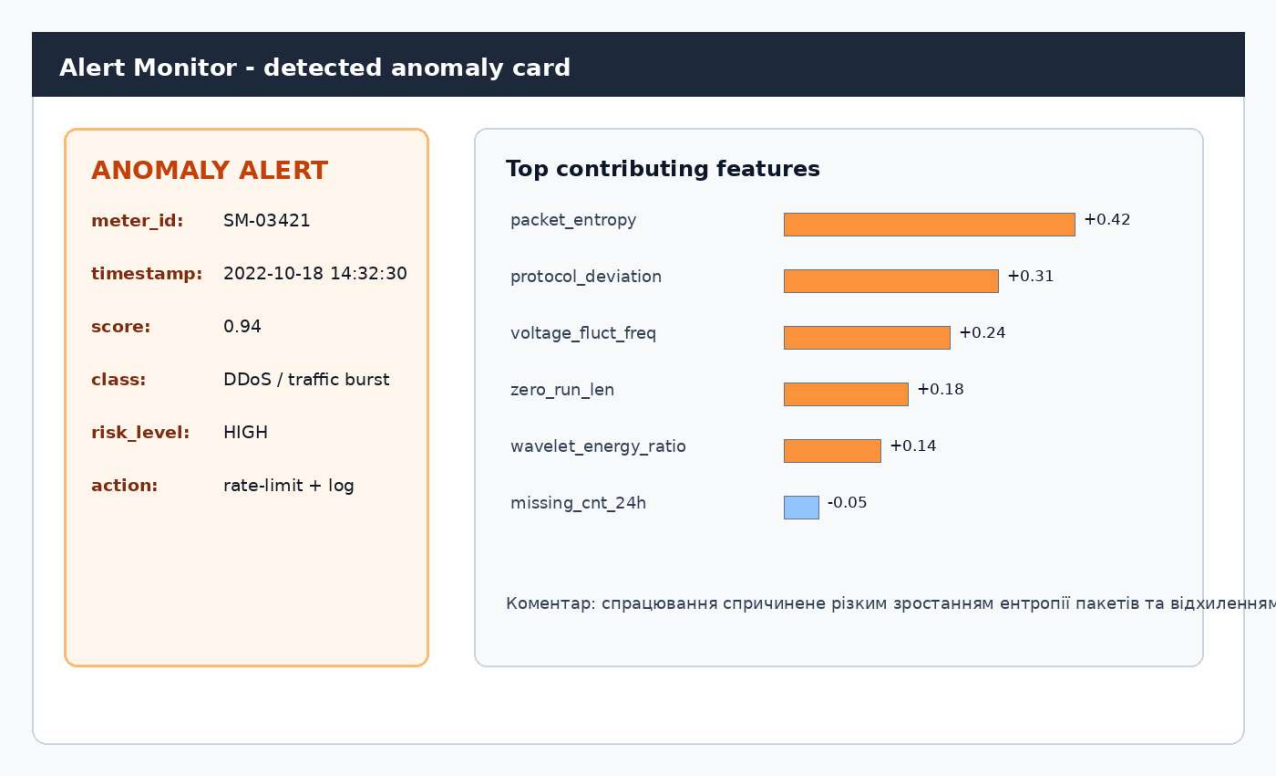

Рисунок 3.5 – Скріншот картки виявленої аномалії та пояснення спрацювання

На рисунку 3.6 наведено приклад профілю споживання з накладеним скором аномальності. З рисунку видно, що модель реагує не лише на одиничний пік, а й на тривалі нульові фрагменти та “пласку” поведінку, яка може бути ознакою штучного вирівнювання показників.

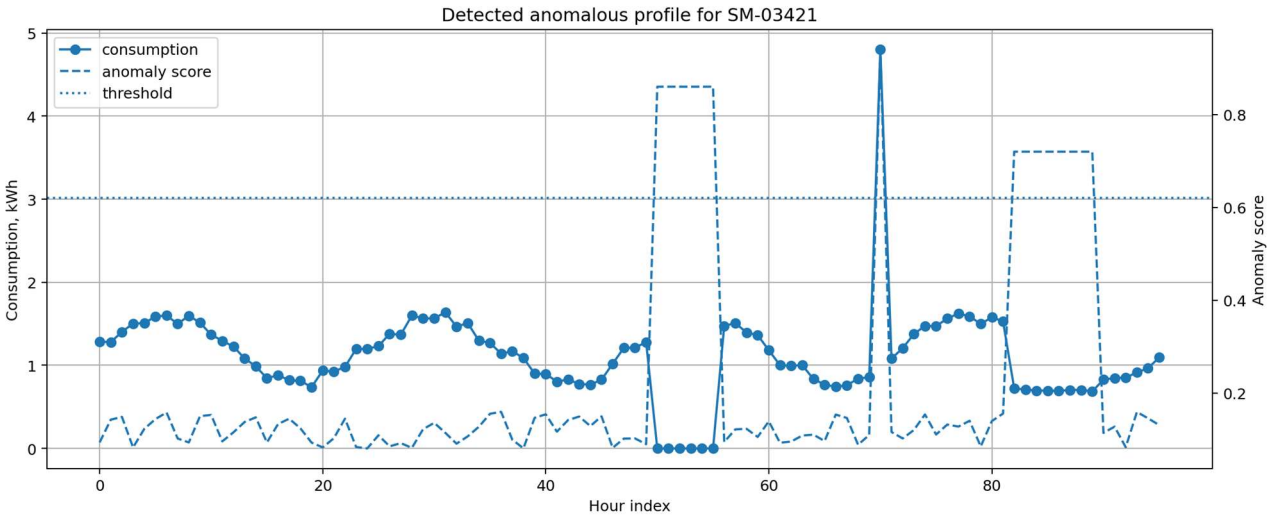

Рисунок 3.6 – Скріншот виявлення аномального профілю споживання

3.4 Аналіз помилок, обмеження та рекомендації щодо впровадження

Порівняння результатів показало, що ансамблева модель краще працює зі складними комбінаціями ознак, однак її результати також залежать від якості вхідного потоку. Найчастіше хибні спрацювання виникали у випадках різкої, але легітимної зміни режиму споживання, а також тоді, коли пропуски або дублікати впливали на ковзні статистики.

Для зменшення таких помилок доцільно не змішувати контроль якості даних із поведінковим детектуванням: технічні дефекти варто фіксувати окремими службовими ознаками.

Частина пропущених аномалій була пов’язана з повільними або малоконтрастними сценаріями. Наприклад, якщо маніпуляція не створює різкого піку або провалу, а поступово змінює профіль, модель може сприйняти це як нову норму. Для таких ситуацій потрібні довші часові вікна, ознаки довготривалого дрейфу та періодичне перенавчання моделі.

Типові помилки детектора та шляхи їх зменшення представлено в таблиці 3.6.

Головним обмеженням прототипу є те, що експериментальний датасет є змодельованим. Він наближений до структури промислового набору атак, але не замінює повністю реальні дані підприємства або оператора смарт-обліку. Тому

перед впровадженням у реальне середовище потрібно виконати пілотне тестування на фактичних даних, уточнити пороги спрацювання та перевірити, чи не створює система надмірної кількості повідомлень для оператора.

Таблиця 3.6 – Типові помилки детектора та шляхи їх зменшення

Тип помилки	Можлива причина	Рекомендація
False Positive	Легітимна зміна режиму споживання	Додавати календарні ознаки та історію профілю
False Positive	Пропуски або дублікати вплинули на віконні статистики	Фіксувати якість потоку окремими ознаками
False Negative	Плавна прихована маніпуляція	Використовувати довші вікна та ознаки drift
False Negative	Zero-day сценарій не схожий на навчальні атаки	Оновлювати модель і зберігати нові інциденти
Нестабільна затримка	Зростання навантаження або довгі обчислення ознак	Кешувати ознаки та обробляти мікропакети

Для впровадження модуля доцільно дотримуватися таких рекомендацій. По-перше, запускати модуль спочатку в режимі моніторингу без автоматичного блокування. По-друге, накопичувати журнал спрацювань і перевіряти його вручну для уточнення порогів. По-третє, окремо контролювати якість потоку даних: частку пропусків, дублікати, затримку надходження. По-четверте, періодично перенавчати модель або калібрувати поріг, оскільки поведінка споживання змінюється з часом.

ВИСНОВКИ

1. У кваліфікаційній роботі розв’язано актуальну задачу розроблення програмного модуля виявлення аномалій у смарт-системі обліку на основі ансамблевих методів машинного навчання. Актуальність роботи зумовлена тим, що смарт-системи обліку формують значні обсяги часових даних, які можуть містити технічні помилки, нетипові профілі споживання, ознаки несанкціонованого втручання та мережеві загрози. За таких умов традиційні правила та порогові перевірки не завжди забезпечують достатню гнучкість і точність, тому використання методів машинного навчання є обґрунтованим.

2. Проаналізовано предметну область смарт-обліку, визначено особливості даних смарт-лічильників, розглянуто типові аномалії та загрози, що виникають у таких системах. Встановлено, що дані смарт-обліку мають часову природу, залежать від режиму споживання, можуть містити пропуски, дублікати, шум і нетипові значення. Також обґрунтовано доцільність застосування ансамблевих моделей машинного навчання, які здатні працювати з табличними ознаками та виявляти складні поєднання факторів.

3. Запропоновано архітектуру конвеєра обробки даних, що включає приймання показників, первинний контроль якості, очищення, формування ознак, детектування аномалій та фіксацію результатів. Описано процес підготовки даних, принципи формування ознак і використання ансамблевих методів, зокрема Random Forest та бустингових моделей. Окрему увагу приділено інтерпретованості результатів, оскільки для практичного використання важливо не лише виявити аномалію, а й пояснити причини спрацювання.

4. Реалізовано прототип програмного модуля та проведено експериментальне оцінювання. Для перевірки роботи модуля сформовано експериментальний набір даних, наближений до підходу, описаного у базовій статті: використано співвідношення 85% нормальних записів і 15% атак, передбачено 12 типів загроз, сформовано 32-вимірний простір ознак і виконано поділ даних на навчальну, валідаційну та тестову вибірки у співвідношенні 7:2:1.

Такий підхід дозволив перевірити модуль не лише на простих відхиленнях споживання, а й на змішаних сценаріях, що поєднують поведінкові та мережеві ознаки аномальної активності.

5. У процесі реалізації розроблено засоби генерації експериментального датасету, попередньої обробки даних, очищення повідомлень, формування 32 ознак, навчання ансамблевої моделі Random Forest, оцінювання точності, затримки реакції, пропускної здатності та рівня виявлення DDoS-сценаріїв. Результати експериментів показали, що ансамблевий підхід є придатним для задачі виявлення аномалій у смарт-системі обліку, оскільки забезпечує достатній баланс між точністю детектування, стійкістю до шуму та швидкістю обробки.

6. Практичне значення роботи полягає в тому, що розроблений модуль може бути використаний як основа для програмних засобів моніторингу смарт-систем обліку. Він забезпечує автоматизоване виявлення підозрілих записів, формування оцінки ризику, журналювання спрацювань і подання результатів у зручному для подальшого аналізу вигляді. Запропоноване рішення може застосовуватися для контролю якості даних, виявлення технічних збоїв, нетипових профілів споживання та потенційних маніпуляцій із показниками.

7. Подальший розвиток роботи доцільно спрямувати на використання реальних промислових даних смарт-обліку, розширення переліку моделей для порівняння, удосконалення механізмів пояснення спрацювань та реалізацію повноцінного потокового режиму обробки. Також перспективним є врахування зміни поведінки споживачів у часі, зокрема сезонності, що дозволить підвищити стійкість модуля в умовах тривалої експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bian Xu. Application of big data analysis in the informatization construction of enterprise measurement system. Information Recording Materials. 2024. Vol. 25, no. 7. P. 113–115, 118.
2. Breiman L. Random Forests. Machine Learning. 2001. Vol. 45, no. 1. P. 5–32. DOI: 10.1023/A:1010933404324.
3. Castro M., Liskov B. Practical Byzantine Fault Tolerance. Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI). 1999. P. 173–186.
4. Chen C., Wu Y., Li J., et al. TBtools-II: A “one for all, all for one” bioinformatics platform for biological big-data mining. Molecular Plant. 2023. Vol. 16, no. 11. P. 1733–1742.
5. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. P. 785–794. DOI: 10.1145/2939672.2939785.
6. Feldman D., Schmidt M., Sohler C. Turning big data into tiny data: Constant-size coresets for k-means, PCA, and projective clustering. SIAM Journal on Computing. 2020. Vol. 49, no. 3. P. 601–657.
7. Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics. 2001. Vol. 29, no. 5. P. 1189–1232. DOI: 10.1214/aos/1013203451.
8. Gama J., Žliobaitė I., Bifet A., Pechenizkiy M., Bouchachia A. A Survey on Concept Drift Adaptation. ACM Computing Surveys. 2014. Vol. 46, no. 4. Article 44.
9. Gunturi S. K., Roy S. Ensemble Machine Learning Models for the Detection of Energy Theft in Smart Grids Using Smart Meter Data. Energy. 2021. Vol. 233. Article 121097. DOI: 10.1016/j.energy.2021.121097.
10. Hyperledger Fabric Documentation (Official documentation). URL: https://hyperledger-fabric.readthedocs.io/ (accessed: 05.03.2026).

11. Imran S., Mahmood T., Morshed A., et al. Big data analytics in healthcare: A systematic literature review and roadmap for practical implementation. IEEE/CAA Journal of Automatica Sinica. 2020. Vol. 8, no. 1. P. 1–22.

12. ISO/IEC 23894:2023. Artificial intelligence — Guidance on risk management. 2023. URL: https://www.iso.org/standard/77304.html (accessed: 05.03.2026).

13. ISO/IEC 42001:2023. Artificial intelligence — Management system. 2023. URL: https://www.iso.org/standard/81230.html (accessed: 05.03.2026).

14. Ji X., Li T. Analysis on Safety Performance Improvement of Smart Metering System Based on Big data. Procedia Computer Science. 2025. Vol. 262. P. 909–918. DOI: 10.1016/j.procs.2025.05.125.

15. Kabalci Y. A survey on smart metering and smart grid communication. Renewable and Sustainable Energy Reviews. 2016. Vol. 57. P. 302–318. DOI: 10.1016/j.rser.2015.12.114.

16. Kafka Documentation (Apache Kafka). URL: https://kafka.apache.org/documentation/ (accessed: 05.03.2026).

17. Kawoosa A. I., et al. Using machine learning ensemble method for detection of energy theft in smart meters. IET Generation, Transmission & Distribution. 2023. (Bibliographic details уточнюються за виданням/випуском).

18. Ke G., Meng Q., Finley T., et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems (NeurIPS). 2017.

19. Kumar P., Lin Y., Bai G., Paverd A., Dong J. S., Martin A. Smart Grid Metering Networks: A Survey on Security, Privacy and Open Research Issues. IEEE Communications Surveys & Tutorials. 2019. Vol. 21, no. 3. P. 2886–2927.

20. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest. Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM). 2008. P. 413–422. DOI: 10.1109/ICDM.2008.17.

21. Liu H., Ong Y. S., Shen X., et al. When Gaussian process meets big data: A review of scalable GPs. IEEE Transactions on Neural Networks and Learning Systems. 2020. Vol. 31, no. 11. P. 4405–4423.

22. Lundberg S. M., Lee S.-I. A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems (NeurIPS). 2017. Vol. 30. P. 4768–4777.

23. Martin I. W. R., Nagel S. Market efficiency in the age of big data. Journal of Financial Economics. 2022. Vol. 145, no. 1. P. 154–177.

24. Misra N. N., Dixit Y., Al-Mallahi A., et al. IoT, big data, and artificial intelligence in agriculture and food industry. IEEE Internet of Things Journal. 2020. Vol. 9, no. 9. P. 6305–6324.

25. NIST. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1. 2023. DOI: 10.6028/NIST.AI.100-1.

26. NIST. Guidelines for Smart Grid Cybersecurity. NISTIR 7628 Revision 1. 2018. URL: https://nvlpubs.nist.gov/nistpubs/ir/2014/nist.ir.7628r1.pdf (accessed: 05.03.2026).

27. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union. 2024. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng (accessed: 05.03.2026).

28. scikit-learn: Machine Learning in Python (Official documentation). URL: https://scikit-learn.org/ (accessed: 05.03.2026).

29. Shu Jie, Zhang Zhijing. Optimization analysis of power marketing measurement system based on big data technology. Integrated Circuit Application. 2024. Vol. 41, no. 2. P. 232–233.

30. Spark Structured Streaming Programming Guide (Apache Spark official documentation). URL: https://spark.apache.org/docs/latest/streaming/index.html (accessed: 05.03.2026).

31. Talwar S., Kaur P., Fosso Wamba S., et al. Big Data in operations and supply chain management: a systematic literature review and future research agenda. International Journal of Production Research. 2021. Vol. 59, no. 11. P. 3509–3534.

32. Tian Jing, Liu Bing, Wu Jianhong, Wang Zhiqiang. Design and test research of LNG ship metering system. Shandong Chemical Industry. 2024. Vol. 53, no. 20. P. 214–216.

33. Wang J., Yang Y., Wang T., et al. Big data service architecture: a survey. Journal of Internet Technology. 2020. Vol. 21, no. 2. P. 393–405.

34. Wang Z., et al. State Analysis of Grouped Smart Meters Driven by Random Forest and Isolation Forest. Electronics. 2025. Vol. 14, no. 15. Article 3105.

35. Yang J., Li Y., Liu Q., et al. Brief introduction of medical database and data mining technology in big data era. Journal of Evidence-Based Medicine. 2020. Vol. 13, no. 1. P. 57–69.

36. Zeng Yan, Ye Xiaoying, Wang Kang, Gao Yanhao, Zheng Hongliang. Design of an intelligent unattended metering system. Industrial Technology and Vocational Education. 2024. Vol. 22, no. 5. P. 1–5.

37. Zhang Yongmin, Liu Shengnan, Yi Ling, An Jiyuan. Power metering system based on filtering to avoid power theft behavior dynamic identification method. Mechanical Design and Manufacturing Engineering. 2024. Vol. 53, no. 3. P. 129–132.

38. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016-07-01. Вид. офіц. Київ : УкрНДНЦ, 2016. 16 с.

39. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

40. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

41. Шорін В.С. Ансамблеві методи машинного навчання для виявлення аномалій у смарт-системі обліку. Інтелектуальні комп’ютерні системи та мережі» (ІКСМ) : матеріали IV Всеукраїнської науково-практичної конференції молодих вчених і студентів (м. Тернопіль, ЗУНУ, 20 травня 2026 р.). Тернопіль, 2026. С. 75-78.

Додаток А

Лістинги програмного забезпечення

Лістинг А.1 – Конфігурація експерименту

```
# config.yaml

experiment:
  random_state: 42
  output_dir: "artifacts"

dataset:
  n_records: 216000
  n_meters: 5000
  normal_ratio: 0.85
  attack_ratio: 0.15
  n_attack_types: 12
  n_features: 32
  start_date: "2025-01-01 00:00:00"
  freq: "30S"

split:
  train_ratio: 0.70
  validation_ratio: 0.20
  test_ratio: 0.10

preprocessing:
  noise_filtering: true
  remove_invalid_messages: true
  fill_missing_values: true
  normalize_features: true

model:
  type: "random_forest"
  n_estimators: 300
  max_depth: 14
  min_samples_leaf: 4
  class_weight: "balanced"

evaluation:
  threshold: 0.50
  batch_size: 500
  ddos_attack_share: 0.25
```

Лістинг А.2 – Генерація гібридного датасету смарт-обліку

```
# dataset_generator.py

import numpy as np
import pandas as pd


ATTACK_TYPES = [
    "sql_injection",
    "ddos_syn_flood",
    "ddos_udp_flood",
    "ddos_http_slow",
    "mitm_arp_spoofing",
    "mitm_dhcp_hijacking",
    "false_data_injection",
    "protocol_fuzzing",
    "ransomware_activity",
    "apt_firmware_update",
```

```
    "flat_consumption_profile",
    "zero_day_attack"
]


def generate_smart_meter_dataset(
    n_records: int = 216000,
    n_meters: int = 5000,
    attack_ratio: float = 0.15,
    random_state: int = 42
) -> pd.DataFrame:
    rng = np.random.default_rng(random_state)

    timestamps = pd.date_range(
        start="2025-01-01 00:00:00",
        periods=n_records,
        freq="30S"
    )

    meter_ids = rng.integers(1, n_meters + 1, size=n_records)
    is_attack = rng.choice(
        [0, 1],
        size=n_records,
        p=[1 - attack_ratio, attack_ratio]
    )

    attack_type = np.where(
        is_attack == 1,
        rng.choice(ATTACK_TYPES, size=n_records),
        "normal"
    )

    consumption = rng.normal(loc=2.4, scale=0.7, size=n_records)
    voltage = rng.normal(loc=220, scale=8, size=n_records)
    current = rng.normal(loc=11, scale=3, size=n_records)

    packet_rate = rng.normal(loc=1200, scale=250, size=n_records)
    bandwidth_usage = rng.normal(loc=2.1, scale=0.5, size=n_records)
    protocol_compliance_index = rng.normal(loc=0.96, scale=0.03, size=n_records)

    # Модифікація параметрів для атак
    attack_idx = is_attack == 1

    consumption[attack_idx] *= rng.uniform(0.1, 4.5, size=attack_idx.sum())
    voltage[attack_idx] += rng.normal(0, 25, size=attack_idx.sum())
    packet_rate[attack_idx] *= rng.uniform(2, 10, size=attack_idx.sum())
    bandwidth_usage[attack_idx] *= rng.uniform(2, 8, size=attack_idx.sum())
    protocol_compliance_index[attack_idx] -= rng.uniform(0.10, 0.45,
size=attack_idx.sum())

    df = pd.DataFrame({
        "meter_id": meter_ids,
        "timestamp": timestamps,
        "consumption": np.clip(consumption, 0, None),
        "voltage": voltage,
        "current": current,
        "packet_rate": np.clip(packet_rate, 0, None),
        "bandwidth_usage": np.clip(bandwidth_usage, 0, None),
        "protocol_compliance_index": np.clip(protocol_compliance_index, 0, 1),
        "attack_type": attack_type,
        "label": is_attack
    })

    return df
```

```
if __name__ == "__main__":
    df = generate_smart_meter_dataset()
    df.to_csv("artifacts/hybrid_smart_meter_dataset.csv", index=False)
    print("Dataset saved:", df.shape)
    print(df["label"].value_counts(normalize=True))
```

Лістинг А.3 – Попередня обробка та очищення даних

```
# preprocessing.py

import numpy as np
import pandas as pd


def clean_invalid_messages(df: pd.DataFrame) -> pd.DataFrame:
    df = df.copy()

    df["timestamp"] = pd.to_datetime(df["timestamp"], errors="coerce")
    df = df.dropna(subset=["timestamp", "meter_id", "consumption"])

    df = df.drop_duplicates(subset=["meter_id", "timestamp"], keep="last")

    numeric_cols = [
        "consumption",
        "voltage",
        "current",
        "packet_rate",
        "bandwidth_usage",
        "protocol_compliance_index"
    ]

    for col in numeric_cols:
        df[col] = pd.to_numeric(df[col], errors="coerce")

    df = df.dropna(subset=numeric_cols)

    df.loc[df["consumption"] < 0, "consumption"] = np.nan
    df.loc[df["voltage"] <= 0, "voltage"] = np.nan
    df.loc[df["current"] < 0, "current"] = np.nan

    df = df.fillna(method="ffill").fillna(0)

    return df


def filter_noise_by_sigma(df: pd.DataFrame, columns: list[str], sigma: float =
3.0) -> pd.DataFrame:
    df = df.copy()

    for col in columns:
        mean_value = df[col].mean()
        std_value = df[col].std()

        lower = mean_value - sigma * std_value
        upper = mean_value + sigma * std_value

        df[col] = df[col].clip(lower=lower, upper=upper)

    return df


def preprocess_dataset(df: pd.DataFrame) -> pd.DataFrame:
    df = clean_invalid_messages(df)

    df = filter_noise_by_sigma(
        df,
```

```
        columns=[
            "consumption",
            "voltage",
            "current",
            "packet_rate",
            "bandwidth_usage"
        ],
        sigma=3.0
    )

    df = df.sort_values(["meter_id", "timestamp"]).reset_index(drop=True)
    return df
```

Лістинг А.4 – Формування 32-вимірного простору ознак

```
# feature_engineering.py

import numpy as np
import pandas as pd


def entropy(values: pd.Series) -> float:
    counts = values.value_counts(normalize=True)
    return float(-(counts * np.log2(counts + 1e-9)).sum())


def build_32_features(df: pd.DataFrame) -> pd.DataFrame:
    df = df.copy()
    df = df.sort_values(["meter_id", "timestamp"])

    grouped = df.groupby("meter_id")

    df["consumption_delta"] = grouped["consumption"].diff().fillna(0)
    df["consumption_rel_delta"] = (
        df["consumption_delta"] /
        (grouped["consumption"].shift(1).fillna(0) + 1e-6)
    )

    for window in [5, 10, 30]:
        rolling = grouped["consumption"].rolling(window, min_periods=1)

        df[f"mean_{window}"] = rolling.mean().reset_index(level=0, drop=True)
        df[f"std_{window}"] = rolling.std().reset_index(level=0,
drop=True).fillna(0)
        df[f"min_{window}"] = rolling.min().reset_index(level=0, drop=True)
        df[f"max_{window}"] = rolling.max().reset_index(level=0, drop=True)

    df["voltage_delta"] = grouped["voltage"].diff().fillna(0)
    df["current_delta"] = grouped["current"].diff().fillna(0)

    df["power_estimate"] = df["voltage"] * df["current"]
    df["load_factor"] = df["consumption"] / (df["power_estimate"] + 1e-6)

    df["packet_rate_delta"] = grouped["packet_rate"].diff().fillna(0)
    df["bandwidth_delta"] = grouped["bandwidth_usage"].diff().fillna(0)

    df["protocol_deviation"] = 1 - df["protocol_compliance_index"]

    df["hour"] = df["timestamp"].dt.hour
    df["day_of_week"] = df["timestamp"].dt.dayofweek
    df["is_weekend"] = (df["day_of_week"] >= 5).astype(int)

    df["zero_consumption_flag"] = (df["consumption"] == 0).astype(int)
    df["high_packet_flag"] = (df["packet_rate"] >
df["packet_rate"].quantile(0.95)).astype(int)
```

```
    df["high_bandwidth_flag"] = (df["bandwidth_usage"] >
df["bandwidth_usage"].quantile(0.95)).astype(int)

    df["time_domain_fluctuation_entropy"] = (
        grouped["consumption"]
        .rolling(10, min_periods=2)
        .apply(lambda x: entropy(pd.Series(np.round(x, 2))), raw=False)
        .reset_index(level=0, drop=True)
        .fillna(0)
    )

    df["frequency_energy_ratio"] = (
        df["std_10"] / (df["mean_10"] + 1e-6)
    )

    df["communication_entropy"] = (
        grouped["packet_rate"]
        .rolling(10, min_periods=2)
        .apply(lambda x: entropy(pd.Series(np.round(x, 0))), raw=False)
        .reset_index(level=0, drop=True)
        .fillna(0)
    )

    feature_columns = [
        "consumption", "voltage", "current", "packet_rate", "bandwidth_usage",
        "protocol_compliance_index", "consumption_delta",
"consumption_rel_delta",
        "mean_5", "std_5", "min_5", "max_5",
        "mean_10", "std_10", "min_10", "max_10",
        "mean_30", "std_30", "min_30", "max_30",
        "voltage_delta", "current_delta", "power_estimate", "load_factor",
        "packet_rate_delta", "bandwidth_delta", "protocol_deviation",
        "hour", "day_of_week", "is_weekend",
        "time_domain_fluctuation_entropy", "frequency_energy_ratio"
    ]

    # Додаткові службові поля залишаються для аналізу результатів
    result = df[["meter_id", "timestamp", "attack_type", "label"] +
feature_columns].copy()

    return result, feature_columns
```

Лістинг А.5 – Навчання ансамблевої моделі Random Forest

```
# train_model.py

import joblib
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score


def split_dataset_by_ratio(df: pd.DataFrame):
    df = df.sort_values("timestamp").reset_index(drop=True)

    n = len(df)
    train_end = int(n * 0.70)
    val_end = int(n * 0.90)

    train_df = df.iloc[:train_end]
    val_df = df.iloc[train_end:val_end]
    test_df = df.iloc[val_end:]

    return train_df, val_df, test_df
```

```
def train_random_forest(train_df, val_df, feature_columns):
    X_train = train_df[feature_columns]
    y_train = train_df["label"]

    X_val = val_df[feature_columns]
    y_val = val_df["label"]

    model = RandomForestClassifier(
        n_estimators=300,
        max_depth=14,
        min_samples_leaf=4,
        class_weight="balanced",
        random_state=42,
        n_jobs=-1
    )

    model.fit(X_train, y_train)

    val_pred = model.predict(X_val)

    metrics = {
        "accuracy": accuracy_score(y_val, val_pred),
        "precision": precision_score(y_val, val_pred),
        "recall": recall_score(y_val, val_pred),
        "f1": f1_score(y_val, val_pred)
    }

    joblib.dump(model, "artifacts/random_forest_model.joblib")
    return model, metrics


if __name__ == "__main__":
    df = pd.read_csv("artifacts/features_32.csv")
    feature_columns = [c for c in df.columns if c not in ["meter_id",
"timestamp", "attack_type", "label"]]

    train_df, val_df, test_df = split_dataset_by_ratio(df)
    model, metrics = train_random_forest(train_df, val_df, feature_columns)

    print("Validation metrics:")
    print(metrics)
```

Лістинг А.6 – Оцінювання точності, затримки та пропускної здатності

```
# evaluation.py

import time
import json
import joblib
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix


def calculate_false_alarm_rate(y_true, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    return fp / (fp + tn + 1e-9)


def evaluate_model(model_path: str, test_df: pd.DataFrame, feature_columns:
list[str]):
    model = joblib.load(model_path)

    X_test = test_df[feature_columns]
    y_test = test_df["label"]
```

```
    start_time = time.perf_counter()
    y_pred = model.predict(X_test)
    end_time = time.perf_counter()

    total_time = end_time - start_time
    response_delay_ms = (total_time / len(test_df)) * 1000
    throughput = len(test_df) / total_time

    results = {
        "accuracy": accuracy_score(y_test, y_pred),
        "precision": precision_score(y_test, y_pred),
        "recall": recall_score(y_test, y_pred),
        "f1": f1_score(y_test, y_pred),
        "false_alarm_rate": calculate_false_alarm_rate(y_test, y_pred),
        "response_delay_ms": response_delay_ms,
        "throughput_records_per_second": throughput
    }

    return results, y_pred


def evaluate_ddos_detection(test_df: pd.DataFrame, y_pred):
    ddos_mask = test_df["attack_type"].isin([
        "ddos_syn_flood",
        "ddos_udp_flood",
        "ddos_http_slow"
    ])

    ddos_df = test_df[ddos_mask]
    ddos_pred = y_pred[ddos_mask]

    if len(ddos_df) == 0:
        return 0.0

    interception_rate = ddos_pred.sum() / len(ddos_df)
    return float(interception_rate)


if __name__ == "__main__":
    df = pd.read_csv("artifacts/features_32.csv")
    feature_columns = [c for c in df.columns if c not in ["meter_id",
"timestamp", "attack_type", "label"]]

    n = len(df)
    test_df = df.iloc[int(n * 0.90):].copy()

    results, y_pred = evaluate_model(
        "artifacts/random_forest_model.joblib",
        test_df,
        feature_columns
    )

    results["ddos_interception_rate"] = evaluate_ddos_detection(test_df, y_pred)

    with open("artifacts/evaluation_results.json", "w", encoding="utf-8") as f:
        json.dump(results, f, ensure_ascii=False, indent=2)

    print(results)
```

Лістинг А.7 – Формування журналу спрацювань і картки аномалії

```
# alert_log.py

import joblib
import pandas as pd
```

```
def create_alert_log(test_df: pd.DataFrame, y_pred, model_path: str,
feature_columns: list[str]):
    model = joblib.load(model_path)

    test_df = test_df.copy()
    test_df["predicted_label"] = y_pred

    if hasattr(model, "predict_proba"):
        test_df["risk_score"] = model.predict_proba(test_df[feature_columns])[:,
1]
    else:
        test_df["risk_score"] = test_df["predicted_label"]

    alerts = test_df[test_df["predicted_label"] == 1].copy()

    feature_importance = pd.Series(
        model.feature_importances_,
        index=feature_columns
    ).sort_values(ascending=False)

    top_features = list(feature_importance.head(5).index)

    alerts["top_features"] = ", ".join(top_features)

    alerts["comment"] = alerts.apply(
        lambda row: build_alert_comment(row, top_features),
        axis=1
    )

    alerts[[
        "meter_id",
        "timestamp",
        "attack_type",
        "risk_score",
        "top_features",
        "comment"
    ]].to_csv("artifacts/alert_log.csv", index=False)

    return alerts


def build_alert_comment(row, top_features):
    comments = []

    if "protocol_deviation" in top_features and row.get("protocol_deviation", 0)
> 0.2:
        comments.append("значне відхилення від протоколу")

    if "packet_rate" in top_features and row.get("packet_rate", 0) > 3000:
        comments.append("підвищена інтенсивність пакетів")

    if "consumption_delta" in top_features and abs(row.get("consumption_delta",
0)) > 2:
        comments.append("різка зміна споживання")

    if "time_domain_fluctuation_entropy" in top_features:
        comments.append("нетипова динаміка часового ряду")

    if not comments:
        comments.append("підозріле поєднання ознак")

    return "; ".join(comments)
```

```
if __name__ == "__main__":
    df = pd.read_csv("artifacts/features_32.csv")
    feature_columns = [c for c in df.columns if c not in ["meter_id",
"timestamp", "attack_type", "label"]]

    n = len(df)
    test_df = df.iloc[int(n * 0.90):].copy()

    model = joblib.load("artifacts/random_forest_model.joblib")
    y_pred = model.predict(test_df[feature_columns])

    alerts = create_alert_log(
        test_df,
        y_pred,
        "artifacts/random_forest_model.joblib",
        feature_columns
    )

    print("Кількість спрацювань:", len(alerts))
```

Лістинг А.8 – Формування зведеної таблиці результатів експерименту

```
# report_results.py

import json
import pandas as pd


def create_experiment_summary():
    with open("artifacts/evaluation_results.json", "r", encoding="utf-8") as f:
        results = json.load(f)

    summary = pd.DataFrame([
        {
            "Модель": "Random Forest",
            "Accuracy": round(results["accuracy"], 4),
            "Precision": round(results["precision"], 4),
            "Recall": round(results["recall"], 4),
            "F1": round(results["f1"], 4),
            "False Alarm Rate": round(results["false_alarm_rate"], 4),
            "Response delay, ms": round(results["response_delay_ms"], 4),
            "Throughput, records/s":
round(results["throughput_records_per_second"], 2),
            "DDoS interception rate": round(results["ddos_interception_rate"],
4)
        }
    ])

    summary.to_csv("artifacts/experiment_summary.csv", index=False)
    return summary


if __name__ == "__main__":
    summary = create_experiment_summary()
    print(summary)
```

Лістинг А.9 – Основний сценарій запуску програмного модуля

```
# main.py

from dataset_generator import generate_smart_meter_dataset
from preprocessing import preprocess_dataset
from feature_engineering import build_32_features
from train_model import split_dataset_by_ratio, train_random_forest
from evaluation import evaluate_model, evaluate_ddos_detection
from alert_log import create_alert_log
```

```
from report_results import create_experiment_summary


def main():
    print("1. Генерація експериментального датасету...")
    df = generate_smart_meter_dataset(
        n_records=216000,
        n_meters=5000,
        attack_ratio=0.15,
        random_state=42
    )
    df.to_csv("artifacts/hybrid_smart_meter_dataset.csv", index=False)

    print("2. Попередня обробка даних...")
    df_clean = preprocess_dataset(df)

    print("3. Формування 32-вимірного простору ознак...")
    df_features, feature_columns = build_32_features(df_clean)
    df_features.to_csv("artifacts/features_32.csv", index=False)

    print("4. Поділ даних на train/validation/test...")
    train_df, val_df, test_df = split_dataset_by_ratio(df_features)

    print("5. Навчання ансамблевої моделі Random Forest...")
    model, val_metrics = train_random_forest(train_df, val_df, feature_columns)
    print("Validation metrics:", val_metrics)

    print("6. Оцінювання моделі на тестовій вибірці...")
    results, y_pred = evaluate_model(
        "artifacts/random_forest_model.joblib",
        test_df,
        feature_columns
    )

    results["ddos_interception_rate"] = evaluate_ddos_detection(test_df, y_pred)
    print("Test results:", results)

    print("7. Формування журналу спрацювань...")
    create_alert_log(
        test_df,
        y_pred,
        "artifacts/random_forest_model.joblib",
        feature_columns
    )

    print("8. Формування зведеної таблиці результатів...")
    create_experiment_summary()

    print("Експеримент завершено.")


if __name__ == "__main__":
    main()
```

Додаток Б

Апробація результатів роботи

Почесні співголови конференції:

Десятнюк О.М., ректор Західноукраїнського національного університету, д-р економічних наук, професор;

Дивак М.П., д-р технічних наук, професор, проректор з наукової роботи ЗУНУ;

Голова конференції:

Березький О.М., д-р технічних наук, професор, професор кафедри комп'ютерної інженерії Західноукраїнський національний університет;

Заступники голови конференції:

Мельник Г.М., к.т.н, доцент, Західноукраїнський національний університет;

Піцун О.Й., к.т.н. доцент, Західноукраїнський національний університет;

ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Склад програмного комітету:

Семанюк В.З., д-р економічних наук, професор, начальник науково-дослідної частини Західноукраїнського національного університету

Антощук С.Г., д.т.н, професор, Національний університет «Одеська політехніка»;

Баловсяк С.В., д.т.н, професор, Чернівецький національний університет

Бармак О.В., д.т.н., професор, Хмельницький національний університет

Батько Ю.М., к.т.н., доцент, Західноукраїнський національний університет

Березька К.М.,к.т.н.,доцент, Західноукраїнський національний університет

Винокурова О.А., д.т.н., професор, Львівський національний університет імені Івана Франка

Возна Н.Я., д.т.н., професор, Західноукраїнський національний університет;

Говорущенко Т.О., д.т.н., професор, Хмельницький національний університет;

Дубчак Л.О., к.т.н., доцент, Західноукраїнський національний університет;

Дунець Р.Б., д.т.н., професор, НУ "Львівська політехніка";

Ізонін І.В., д.т.н., доцент, НУ "Львівська політехніка";

Комар М.П., д.т.н, професор, Західноукраїнський національний університет;

Литвиненко В.І., д.н.т, професор, Національний технічний університет України "Київський політехнічний інститут";

Лупенко С.А., д.т.н., професор, Опольський технологічний університет, Польща;

Лящинський П.Б., доктор філософії з комп'ютерних наук, НУ "Львівська політехніка" ;

Мельникова Н.І., д.т.н., професор, НУ "Львівська політехніка" ;

Мулеса О.Ю., д.т.н., професор, Пряшівський університет, Словаччина

Пелешко Д.Д., д.т.н., професор, Львівський національний університет імені Івана Франка;

Сельський П.Р., д.м.н., професор, Тернопільський національний медичний університет імені І. Я. Горбачевського ;
Субботін С.О., д.т.н., професор, Національний університет «Запорізька політехніка» ;
Теслюк В.М., д.т.н., професор, НУ "Львівська політехніка" ;
Тимченко Л.І., д.т.н., професор, Державний університет інфраструктури та технологій;
Цмоць І.Г., д.т.н., професор, НУ "Львівська політехніка" ;
Якименко І.З., к.т.н., доцент, Західноукраїнський національний університет;
Яровий А.А., д.т.н., професор, Вінницький національний технічний університет ;
Яцків В.В., д.т.н., професор, Західноукраїнський національний університет.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Склад організаційного комітету:

Мельник Г.М. к.т.н., доцент, Західноукраїнський національний університет

Піцун О.Й. к.т.н., доцент, Західноукраїнський національний університет

Технічна підтримка:

Зінькевич О. В. студентка, Західноукраїнський національний університет

Кіт М. О. студентка, Західноукраїнський національний університет

Лебединський Т.В. студент,Західноукраїнський національний університет;

Кришталь Е.Р., студент, Західноукраїнський національний університет;

Метою конференції є представлення та обговорення наукових і практичних результатів, сприяння активізації творчої і інноваційної діяльності студентів, аспірантів і молодих вчених.
Конференція проводиться із залученням Ради Молодих Вчених та Студентського Наукового Товариства ФКІТ ЗУНУ.

IV Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі» (ІКСМ весна 2026), м. Тернопіль, ЗУНУ, 29 травня 2026 р. Тернопіль, 2026

Адреса:
Вул. О. Теліги, 8, корпус №6, Тернопіль
URL: https://ki.wunu.edu.ua/conference/
e-mail: kistudconference@gmail.com

Тези доповідей подаються за оригіналом рукопису

Вовк В.С.
Генерування художніх зображень за текстовим описом 58
Нагина М. В., Зварич В. І.
Позитивно-класовий підхід до виявлення порушень носіння засобів індивідуального захисту 61
Бойко Н.Р.
Виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання 64
Грицюк Г.І.
Методи машинного навчання та аналізу великих даних для класифікації програмних проєктів 68
Заєць О.Ю.
Програмна система аналізу мережевих логів з використанням NLP-підходу для виявлення аномалій 72
Шорін В.С.
Ансамблеві методи машинного навчання для виявлення аномалій у смарт-системі обліку 75
Ярунів М.А.
Програмний модуль розподіленого зберігання та паралельної обробки фінансових даних 78
Пугінець А. Ю., Зварич В. І.
Смарт-система для догляду за кімнатними рослинами 82
Вишинська Н.М.
Інформаційна система рекомендацій закладів харчування з використанням методів машинного навчання 84
Тарбай Ю.О.
Система підтримки прийняття рішень для оцінки фінансових ризиків підприємства на основі технології аналізу великих даних 86
Григорян Д. М.
Методи та засоби класифікації акне на основі методів комп'ютерного зору 89
Василиків В.Д.
Інформаційна система визначення фізичних характеристик тварин з використанням технологій комп'ютерного зору 91
Askerov V.V.
Risk-aware architecture for adaptive management of secure transactions in permissioned blockchain systems 94
Вім Р.В.
Формування репрезентативного набору макрозображень тканин для застосування штучного інтелекту в циркулярній економіці 97
Тимофієв І.А.
Інтелектуальна інформаційна система для виявлення соціально-деструктивних і депресивних наративів у текстовому контенті 100
Шурипа М.О.
Метод валідації антропометричної пози людини для фотограмметричного зняття мірок на основі комп'ютерного зору 103
Юрченко Д.Ю.
Дослідження методу візуального пояснення результатів нейромережевого виявлення маніпулятивних технік у текстовому контенті 106

частку правильних рішень, незначну кількість хибних спрацьовувань і добру здатність системи виявляти аномальні події. Аналіз матриці помилок також показав, що основна частина записів класифікується правильно, а кількість пропущених аномалій залишається низькою.

Порівняння із базовими підходами, зокрема Decision Tree, GBM і CNN-BiLSTM, показало, що запропонована система має кращі підсумкові результати. Найважливішою перевагою є підвищення recall, тобто покращення здатності виявляти аномальні мережеві події, що має особливе значення для задач кібербезпеки. Це підтвердило доцільність використання NLP-підходу, за якого мережевий лог розглядається як цілісний текстовий опис події.

Отже, поставлену задачу розв'язано: розроблено концепцію програмної системи аналізу мережевих логів, обґрунтовано її архітектуру та підтверджено ефективність запропонованого підходу експериментальними результатами. Отримані висновки засвідчили, що використання NLP-підходу є перспективним напрямом для побудови інтелектуальних систем виявлення аномалій у мережевих логах.

Список літератури

1. Hubballi N., Suryanarayanan V. False alarm minimization techniques in signature-based intrusion detection systems: a survey. *Computer Communications*. 2014. Vol. 49. P. 1–17.

2. Masdari M., Khezri H. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing*. 2020. Vol. 92. Article 106301.

3. Louk M. H. L., Tama B. A. Dual-IDS: a bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. *Expert Systems with Applications*. 2023. Vol. 213. Article 119030.

4. Rai K., Syamala Devi M., Guleria A. Decision tree based algorithm for intrusion detection. *International Journal of Advanced Networking and Applications*. 2016. Vol. 7, no. 4. P. 2828–2834.

5. Sinha J., Manollas M. Efficient deep CNN-BiLSTM model for network intrusion detection. In: *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*. 2020.

Шорін В.С.
студент 4 курсу ФКІТ ЗУНУ
Науковий керівник д.т.н., професор Комар М.П., кафедра ІОСУ ЗУНУ

АНСАМБЛЕВІ МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У СМАРТ-СИСТЕМІ ОБЛІКУ

Вступ. Смарт-системи обліку застосовуються для автоматизованого збору показників споживання електроенергії, газу чи води та подальшого використання цих даних у розрахунках і керуванні інфраструктурою [1]. З технічного погляду такі рішення поєднують лічильники, канали зв'язку, серверні компоненти та програмні підсистеми, що забезпечують приймання, збереження й аналіз вимірювань [2]. Практика впровадження подібних систем демонструє, що ключовим ресурсом стають саме дані: вони надходять потоково, накопичуються у великих обсягах, потребують очищення та контролю якості [3, 4].

Постановка задачі. Результати аналізу предметної області та класифікації аномалій і загроз свідчать, що дані смарт-обліку є потоковими, часовими та схильними до помилок як технічного, так і безпекового характеру. За таких умов завдання виявлення аномалій доцільно формулювати не як одноразову "перевірку файлу", а як програмний модуль, який може бути вбудований у контур збору та аналізу вимірювань і працювати з регулярними надходженнями показників. Оскільки виявлення аномалій часто використовується як елемент моніторингу та реагування, важливо заздалегідь визначити, що саме вважати "аномалією", які дані є на вході, який формат результату потрібен на виході, а також якими метриками оцінюватиметься ефективність рішення [1, 4].

Задача дослідження полягає в тому, щоб проєктувати прототип програмного модуля, який на основі даних смарт-лічильників автоматично визначає підозрілі або нетипові

вимірювання/інтервали та формує рішення про наявність аномалії. Враховуючи поширені сценарії викривлення даних, модуль має бути орієнтований як на виявлення збоїв якості даних (пропуски, дублікати, неможливі значення), так і на виявлення поведінкових відхилень, що можуть бути пов'язані з маніпуляціями або крадіжками ресурсу [5].

Об'єктом дослідження є процеси збору та аналізу даних у смарт-системах обліку. Предметом дослідження є методи та програмні засоби виявлення аномалій у даних смарт-обліку з використанням ансамблевих моделей машинного навчання. Метою дослідження є проєктування програмного модуля виявлення аномалій у смарт-системі обліку на основі ансамблевих методів машинного навчання та оцінювання його ефективності на даних обліку.

Основний матеріал. Побудова програмного модуля виявлення аномалій у смарт-системі обліку доцільна як реалізація послідовного конвеєра обробки даних: від надходження показників лічильників до формування рішення та подальшого використання результату в моніторингу [4]. Такий підхід відповідає природі смарт-обліку, де дані надходять регулярно, мають часовий контекст і можуть містити пропуски або викривлення [1]. Додатково враховується вимога до масштабованості, оскільки обсяги вимірювань у великих системах набувають ознак "big data".

Запропонований підхід ґрунтується на п'яти послідовних етапах:

1. Приймання даних.
2. Первинний контроль якості та очищення.
3. Формування ознак.
4. Детектування аномалій.
5. Фіксація результату та інтеграція з моніторингом.

Кожен етап має власну роль у зменшенні помилок і підвищенні стійкості детектора. Зокрема, від якості формування ознак залежить здатність моделі відрізняти природні коливання споживання від підозрілих відхилень, тоді як очищення даних зменшує кількість "хибних" аномалій, що виникають через технічні дефекти потоку [4].

На рисунку 1 представлено узагальнену архітектуру підходу у вигляді потоку даних між логічними компонентами.

Рисунок 1 – Архітектура підходу до виявлення аномалій у смарт-обліку

Компоненти та їх функції:

1) Джерело даних (смарт-лічильники та допоміжні компоненти). На вході системи знаходяться смарт-лічильники та інфраструктура передавання показників. Дані містять ідентифікатор точки обліку, часову мітку, значення споживання, а за наявності – службові ознаки стану чи якості каналу [1, 4]. У рамках роботи припускається, що ці дані надходять у вигляді потоку записів або пакета записів за певний період.

2) Компонент приймання. Залежно від сценарію реалізації приймання може здійснюватися:

- пакетно (читання файлів/таблиць);
- потоково через брокер повідомлень (наприклад, Kafka) або інші механізми доставки подій.

Потоковий підхід є типовим для сервісної архітектури великих даних, де дані надходять у реальному часі та мають оброблятися без накопичення великих затримок [4]. У разі використання Spark Structured Streaming можливе формування мікропакетів та обробка за принципом "near real-time".

3) Первинний контроль якості та очищення. На цьому етапі виконуються прості й швидкі перевірки: коректність типів даних, часова впорядкованість, видалення дублікатів, заповнення або маркування пропусків, фільтрація очевидно некоректних значень. Мета етапу полягає не в "пошуку аномалій" у змістовому сенсі, а в усуненні технічного сміття, яке інакше спричинить зайві спрацювання детектора. Така логіка узгоджується з практикою використання правил та фільтрацій як базового шару контролю [4, 5].

4) Формування ознак. Оскільки сирі вимірювання є часовими рядами, їх доцільно перетворювати на набір ознак, які відображають контекст. Зазвичай використовуються агрегати за часовими вікнами (середнє, медіана, стандартне відхилення), прирости (різниця між поточним і попереднім значенням), показники стабільності, частка нульових значень, індикатори "надмірної регулярності" тощо [4]. Такий підхід дозволяє подати задачу у формі табличного набору ознак, для якого деревові ансамблі є природним вибором.

5) Модель детектування (ансамблевий класифікатор/детектор). На цьому етапі застосовується навчена модель, наприклад Random Forest [6] або бустингові дерева (Gradient Boosting, XGBoost, LightGBM) [7-9]. Виходом є скор (оцінка ризику) і/або мітка "аномалія/норма". У прикладних задачах, зокрема для виявлення енергетичних крадіжок, ансамблеві моделі розглядаються як практичний інструмент через стійкість до шуму та здатність уловлювати складні комбінації ознак.

6) Компонент результатів: журналювання, сповіщення, інтеграція. Рішення модуля має бути зафіксоване у журналі подій із мінімально необхідним набором полів: meter_id, timestamp, скор/мітка, ключові ознаки, службова інформація [4]. За потреби формується сповіщення для оператора або автоматизованої системи реагування. З позиції кібербезпеки та управління ризиками важливо забезпечити аудит подій і можливість розслідування причин спрацювань.

У рамках роботи можливі два режими, які відрізняються переважно способом приймання даних:

1. Пакетний режим. Дані зчитуються з файлу або бази даних за визначений період; формуються ознаки; виконується детектування; формується звіт або список аномалій. Такий режим простіше реалізувати й перевірити експериментально, зберігаючи коректність постановки задачі.
2. Потоковий режим. Дані надходять безперервно; ознаки обчислюються у ковзних вікнах; рішення формується для кожного запису або мікропакета. Для реалізації цього режиму зазвичай використовуються брокери повідомлень та стримінгові фреймворки. Перевага полягає в оперативності, однак зростають вимоги до синхронізації часу, обліку затримок і обробки пропусків.

Отже, архітектура підходу базується на конвеєрі "дані – якість – ознаки – ансамбль – рішення", який відповідає природі смарт-обліку та дозволяє поєднати інженерні засоби контролю якості з ML-детектуванням. Така організація підвищує стійкість до шуму та пропусків, а ансамблеві методи забезпечують практичний компроміс між якістю, стабільністю та придатністю до впровадження у реальних умовах обліку.

Висновки. Запропонований підхід базується на послідовному конвеєрі обробки даних: приймання показників, первинний контроль якості, очищення, формування ознак, застосування ансамблевої моделі та фіксація результату. Така структура є логічною, оскільки кожен етап виконує окрему функцію: очищення зменшує кількість хибних спрацювань, формування ознак перетворює часові ряди на зручний для аналізу табличний формат, а модель детектування оцінює підозрілість вимірювання. Особливу увагу приділено формуванню ознак, оскільки саме вони дозволяють відобразити поведінку споживання у зрозумілому для моделі вигляді. Доцільними визначено ознаки, що описують середні значення, медіану, стандартне відхилення, прирости, стабільність, частку нульових значень та інші характеристики часових вікон. Це дає змогу відрізняти природні коливання споживання від нетипових відхилень. Для детектування аномалій обґрунтовано використання ансамблевих методів машинного навчання, зокрема Random Forest, Gradient Boosting, XGBoost та LightGBM. Такі моделі є придатними для роботи з табличними ознаками, мають достатню стійкість до шуму та можуть виявляти складні комбінації факторів, які не завжди помітні за допомогою простих правил або порогових

перевірок. Також визначено два можливі режими роботи модуля: пакетний і потоковий. Пакетний режим є простішим для реалізації та експериментальної перевірки, тоді як потоковий режим краще відповідає умовам реального смарт-обліку, де дані надходять безперервно. Водночас потокова реалізація потребує додаткової уваги до синхронізації часу, затримок, пропусків і стабільності обробки.

Список літератури

1. Kabalci Y. A survey on smart metering and smart grid communication. Renewable and Sustainable Energy Reviews. 2016. Vol. 57. P. 302–318.
2. Zeng Yan, Ye Xiaoying, Wang Kang, Gao Yanhao, Zheng Hongliang. Design of an intelligent unattended metering system. Industrial Technology and Vocational Education. 2024. Vol. 22, no. 5. P. 1–5.
3. Ji X., Li T. Analysis on Safety Performance Improvement of Smart Metering System Based on Big data. Procedia Computer Science. 2025. Vol. 262. P. 909–918.
4. Wang J., Yang Y., Wang T., et al. Big data service architecture: a survey. Journal of Internet Technology. 2020. Vol. 21, no. 2. P. 393–405.
5. Zhang Yongmin, Liu Shengnan, Yi Ling, An Jiyuan. Power metering system based on filtering to avoid power theft behavior dynamic identification method. Mechanical Design and Manufacturing Engineering. 2024. Vol. 53, no. 3. P. 129–132.
6. Breiman L. Random Forests. Machine Learning. 2001. Vol. 45, no. 1. P. 5–32.
7. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. P. 785–794.
8. Ke G., Meng Q., Finley T., et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems (NeurIPS). 2017.
9. Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics. 2001. Vol. 29, no. 5. P. 1189–1232.