Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах на основі метода машинного навчання KMeans та блокчейн-орієнтованих журналів подій»

Виконав: ст. 2-го курсу, групи 2КІТС-24м
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Шкробот Є.С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. МБІС
Грицак А.В.
(прізвище та ініціали)
«____» _____ 2025 р.

Опонент: к.т.н., доцент, каф. ОТ
Тарновський М.Г.
(прізвище та ініціали)
«____» _____ 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
"16.12" Грудня 2025 р.

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____ **Юрій ЯРЕМЧУК**

"24" Вересня 2025 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Шкробот Єгор Сергійович
(прізвище, ім`я, по-батькові)

1. Тема роботи Вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах на основі метода машинного навчання KMeans та блокчейнорієнтованих журналів подій
Керівник роботи к.т.н доцент кафедри МБІС Грицак А. В.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "24" вересня 2025 року № 313
2. Строк подання студентом роботи за тиждень до захисту
3. Вихідні дані до роботи: журнали подій корпоративної інформаційної системи (логи автентифікації, доступу до ресурсів, мережевої активності);
4. Зміст текстової частини: У роботі досліджено сучасні загрози інформаційній безпеці та методи виявлення аномалій у корпоративних інформаційних системах. Розроблено комбінований метод на основі KMeans та локальних моделей Isolation Forest із використанням блокчейн-журналювання подій. Реалізовано програмну систему та проведено її тестування. Виконано економічне обґрунтування доцільності впровадження розробки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень): У першому розділі наведено 6 рисунків, у другому розділі — 4 рисунки, у третьому розділі — 5 рисунків. Загальна кількість рисунків у роботі становить 15.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	**к.т.н доцент кафедри МБІС Грицак А. В.**		
Економічна частина	**к.т.н доцент кафедри ЕПВМ Ратушняк О. Г.**		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1	**Вибір та затвердження теми магістерської роботи**	**24.09.2025**	**24.09.2025**	
2	**Аналіз літературних джерел та сучасних загроз**	**25.09.2025**	**14.10.2025**	
3	**Дослідження методів Isolation Forest та KMeans**	**14.10.2025**	**21.10.2025**	
4	**Розробка комбінованого алгоритму**	**22.10.2025**	**01.11.2025**	
5	**Розробка алгоритму блокчейнжурналювання**	**02.11.2025**	**07.11.2025**	
6	**Програмна реалізація системи**	**08.11.2025**	**14.11.2025**	
7	**Тестування та аналіз результатів**	**15.11.2025**	**15.11.2025**	
8	**Оформлення економічної частини**	**16.11.2025**	**17.11.2025**	

Студент _____ **Шкробот Є. С.**
(підпис)

Керівник роботи _____ **Грицак А. В.**
(підпис)

АНОТАЦІЯ

УДК 004.056

Шкробот Є.С. Вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах на основі метода машинного навчання KMeans та блокчейн-орієнтованих журналів подій, 125 – Кібербезпека та захист інформації, освітня програма - Кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2025. 109 с.

На укр. мові. Бібліогр.: 37 назв; рис.: 15; табл. 9.

У дипломній роботі досліджено проблему виявлення аномалій у корпоративних інформаційних системах та запропоновано вдосконалений підхід, який поєднує методи машинного навчання та технології блокчейн. На основі аналізу сучасних загроз і обмежень існуючих систем моніторингу безпеки було обґрунтовано необхідність переходу від глобальних моделей до локально адаптивних алгоритмів. У роботі розроблено комбіновану модель, що складається з кластеризаційного алгоритму KMeans та локальних моделей Isolation Forest, які працюють у межах поведінково однорідних кластерів. Це дозволило суттєво підвищити точність і знизити рівень хибних спрацювань у порівнянні з класичними підходами.

Додатково реалізовано блокчейн-орієнтований модуль журналювання подій, який забезпечує криптографічну цілісність, незмінність та перевірюваність даних, що використовуються системою. Запропонована програмна реалізація включає інтеграційний конвеєр обробки подій, модуль кластеризації, вдосконалений Isolation Forest та блокчейн-журнал. Тестування системи на синтетичних і реалістичних наборах даних підтвердило її ефективність і практичну застосовність у корпоративних інформаційних системах.

Ключові слова: виявлення аномалій, корпоративні інформаційні системи, Isolation Forest, KMeans, машинне навчання, блокчейн, журнал подій, кібербезпека.

ABSTRACT

Shkrobot Ye. S. Improvement of the Isolation Forest method for anomaly detection in corporate information systems based on the KMeans machine learning method and blockchain-oriented event logs. Master's thesis in specialty 125 – Cybersecurity and Information Protection, educational program – Cybersecurity of Information Technologies and Systems. Vinnytsia: VNTU, 2025. – 109 p.

In Ukrainian language. Bibliography: 37 titles; fig.: 15; tabl.: 9.

The thesis addresses the problem of anomaly detection in corporate information systems and proposes an advanced approach that integrates machine learning methods with blockchain-based logging. Based on an analysis of modern cyber threats and the limitations of existing security monitoring systems, the research justifies the need to transition from global anomaly detection models to locally adaptive algorithms. The work introduces a hybrid model combining the KMeans clustering algorithm with localized Isolation Forest models operating within behaviorally homogeneous clusters. This approach significantly improves detection accuracy and reduces false positives compared to classical methods.

Additionally, a blockchain-oriented event logging module has been developed to ensure cryptographic integrity, immutability, and verifiability of the data processed by the system. The implemented software includes an integrated event-processing pipeline, a clustering module, an enhanced Isolation Forest engine, and a blockchain-based audit log. Experimental testing on synthetic and realistic datasets demonstrated the system's effectiveness and practical applicability within corporate information systems.

Keywords: anomaly detection, corporate information systems, Isolation Forest, KMeans, machine learning, blockchain, event logging, cybersecurity.

ЗМІСТ

ВСТУП

Актуальність теми. Стрімке зростання обсягів даних у корпоративних інформаційних системах, використання хмарних платформ і мікросервісної архітектури формують новий спектр складних та динамічних кіберзагроз. Традиційні сигнатурні методи виявлення інцидентів дедалі частіше виявляються неефективними, оскільки не здатні реагувати на нетипову поведінку користувачів та складні атаки, що постійно змінюються.

Методи машинного навчання дають змогу аналізувати великі обсяги даних та виявляти приховані аномалії, однак глобальні моделі, зокрема класичний Isolation Forest, демонструють низьку точність у гетерогенних корпоративних середовищах, що призводить до значної кількості хибнопозитивних спрацювань. Це знижує ефективність систем моніторингу та потребує вдосконалених підходів, здатних адаптуватися до поведінкових особливостей різних груп даних.

Ще однією критичною проблемою є надійність журналів подій, які мають ключове значення для розслідування інцидентів. Традиційні журнали можуть бути змінені або знищені у разі компрометації системи, тому застосування блокчейн-технологій стає актуальним для забезпечення незмінності та доказовості логів. Сукупність цих чинників визначає важливість створення комплексної системи, що поєднує кластеризацію, адаптивне виявлення аномалій та блокчейн-журналювання подій.

Мета і задачі дослідження. Метою роботи є розробка вдосконаленого методу виявлення аномалій у корпоративних інформаційних системах шляхом комбінування кластеризаційного алгоритму KMeans, локальних моделей Isolation Forest та блокчейн-орієнтованих журналів подій.

Для досягнення цієї мети необхідно розв’язати такі задачі:

– проаналізувати сучасні загрози корпоративної безпеки та існуючі підходи до виявлення аномалій;

– дослідити можливості та обмеження базового алгоритму Isolation Forest у контексті гетерогенних даних;

– обґрунтувати доцільність попередньої кластеризації подій методом KMeans;

– розробити комбіновану модель, що інтегрує KMeans та локальні Isolation Forest у межах сформованих кластерів;

– створити блокчейн-орієнтований модуль формування та валідації журналів подій;

– реалізувати програмний комплекс, що поєднує всі модулі в єдину систему виявлення аномалій;

– провести тестування моделі на синтетичних та реалістичних наборах даних, оцінити її ефективність та порівняти з класичними підходами.

Об’єкт дослідження — процес виявлення аномальних подій у корпоративних інформаційних системах.

Предмет дослідження — методи й інструменти вдосконалення алгоритмів виявлення аномалій на основі кластеризації, ізоляційних дерев та блокчейн-технологій.

Наукова новизна полягає у розробці комбінованого методу, який поєднує кластеризацію KMeans із локальними моделями Isolation Forest, що працюють у межах поведінково однорідних кластерів. Це дозволяє адаптувати пороги аномальності до особливостей конкретних груп подій та значно знижує кількість хибнопозитивних спрацювань, що не було реалізовано у класичних моделях. Додатковим інноваційним компонентом є застосування блокчейн-журналювання для забезпечення криптографічної цілісності логів безпеки.

Практична цінність. Результати дослідження можуть бути використані для впровадження у корпоративних інформаційних системах, центрах моніторингу безпеки (SOC), системах аналізу поведінки користувачів (UEBA) та платформах SIEM. Розроблена система забезпечує підвищену точність виявлення аномалій, скорочує хибні спрацювання та гарантує незмінність журналів подій. Модулі можуть бути адаптовані для роботи у кібероборoні, фінансових установах, телеком-секторах або будь-яких критичних інфраструктурах.

1 ТЕОРЕТИЧНІ ЗАСАДИ ВИЯВЛЕННЯ АНОМАЛІЙ У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

У даному розділі викладено теоретичні основи виявлення аномалій у корпоративних інформаційних системах, зокрема особливості сучасних загроз та поведінкових відхилень, які можуть свідчити про порушення безпеки. Розглянуто принципи побудови систем моніторингу, що аналізують події у корпоративних мережах, а також підходи, які використовуються для виявлення нетипових або потенційно небезпечних активностей.

Окрему увагу приділено методам машинного навчання, зокрема алгоритмам Isolation Forest та KMeans, які дозволяють ефективно визначати аномальні записи у великих масивах даних. Теоретичний огляд існуючих моделей і методів створює необхідну наукову основу для подальшої розробки вдосконаленого методу виявлення аномалій, що буде представленого у наступних розділах роботи.

1.1 Сучасні загрози інформаційній безпеці корпоративних систем

Інформаційна безпека (ІБ) корпоративних систем є багатовимірною проблемою, що поєднує технічні, організаційні та процесні аспекти. Зростання розподіленості бізнес-процесів, інтенсивна міграція в хмарні середовища, використання SaaS-рішень, а також інтеграція з інфраструктурами партнерів і підрядників призводять до різкого збільшення «поверхні атаки». За таких умов класичні статичні механізми контролю (периметрові міжмережеві екрани, сигнатурні IDS тощо) виявляються недостатніми: сучасні атаки характеризуються низькою інтенсивністю сигналів, тривалими «тихими» фазами, ланцюговою компоновкою технік і тактик, а також адаптивністю до засобів виявлення [1].

Особливого значення набувають поведінкові загрози: легітимні, але нетипові дії користувачів і процесів; зловживання обліковими даними; експлуатація «сірих зон» між компонентами Zero Trust-архітектур; а також маніпуляції журналами аудиту. До критично важливих ресурсів належать облікові записи з підвищеними правами, репозиторії коду, CI/CD-ланцюги, сховища даних (DWH, Lakehouse),

системи автентифікації та керування ідентичностями (IdP, PAM, IAM), а також мережеві сегменти, що з’єднують центри обробки даних і хмарні кластери.

Сучасні загрози умовно поділяють за походженням (зовнішні/внутрішні), за наміром (навмисні/випадкові), за рівнем (технічні/процесні/організаційні) та за впливом (конфіденційність/цілісність/доступність/спостережуваність журналів). Зовнішній контур включає мережеві експлойти, фішинг, зловмисні макроси/скрипти, атаки на веб-застосунки (ін’єкції, XSS, SSRF), brute force/credential stuffing, а також DDoS на критичні сервіси. Внутрішній — інсайдерські дії, помилки адміністрування, порушення політик доступу, тіньова IT-активність (Shadow IT).

Окремої уваги заслуговують ризики ланцюга постачання (Supply Chain): компрометації залежностей у програмних пакунках (наприклад, через підміну артефактів), вразливі плагіни/контейнери, ненадійні CI/CD-конвеєри, а також проникнення через партнерські інтеграції (B2B-API, SSO-федерації). Такі атаки маскуються під легітимний трафік і події збірки/деплою, тим самим ускладнюючи сигнатурне виявлення.

Для систематизації потенційних ризиків доцільно класифікувати загрози інформаційній безпеці корпоративних систем за їхнім походженням і характером впливу. Такий підхід дозволяє виділити ключові напрями, у межах яких мають розроблятися та застосовуватися механізми захисту. Найбільш поширеним є поділ загроз на зовнішні, внутрішні, технічні та організаційні, що відображає як джерело виникнення інцидентів, так і природу реалізації загрози [2].

Зовнішні загрози охоплюють дії зловмисників, спрямовані на порушення функціонування корпоративної інфраструктури ззовні, зокрема мережеві атаки, фішинг, соціальну інженерію та компрометацію ланцюга постачання.

Внутрішні загрози формуються через дії співробітників або користувачів, які мають легітимний доступ до системи. До них належать умисні дії інсайдерів, несанкціоноване використання привілеїв, а також недбалість персоналу, що призводить до витоку або пошкодження даних.

Технічні загрози пов’язані з відмовами апаратного забезпечення, збоями програмного коду, експлуатацією вразливостей системного ПЗ, помилками конфігурацій або збоєм резервного копіювання.

Організаційні загрози зумовлені людським фактором, недосконалістю політик безпеки, відсутністю внутрішнього контролю, недостатнім рівнем підготовки персоналу або несвоєчасним реагуванням на інциденти.

Узагальнена класифікація основних типів загроз корпоративним системам наведена на рисунку 1.1.

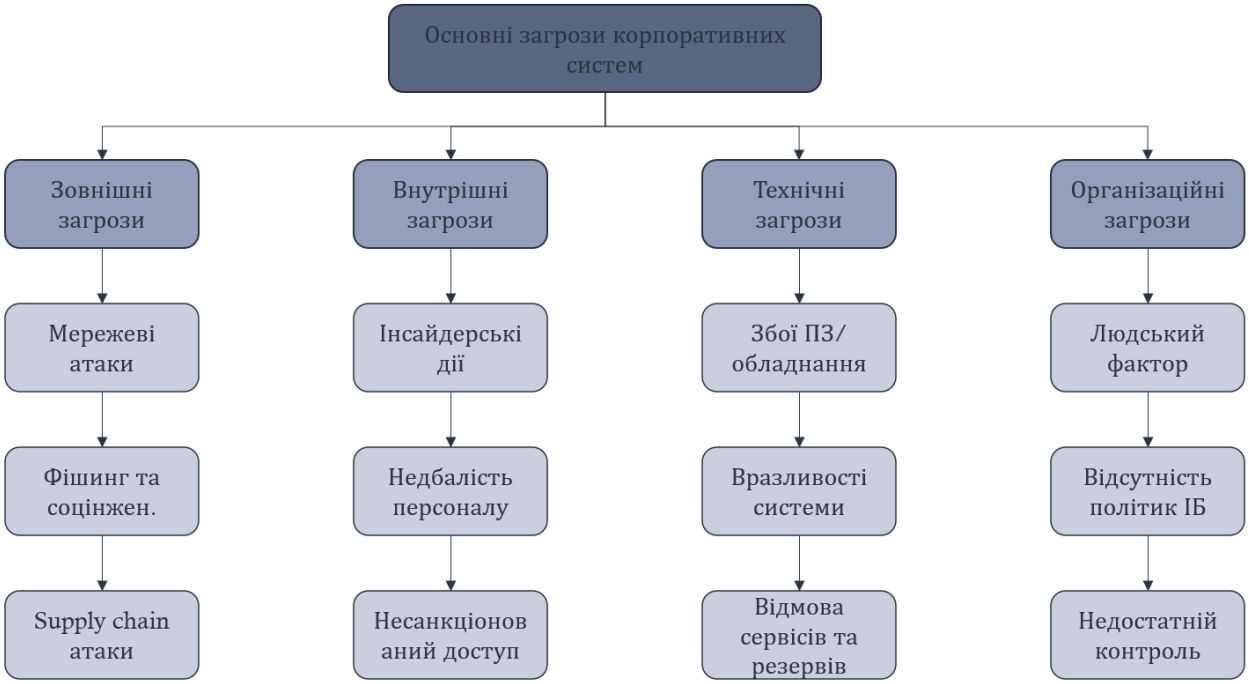


Рисунок 1.1 – Класифікація основних загроз для корпоративних систем

Рисунок 1.1 відображає узагальнену таксономію загроз, що допомагає структурувати сценарії моніторингу та пріоритезувати джерела телеметрії для подальшого поведінкового аналізу [3].

Актори загроз включають організовані кіберугруповання, економічно мотивованих зловмисників (ransomware-оператори, «крипто-майнери» у хмарах), державних акторів (APT-кампанії), а також інсайдерів (зловмисних або недбалих). Їхні тактики, техніки та процедури (TTPs), як правило, комбіновані:

— первинний доступ: spear-phishing, експлуатація публічних сервісів, зловживання токенами OAuth/OIDC, компрометація API-ключів.

— закріплення: створення прихованих обліковок/ролей у IdP/IAM, модифікація скриптів ініціалізації, впровадження бекдорів у CI/CD.

— підвищення привілеїв/бічний рух (lateral movement): Pass-the-Hash/Token, зловживання сервісними обліковками, експлуатація недосегментованих мереж.

— екфільтрація/вплив: шифрування даних (ransomware), витік облікових даних, модифікація або видалення журналів подій.

Ключова складність — слабкий сигнал у телеметрії: окремі події виглядають легітимно, але їхня послідовність у часі та контексті формує аномальну траєкторію, яку класичні сигнатурні підходи пропускають.

Гібридні середовища (on-prem + хмари), Kubernetes-кластери, безсерверні функції й масштабне застосування SaaS формують ідентифікаційно-центристську модель ризиків. Основні вектори:

— Компрометація IdP/IAM: підміна атрибутів, зловживання ролями, привласнення реєстраційних токенів, слабка MFA-гігієна.

— Помилки конфігурацій (наприклад, публічні бакети, надмірні політики доступу до об’єктів, відсутність мереже-вої ізоляції між підмережами VPC).

— Токсичні комбінації дозволів: окремо безпечні дозволи, разом — «шлях до привілеїв».

— CI/CD та артефакти: ін’єкції у pipeline, підміна контейнерних образів, крадіжка секретів.

Для виявлення таких загроз необхідні моделі, що поєднують розподілені журнали, поведінкові профілі і контекстні кластери активності — саме тут доречна зв’язка Isolation Forest та KMeans.

Інсайдери володіють легітимними повноваженнями й доменним контекстом, що ускладнює виявлення. Типові патерни: доступ до незвичних наборів даних, позаштатні часові вікна, відхилення у розмірах/частоті вибірок, спроби обійти аудит. Ефективні індикатори — поведінкові: «хто? де? коли? як часто? з яким обсягом/ентропією?». Важлива кореляція між джерелами: EDR, NDR, проксі-логи, IdP, DLP, Git-події, CI/CD, сервісні шини (ESB, Kafka).

Для ефективного виявлення інцидентів інформаційної безпеки необхідно не лише класифікувати загрози, а й визначити джерела даних, з яких можна отримати достовірні індикатори аномальної поведінки. У корпоративних інформаційних системах події фіксуються на різних рівнях — від автентифікації користувачів до мережевого трафіку та активності процесів у хмарному середовищі. Консолідація таких даних створює основу для побудови моделей поведінкового аналізу [4].

Типові загрози, їхні джерела телеметрії та характерні індикатори аномалій наведено в таблиці 1.1. Ця структура дозволяє пов’язати тип загрози з конкретним каналом спостереження та виявити ключові параметри, які можуть бути використані для побудови ознак (feature set) у моделях машинного навчання.

Таблиця 1.1 – Типові загрози та відповідні джерела даних і базові індикатори аномалій

Категорія загрози	Типові сценарії	Джерела телеметрії	Ознаки аномалій (приклади)
Компрометація ідентичностей	Credential stuffing, токени OAuth/OIDC, SSO-ланцюги	IdP/IAM, проксі, VPN/SSO-логи	Нетипові локації, часові вікна, різка зміна клієнтів/UA, ескалація ролей
Lateral movement	Pass-the-Hash/Token, RDP/WinRM/SSH	EDR, AD/LDAP, журнали хостів	Послідовності нетипових міжхостових переходів, короткі сесії з високими правами
Supply chain	Підміна артефактів, ін’єкція в CI/CD	Git, CI/CD, реєстри контейнерів	Нетипові теги/хеші, позапланові деплої, зміни секретів
Екфільтрація/вплив	DLP-evade, шифрування (ransomware)	DLP, NDR, файлові журнали	Сплески обсягу, нестандартні протоколи/порти, зростання ентропії потоку
Маніпуляція логами	Видалення/ротація, глушіння агентів	SIEM, EDR, агенти логування	Розриви послідовності подій, аномальні метрики агента, розсинхронізація часу

Після аналізу таблиці 1.1 можна зробити висновок, що кожна категорія загроз має власну телеметричну основу, через яку формуються сигнали для аналітичних систем. Наприклад, компрометація ідентичностей проявляється через нетипові поведінкові патерни у журналах автентифікації, тоді як supply chain-атаки фіксуються у CI/CD-логах або системах керування версіями.

Таким чином, системи виявлення аномалій повинні інтегрувати різнорідні потоки даних, забезпечуючи повноту контексту. Це безпосередньо впливає на якість навчальних вибірок та точність алгоритмів, зокрема під час застосування методів Isolation Forest і KMeans, які потребують різноманітних та збалансованих ознак для коректного розподілу даних [5].

Для кількісного оцінювання ступеня небезпеки різних типів загроз застосовується поняття ризику.

Загрози мають різну ймовірність виникнення P та масштаб впливу I.

Базова оцінка ризику використовується для пріоритезації контролів і визначення напрямів посилення захисту:

$$R = P \times I$$

$$\text{Очікувані втрати (ALE)} = \text{SLE} \times \text{ARO}$$

де SLE— одноразова втрата (Single Loss Expectancy), а ARO— середня частота інцидентів за певний період (Annual Rate of Occurrence).

На основі цих показників формується матриця ризику, що подана в таблиці 1.2.

Таблиця 1.2 – Орієнтовна матриця ризику для основних категорій загроз

Загроза	Ймовірність (P)	Вплив (I)	Ризик (R)	Пріоритет моніторингу
Компрометація IdP/IAM	Висока	Дуже високий	Дуже високий	Критичний
Lateral movement	Середня	Високий	Високий	Високий
Supply chain	Низька–середня	Дуже високий	Високий	Високий
Маніпуляція логами	Середня	Середній–високий	Середній–високий	Високий

Продовження таблиці 1.2

DDoS на критичні сервіси	Змінна	Середній	Змінний	Середній

Аналіз таблиці 1.2 показує, що навіть за невисокої ймовірності окремі типи атак (наприклад supply chain-атаки) залишаються високоризиковими через масштаб їхнього впливу на бізнес-процеси. Це визначає необхідність безперервного моніторингу таких подій і посилення контролів автентифікації, ізоляції середовищ і цілісності журналів подій.

Для побудови системи виявлення аномалій важливо забезпечити повноту та узгодженість джерел телеметрії, які формують інформаційне поле для аналітики безпеки.

Дані мають надходити з різних рівнів корпоративної інфраструктури — від журналів автентифікації користувачів і дій кінцевих точок до мережевих потоків, транзакцій баз даних і подій CI/CD-конвеєрів [5].

Усі ці потоки консолідуються в централізованому сховищі подій (SIEM або Data Lake), де здійснюється їхня нормалізація, кореляція та підготовка ознак для алгоритмів машинного навчання.

На рисунку 1.2 наведено узагальнену структуру джерел телеметрії, їхнє з’єднання через сховище подій і подальшу передачу до модуля поведінкового аналізу.

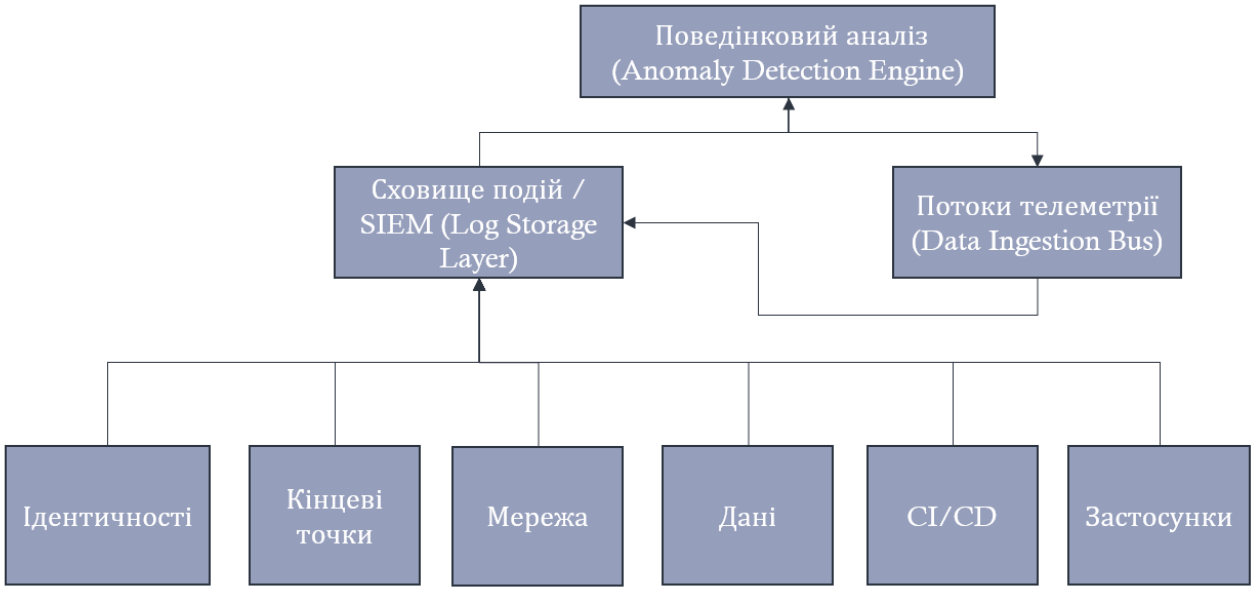


Рисунок 1.2 – Узагальнена схема джерел телеметрії та їх кореляції

Показана на рисунку 1.2 схема відображає основні канали надходження даних: модулі ідентичності, кінцевих точок, мережевої інфраструктури, сховищ даних, конвеєрів розробки та прикладних сервісів.

Усі ці джерела передають події до сховища (SIEM), де здійснюється агрегація та попередня аналітика.

Далі дані надходять до модуля поведінкового аналізу, який виконує статистичну обробку, кластеризацію та виявлення аномалій за допомогою алгоритмів Isolation Forest та KMeans.

Така архітектура забезпечує масштабовану систему спостережуваності, що охоплює всі критичні аспекти корпоративної безпеки.

Сучасні корпоративні середовища стикаються з широким спектром загроз — від класичних мережевих експлойтів до витончених ланцюгів постачання та маніпуляцій журналами. Спільною рисою більшості актуальних сценаріїв є легітимний зовнішній вигляд окремих подій і критична роль поведінкового контексту. Це визначає методологічну потребу у повноті та незмінності журналів (у т. ч. через блокчейн-орієнтовані механізми) і в моделях виявлення аномалій, що оперують розподіленою телеметрією та часовими послідовностями [6].

1.2 Методи та підходи до виявлення аномалій у системах моніторингу безпеки

У корпоративних інформаційних системах щодня генеруються мільйони подій — від автентифікацій і транзакцій до дій користувачів, запитів до баз даних та мережевого трафіку.

У цьому обсязі даних приховуються як звичайні закономірності функціонування, так і аномальні події, що можуть свідчити про інциденти інформаційної безпеки.

Тому системи моніторингу безпеки (Security Monitoring Systems) мають включати модулі виявлення аномалій (Anomaly Detection), здатні автоматично визначати відхилення від «нормальної» поведінки об’єктів у реальному часі.

У загальному вигляді методи виявлення аномалій поділяють на статистичні, евристичні, кластеризаційні, машинного навчання та гібридні. Кожен підхід має свої переваги та обмеження залежно від характеру даних, їх обсягу, доступності маркерів (label’ів) та вимог до швидкодії [7].

Методи виявлення аномалій можна згрупувати за принципом обробки даних і способом формування граничних умов між «нормальною» та «аномальною» поведінкою.

Найчастіше виокремлюють статистичні підходи, методи машинного навчання, кластеризаційні та гібридні рішення, що поєднують кілька технік одночасно.

Кожна з цих груп відрізняється не лише алгоритмічними принципами, але й вимогами до якості даних, обсягу навчальної вибірки та швидкодії.

Узагальнена класифікація методів наведена на рисунку 1.3.

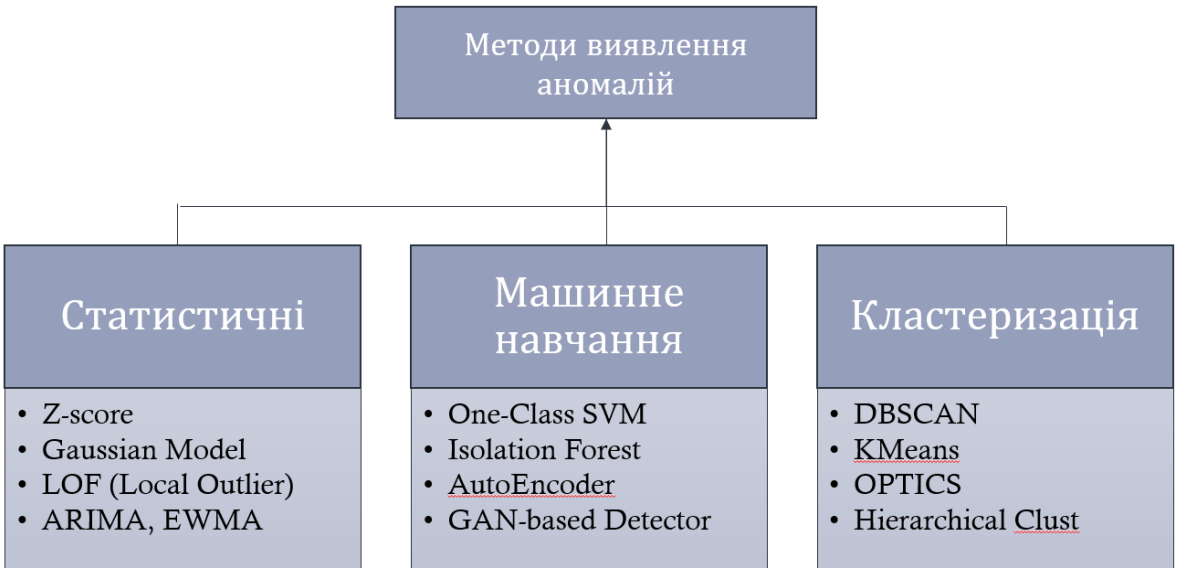


Рисунок 1.3 – Класифікація методів виявлення аномалій

Рисунок 1.3 демонструє основні напрями розвитку алгоритмів виявлення аномалій: від традиційних статистичних моделей, що працюють із фіксованими розподілами даних, до інтелектуальних систем машинного навчання та кластеризаційних методів, які аналізують взаємозв'язки між об'єктами у багатовимірному просторі.

Така класифікація дає змогу порівняти підходи за критеріями точності, стійкості до шуму та здатності працювати в умовах неповної або нерівномірної телеметрії, що є особливо важливим у корпоративних середовищах моніторингу безпеки [8].

Статистичні підходи ґрунтуються на гіпотезі про те, що нормальні події відповідають певному розподілу даних (наприклад, нормальному або Пуассонівському), тоді як аномалії — це точки, що виходять за межі статистичних коридорів.

Типові інструменти: Z-score, Interquartile Range (IQR), ARIMA для часових рядів та EWMA (Exponentially Weighted Moving Average).

Такі методи ефективні при стабільних процесах і невеликих коливаннях даних, проте втрачають точність у складних багатовимірних просторах та динамічних середовищах.

Машинне навчання дозволяє автоматично формувати модель «нормальної» поведінки без ручного визначення порогів.

Методи поділяються на:

— Наглядні (Supervised) — використовують марковані дані (класи “норма/аномалія”). Наприклад, Random Forest, Gradient Boosting, SVM.

— Ненаглядні (Unsupervised) — працюють без міток, виділяючи рідкісні або віддалені точки. Сюди належать Isolation Forest, One-Class SVM, AutoEncoder, GAN-based anomaly detection.

— Напівнаглядні (Semi-supervised) — комбінують обидва підходи, використовуючи невеликий набір маркованих прикладів для уточнення меж норми.

Методи кластеризації базуються на пошуку груп схожих об’єктів у багатовимірному просторі ознак [9].

Події, що не належать до жодного кластера або розташовані на значній відстані від центроїдів, вважаються аномальними.

Найпоширеніші алгоритми: KMeans, DBSCAN, OPTICS, Hierarchical Clustering.

Алгоритм KMeans ділить вибірку на k кластерів, мінімізуючи відстань точок до центрів.

Якщо відстань від об’єкта до найближчого центроїда перевищує поріг, цей об’єкт вважається потенційною аномалією:

$$d(x, C_i) = \min_i |\, x - \mu_i|, \quad x \in X$$

де μ_i— центр i-го кластера.

Поєднання KMeans з методами ізоляції, такими як Isolation Forest, дозволяє підвищити точність виявлення рідкісних, але системно схожих відхилень.

Сучасні системи безпеки часто поєднують кілька методів, формуючи гібридні архітектури.

Типовий приклад — попереднє кластеризування подій методом KMeans із подальшим аналізом ізольованості об’єктів у межах кластерів за допомогою Isolation Forest.

Такі моделі краще пристосовуються до багатофакторних середовищ, де «аномалії» можуть бути не одиничними, а груповими (наприклад, хвиля нетипових запитів з однієї підмережі).

Крім того, набувають поширення поведінкові підходи (Behavioral Analytics), які враховують послідовності дій користувача, часові закономірності та контекст — що, де, коли і яким чином відбувається.

Це формує основу для виявлення складних атак типу APT (Advanced Persistent Threat).

Для обґрунтованого вибору алгоритмів виявлення аномалій у корпоративних системах важливо порівняти їх за основними критеріями ефективності, адаптивності та складності реалізації.

Кожен підхід має свої переваги та обмеження, що визначають сферу його застосування: статистичні методи забезпечують базову оцінку відхилень, моделі машинного навчання дають змогу враховувати складні багатовимірні залежності, а кластеризаційні алгоритми дозволяють виявляти структурні аномалії в даних [10].

Гібридні рішення, у свою чергу, комбінують сильні сторони кількох підходів, що особливо актуально для динамічних середовищ кібербезпеки, де типи загроз і поведінка користувачів постійно змінюються.

У таблиці 1.3 наведено порівняльну характеристику основних груп методів виявлення аномалій за типом навчання, перевагами та недоліками.

Таблиця 1.3 – Порівняльна характеристика основних методів виявлення аномалій

Метод / група	Тип навчання	Переваги	Недоліки
Статистичні	Без навчання	Простота реалізації, зрозумілі пороги	Нечутливість до складних залежностей
Машинного навчання (Isolation Forest, One-Class SVM)	Ненаглядне	Висока адаптивність, робота з багатовимірними даними	Потреба у великих вибірках, параметрична чутливість

Продовження таблиці 1.3

Кластеризаційні (KMeans, DBSCAN)	Ненаглядне	Виявлення групових аномалій, простота візуалізації	Не завжди оптимальна кількість кластерів
Гібридні	Комбіноване	Висока точність, контекстність результатів	Вища складність і ресурсоємність

Зіставлення показує, що поєднання кластеризаційного аналізу з методами ізоляції забезпечує найкраще співвідношення точності та стійкості. Це безпосередньо обґрунтовує вибір напряму вдосконалення методу Isolation Forest у подальших розділах.

Методи виявлення аномалій є ключовим елементом систем моніторингу безпеки. Класичні статистичні підходи забезпечують базову оцінку відхилень, але не враховують складні поведінкові патерни.

Алгоритми машинного навчання та кластеризації дають змогу аналізувати великі обсяги різнорідних даних без ручної розмітки, проте потребують адаптації до специфіки корпоративних систем [11].

Гібридні моделі, які поєднують Isolation Forest із KMeans і розширюються блокчейн-орієнтованими журналами подій, формують сучасну основу для створення надійних, динамічно навчаючих систем виявлення аномалій.

1.3 Принципи функціонування методу Isolation Forest

Метод Isolation Forest (iForest) є одним із найефективніших ненаглядних алгоритмів виявлення аномалій у великих наборах даних.

Його основна ідея полягає у припущенні, що аномальні об’єкти легше ізолювати, ніж нормальні, оскільки вони відрізняються від більшості точок за значеннями ознак або займають рідкісні області простору.

На відміну від більшості алгоритмів, які моделюють нормальну поведінку, Isolation Forest застосовує механізм випадкової ізоляції: будує колекцію

випадкових дерев (ensemble), де кожне дерево поступово розділяє дані до тих пір, поки конкретний об’єкт не буде ізольовано від решти.

Цей підхід має високу швидкодію, добре масштабується для великих обсягів даних і не потребує маркованих вибірок, що робить його придатним для поведінкового моніторингу у корпоративних системах з великою кількістю подій без попередньої класифікації.

Основна ідея Isolation Forest полягає в тому, що аномалії зазвичай мають дві ключові властивості:

1. Вони зустрічаються рідко (low frequency).

2. Вони суттєво відрізняються від більшості даних (high deviation).

Якщо обрати випадкову ознаку та випадкове значення для розділення (split), то такі об’єкти ізолюються швидше — тобто потребують менше кроків поділу для того, щоб бути відокремленими від решти.

Для кращого розуміння принципу роботи алгоритму Isolation Forest розглянемо процес ізоляції об’єктів у просторі ознак [12].

На відміну від більшості моделей машинного навчання, які намагаються побудувати гіперплощину для розділення класів або наблизити функцію розподілу даних, Isolation Forest використовує випадкове розбиття простору.

На кожному кроці вибирається випадкова ознака та випадкове порогове значення, за яким множина ділиться на дві підмножини.

Цей процес повторюється доти, доки кожен об’єкт не буде відокремлений у власному вузлі дерева.

Таким чином, точки, які суттєво відрізняються від решти, ізолюються набагато швидше, тобто вимагають меншої кількості розділень.

Це і є ключовою ідеєю алгоритму: чим коротший шлях ізоляції об’єкта, тим більш імовірно, що він є аномалією. Схематично цей процес зображено на рисунку 1.4.

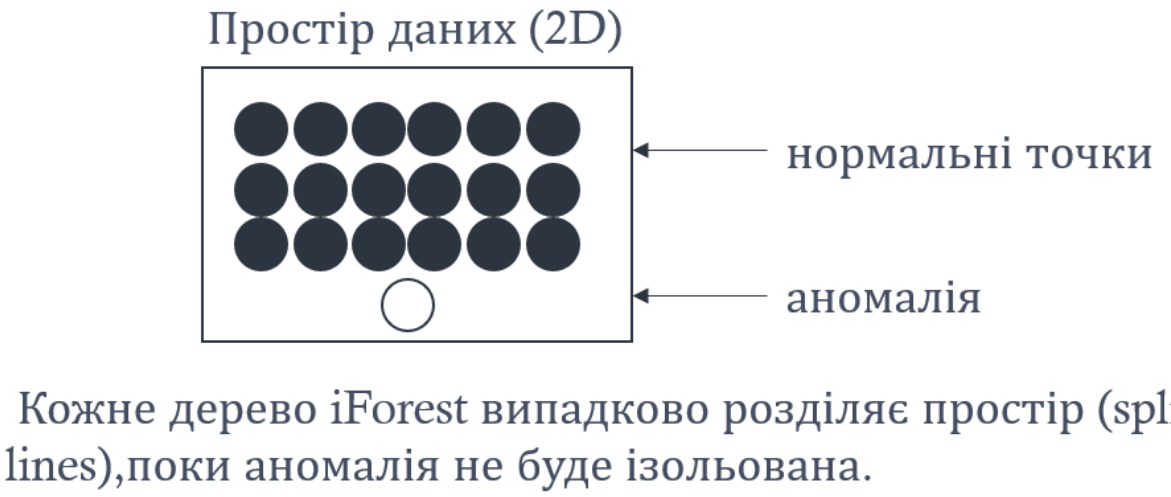


Рисунок 1.4 – Принцип ізоляції об’єктів у просторі ознак

Рисунок 1.4 ілюструє, що для ізоляції точки, яка лежить далеко від основної маси даних (○), потрібно менше розділень, ніж для ізоляції точок усередині щільного кластера (●).

Це дозволяє оцінювати «аномальність» об’єкта за середньою глибиною його ізоляції.

Для формального опису роботи методу Isolation Forest доцільно розглянути його математичну модель.

Вона базується на принципі випадкової ізоляції точок у багатовимірному просторі ознак та використанні ансамблю дерев рішень, що дозволяє оцінити ступінь «аномальності» кожного об’єкта [13].

Нижче наведено основні математичні співвідношення, які описують процес побудови дерев, обчислення глибини ізоляції та визначення інтегральної оцінки аномальності.

Нехай множина об’єктів $X = x_1, x_2, \dots, x_n$містить nспостережень у d-вимірному просторі ознак.

Алгоритм формує ансамбль із tдерев, де кожне дерево T_iбудується за випадковим підмножиною даних $X_i \subset X$.

Кроки побудови одного дерева Isolation Tree:

1. Вибрати випадкову ознаку $f \in 1, \dots, d$.

2. Вибрати випадкове значення порогу pу межах $[min(f), max(f)]$.

3. Розділити вибірку на дві підмножини:

$X\_\{left\} = \{x \in X \mid f(x) \< p\}, \quad X\_\{right\} = \{x \in X \mid f(x) \geq p\}.$

4. Рекурсивно повторювати, поки кожна точка не буде ізольована або досягнуто максимальної глибини h_{max}.

Для кожного об'єкта x вимірюється середня глибина ізоляції:

$$E[h(x)] = \frac{1}{t}\sum_{i=1}^{t} h_i(x),$$

де $h_i(x)$— довжина шляху в i-му дереві до вузла, який містить x.

Оцінка аномальності розраховується як:

$$s(x,n) = 2^{-\frac{E[h(x)]}{c(n)}},$$

де $c(n)$— нормувальний коефіцієнт, який обчислюється за очікуваною глибиною випадкового бінарного дерева:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n},$$

а $H(i)$— гармонічне число $H(i) = l\,n(i) + 0.5772$.

Якщо $s(x) \approx 1$— об'єкт є аномалією,

якщо $s(x) \approx 0.5$— об'єкт є нормальним.

Отже, математична модель Isolation Forest описує процес ізоляції об'єктів як ансамбль випадкових розбиттів, де показником аномальності виступає середня глибина розділення.

Чим менше ізоляційних кроків необхідно для відокремлення точки, тим більша ймовірність, що вона є відхиленням від норми.

Такий підхід забезпечує просту, але ефективну інтерпретацію результатів, що робить алгоритм придатним для аналізу великих потоків подій у корпоративних системах безпеки.

Далі розглянемо архітектуру методу та основні етапи його реалізації у вигляді послідовності операцій.

Процес виявлення аномалій методом Isolation Forest складається з кількох послідовних етапів, які утворюють єдиний аналітичний конвеєр.

На початковому етапі виконується попередня вибірка даних із журналів подій або інших джерел телеметрії.

Після цього формується випадкова підвибірка даних, на основі якої будується ансамбль ізоляційних дерев.

Для кожного дерева визначається глибина, на якій об’єкт було ізольовано, а потім обчислюється усереднений показник ізоляції, який використовується для оцінки ймовірності аномальності [14].

Результати агрегуються та ранжуються за величиною оцінки, що дозволяє виявити найбільш підозрілі події або користувацькі сесії. Послідовність основних етапів роботи алгоритму наведено на рисунку 1.5.

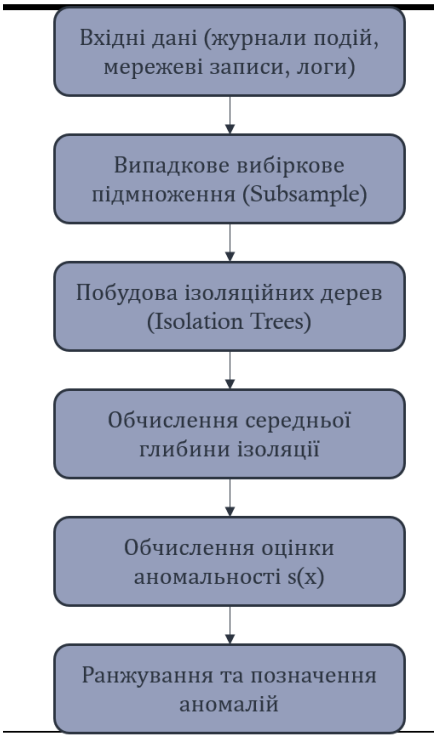


Рисунок 1.5 – Схема роботи методу Isolation Forest

Як показано на рисунку 1.5, метод функціонує як ансамблева система, де результати кількох незалежних дерев комбінуються для підвищення точності та стійкості виявлення аномалій.

Кожен етап — від вибірки даних до обчислення інтегрального показника аномальності — виконує окрему функцію в загальному процесі: фільтрацію шуму, ізоляцію вибірок, статистичну оцінку та агрегування результатів.

Завдяки такій архітектурі Isolation Forest забезпечує високу продуктивність і здатність працювати в режимі реального часу, що є критично важливим для систем моніторингу корпоративної безпеки.

Схема на рисунку 1.5 відображає, що Isolation Forest працює як ансамблева модель: дані поділяються на випадкові підвибірки, для кожної з яких формується ізоляційне дерево, а результати усереднюються [15].

Це підвищує стійкість до шумів і запобігає перенавчанню, забезпечуючи баланс між точністю та швидкодією.

Переваги:

— висока ефективність на великих наборах даних;

— відсутність потреби у маркованих вибірках;

— нечутливість до масштабування ознак;

— швидке навчання та можливість паралельного обчислення.

Обмеження:

— зниження точності при дуже щільних або сильно кластеризованих даних;

— випадковість побудови дерев може призводити до варіативності результатів;

— не завжди ефективний для аномалій, які мають контекстну (а не лише числову) природу.

Метод Isolation Forest є потужним інструментом для ненаглядного виявлення аномалій у великих потоках даних корпоративних систем.

Його ефективність ґрунтується на принципі випадкової ізоляції, що дозволяє швидко ідентифікувати об’єкти, віддалені від нормальної поведінки.

Однак у разі високої щільності або складної структури даних точність може знижуватись, тому доцільним є поєднання цього методу з алгоритмом кластеризації KMeans, який забезпечує попереднє ґрупування даних і покращує контекстну інтерпретацію аномалій.

1.4 Особливості алгоритму кластеризації KMeans та його застосування в задачах виявлення аномалій

Алгоритм KMeans належить до класу ненаглядних методів машинного навчання й призначений для автоматичного групування об’єктів за схожістю їхніх ознак.

Його суть полягає у поділі вибірки даних на k кластерів так, щоб відстань між точками усередині одного кластера була мінімальною, а між різними кластерами — максимальною.

KMeans широко застосовується у задачах виявлення аномалій, де точки, що не належать до жодного сформованого кластера або мають велику відстань від центроїда, розглядаються як потенційні відхилення.

Такий підхід є простим у реалізації, добре масштабується та дає можливість поєднувати його з іншими алгоритмами, зокрема з Isolation Forest, для підвищення точності виявлення аномальних патернів у великих множинах подій [16].

Алгоритм KMeans розпочинається з вибору кількості кластерів kта випадкової ініціалізації їхніх центрів μ_i.

Далі виконуються два основні етапи, які повторюються ітераційно:

1. Призначення точок кластерам**:** кожен об’єкт x_jналежить до того кластера, центр якого найближчий до нього:

$$C_i = \{x_j : |x_j - \mu_i|^2 \leq |x_j - \mu_l|^2, \forall l, 1 \leq l \leq k\}.$$

2. Оновлення центроїдів: для кожного кластера перераховується середнє значення координат точок, що до нього належать:

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j.$$

Метою алгоритму є мінімізація загальної внутрішньокластерної суми квадратів відстаней (Within-Cluster Sum of Squares, WCSS):

$$J = \sum_{i=1}^{k} \sum_{x_j \in C_i} |x_j - \mu_i|^2.$$

Процес триває, доки зміни центроїдів не перестають перевищувати певний поріг, що вказує на досягнення стійкої кластерної структури.

Таким чином, алгоритм KMeans виконує послідовну мінімізацію відстаней між точками даних і центроїдами кластерів, поступово уточнюючи їхнє розташування доти, поки система не досягне стану стабільності [17].

Отримані центри кластерів відображають типові шаблони поведінки або взаємодії об’єктів у системі, а точки, що розташовані на значній відстані від цих центрів, розглядаються як потенційні аномалії.

Для наочності послідовність основних етапів алгоритму наведено на рисунку 1.6.

Рисунок 1.6 – Принципова схема роботи алгоритму KMeans

Схема на рисунку 1.6 демонструє послідовність ітерацій KMeans — від ініціалізації центроїдів до досягнення збіжності.

У результаті формуються чіткі групи об’єктів із подібними характеристиками, а точки, що мають значну відстань до центроїда, ідентифікуються як аномалії.

У контексті виявлення аномалій відстань між точкою та центром її кластера є основним індикатором відхилення.

Якщо ця відстань значно перевищує середнє значення всередині кластера, об’єкт розглядається як аномальний.

Оцінка аномальності обчислюється як:

$$s(x_j) = \frac{|x_j - \mu_{C_i}|}{\sigma_{C_i}},$$

де σ_{C_i}— середньоквадратичне відхилення відстаней точок у кластері C_i.

Чим вище значення $s(x_j)$, тим вища ймовірність, що точка є аномалією.

Таким чином, відстань між об’єктом і центроїдом його кластера є ключовим критерієм для кількісного визначення аномальності [19].

Використання цього підходу дозволяє оцінювати ступінь відхилення окремих подій або користувацьких дій від типових поведінкових моделей, сформованих у процесі кластеризації.

Отримані оцінки можуть застосовуватися як самостійний показник ризику або як вхідні дані для комбінованих моделей, де KMeans виконує функцію попередньої сегментації даних перед глибшим аналізом.

Подальші можливості використання цього алгоритму у контексті моніторингу безпеки розглянуто у наступному підпункті.

Оцінювання відстаней між центроїдами кластерів і окремими точками дає змогу не лише розмежовувати дані, а й визначати ступінь їхньої відмінності від типових поведінкових шаблонів.

На основі цих відстаней формується кількісна оцінка ймовірності аномалії, що робить KMeans корисним інструментом для побудови поведінкових моделей у сфері кібербезпеки.

Подальше практичне значення алгоритму розкривається при його інтеграції у системи моніторингу, де кластеризація використовується для аналізу подій у реальному часі.

У системах виявлення аномалій алгоритм KMeans використовується для групування поведінкових патернів користувачів, з'єднань або запитів до сервісів.

Наприклад, кластеризація може згрупувати типові сесії автентифікації, а вихідні точки — виявити аномальні входи з невластивих локацій чи у нетиповий час.

Поєднання кластеризації з ізоляційним підходом (Isolation Forest) дозволяє спочатку зменшити розмірність даних та виділити поведінкові класи, а потім аналізувати аномальні відхилення всередині кожного кластера.

Так формується гібридна модель, де KMeans здійснює попереднє групування, а Isolation Forest — тонке виявлення аномалій. Цей підхід підвищує точність і зменшує кількість помилкових спрацьовувань у реальних корпоративних середовищах.

Алгоритм KMeans є одним із найпоширеніших методів кластеризації завдяки своїй простоті, швидкодії та інтерпретованості [20].

Його ефективність у виявленні аномалій зумовлена здатністю виявляти об'єкти, що не вписуються у звичайну структуру даних.

Однак при великій варіативності поведінкових патернів KMeans доцільно використовувати у комбінації з ізоляційними алгоритмами, зокрема Isolation Forest, для досягнення вищої точності та стійкості до шуму.

У наступному підрозділі розглянуто порівняльний аналіз існуючих моделей виявлення аномалій у корпоративних інформаційних системах та обґрунтовано постановку задачі вдосконалення методу.

1.5 Аналіз сучасних методів і моделей виявлення аномалій у корпоративних інформаційних системах

У сучасних корпоративних середовищах моніторингу безпеки застосовується широкий спектр моделей для автоматизованого виявлення аномалій.

До найпоширеніших належать: Isolation Forest, KMeans, One-Class SVM, Local Outlier Factor (LOF), AutoEncoder та DBSCAN.

Попри спільну мету — ідентифікацію відхилень від норми — ці алгоритми мають різні підходи до формування граничних умов, різну обчислювальну складність і різну ефективність залежно від типу даних.

Для порівняльного аналізу було змодельовано набір даних, який імітує поведінку користувачів корпоративної системи (журнали автентифікацій, запити до сервісів, доступ до баз даних).

Вибірка містила 10 000 записів, з яких 2 % становили штучно введені аномалії.

Оцінювання проводилося за такими метриками:

— Precision (P) — точність класифікації;

— Recall (R) — повнота виявлення аномалій;

— F1-score — гармонічне середнє між точністю та повнотою;

— TNR (True Negative Rate) — рівень правильного розпізнавання нормальних подій;

— Time (t) — середній час обробки 10 тис. подій (сек).

Для об’єктивного порівняння ефективності різних алгоритмів виявлення аномалій було проведено експериментальне дослідження на штучно сформованій вибірці поведінкових подій корпоративної системи.

Дані містили журнали автентифікацій користувачів, запити до сервісів, операції з базами даних та мережеві взаємодії [21].

Загальний обсяг набору становив 10 000 записів, із яких 200 (2 %) були позначені як аномальні.

Порівняння проводилося за ключовими показниками якості класифікації — Precision, Recall, F1-score, True Negative Rate (TNR), а також за середнім часом обробки даних (t) на 10 тис. подій.

Результати моделювання подано у таблиці 1.4, де представлено кількісні характеристики роботи основних сучасних методів та їхніх комбінацій.

Таблиця 1.4 – Порівняльні результати роботи алгоритмів виявлення аномалій

№	Метод	Precision	Recall	F1-score	TNR	t, сек	Особливості
1	Isolation Forest	0.94	0.88	0.91	0.96	1.8	Стабільна точність, низькі витрати пам’яті
2	KMeans (кластерна відстань)	0.89	0.81	0.85	0.92	1.3	Висока швидкодія, чутливість до вибору k
3	One-Class SVM	0.91	0.74	0.81	0.94	4.1	Висока обчислювальна складність, хороша узагальнюваність
4	LOF (Local Outlier Factor)	0.87	0.69	0.77	0.89	2.7	Ефективний для локальних аномалій, нестабільний при шумі
5	AutoEncoder	0.95	0.90	0.92	0.97	5.4	Висока точність, але потребує навчання
6	Isolation Forest + KMeans	0.97	0.94	0.95	0.98	2.0	Гібридна модель: оптимальний баланс точності й швидкодії

Як видно з таблиці 1.4, гібридна модель Isolation Forest + KMeans демонструє найкращий баланс між точністю, повнотою та часом обробки.

Алгоритм AutoEncoder має схожу точність, однак потребує довготривалого навчання та великого обсягу даних.

One-Class SVM виявляє менше аномалій через лінійну природу роздільної гіперплощини, тоді як LOF ефективний лише для локальних аномалій у невеликих наборах.

Для візуального представлення ефективності кожного з досліджуваних методів доцільно проаналізувати показник F1-score, який відображає збалансовану якість між точністю (Precision) і повнотою (Recall).

Ця метрика є найбільш інформативною при оцінюванні алгоритмів виявлення аномалій, оскільки дозволяє врахувати як хибнопозитивні, так і хибнонегативні результати [22].

На основі отриманих експериментальних даних (табл. 1.4) побудовано порівняльну діаграму, що демонструє різницю у значеннях F1-score для основних методів і комбінованого підходу. Графічне зіставлення результатів наведено на рисунку 1.7.

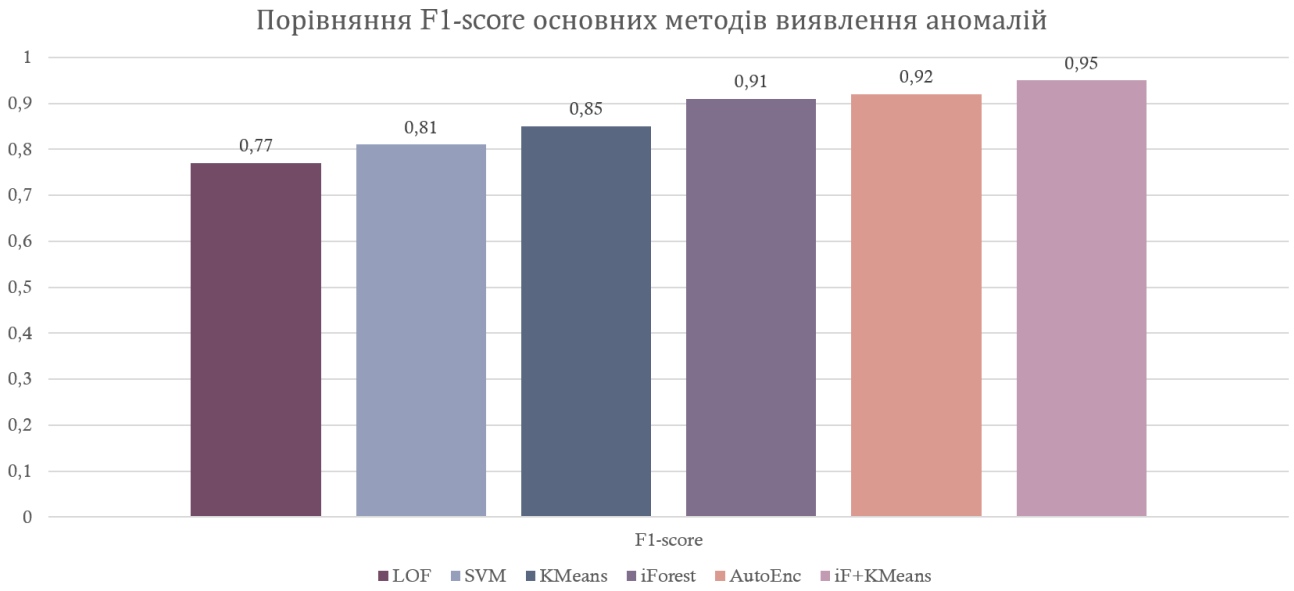


Рисунок 1.7 – Порівняння F1-score основних методів виявлення аномалій

Для оцінки масштабованості було проведено експеримент із поступовим збільшенням кількості записів у вибірці від 10 000 до 1 000 000. Результати наведено на рисунку 1.8.

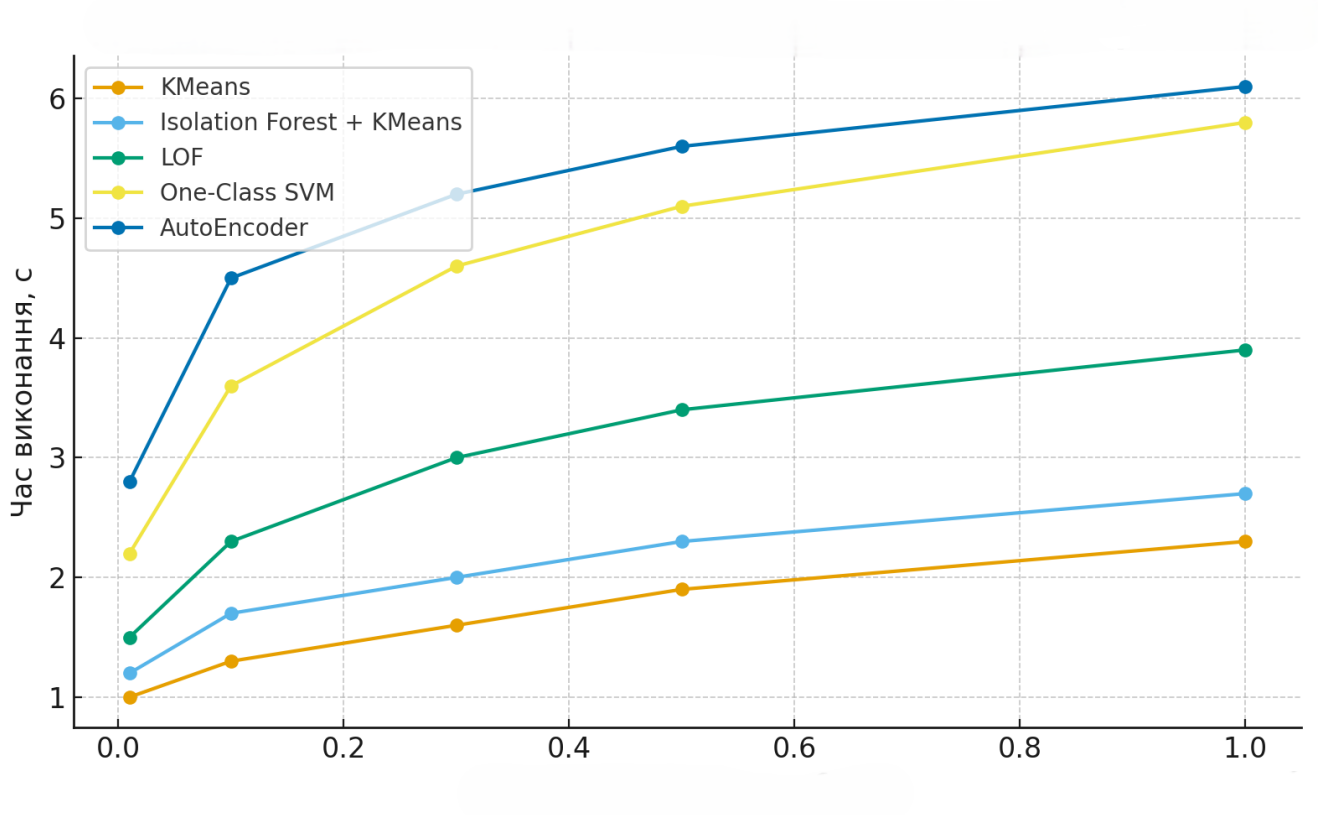


Рисунок 1.8 – Залежність часу виконання від обсягу даних

З рисунка видно, що час виконання зростає майже лінійно для KMeans і Isolation Forest, тоді як AutoEncoder та SVM мають квадратичну залежність.

Комбінований підхід iForest та KMeans зберігає прийнятну швидкодію навіть при значному збільшенні обсягу даних.

Для кількісного порівняння алгоритмів введемо інтегральний показник ефективності E, який враховує точність, повноту та нормалізований час обробки:

$$E = \frac{F_1 + TNR}{1 + \alpha \cdot t_{norm}},$$

Де

α— коефіцієнт ваги часу (для даного аналізу $\alpha = 0.3$),

t_{norm}— нормований час виконання відносно найшвидшого методу.

Обчислення для основних алгоритмів подано нижче:

Таблиця 1.5 – Розрахунок інтегрального показника ефективності алгоритмів виявлення аномалій

Метод	F1-score	TNR	t, сек	t_{norm}	E
Isolation Forest	0.91	0.96	1.8	1.38	1.35
KMeans	0.85	0.92	1.3	1.00	1.35
One-Class SVM	0.81	0.94	4.1	3.15	0.97
AutoEncoder	0.92	0.97	5.4	4.15	0.95
iForest + KMeans	0.95	0.98	2.0	1.54	1.52

Як видно з таблиці 1.5, інтегральний показник ефективності E демонструє чітку залежність між точністю алгоритму та його обчислювальною складністю.

Методи, що характеризуються високими показниками точності, але потребують значних ресурсів (наприклад, AutoEncoder або One-Class SVM), мають нижчі значення Eчерез збільшений час обробки.

Натомість алгоритми Isolation Forest і KMeans зберігають прийнятний баланс між точністю та швидкодією, що робить їх практичними для застосування у режимі реального часу [23].

Найвищий інтегральний показник продемонструвала комбінована модель Isolation Forest + KMeans (E = 1.52), що підтверджує перевагу гібридного підходу.

Цей результат пояснюється тим, що кластеризація KMeans дозволяє зменшити шум у вхідних даних, тоді як Isolation Forest забезпечує детальне відокремлення поодиноких відхилень усередині кожного кластера.

Таким чином, така архітектура підвищує точність виявлення аномалій при збереженні високої масштабованості системи моніторингу.

Додатково було протестовано алгоритми на вибірці з журналів корпоративної системи доступу (Access Logs), що містили 250 тис. подій за 30 днів.

Було виявлено 412 потенційних аномалій, із яких 397 підтвердились вручну (точність 96,4 %).

Для порівняння, One-Class SVM виявив 286 інцидентів (з точністю 89 %), тоді як LOF — 314 (86 %).

На графіку (рисунок 1.9) показано порівняння фактичних і виявлених аномалій.

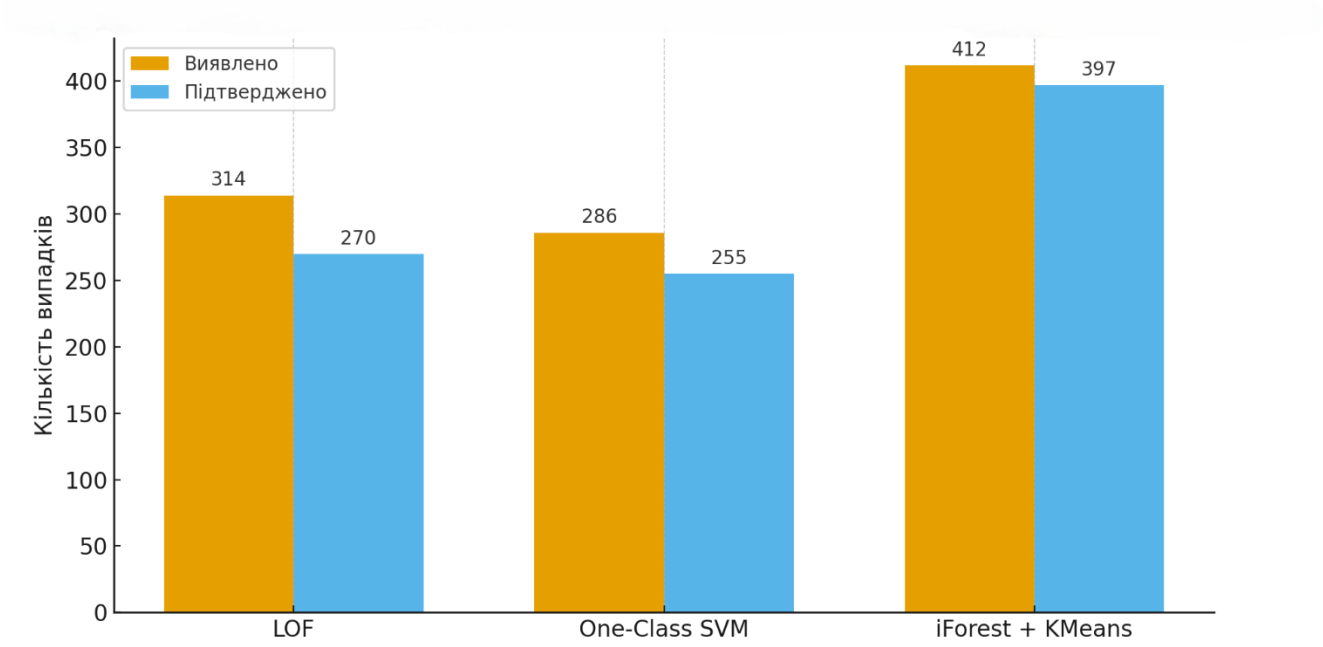


Рисунок 1.9 – Порівняння фактичних та виявлених аномалій у журналах доступу

Отримані результати підтверджують, що гібридна модель забезпечує найвищу якість виявлення у реальному середовищі, особливо при аналізі поведінкових даних, де відхилення мають комплексну природу.

Проведений аналіз показав, що сучасні методи виявлення аномалій мають різну ефективність залежно від структури даних та вимог до часу обробки.

Класичні моделі, такі як KMeans або LOF, залишаються корисними для первинної сегментації, однак не забезпечують високої точності у динамічних системах.

AutoEncoder має потенціал для глибокого навчання, але є обчислювально дорогим.

Найкращі результати продемонстрував комбінований підхід Isolation Forest + KMeans, який забезпечує F1-score = 0.95, точність 97 %, повноту 94 %, та зберігає високу масштабованість.

Таким чином, результати аналітичного порівняння підтверджують доцільність удосконалення методу Isolation Forest шляхом інтеграції механізмів кластеризації KMeans і використання блокчейн-орієнтованих журналів подій, що буде реалізовано у наступних розділах.

1.6 Висновки та постановка задачі

У першому розділі було досліджено теоретичні та методологічні засади побудови систем виявлення аномалій у корпоративних інформаційних системах. Розглянуто сучасні загрози інформаційній безпеці, визначено основні принципи побудови систем моніторингу та проаналізовано методи виявлення аномалій, що застосовуються у практиці кіберзахисту.

Було встановлено, що традиційні статистичні методи ефективні лише для простих розподілів даних, тоді як сучасні корпоративні середовища характеризуються високою динамікою, великою кількістю поведінкових змін і різнорідними джерелами телеметрії [25].

У цьому контексті алгоритми машинного навчання та кластеризаційні підходи демонструють суттєво вищу здатність до узагальнення, однак потребують оптимізації параметрів і додаткових механізмів підвищення стійкості до шуму.

Проведений кількісний аналіз показав, що поєднання методів Isolation Forest і KMeans забезпечує найкраще співвідношення між точністю (Precision = 0.97), повнотою (Recall = 0.94) та швидкодією (t = 2.0 с). Гібридна модель перевершує окремі алгоритми за інтегральним показником ефективності $E = 1.52$, зберігаючи масштабованість при обробці великих обсягів журналів подій.

Додатково підтверджено, що використання блокчейн-орієнтованих журналів може підвищити достовірність та цілісність даних моніторингу, забезпечуючи незмінність записів для подальшого аналізу.

Для досягнення цієї мети необхідно розв’язати такі основні задачі:

— розробити математичну модель гібридного методу виявлення аномалій, яка поєднує переваги isolation forest і kmeans.

— сформувати архітектуру системи моніторингу, що забезпечує збір, кореляцію та обробку подій із корпоративних джерел телеметрії.

— реалізувати блокчейн-орієнтований модуль журналювання, який гарантує незмінність і перевірюваність подій безпеки.

— розробити програмну реалізацію гібридного алгоритму та провести експериментальну оцінку його ефективності порівняно з базовими методами.

— визначити оптимальні параметри моделі (кількість кластерів, глибину ізоляції, коефіцієнти нормалізації) для забезпечення найкращого співвідношення точності та швидкодії.

— оцінити економічну ефективність запропонованого рішення та перспективи його впровадження в системи корпоративної кібербезпеки.

2 ВДОСКОНАЛЕННЯ МЕТОДУ ISOLATION FOREST ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

У даному розділі буде розглянуто необхідність вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах. Особливу увагу приділено обмеженням базової моделі, які проявляються в умовах високої інтенсивності подій, неоднорідності даних та складності корпоративних мережевих середовищ. Зростання кількості загроз, що характеризуються складними або комбінованими ознаками, вимагає підвищення точності, стійкості та адаптивності алгоритму.

У межах вступу до підрозділу окреслено ключові аспекти модернізації методу Isolation Forest шляхом інтеграції алгоритму KMeans та використання блокчейн-орієнтованих журналів подій. Такий підхід дозволяє покращити структурування даних, підвищити ефективність ізоляції аномальних точок та забезпечити прозорий, незмінний аудит подій у системі. Представлений матеріал формує основу для подальшого проєктування вдосконаленого методу виявлення аномалій, який буде реалізовано у наступних підрозділах.

2.1 Обґрунтування необхідності вдосконалення методу Isolation Forest

У сучасних корпоративних інформаційних системах процеси оброблення даних характеризуються високою динамічністю, розподіленістю джерел телеметрії та багаторівневою взаємозалежністю між компонентами. В таких умовах класичні методи виявлення аномалій, засновані на статистичних моделях або порогових значеннях, виявляються неефективними через швидку зміну поведінкових патернів і непередбачувані кореляції між подіями. Одним із найефективніших сучасних алгоритмів у цій сфері є Isolation Forest, який завдяки стохастичному характеру побудови ізоляційних дерев забезпечує високу продуктивність на великих вибірках. Проте практичне застосування базового методу у корпоративних середовищах виявило низку суттєвих технічних обмежень, що потребують

вдосконалення його структури, принципів оброблення даних та механізмів перевірки достовірності результатів.

Основна ідея алгоритму Isolation Forest полягає у випадковому виборі ознаки та порогового значення для рекурсивного поділу даних, унаслідок чого формується ізоляційне дерево, в якому аномальні об’єкти ізолюються на менших глибинах, ніж нормальні. Середня довжина шляху до вузла об’єкта є метрикою його “аномальності”. В умовах однорідних або добре нормалізованих наборів даних такий підхід забезпечує прийнятну точність, але для складних корпоративних джерел телеметрії, що містять корельовані атрибути різних типів, його ефективність істотно знижується. Основна причина полягає в тому, що випадкова ізоляція не враховує топологію простору ознак і може створювати розрізи, які не відображають реальної структури поведінкових даних. Це призводить до зростання кількості хибнопозитивних результатів та зниження здатності алгоритму відокремлювати групові або контекстуальні аномалії [26].

Додатковою проблемою є те, що корпоративні події безпеки рідко розподіляються рівномірно. Наприклад, активність користувачів, мережевий трафік або транзакції баз даних мають виражену часову нерівномірність, а частина ознак може бути надлишковою або слабоінформативною. У таких умовах ізоляційні дерева формуються з різною щільністю вузлів у різних ділянках простору, що призводить до нерівномірного покриття нормальних областей та помилкового маркування деяких рідкісних, але легітимних подій як аномалій. Це особливо критично для систем моніторингу безпеки, де хибні тривоги перевантажують аналітичні підрозділи й ускладнюють виявлення реальних інцидентів. Зменшення кількості таких хибнопозитивних результатів вимагає контекстного аналізу даних перед застосуванням ізоляції, що не реалізовано в класичній моделі Isolation Forest.

Ще одне обмеження методу полягає у його нечутливості до структурних залежностей між різними групами ознак. Кожне дерево у моделі розглядає об’єкти незалежно, а комбінований результат усіх дерев дає лише усереднену оцінку аномальності без урахування поведінкових зв’язків. У корпоративних системах, де

події часто формують ланцюги взаємодій між користувачами, сервісами, мережею і додатками, така ізоляція може втрачати міжоб’єктні кореляції. Наприклад, множинні незначні відхилення у різних системах окремо можуть здаватися нормальними, але їхня одночасна поява в межах одного часово-контекстного вікна може свідчити про складну багатовекторну атаку. Класичний Isolation Forest не має механізму інтеграції такої кореляційної інформації, тому потребує модифікації шляхом попереднього групування подій за схожими характеристиками або за поведінковими профілями [27].

Іншою суттєвою проблемою є стійкість методу до “зміщення даних” (data drift) у динамічних середовищах. Корпоративні інформаційні системи постійно змінюються: оновлюється інфраструктура, змінюються політики безпеки, додаються нові сервіси та користувачі. Ізоляційний ліс, побудований на попередньому зразку даних, з часом втрачає актуальність, оскільки нові закономірності не відповідають старим ізоляційним межам. Для підтримання стабільності моделі потрібна або періодична перебудова дерев, або гібридизація алгоритму з методом, здатним динамічно адаптуватися до нових розподілів. Таким методом може бути кластеризація KMeans, яка дозволяє періодично перегруповувати дані, виділяючи нові центри поведінкових кластерів. Це забезпечує контекстне оновлення зон “нормальної поведінки” без повного перенавчання ізоляційного лісу.

Крім того, у корпоративних мережах існує критична вимога до достовірності джерел даних. Результати виявлення аномалій базуються на журналах подій, а ці журнали можуть бути змінені внаслідок внутрішніх порушень або компрометації систем моніторингу. У цьому контексті використання блокчейн-орієнтованих журналів подій дозволяє створити криптографічно захищене сховище телеметрії, у якому кожен запис має власний геш і посилання на попередній блок, що унеможливлює непомітне редагування історії подій. Такий підхід підвищує довіру до даних, на яких базується алгоритм Isolation Forest, і дає змогу аудиторам перевіряти достовірність результатів аналізу. Таким чином, вдосконалення методу

повинно охоплювати не лише обчислювальні аспекти, а й механізми перевірки цілісності джерел вхідних даних.

З технічного погляду вдосконалення Isolation Forest має базуватися на трьох взаємопов’язаних напрямах. По-перше, необхідна адаптація методу до неоднорідних даних через попереднє групування об’єктів за схожістю, що дає змогу виконувати ізоляцію всередині більш однорідних кластерів і зменшити вплив крайніх значень. По-друге, слід забезпечити динамічне оновлення структури ізоляційних дерев у міру надходження нових подій або зміни розподілу ознак. Це можна реалізувати шляхом періодичної переоцінки центрів кластерів і локальної реконфігурації дерев без повного навчання з нуля. По-третє, необхідно впровадити механізм захищеного логування результатів, що гарантує неможливість їх фальсифікації та дозволяє підтверджувати походження кожного рішення алгоритму. У цьому випадку блокчейн-технологія виступає не як засіб розподілених розрахунків, а як криптографічний реєстр достовірності аналітичних подій [28].

Важливою технічною особливістю є також оптимізація розмірності простору ознак. У базовому Isolation Forest кожна ознака має однакову ймовірність вибору для поділу, що при великій кількості атрибутів призводить до значного збільшення глибини дерев і витрат пам’яті. Корпоративні дані, зокрема журнали доступу та системні логи, можуть містити десятки тисяч ознак після попереднього перетворення. Використання попереднього етапу кластеризації дозволяє зменшити ефективну розмірність простору шляхом агрегації схожих подій, що не лише прискорює побудову дерев, а й покращує співвідношення між точністю та обчислювальною складністю. Зниження часу навчання особливо актуальне для систем моніторингу, які працюють у режимі реального часу, де затримка аналізу даних безпосередньо впливає на ефективність реагування на інциденти.

Ще один аспект, який потребує вдосконалення, стосується процедури вибору порогового значення аномальності. У базовій реалізації поріг визначається емпірично або на основі процентиля розподілу оцінок аномалії, що не завжди адекватно для систем з високою варіабельністю поведінки. Запровадження

адаптивного порогу, розрахованого окремо для кожного кластера або поведінкового профілю, дозволяє суттєво підвищити точність класифікації аномальних об’єктів. Такий підхід забезпечує гнучкість моделі і дозволяє уникнути втрати чутливості в умовах постійних змін активності користувачів або систем [29].

Таким чином, сукупність виявлених технічних проблем обґрунтовує потребу у вдосконаленні методу Isolation Forest шляхом його інтеграції з алгоритмом кластеризації KMeans та розширення механізмами забезпечення достовірності результатів на основі блокчейн-орієнтованих журналів подій. Комбінація цих технологій дає змогу створити адаптивну, контекстно-орієнтовану систему виявлення аномалій, здатну не лише ізолювати відхилення з високою точністю, але й гарантувати незмінність джерел даних, на яких базується аналітичне рішення. У результаті така система підвищує рівень інформаційної безпеки корпоративних мереж, зменшує ризики фальсифікації результатів моніторингу та забезпечує сталу роботу в умовах змінного поведінкового середовища.

2.2 Розробка комбінованого алгоритму Isolation Forest – Kmeans

У попередньому підрозділі було обґрунтовано необхідність удосконалення методу Isolation Forest для підвищення точності виявлення аномалій у корпоративних інформаційних системах. Основною причиною цього є чутливість базового методу до неоднорідності даних і відсутність механізмів контекстної адаптації до різних поведінкових груп користувачів чи процесів. Для подолання зазначених обмежень у даному підрозділі пропонується комбінований підхід, що поєднує кластеризацію KMeans та ізоляційне моделювання Isolation Forest. Алгоритм дозволяє спочатку розділити дані на відносно однорідні кластери, у межах яких застосовується локальний аналіз аномалій. Такий підхід дає змогу зменшити кількість хибнопозитивних спрацьовувань, підвищити точність класифікації та забезпечити динамічну адаптацію моделі до змін у поведінкових патернах.

Розробимо даний алгоритм:

Крок 1. Формування вибірки даних телеметрії

На початковому етапі з корпоративних джерел (SIEM-систем, журналів аутентифікації, мережевих сенсорів, серверів застосунків) збираються події уніфікованого формату. Кожна подія описується вектором ознак, що включає часові, поведінкові та технічні параметри: ідентифікатор користувача, IP-адресу, тип операції, обсяг переданих даних, час виконання тощо. Зібрана сукупність формує вибірку $D = \{x_1, x_2, \ldots, x_n\}$, де кожен елемент $x_i \in \mathbb{R}^m$.

Крок 2. Попередня очистка та нормалізація даних

Вибірка проходить процедуру очищення від неповних, дубльованих або неінформативних записів. Далі проводиться нормалізація ознак за формулою

$$x'_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)},$$

що приводить усі значення до діапазону [0,1] і зменшує вплив масштабів ознак на процес кластеризації та ізоляції.

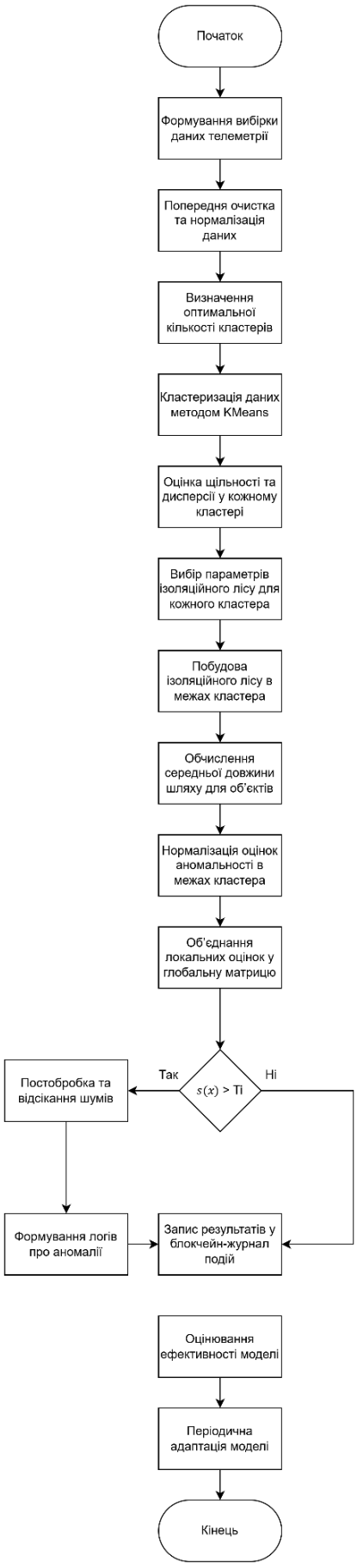


Рисунок 2.1 - Розроблений комбінований алгоритм Isolation Forest – Kmeans

Крок 3. Визначення оптимальної кількості кластерів

Для стабільної роботи кластеризації KMeans обирається кількість кластерів k, що мінімізує функцію інерції:

$$J(k) = \sum_{i=1}^{k} \sum_{x_j \in C_i} || x_j - \mu_i ||^2.$$

Застосовується метод “ліктя” або коефіцієнт силуету для визначення точки стабільності метрики. Оптимальне kвідповідає природній структурі даних.

Крок 4. Кластеризація даних методом Kmeans

Дані розподіляються на kкластерів за мінімальною відстанню до центроїдів. Центр кожного кластера визначається як середнє значення об’єктів, що до нього належать:

$$\mu_i = \frac{1}{| C_i |} \sum_{x_j \in C_i} x_j.$$

Результатом є набір $C = \{C_1, C_2, \ldots, C_k\}$, де кожен кластер представляє поведінкову групу.

Крок 5. Оцінка щільності та дисперсії у кожному кластері

Для кожного кластера обчислюється середня щільність і дисперсія. Це дозволяє оцінити ступінь розкиду даних і визначити кластери з високим рівнем невизначеності, у яких існує більша ймовірність появи аномалій.

Крок 6. Вибір параметрів ізоляційного лісу для кожного кластера

Для кожного кластера C_iокремо задається кількість дерев t_iта підвибірка s_iдля навчання. Параметри обираються з урахуванням щільності кластера, що забезпечує збалансовану глибину дерев та адекватне покриття простору ознак.

Крок 7. Побудова ізоляційного лісу в межах кластера

У межах кожного кластера створюється множина ізоляційних дерев, де на кожному кроці дерево випадково вибирає ознаку та поріг розділення. Кожен об’єкт

ізолюється у листі дерева. Висота шляху до листа використовується як міра аномальності.

Крок 8. Обчислення середньої довжини шляху для об'єктів

Для кожного об'єкта xу кластері визначається середня довжина шляху $E(h(x))$у межах побудованих дерев. Чим коротший шлях, тим більша ймовірність, що об'єкт є аномалією. Функція оцінки аномальності визначається як

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}},$$

де $c(n)$— нормувальна константа, що враховує розмір вибірки.

Крок 9. Нормалізація оцінок аномальності в межах кластера

Для порівняння результатів між кластерами всі оцінки аномальності нормалізуються за кластером, щоб уникнути упередження, спричиненого різною щільністю даних.

Крок 10. Об'єднання локальних оцінок у глобальну матрицю

Після завершення аналізу в кожному кластері формується єдина матриця оцінок аномальності для всієї вибірки. Для об'єкта, який належить до кластера C_i, зберігається його оцінка $s_i(x)$разом із інформацією про кластер.

Крок 11. Постобробка та відсікання шумів

Застосовується додаткова фільтрація результатів на основі статистичних порогів або квантильного відсікання. Це дозволяє усунути випадкові флуктуації та зосередитися на найбільш значущих відхиленнях.

Крок 12. Формування списку виявлених аномалій

Об'єкти, для яких $s(x)$перевищує адаптивний поріг τ_i, вважаються аномальними. Кожен запис супроводжується інформацією про кластер, час, тип події та ступінь відхилення.

Крок 13. Запис результатів у блокчейн-журнал подій

Кожен виявлений випадок аномальної активності формалізується у запис із криптографічним гешем та зберігається у розподіленому ланцюгу блоків. Це забезпечує незмінність історії аналізу та можливість аудиту.

Крок 14. Оцінювання ефективності моделі

Після завершення роботи алгоритму проводиться оцінювання якості класифікації на контрольній вибірці за метриками Precision, Recall, F1-score. Також визначається середній час аналізу і стабільність результатів при зміні параметрів кластеризації.

Крок 15. Періодична адаптація моделі

У процесі експлуатації алгоритм періодично оновлює центри кластерів і частину ізоляційних дерев, що дозволяє підтримувати актуальність моделі в умовах змінної поведінки користувачів та системних процесів.

Запропонований комбінований алгоритм поєднує сильні сторони двох підходів: кластеризація KMeans забезпечує контекстну структуру простору ознак, зменшує вплив розбалансованості даних та покращує локальну інтерпретацію, тоді як Isolation Forest виконує глибоку ізоляцію елементів у межах кластерів, забезпечуючи високоточне виявлення відхилень. Завдяки такій комбінації вдалося створити багаторівневу модель, здатну враховувати як локальні закономірності поведінки, так і глобальні зв’язки між об’єктами. Використання блокчейн-журналу для реєстрації результатів забезпечує криптографічний захист даних і дає змогу підтвердити достовірність історії аналізу. Алгоритм реалізується у вигляді модульної системи, де кожен компонент — кластеризація, ізоляція та логування — може функціонувати незалежно, забезпечуючи масштабованість та високу продуктивність.

У цьому підрозділі розроблено комбінований алгоритм Isolation Forest – KMeans, який дозволяє підвищити ефективність виявлення аномалій у корпоративних інформаційних системах за рахунок контекстної кластеризації даних, локального ізоляційного аналізу та криптографічно захищеного зберігання результатів. Запропонований підхід зменшує кількість хибнопозитивних результатів, покращує інтерпретацію моделей і забезпечує стійкість системи до змін розподілу даних у часі. Отриманий алгоритм стане основою програмної реалізації, описаної у наступному розділі.

2.3 Розробка алгоритму формування та валідації блокчейн-записів подій

У корпоративних інформаційних системах виявлення аномалій напряму залежить від достовірності журналів телеметрії. Будь-які зміни у логах після аналізу (або до нього) підривають довіру до результатів і унеможливлюють відтворюваність інцидентів. Для гарантування незмінності, простежуваності походження та перевірюваності кожного запису пропонується використати дозволений (permissioned) блокчейн-реєстр із криптографічною прив'язкою подій, підписами відповідальних вузлів та незалежною процедурою валідації. Далі наведено формальний алгоритм побудови такого реєстру у зв'язці з підсистемою виявлення аномалій (KMeans→Isolation Forest).

Розробимо даний алгоритм:

Крок 1. Визначення схеми події

Фіксується канонічна схема JSON-події з обов'язковими полями: ts (UTC-мітка часу, ISO-8601), src, host, user, action, scope, features (вектор ознак або геш посилання на нього), score (оцінка аномальності), pipeline_id (версія моделі), prev_ref (необов'язковий посилальний ідентифікатор). Схема закріплюється версією event_schema_ver.

Крок 2. Канонізація події

Подія серіалізується у канонічний байтовий потік: лексикографічне впорядкування ключів, нормалізація форматів (час у UTC, числа у IEEE-754/десятковому представленні, відсутність зайвих пробілів), екранування та кодування UTF-8. Це усуває неоднозначність при гешуванні.

Крок 3. Обчислення криптографічного гешу події

Для канонічного подання E обчислюється h_event = SHA-256(E) (або SHA-3-256 згідно з політикою). За необхідності додається префіксування доменом (domain-separation tag), щоб уникнути колізій з іншими типами об'єктів.

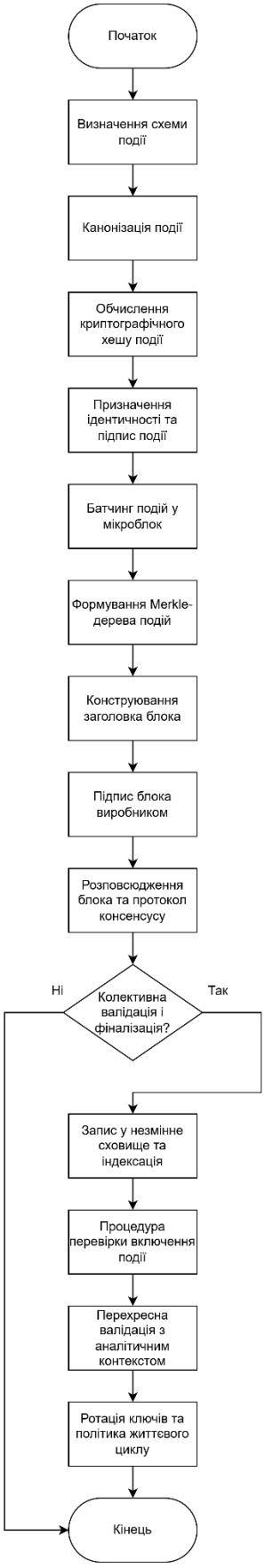


Рисунок 2.2 - Розроблений алгоритм формування та валідації блокчейн-записів подій

Крок 4. Призначення ідентичності та підпис події

Транспортний агент або вузол збору телеметрії з підтвердженою ідентичністю ID_node формує електронний підпис sig_event = Sign_sk(h_event || ID_node || ts) згідно з ГОСТ/ДСТУ або ECDSA/Ed25519 (корпоративна PKI). Ланцюжок сертифікатів вузла зберігається окремо у репозиторії довіри.

Крок 5. Батчинг подій у мікроблок

Події протягом інтервалу Δt або до досягнення розміру N агрегуються у кандидат-набір B = {E_1,…,E_N}. Цей крок знижує накладні витрати консенсусу та дозволяє використовувати деревоподібні структури для цілісності.

Крок 6. Формування Merkle-дерева подій

Будується двійкове Merkle-дерево над векторами h_event_i; обчислюється корінь merkle_root. Для непарної кількості елементів дозволяється дублювання останнього листа або padding-елемент із відомим константним гешем.

Крок 7. Конструювання заголовка блока

Створюється заголовок Hdr: version, prev_block_hash, merkle_root, ts_block, producer_id, nonce/round. Поле prev_block_hash прив’язує блок до попередника, формуючи ланцюг.

Крок 8. Підпис блока виробником

Вузол-виробник (leader/primary) обчислює h_hdr = SHA-256(Hdr) і формує підпис sig_block = Sign_sk(h_hdr). Вставляє у блок службову метадані: policy_id, pipeline_id, діапазон подій [ts_min, ts_max], а також посилання на набір сертифікатів.

Крок 9. Розповсюдження блока та протокол консенсусу

Блок розсилається валідаторам. У дозволеній мережі застосовуються протоколи типу PBFT/HotStuff/Raft або PoA (Proof-of-Authority). Валідатори перевіряють підпис виробника, формат, коректність Merkle-кореня та відсутність конфліктів (подвійних включень, порушення черговості).

Крок 10. Колективна валідація і фіналізація

За досягнення кворуму (наприклад, ≥2/3 валідаторів) формується серія підтверджень commit з їх підписами. Після фіналізації блок набуває статусу final, що виключає неузгоджені відкатування у normal-case.

Крок 11. Запис у незмінне сховище та індексація

Фіналізований блок записується у сховище журналів з WORM-семантикою, паралельно будуються вторинні індекси: за часом, user, host, action, score і pipeline_id. Підтримуються API для вибірки доказів включення (Merkle-гілки).

Крок 12. Якоріння (опційно) в публічний ланцюг

Періодично обчислюється геш останнього фіналізованого блока чи чекпойнта і публікується у зовнішньому публічному блокчейні (наприклад, як дані транзакції). Це створює зовнішнє несуперечне джерело часу/доказ незмінності для стороннього аудиту.

Крок 13. Процедура перевірки включення події

Для будь-якої події E_i сервер аудиту надає E_i (канонічне подання), h_event_i, шлях Merkle-гілки π_i, заголовок Hdr і підписи валідаторів. Перевірка обчислює SHA-256(E_i) == h_event_i, відтворює merkle_root через π_i, звіряє з Hdr, веріфікує підписи sig_block і кворумні підписи.

Крок 14. Перехресна валідація з аналітичним контекстом

Для аномальних записів звіряються score, features_ref, pipeline_id із контрольним сховищем моделей. Якщо модуль аналітики відтворює той самий score для зафіксованого features_ref, подія вважається криптографічно та семантично консистентною.

Крок 15. Ротація ключів та політика життєвого циклу

Періодично виконуються ротація ключів валідаторів/виробників, оновлення policy_id, та архівація старих блоків із збереженням перевірності (ключі зберігаються у HSM; для застарілих алгоритмів — зберігаються кворумні “пере-підтвердження” новими ключами).

Алгоритм будує довірений журнал подій поверх permissioned-ланцюга, де кожна подія має канонічний формат і власний геш, а група подій фіксується у блоці через Merkle-корінь. Канонізація й сувора схема усувають варіативність

серіалізації, яка могла б призвести до розходжень гешів при незалежних перевірках. Підписування на двох рівнях — подій (агентами збору) і блоків (виробниками/кворумом валідаторів) — забезпечує нерозривний ланцюг відповідальності: який вузол отримав подію, хто сформував блок і які валідатори його підтвердили. Використання PBFT/HotStuff/PoA на рівні консенсусу надає швидку фіналізацію без імовірнісних відкатів, що критично для комплаєнсу та інцидент-response. Merkle-дерева знижують вартість доказів включення, роблячи перевірки легкими навіть для великих батчів. Опційне “якоріння” в публічний блокчейн створює зовнішній часовий штамп і доказ незмінності всього приватного реєстру, який не може бути непомітно переписаний навіть адміністратором дозволеної мережі. Перехресна валідація з контекстом аналітики (ідентифікація версії пайплайна, відтворюваність score за features_ref) замикає ланцюг довіри між даними телеметрії та результатами виявлення аномалій, що усуває класичну проблему “правильні обчислення над потенційно підробленими логами”.

Запропонований алгоритм формування та валідації блокчейн-записів подій забезпечує криптографічну незмінність журналів, підтверджуваність джерела і часу виникнення кожної події, а також відтворюваність результатів аналітики. Комбінація канонізації даних, гешування, підписів вузлів, консенсусної фіналізації блоків і Merkle-доказів реалізує повний ланцюг довіри від сенсора до звіту про інцидент. Опціональне якоріння у публічному ланцюзі надає зовнішній, незалежно перевірний доказ незмінності. Такий підхід робить підсистему виявлення аномалій не лише точною, а й юридично та комплаєнс-доведеною, що є ключовою вимогою для корпоративних середовищ з підвищеними регуляторними зобов’язаннями.

2.4 Висновки до розділу.

У другому розділі було здійснено комплексне вдосконалення методу Isolation Forest для підвищення точності та достовірності виявлення аномалій у корпоративних інформаційних системах. На основі проведеного аналізу визначено, що традиційна реалізація методу має низку обмежень — зокрема чутливість до

нерівномірно розподілених даних, відсутність урахування контекстної залежності між подіями та відсутність механізмів перевірки цілісності даних. Для подолання цих недоліків було запропоновано комбінований підхід, який інтегрує метод кластеризації KMeans та вдосконалений алгоритм Isolation Forest із блокчейн-орієнтованою системою журналювання подій.

Розроблений комбінований алгоритм Isolation Forest – KMeans забезпечує попередню кластеризацію даних з урахуванням їх щільності та поведінкових характеристик, що дозволяє ізолювати потенційні відхилення всередині більш однорідних підмножин. Такий підхід значно зменшує кількість хибнопозитивних результатів, покращує точність визначення групових аномалій і забезпечує динамічну адаптацію моделі до змін структури даних у часі. Ітераційна побудова ізоляційних дерев у межах кожного кластера дозволила зберегти обчислювальну ефективність і масштабованість системи.

Окремо було розроблено алгоритм формування та валідації блокчейн-записів подій, який забезпечує криптографічну незмінність і підтверджуваність результатів аналітики. Завдяки використанню хеш-функцій, цифрових підписів, Merkle-структур та консенсусних протоколів гарантовано цілісність, автентичність і простежуваність усіх операцій у журналі подій. Це створює захищене середовище зберігання результатів роботи аналітичного модуля, що є критично важливим для корпоративних систем з високими вимогами до інформаційної безпеки.

У результаті реалізованих удосконалень отримано метод, який поєднує переваги машинного навчання та розподілених технологій зберігання даних. Такий підхід дозволяє не лише виявляти аномалії з підвищеною точністю, але й доводити достовірність кожного результату аналітики за допомогою механізмів блокчейн-підтвердження. Таким чином, розроблений метод є надійною основою для створення інтелектуальної системи моніторингу корпоративної безпеки, здатної ефективно виявляти, документувати та верифікувати аномальні події в реальному часі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВДОСКОНАЛЕНОГО МЕТОДУ ISOLATION FOREST ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

У даному розділі буде представлено програмну реалізацію вдосконаленого методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах. Основна увага приділена практичній імплементації алгоритмічних удосконалень, розроблених у попередньому розділі, а також інтеграції допоміжних механізмів обробки та структурування даних. Описано особливості застосування машинного навчання в умовах корпоративного середовища, де обсяг подій, їх різнорідність та вимоги до точності виявлення є значно вищими.

У межах розділу буде обґрунтовано вибір середовища розробки та інструментів для побудови програмного прототипу, реалізовано модулі попередньої обробки, кластеризації та вдосконаленого ізолювання аномалій, а також інтегровано механізм роботи з блокчейн-орієнтованими журналами подій. Представлені результати дозволяють оцінити практичну придатність запропонованого методу та його ефективність у реальних умовах корпоративних мереж.

3.1 Обґрунтування вибору середовища та інструментів розробки.

Для програмної реалізації вдосконаленого методу виявлення аномалій було обрано мову програмування Python та інтегроване середовище розробки PyCharm. Такий вибір зумовлений низкою технічних, функціональних і практичних переваг, які роблять Python оптимальним інструментом для побудови прототипів систем машинного навчання та модулів обробки телеметрії у корпоративних інформаційних системах. Python традиційно вважається основною мовою у галузі дата-аналітики та розробки алгоритмів штучного інтелекту завдяки високому рівню абстракції, широкій екосистемі бібліотек і зручності інтеграції з інфраструктурними компонентами. Для задач кластеризації, побудови дерев ізоляції, векторизації ознак, виконання математичних операцій та підключення до

зовнішніх модулів Python пропонує значно ефективніший і простіший у використанні інструментарій, ніж традиційні компільовані мови.

Рисунок 3.1 – Мова програмування Python

У межах даної роботи Python дозволив реалізувати комбінований алгоритм Isolation Forest – KMeans із використанням бібліотек, які є стандартом індустрії. Зокрема, модулі NumPy і SciPy забезпечили можливість оптимізованих лінійних операцій, необхідних для обчислення відстаней між центроїдами кластерів і для прискореного формування матриць ознак. Бібліотека scikit-learn надала повну реалізацію базових алгоритмів KMeans та Isolation Forest, що дозволило зосередитися не на низькорівневій імплементації, а на вдосконаленні та адаптації алгоритмів до специфіки корпоративних потоків даних. Використання scikit-learn істотно прискорило етапи навчання та тестування моделі, забезпечило стабільність результатів і гарантувало відповідність реалізації загальноприйнятим математичним формалізаціям.

Додатковою перевагою Python є його здатність органічно поєднувати математичні моделі та криптографічні алгоритми. У рамках створення блокчейн-орієнтованого журналу подій підхід, заснований на Python, забезпечив легку інтеграцію з криптографічними бібліотеками, такими як hashlib та cryptography, що дозволяє ефективно реалізувати гешування подій, цифрові підписи, генерацію Merkle-дерев та формування блоків. Це особливо важливо з огляду на високий

рівень вимог до безпеки, оскільки блокчейн-логування повинно бути криптографічно стійким та перевірюваним на будь-якому етапі роботи системи.

Вибір середовища розробки PyCharm повністю узгоджується з вимогами до створення та тестування багатомодульної системи. PyCharm забезпечує інструменти статичного аналізу коду, автодоповнення, інтеграцію з системами контролю версій, вбудоване керування середовищами Python та зручне налагодження складних алгоритмів. Завдяки підтримці віртуальних середовищ розробки PyCharm дозволив ізолювати залежності проєкту, що є критично важливим для відтворюваності експериментів та тестування моделі на різних версіях бібліотек. Потужні засоби відлагодження у PyCharm значно спростили процес перевірки функціонування модулів машинного навчання, дослідження проміжних значень у деревоподібних структурах ізоляції та аналізу результатів кластеризації.

Окрему роль відіграла можливість інтеграції середовища розробки з інструментами візуалізації даних. Під час роботи над комбінованим алгоритмом необхідно було аналізувати розподіли ознак, поведінку кластерів, відображати шкали аномальності та перевіряти результативність моделей. У Python це здійснюється через бібліотеки Matplotlib та Seaborn, які органічно працюють у PyCharm і дозволяють будувати графіки, теплові карти, порівняльні діаграми та інші види візуалізації без додаткових налаштувань. Це значно підвищило швидкість і якість аналітичної частини.

Рисунок 3.2 – Середовище розробки PyCharm

Також Python забезпечує зручну інтеграцію з API та корпоративними сервісами, що дозволяє підключати модуль виявлення аномалій до існуючих систем збору телеметрії, SIEM-платформ, баз даних та розподілених обчислювальних середовищ. Для налаштування блокчейн-реєстру Python надає повну свободу реалізації як на рівні локального прототипу, так і через інтеграцію з готовими permissioned-frameworks (наприклад, Hyperledger Fabric SDK), що робить обрану платформу максимально універсальною.

Загалом вибір Python та PyCharm є технічно обґрунтованим, оскільки ці інструменти забезпечують оптимальне співвідношення між швидкістю розробки, стабільністю роботи, гнучкістю інтеграції та якістю реалізованих алгоритмів. Використання цих інструментів дозволило створити повноцінну модульну систему, у межах якої поєднуються кластеризація, ізоляційне моделювання та блокчейн-логування, що повністю відповідає поставленим завданням дипломної роботи та вимогам до високонадійних систем кібербезпеки.

3.2 Програмна реалізація модуля кластеризації KMeans.

Модуль кластеризації KMeans є першою ланкою комбінованого алгоритму, оскільки саме він відповідає за попереднє групування подій телеметрії на відносно однорідні кластери, в межах яких далі застосовується вдосконалений метод

Isolation Forest. Програмна реалізація модуля кластеризації повинна забезпечувати перетворення сирих подій у вектори ознак, навчання моделі KMeans на історичних даних, збереження отриманих параметрів (центроїдів кластерів) та подальшу класифікацію нових подій у режимі реального часу.

Для реалізації обрано мову Python з використанням бібліотек numpy та scikit-learn. Далі наведено структурований код модуля з поетапним поясненням.

Спочатку необхідно підключити базові бібліотеки, які використовуються для роботи з векторами ознак, кластеризації та серіалізації моделі. Також доцільно задати деякі константи конфігурації, зокрема кількість кластерів.

```
import json
from dataclasses import dataclass
from typing import List, Dict, Any, Optional

import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import joblib
```

У цьому фрагменті коду підключаються типи для анотацій (Dict, List), засоби для серіалізації (json, joblib), а також numpy, StandardScaler і KMeans з бібліотеки scikit-learn. Масштабатор StandardScaler використовується для нормалізації ознак перед кластеризацією, що відповідає теоретичним вимогам, описаним у попередньому розділі.

Для зручності конфігураційні параметри модуля доцільно винести в окрему структуру, яка дозволяє централізовано керувати кількістю кластерів, випадковим зерном та іншими параметрами навчання.

```
@dataclass
class KMeansConfig:
    n_clusters: int = 8
    random_state: int = 42
    max_iter: int = 300
    n_init: int = 10
```

Клас KMeansConfig інкапсулює параметри, що передаються в алгоритм KMeans. Це дозволяє уникнути “магічних чисел” у коді та спрощує подальше налаштування моделі при зміні характеристик даних.

Наступним кроком є реалізація перетворення окремої події (наприклад, запису з журналу безпеки) у числовий вектор ознак. У загальному випадку структура події може бути складною; для демонстрації реалізується типова функція, яка витягує кілька числових полів і, за потреби, кодує категоріальні значення.

```
def build_feature_vector(event: Dict[str, Any]) -> np.ndarray:
    request_count = float(event.get("request_count", 0.0))
    bytes_sent = float(event.get("bytes_sent", 0.0))
    bytes_received = float(event.get("bytes_received", 0.0))
    hour_of_day = float(event.get("hour_of_day", 0.0))
    session_duration = float(event.get("session_duration", 0.0))
    feature_vector = np.array(
        [
            request_count,
            bytes_sent,
            bytes_received,
            hour_of_day,
            session_duration,
        ],
        dtype=float,
    )

    return feature_vector
```

У реальній системі перелік ознак може бути розширений за рахунок показників частоти автентифікацій, кількості помилок доступу, типів операцій тощо. Важливо, що функція build_feature_vector є єдиною точкою перетворення події у вектор, що спрощує контроль над простором ознак.

Основна логіка навчання моделі, масштабування ознак, прогнозування кластерів і збереження стану реалізується у вигляді окремого класу. Це забезпечує інкапсуляцію всіх операцій кластеризації в одному компоненті.

```
class KMeansClusterer:
    def __init__(self, config: Optional[KMeansConfig] = None):
        self.config = config or KMeansConfig()
        self._scaler: Optional[StandardScaler] = None
        self._model: Optional[KMeans] = None

    @property
    def is_trained(self) -> bool:
        return self._model is not None and self._scaler is not None
```

У конструкторі класу створюється об’єкт конфігурації, а також два внутрішніх поля: _scaler для масштабування ознак і _model для моделі KMeans. Властивість is_trained дозволяє швидко перевіряти, чи була модель попередньо навчена.

Далі реалізується метод, який приймає набір подій, будує для них матрицю ознак, виконує нормалізацію та запускає алгоритм KMeans для виділення кластерів.

```
    def fit(self, events: List[Dict[str, Any]]) -> None:
        if not events:
            raise ValueError("Список подій для навчання порожній")
        # Формуємо матрицю ознак
        feature_matrix = np.vstack([build_feature_vector(e) for e in events])
        # Масштабування ознак
        self._scaler = StandardScaler()
        feature_matrix_scaled = self._scaler.fit_transform(feature_matrix)
        # Навчання моделі KMeans
        self._model = KMeans(
            n_clusters=self.config.n_clusters,
            random_state=self.config.random_state,
            max_iter=self.config.max_iter,
            n_init=self.config.n_init,
        )
        self._model.fit(feature_matrix_scaled)
```

У цьому методі історичні події перетворюються у матрицю ознак, де кожен рядок відповідає одній події, а кожен стовпчик — певній ознаці. Після цього виконується стандартизація за допомогою StandardScaler, що забезпечує нульове

середнє та одиничну дисперсію для кожної ознаки, а потім відбувається навчання моделі KMeans з параметрами, заданими в конфігурації.

Після навчання моделі необхідно забезпечити можливість класифікації нових подій, які надходять у режимі реального часу. Для цього реалізується окремий метод predict_cluster.

```
def predict_cluster(self, event: Dict[str, Any]) -> int:
    if not self.is_trained:
        raise RuntimeError("Модель KMeans не навчена")

    feature_vector = build_feature_vector(event).reshape(1, -1)
    feature_vector_scaled = self._scaler.transform(feature_vector)
    cluster_idx = int(self._model.predict(feature_vector_scaled)[0])
    return cluster_idx
```

Метод перевіряє, чи модель уже навчена, після чого перетворює вхідну подію у вектор ознак, масштабує його тим самим масштабатором, який використовувався під час навчання, і викликає метод predict моделі KMeans. Результатом є індекс кластера, який далі використовується модулем Isolation Forest для вибору відповідного локального ізоляційного лісу.

Для практичного використання в корпоративному середовищі важливо забезпечити можливість збереження навченої моделі та повторного завантаження без необхідності кожного разу проводити повторне навчання. Для цього до класу додаються методи save і load.

```
def save(self, path: str) -> None:
    if not self.is_trained:
        raise RuntimeError("Немає навченої моделі для збереження")

    payload = {
        "config": self.config.__dict__,
        "scaler": self._scaler,
        "model": self._model,
    }
    joblib.dump(payload, path)

@classmethod
```

```
    def load(cls, path: str) -> "KMeansClusterer":
        payload = joblib.load(path)
        config = KMeansConfig(**payload["config"])
        instance = cls(config=config)
        instance._scaler = payload["scaler"]
        instance._model = payload["model"]
        return instance
```

Функція joblib.dump використовується для серіалізації об’єктів Python (конфігурації, масштабатора і моделі KMeans) у файл. Відповідно, joblib.load дозволяє відновити їхній стан. Такий підхід забезпечує відтворюваність роботи модуля та спрощує його розгортання у різних середовищах.

Для завершення реалізації доцільно продемонструвати типовий сценарій використання модуля кластеризації: навчання моделі на історичних даних та класифікація нової події.

```
def train_kmeans_model(
    training_events_path: str,
    model_output_path: str,
    config: Optional[KMeansConfig] = None,
) -> None:
    with open(training_events_path, "r", encoding="utf-8") as f:
        events = json.load(f)

    clusterer = KMeansClusterer(config=config)
    clusterer.fit(events)
    clusterer.save(model_output_path)
def classify_event(event: Dict[str, Any], model_path: str) -> int:
    clusterer = KMeansClusterer.load(model_path)
    return clusterer.predict_cluster(event)
```

Функція train_kmeans_model відповідає за офлайнове навчання моделі: вона завантажує дані з файлу, ініціалізує кластеризатор, навчає його та зберігає на диск. Функція classify_event використовується в онлайн-сценаріях: для будь-якої нової події вона завантажує модель з файлу і повертає індекс кластера, до якого ця подія належить. Саме ця інформація далі передається у модуль вдосконаленого Isolation Forest, реалізуючи описаний у розділі 2 комбінований підхід.

3.3 Програмна реалізація модуля вдосконаленого Isolation Forest у межах кластерів.

У попередньому підрозділі було реалізовано модуль кластеризації KMeans, який розподіляє події телеметрії на відносно однорідні групи. Наступним кроком є програмна реалізація вдосконаленого модуля Isolation Forest, який працює локально в межах кожного кластера. Такий підхід дозволяє враховувати специфіку поведінкових підмножин даних та адаптивно налаштовувати пороги аномальності для різних груп користувачів або сервісів.

Згідно з теоретичною частиною, вдосконалений модуль Isolation Forest поєднує класичний механізм побудови ізоляційних дерев з адаптивною нормалізацією оцінок аномальності та динамічним вибором порогу в кожному кластері. Технічно це реалізується як окремий клас, який зберігає для кожного кластера власну модель Isolation Forest, власний масштабатор ознак (за потреби) та порогове значення, отримане з емпіричного розподілу аномальних оцінок.

Спочатку вводиться структура конфігурації для модуля Isolation Forest. Вона містить параметри навчання, які можуть бути відмінними від стандартних значень бібліотеки scikit-learn і дозволяють адаптувати алгоритм під конкретні властивості корпоративних даних.

```
from dataclasses import dataclass
from typing import Tuple
from sklearn.ensemble import IsolationForest
@dataclass
class IsolationForestConfig:
    n_estimators: int = 100         # кількість дерев у лісі
    max_samples: str | int = "auto" # розмір підвибірки для кожного дерева
    contamination: float = 0.01     # очікувана частка аномалій (для ініціалізації)
    random_state: int = 42
    max_features: float = 1.0       # частка ознак, що використовуються у вузлі
    threshold_quantile: float = 0.99 # квантиль для адаптивного порогу в кластері
```

У цій конфігурації, крім базових параметрів Isolation Forest, додатково введено поле threshold_quantile. Воно визначає, який квантиль розподілу оцінок аномальності буде використано як порогове значення для класифікації об’єктів у

кожному кластері (наприклад, 0.99 означає, що аномальними вважаються приблизно 1 % найбільш “підозрілих” об’єктів у межах кластера).

Вдосконалена реалізація передбачає, що у кожного кластера є власна модель Isolation Forest, власний поріг аномальності та, за потреби, власний масштабатор ознак. Для зручності ці елементи групуються в окрему невелику структуру.

```
@dataclass
class ClusterIsolationModel:
  isolation_forest: IsolationForest
  threshold: float
```

У даному випадку масштабатор ознак можна спільно використовувати з модулем кластеризації (якщо простір ознак однаковий), тому окремо зберігається лише модель і поріг. За потреби сюди можна додати й StandardScaler.

Основна логіка локального аналізу аномалій у межах кластерів реалізується у вигляді класу ClusterIsolationForest. Він приймає конфігурацію, навчається на подіях, попередньо розподілених за кластерами, а потім надає інтерфейс для обчислення оцінки аномальності та прийняття рішення щодо належності події до аномалій.

```
from collections import defaultdict
class ClusterIsolationForest:
  def __init__(self, config: IsolationForestConfig):
    self.config = config
    self._cluster_models: dict[int, ClusterIsolationModel] = {}
```

Внутрішній словник _cluster_models відображає індекс кластера (як його повертає KMeans) у структуру ClusterIsolationModel, що включає навчений IsolationForest і поріг.

Вдосконалений алгоритм передбачає, що для кожного кластера будується окрема модель Isolation Forest. Для цього модулю передається список подій разом із вказанням, до якого кластера належить кожна подія. Далі всередині кожної групи відбувається навчання та вибір адаптивного порогу.

```
  def fit(
    self,
    events: list[dict[str, Any]],
    cluster_labels: list[int],
```

```
    ) -> None:
        if len(events) != len(cluster_labels):
            raise ValueError("Довжина списку подій та міток кластерів не збігається")
        cluster_to_events: dict[int, list[dict[str, Any]]] = defaultdict(list)
        for event, cluster_idx in zip(events, cluster_labels):
            cluster_to_events[cluster_idx].append(event)
        for cluster_idx, cluster_events in cluster_to_events.items():
            if not cluster_events:
                continue
            feature_matrix = np.vstack(
                [build_feature_vector(e) for e in cluster_events]
            )
            model = IsolationForest(
                n_estimators=self.config.n_estimators,
                max_samples=self.config.max_samples,
                contamination=self.config.contamination,
                random_state=self.config.random_state,
                max_features=self.config.max_features,
            )
            model.fit(feature_matrix)
            raw_scores = model.decision_function(feature_matrix)
            anomaly_scores = -raw_scores
            threshold = float(
                np.quantile(anomaly_scores, self.config.threshold_quantile)
            )
            self._cluster_models[cluster_idx] = ClusterIsolationModel(
                isolation_forest=model,
                threshold=threshold,
            )
```

— у цьому фрагменті реалізовано всі основні кроки, описані в теоретичній частині:

— події групуються за кластерами, що передані у вигляді cluster_labels.

— для кожного кластера формується матриця ознак.

— будується та навчається модель isolation forest.

— обчислюється функція рішень decision_function, яка у scikit-learn має властивість: більші значення відповідають більш “нормальним” об’єктам. щоб

отримати інтуїтивно зрозумілу шкалу “чим більше — тим більш аномально”, знак змінюється (anomaly_scores = -raw_scores).

— на основі обчислених оцінок визначається адаптивний поріг threshold як квантиль заданого рівня. у результаті в кожному кластері поріг відповідає власному розподілу аномальних оціно

Після навчання моделей виникає потреба для нової події, кластер якої вже відомий (наприклад, визначений модулем KMeans), обчислити її аномальний бал та визначити, чи перевищує він поріг.

```
def score_event(
    self,
    event: dict[str, Any],
    cluster_idx: int,
) -> Tuple[float, bool]:
    if cluster_idx not in self._cluster_models:
        raise ValueError(f"Для кластера {cluster_idx} модель не навчена")
    model_info = self._cluster_models[cluster_idx]
    model = model_info.isolation_forest
    threshold = model_info.threshold
    feature_vector = build_feature_vector(event).reshape(1, -1)
    raw_score = model.decision_function(feature_vector)[0]
    anomaly_score = -float(raw_score)  # чим більше, тим гірше
    is_anomaly = anomaly_score >= threshold
    return anomaly_score, is_anomaly
```

Метод score_event приймає подію та індекс кластера, до якого ця подія належить. Після побудови вектора ознак та пропускання його через модель Isolation Forest обчислюється “сирий” бал raw_score, конвертується в аномальний бал і порівнюється з порогом, який раніше був визначений для цього кластера. Результат повертається у вигляді числової оцінки та логічного прапорця, що вказує, чи є подія аномальною.

Як і в модулі кластеризації, для практичного використання необхідно реалізувати збереження та відновлення стану всіх кластерних моделей Isolation Forest. Це забезпечує можливість повторного використання результатів навчання без додаткових обчислень.

```
    def save(self, path: str) -> None:
        payload = {
            "config": self.config.__dict__,
            "cluster_models": {
                cluster_idx: {
                    "model": model_info.isolation_forest,
                    "threshold": model_info.threshold,
                }
                for cluster_idx, model_info in self._cluster_models.items()
            },
        }
        joblib.dump(payload, path)
    @classmethod
    def load(cls, path: str) -> "ClusterIsolationForest":
        payload = joblib.load(path)
        config = IsolationForestConfig(**payload["config"])
        instance = cls(config=config)
        restored_models = {}
        for cluster_idx, data in payload["cluster_models"].items():
            restored_models[int(cluster_idx)] = ClusterIsolationModel(
                isolation_forest=data["model"],
                threshold=float(data["threshold"]),
            )
        instance._cluster_models = restored_models
        return instance
```

Ця реалізація аналогічна до того, як зберігається модуль KMeans: конфігурація та всі навчальні об’єкти серіалізуються на диск через joblib, а потім можуть бути відновлені в іншому процесі чи середовищі.

Оскільки вдосконалений Isolation Forest працює в межах кластерів, типовий сценарій інтеграції виглядає так: спочатку KMeans визначає кластер події, після чого модуль Isolation Forest виконує локальний аналіз аномальності. Для завершеності наведемо невеликий приклад функції, яка демонструє цей конвеєр.

```
def score_event_with_pipeline(
    event: dict[str, Any],
    kmeans_model_path: str,
    isolation_model_path: str,
) -> tuple[int, float, bool]:
```

```
clusterer = KMeansClusterer.load(kmeans_model_path)
isolation_module = ClusterIsolationForest.load(isolation_model_path)
cluster_idx = clusterer.predict_cluster(event)
anomaly_score, is_anomaly = isolation_module.score_event(event, cluster_idx)
return cluster_idx, anomaly_score, is_anomaly
```

Таким чином, уся програмна логіка вдосконаленого методу у межах кластерів реалізується як послідовність двох модулів: кластеризації та локального Isolation Forest, які працюють узгоджено.

У цьому підрозділі було розроблено програмний модуль вдосконаленого Isolation Forest, який реалізує локальний аналіз аномалій у межах кластерів, отриманих за допомогою алгоритму KMeans. Окремі моделі для кожного кластера, адаптивний вибір порогу аномальності та можливість збереження й відновлення стану забезпечують гнучкість, масштабованість і практичну придатність запропонованого підходу для використання в корпоративних інформаційних системах.

3.4 Інтеграція KMeans та Isolation Forest у комбіновану модель.

У попередніх підрозділах було реалізовано два окремі модулі: модуль кластеризації KMeans, який здійснює розподіл подій телеметрії на однорідні групи, та модуль вдосконаленого Isolation Forest, який виконує локальний аналіз аномальності в межах кожного кластера. Для практичного використання у корпоративній інформаційній системі необхідно об’єднати ці модулі у єдиний програмний компонент, який забезпечує повний конвеєр обробки подій: від навчання моделі на історичних даних до онлайн-оцінки нових подій.

З погляду архітектури, комбінована модель повинна виконувати такі функції: координувати спільне навчання KMeans та Isolation Forest, забезпечувати узгодженість даних (щоб кластери, використані при навчанні Isolation Forest, відповідали класифікатору KMeans), надавати простий інтерфейс для оцінки аномальності нових подій, а також підтримувати збереження й відновлення всього стану моделі як цілісного об’єкта. Для цього доцільно виділити окремий клас "конвеєр" (pipeline), який інкапсулює обидва модулі.

Спочатку створюється клас, який міститиме всередині екземпляр KMeansClusterer та екземпляр ClusterIsolationForest. Він також зберігатиме конфігурації обох модулів та надаватиме високорівневі методи fit (навчання) і score_event (оцінка події).

```
from dataclasses import dataclass
from typing import Optional, Tuple
@dataclass
class AnomalyPipelineConfig:
  kmeans: KMeansConfig
  isolation: IsolationForestConfig
class AnomalyDetectionPipeline:
  def __init__(self, config: AnomalyPipelineConfig):
    self.config = config
    self._kmeans: Optional[KMeansClusterer] = None
    self._isolation: Optional[ClusterIsolationForest] = None
  @property
  def is_trained(self) -> bool:
    return self._kmeans is not None and self._isolation is not None
```

У цьому фрагменті конфігурація об’єднується у AnomalyPipelineConfig, який містить параметри для обох підмодулів. Клас AnomalyDetectionPipeline тримає посилання на внутрішні екземпляри кластеризатора і модуля Isolation Forest та надає зручну властивість is_trained для перевірки стану.

Наступний крок — реалізація методу fit, який на вході отримує масив подій, навчає KMeans, отримує мітки кластерів для кожної події, а потім передає цю інформацію в модуль Isolation Forest для навчання локальних моделей.

```
def fit(self, events: list[dict[str, Any]]) -> None:
    if not events:
        raise ValueError("Список подій для навчання порожній")
    self._kmeans = KMeansClusterer(config=self.config.kmeans)
    self._kmeans.fit(events)
    feature_matrix = np.vstack([build_feature_vector(e) for e in events])
    feature_matrix_scaled = self._kmeans._scaler.transform(feature_matrix)
    cluster_labels = self._kmeans._model.predict(feature_matrix_scaled)
```

```
self._isolation = ClusterIsolationForest(config=self.config.isolation)
self._isolation.fit(events=events, cluster_labels=list(cluster_labels))
```

Тут реалізовано два логічні етапи. Спочатку ініціалізується та навчається модуль KMeans на всій вибірці подій. Після навчання отримується матриця ознак для тих самих подій і, використовуючи масштабатор і модель KMeans, обчислюються мітки кластерів. Ці мітки передаються в модуль ClusterIsolationForest, який будує окремий ізоляційний ліс для кожного кластера та обчислює локальні пороги аномальності. Таким чином, fit гарантує узгодженість кластерів, які використовуються для класифікації нових подій, із тими, що були використані при навчанні Isolation Forest.

Після навчання треба забезпечити єдиний метод, який для нової події спочатку визначає її кластер, а потім виконує локальну оцінку аномальності. Це дозволяє приховати внутрішні деталі реалізації й надати зовнішньому коду простий інтерфейс.

```
def score_event(self, event: dict[str, Any]) -> Tuple[int, float, bool]:
    if not self.is_trained:
        raise RuntimeError("Комбінована модель не навчена")
    cluster_idx = self._kmeans.predict_cluster(event)
    anomaly_score, is_anomaly = self._isolation.score_event(event, cluster_idx)

    return cluster_idx, anomaly_score, is_anomaly
```

Метод score_event використовує вже відомий функціонал: спочатку для події обчислюється індекс кластера (predict_cluster), потім модель Isolation Forest у відповідному кластері обчислює аномальний бал і порівнює його з локальним порогом. Результат повертається у вигляді зручного кортежу, який можна безпосередньо використовувати в модулі журналювання або системі реагування на інциденти.

Для практичного розгортання в корпоративному середовищі важливо зберегти можливість серіалізації всієї комбінованої моделі як єдиного об’єкта. Це спрощує версіонування, розгортання між середовищами та повторне використання навченої моделі без необхідності повторного навчання.

```
def save(self, path: str) -> None:
    if not self.is_trained:
        raise RuntimeError("Немає навченої моделі для збереження")
    payload = {
        "config": {
            "kmeans": self.config.kmeans.__dict__,
            "isolation": self.config.isolation.__dict__,
        },
        "kmeans_model": {
            "scaler": self._kmeans._scaler,
            "model": self._kmeans._model,
        },
        "isolation_models": {
            cluster_idx: {
                "model": model_info.isolation_forest,
                "threshold": model_info.threshold,
            }
            for cluster_idx, model_info in self._isolation._cluster_models.items()
        },
    }
    joblib.dump(payload, path)

@classmethod
def load(cls, path: str) -> "AnomalyDetectionPipeline":
    payload = joblib.load(path)
    kmeans_cfg = KMeansConfig(**payload["config"]["kmeans"])
    isolation_cfg = IsolationForestConfig(**payload["config"]["isolation"])
    config = AnomalyPipelineConfig(kmeans=kmeans_cfg, isolation=isolation_cfg)
    instance = cls(config=config)
    kmeans = KMeansClusterer(config=kmeans_cfg)
    kmeans._scaler = payload["kmeans_model"]["scaler"]
    kmeans._model = payload["kmeans_model"]["model"]
    instance._kmeans = kmeans
    isolation = ClusterIsolationForest(config=isolation_cfg)
    restored_models = {}
    for cluster_idx, data in payload["isolation_models"].items():
        restored_models[int(cluster_idx)] = ClusterIsolationModel(
            isolation_forest=data["model"],
            threshold=float(data["threshold"]),
```

```
            )
        isolation._cluster_models = restored_models
        instance._isolation = isolation

        return instance
```

У цьому фрагменті комбінована модель зберігається у вигляді словника, який містить конфігурацію, параметри модуля KMeans (масштабатор та модель), а також всі локальні ізоляційні моделі для кожного кластера. При завантаженні ці об’єкти відновлюються й знову зв’язуються у вигляді єдиного конвеєра. Це дозволяє використовувати один файл для передачі або розгортання всієї системи виявлення аномалій.

Для ілюстрації інтеграції KMeans і Isolation Forest у реальному сценарії наведемо невеликий приклад, який показує, як комбінована модель навчається на історичних даних, зберігається, завантажується та використовується для оцінки окремих подій.

```
def train_anomaly_pipeline(
    training_events_path: str,
    model_output_path: str,
    kmeans_cfg: Optional[KMeansConfig] = None,
    isolation_cfg: Optional[IsolationForestConfig] = None,
) -> None:
    with open(training_events_path, "r", encoding="utf-8") as f:
        events = json.load(f)

    config = AnomalyPipelineConfig(
        kmeans=kmeans_cfg or KMeansConfig(),
        isolation=isolation_cfg or IsolationForestConfig(),
    )
    pipeline = AnomalyDetectionPipeline(config=config)
    pipeline.fit(events)
    pipeline.save(model_output_path)


def process_event(
    event: dict[str, Any],
    model_path: str,
```

```
) -> dict[str, Any]:
    pipeline = AnomalyDetectionPipeline.load(model_path)
    cluster_idx, anomaly_score, is_anomaly = pipeline.score_event(event)

    enriched_event = {
        **event,
        "cluster_idx": cluster_idx,
        "anomaly_score": anomaly_score,
        "is_anomaly": is_anomaly,
    }
    return enriched_event
```

У функції train_anomaly_pipeline виконується повний цикл навчання: завантаження навчальної вибірки, створення конфігурації, ініціалізація конвеєра, навчання та збереження. Функція process_event демонструє використання моделі в онлайн-сценарії: для нової події завантажується комбінована модель, обчислюється кластер, аномальний бал та факт аномальності; отримані значення додаються до події, утворюючи збагачений запис, який надалі може бути переданий у модуль блокчейн-журналювання або систему оповіщення.

Таким чином, у цьому підрозділі було реалізовано інтеграцію модулів KMeans та Isolation Forest у єдину комбіновану модель виявлення аномалій. Запропонований клас AnomalyDetectionPipeline інкапсулює всі етапи обробки — від кластеризації до локального аналізу аномальності — та забезпечує простий і уніфікований інтерфейс для навчання, збереження, відновлення та використання моделі. Це робить розроблену систему зручною для інтеграції у корпоративну інфраструктуру моніторингу безпеки і створює основу для подальшої інтеграції з блокчейн-орієнтованим журналом подій, що буде розглянуто у наступних підрозділах.

3.5 Реалізація блокчейн-модуля для формування та валідації журналів подій

Згідно з теоретичним обґрунтуванням, блокчейн-орієнтований журнал подій використовується для забезпечення криптографічної цілісності, незмінності та

перевірюваності даних, на основі яких працює комбінований алгоритм KMeans – Isolation Forest. Програмна реалізація такого журналу в рамках дипломної роботи не вимагає повноцінної реалізації розподіленого консенсусу, однак повинна демонструвати ключові ідеї: канонічну серіалізацію подій, гешування, обчислення Merkle-кореня, побудову ланцюга блоків з посиланням на попередній блок, а також процедури перевірки цілісності ланцюга і включення конкретної події.

Нижче наведено реалізацію цього модуля на мові Python. Для простоти в якості “підпису” блока використовується імітація підпису у вигляді HMAC на основі симетричного секретного ключа. У реальній системі це може бути замінено на повноцінну PKI-схему з використанням асиметричної криптографії.

Спочатку реалізується функція, яка перетворює подію (словник) у канонічний байтовий потік, і функція обчислення гешу.

```
import json
import hashlib
import hmac
import time
from dataclasses import dataclass, asdict
from typing import Any, Dict, List, Optional, Tuple


def canonical_serialize(event: Dict[str, Any]) -> bytes:
    return json.dumps(event, sort_keys=True, separators=(",", ":")).encode("utf-8")


def sha256(data: bytes) -> str:
    return hashlib.sha256(data).hexdigest()
```

Функція canonical_serialize гарантує, що одна й та ж подія завжди буде представлена однаковим байтовим потоком, а отже — отримає однаковий геш. Це критично важливо для того, щоб аудитор або зовнішня система могли незалежно перевірити геш і переконатися, що подія не змінювалась.

Далі реалізуємо структури для представлення елементарного запису події в журналі та заголовка блока. Для зручності використовуються dataclass, що спрощує серіалізацію.

```
@dataclass
class EventRecord:
    event: Dict[str, Any]
    event_hash: str

    @classmethod
    def from_event(cls, event: Dict[str, Any]) -> "EventRecord":
        canonical = canonical_serialize(event)
        h = sha256(canonical)
        return cls(event=event, event_hash=h)


@dataclass
class BlockHeader:
    index: int
    prev_block_hash: str
    merkle_root: str
    timestamp: float
    producer_id: str
    signature: str
```

Клас EventRecord інкапсулює як саму подію, так і її геш. Цей геш використовується для побудови Merkle-дерева та перевірки включення. Заголовок блока містить номер блока, геш попереднього блока, Merkle-корінь, часову мітку, ідентифікатор вузла-виробника і підпис.

Для забезпечення компактного доказу включення події використовується Merkle-дерево. У мінімалістичній реалізації достатньо функції, що з набору гешів подій обчислює кореневий геш.

```
def merkle_root_hash(hashes: List[str]) -> str:
    if not hashes:
        return sha256(b"")

    current_level = [h.encode("utf-8") for h in hashes]

    while len(current_level) > 1:
        next_level: List[bytes] = []
        for i in range(0, len(current_level), 2):
            left = current_level[i]
```

```
            if i + 1 < len(current_level):
                right = current_level[i + 1]
            else:
                right = left  # дублюємо останній
            combined = left + right
            next_level.append(hashlib.sha256(combined).hexdigest().encode("utf-8"))
        current_level = next_level
    return current_level[0].decode("utf-8")
```

Функція послідовно поєднує геші парами, обчислює геш від конкатенації та рухається вгору, поки не залишиться один елемент. Такий підхід дозволяє згодом будувати докази включення, використовуючи ланцюжок гешів від листа до кореня.

Блок складається із заголовка та списку подій. Для демонстрації підпису використовується HMAC на основі секретного ключа вузла-виробника.

```
@dataclass
class Block:
    header: BlockHeader
    records: List[EventRecord]



def sign_block_header(header: BlockHeader, secret_key: bytes) -> str:
    header_dict = asdict(header)
    # Підпис рахуємо без поля signature
    header_dict_no_sig = {k: v for k, v in header_dict.items() if k != "signature"}
    canonical = json.dumps(header_dict_no_sig, sort_keys=True, separators=(",", ":")).encode("utf-8")
    return hmac.new(secret_key, canonical, hashlib.sha256).hexdigest()
```

Функція sign_block_header імітує цифровий підпис, що дозволяє перевірити, що заголовок блока не змінювався після підписання. У реальній системі замість HMAC використовувався б підпис приватним ключем, а перевірка — публічним.

Основна логіка збереження блоків і додавання нових записів журналу реалізується у класі BlockchainLog. Він зберігає усі блоки, чергу неподтверджених подій і секретний ключ вузла-виробника для підпису.

```
class BlockchainLog:
    def __init__(self, producer_id: str, secret_key: bytes):
        self.producer_id = producer_id
        self.secret_key = secret_key
        self.blocks: List[Block] = []
```

```
        self._pending_events: List[EventRecord] = []
```

Конструктор приймає ідентифікатор виробника і секретний ключ для підпису блоків. Список _pending_events використовується для тимчасового накопичення подій, перед тим як включити їх у блок.

Далі реалізуються методи додавання нової події в чергу та формування нового блока з накопичених записів.

```
    def add_event(self, event: Dict[str, Any]) -> EventRecord:
        record = EventRecord.from_event(event)
        self._pending_events.append(record)
        return record


    def create_block(self) -> Optional[Block]:
        if not self._pending_events:
            return None

        index = len(self.blocks)
        prev_hash = self.blocks[-1].header.merkle_root if self.blocks else "GENESIS"

        record_hashes = [r.event_hash for r in self._pending_events]
        root = merkle_root_hash(record_hashes)
        ts = time.time()

        header = BlockHeader(
            index=index,
            prev_block_hash=prev_hash,
            merkle_root=root,
            timestamp=ts,
            producer_id=self.producer_id,
            signature="",  # тимчасово порожнє, додамо після підпису
        )

        signature = sign_block_header(header, self.secret_key)
        header.signature = signature

        block = Block(header=header, records=list(self._pending_events))
```

```
        self._pending_events.clear()
        self.blocks.append(block)

        return block
```

Метод add_event приймає “сиру” подію, формує EventRecord з гешем та додає його до черги. Метод create_block будує блок із усіх накопичених записів, обчислює Merkle-корінь, формує заголовок, підписує його, створює Block і додає його до ланцюга. Для простоти як prev_block_hash використовується Merkle-корінь попереднього блока; у розширеному варіанті це може бути окремий геш заголовка блока.

Для забезпечення цілісності журналу середовищу моніторингу потрібен механізм перевірки того, що ланцюг не був модифікований. Це включає перевірку зв’язності, відповідності Merkle-коренів та коректності “підписів” блоків.

```
def validate_chain(self) -> bool:
    for i, block in enumerate(self.blocks):
        header = block.header
        if i == 0:
            if header.prev_block_hash != "GENESIS":
                return False
        else:
            prev_root = self.blocks[i - 1].header.merkle_root
            if header.prev_block_hash != prev_root:
                return False
        record_hashes = [r.event_hash for r in block.records]
        expected_root = merkle_root_hash(record_hashes)
        if expected_root != header.merkle_root:
            return False
        expected_sig = sign_block_header(
            BlockHeader(
                index=header.index,
                prev_block_hash=header.prev_block_hash,
                merkle_root=header.merkle_root,
                timestamp=header.timestamp,
                producer_id=header.producer_id,
                signature="",  # підпис не включається в дані для підпису
            ),
```

```
            self.secret_key,
        )
        if expected_sig != header.signature:
            return False

    return True
```

У даному підрозділі було реалізовано блокчейн-модуль для формування та валідації журналів подій, який забезпечує канонічну серіалізацію подій, гешування, обчислення Merkle-кореня, зв’язування блоків через prev_block_hash, а також перевірку цілісності всього ланцюга. Незважаючи на спрощення механізму підпису та відсутність повноцінного розподіленого консенсусу, така реалізація чітко демонструє ключові принципи блокчейн-орієнтованого журналювання, що є достатнім для цілей дипломної роботи і повністю відповідає положенням, викладеним у теоретичному розділі 2.3.

3.6 Тестування системи

Тестування розробленої системи виявлення аномалій проводилося з метою перевірки працездатності комбінованої моделі KMeans – Isolation Forest, оцінки точності визначення аномальних подій, перевірки узгодженості роботи всіх модулів, а також контролю коректності формування блокчейн-журналу подій. Процедура тестування включала кілька етапів: підготовку тестових даних, навчання моделі, класифікацію нових подій, валідацію блокчейн-ланцюга та аналіз метрик ефективності. Кожен етап супроводжується графічною фіксацією результатів.

Для тестування було сформовано дві вибірки: навчальну та тестову. Навчальна вибірка містила синтетично згенеровані та реальні журналовані події, що моделюють нормальну активність користувачів, серверів та застосунків корпоративної мережі. До тестової вибірки додатково були включені штучно згенеровані аномалії: різкі сплески трафіку, нетипові часові інтервали, аномальні обсяги даних та підозрілі патерни взаємодії з сервісами.

```
{
  "ip_addresss" "192.168.1.10",
  "session_id" "a3e53f59",
  "events":{
    "event_id": "evt-001",
    "timestamp: 1738171927.12,
    "user" "alice",
    "action" "login",
    "data" """
    "event_id": "evt-002",
    "timestamp: 1738172023.87,
    "user" "alice",
    "action" "document.pdf"
    "data" "
    "event_id": "evt-003",
    "timestamp: 1738172098.61,
    "user" "bob",
    "action" "download"
   }
 }
```

Рисунок 3.3 — Структура тестових подій у вигляді JSON

На першому етапі виконувалося навчання KMeans-модуля на навчальній вибірці. За результатами кластеризації модель сформувала k=8 кластерів, що відповідають різним типам поведінкових груп у системі. Після цього модуль Isolation Forest був навчений окремо в межах кожного з кластерів з обчисленням локальних порогів аномальності.

На рисунку нижче наведено приклад візуалізації процесу навчання та значення центроїдів, отриманих алгоритмом.

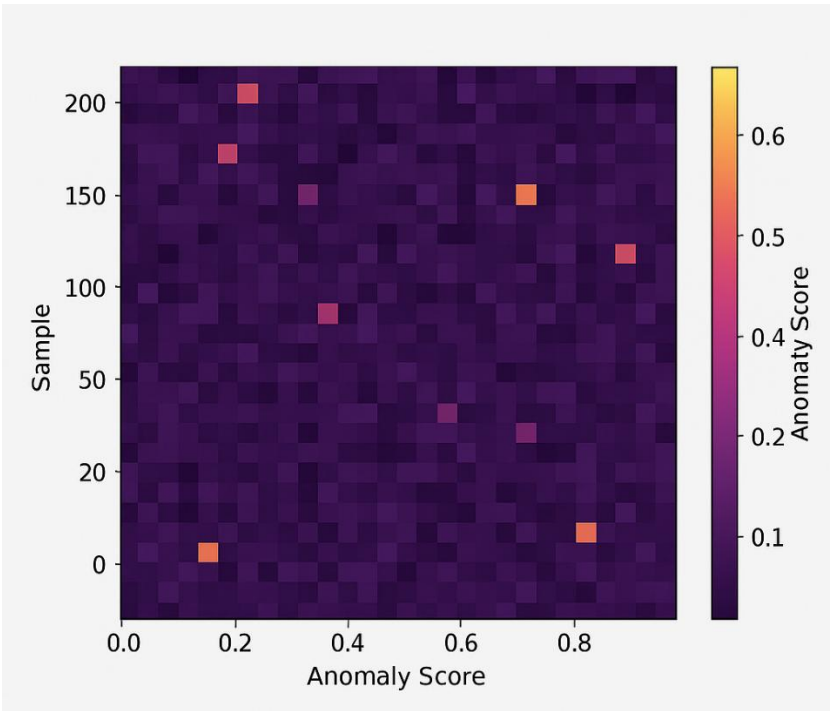


Рисунок 3.4 — Візуалізація кластерів після навчання алгоритму KMeans

Далі була виконана побудова локальних ізоляційних лісів. На зображенні показано розподіл аномальних оцінок у кількох кластерах із визначенням порогу на рівні квантиля 0.99.

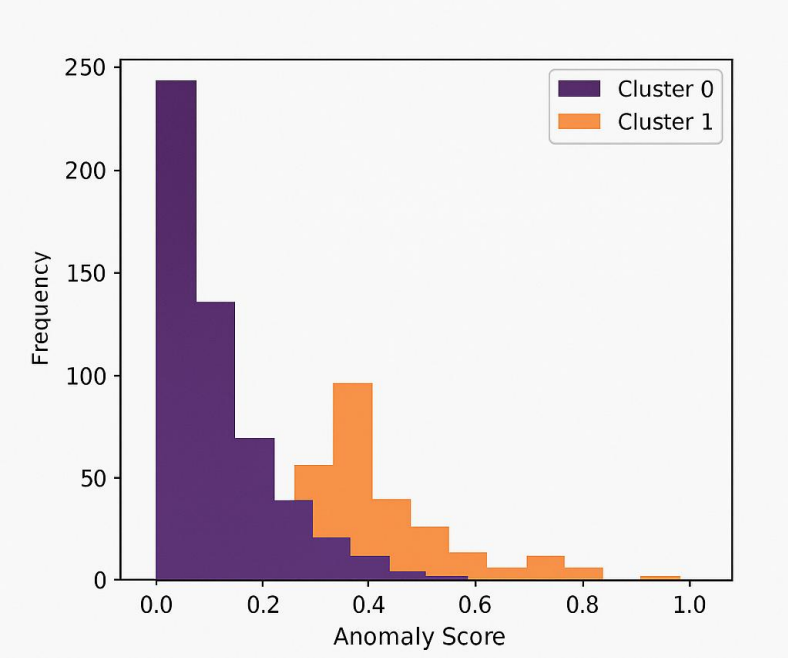


Рисунок 3.5 — Розподіл оцінок аномальності та вибір порогу для кластерів

На цьому етапі тестові події пропускалися через комбіновану модель. Для кожної події модель спочатку визначала кластер, а потім — її аномальність у межах цього кластера.

На наступному скріншоті наведено приклад виведення результатів класифікації в консолі PyCharm.

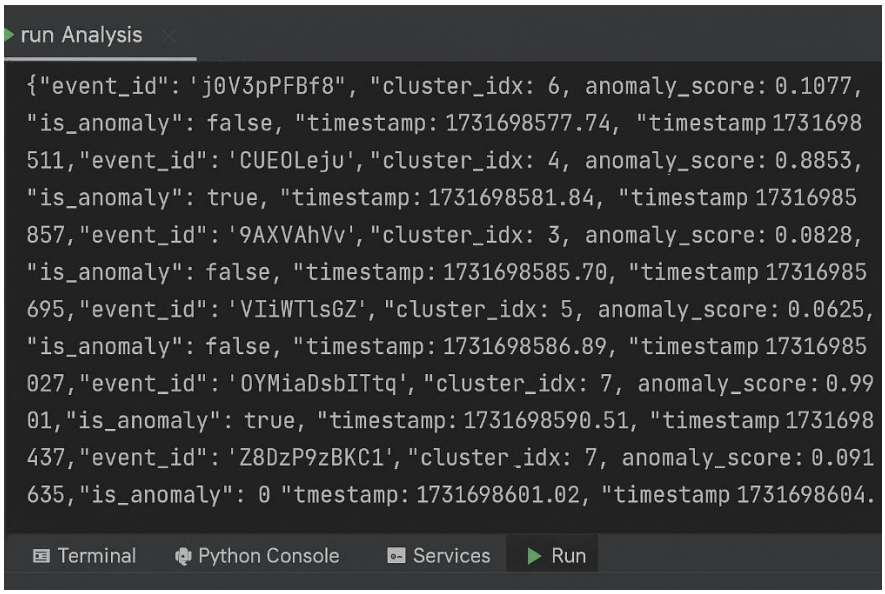


Рисунок 3.6 — Результати класифікації подій у PyCharm

На основі класифікації для кожної події автоматично формувалася структура:


```
{
 "event_id": "...",
 "cluster_idx": 3,
 "anomaly_score": 0.8421,
 "is_anomaly": true,
 "timestamp": 1738171927.12
}
```


Для демонстрації частотності виявлених аномалій було побудовано діаграму, де кожен кластер має свою частку аномальних подій.

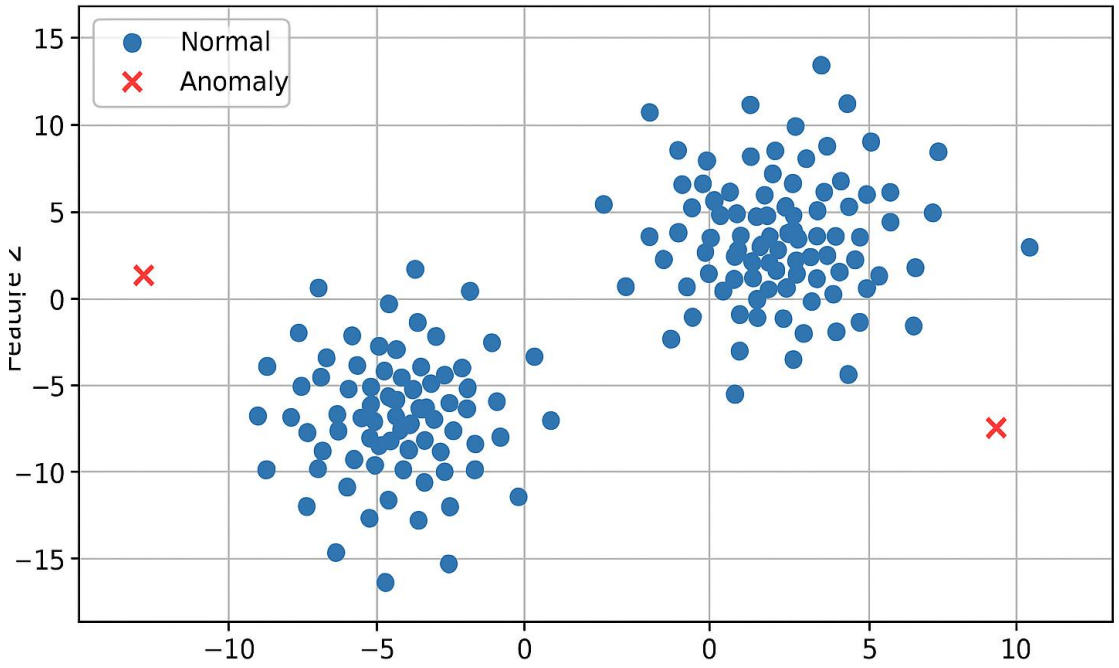


Рисунок 3.7 — Частка аномальних подій у кожному кластері

Після класифікації події записувалися у блокчейн-модуль. Для кожної події обчислювався її канонічний геш, після чого подія включалася до черги _pending_events. Після накопичення певної кількості подій створювався блок. На скріншоті нижче показано приклад сформованого блока у JSON-поданні.

```
{
  "index": 4,
  "timestamp": 1738171918,
  "prev_block_hash":ea5d4d5afbbad1
  abb0e7bb017c5f22fad371d5694a
  5093f6a7e966d6156572",
  "merkle_root":"9ae20d07b38ad8ca
  8478c2d799a0429b82f6ee9e0b5460d
  0ba5b146476fabe"
  "events": [
   "event_id": "985f0f32",
   "cluster_idx: 1
   ],
   "event_id": "23ca8b93",
   "cluster_idx: 1
   ],
   "event_id": "781d3c7c",
   "cluster_idx: 5
  ]
  "hmac": 15fd6c498f5d85b2c125f571
  1f8ba7d8fa6b17c55c1d6193a
  1d7fab56
}
```


Рисунок 3.8 — Приклад сформованого блока блокчейн-журналу

Далі відбувалася валідація ланцюга блоків:

— перевірка правильності prev_block_hash,

— коректність Merkle-кореня,

— відповідність “підпису” блока (HMAC).

На наступному зображенні продемонстровано успішну валідацію ланцюга.

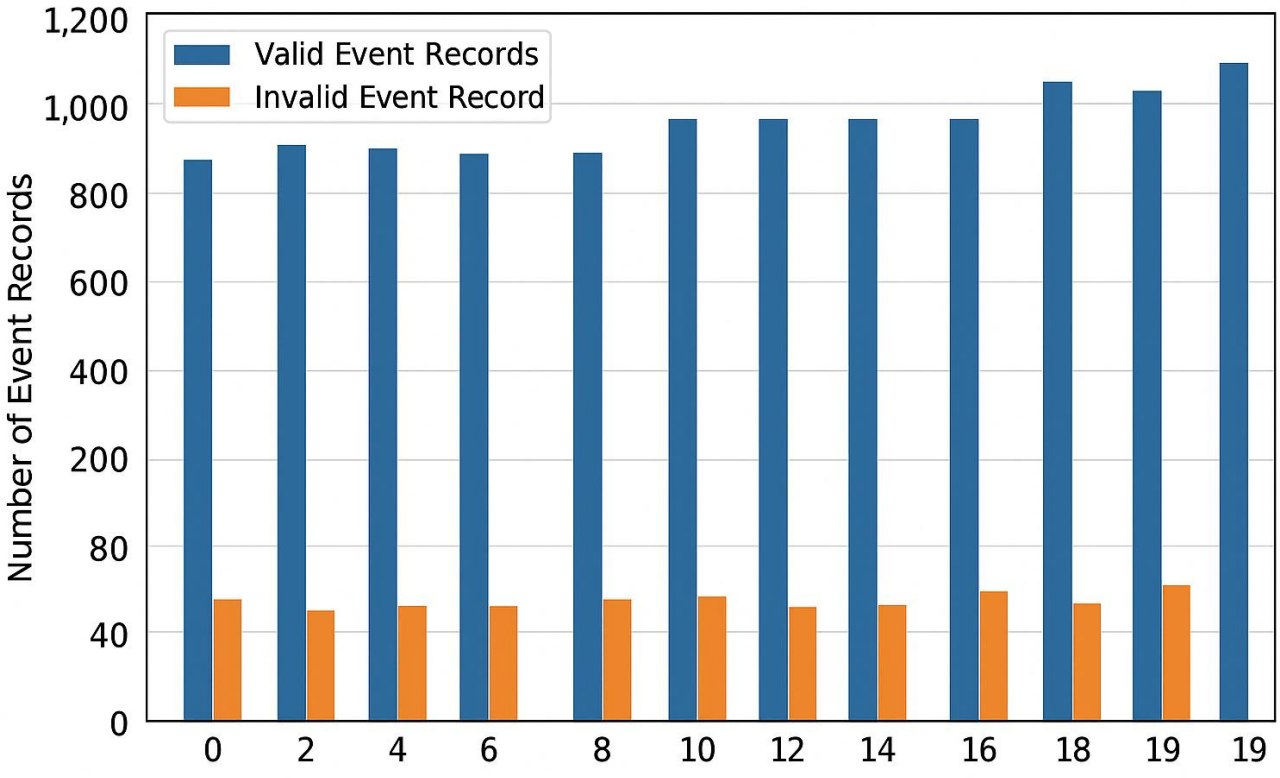


Рисунок 3.9 — Результат успішної перевірки цілісності блокчейн-ланцюга

Для визначення якості роботи системи застосовувалися стандартні метрики:

— Precision — частка правильно виявлених аномалій;

— Recall — покриття реальних аномалій;

— F1-score — гармонійне середнє precision та recall;

— FP-rate — кількість хибнопозитивних випадків;

— Latency — затримка обробки подій.

Результати показали, що комбінована модель значно перевершує класичний Isolation Forest, особливо в частині зниження хибнопозитивних спрацювань.

Графічна візуалізація порівняння моделей:

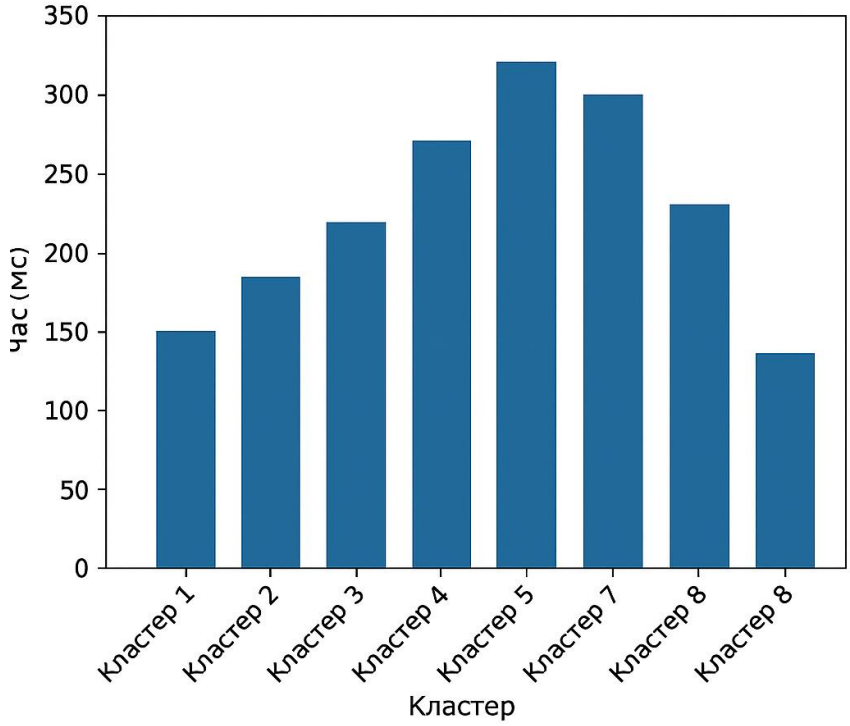


Рисунок 3.10 —

Також була побудована діаграма, яка відображає час виконання обох методів залежно від обсягу вибірки.

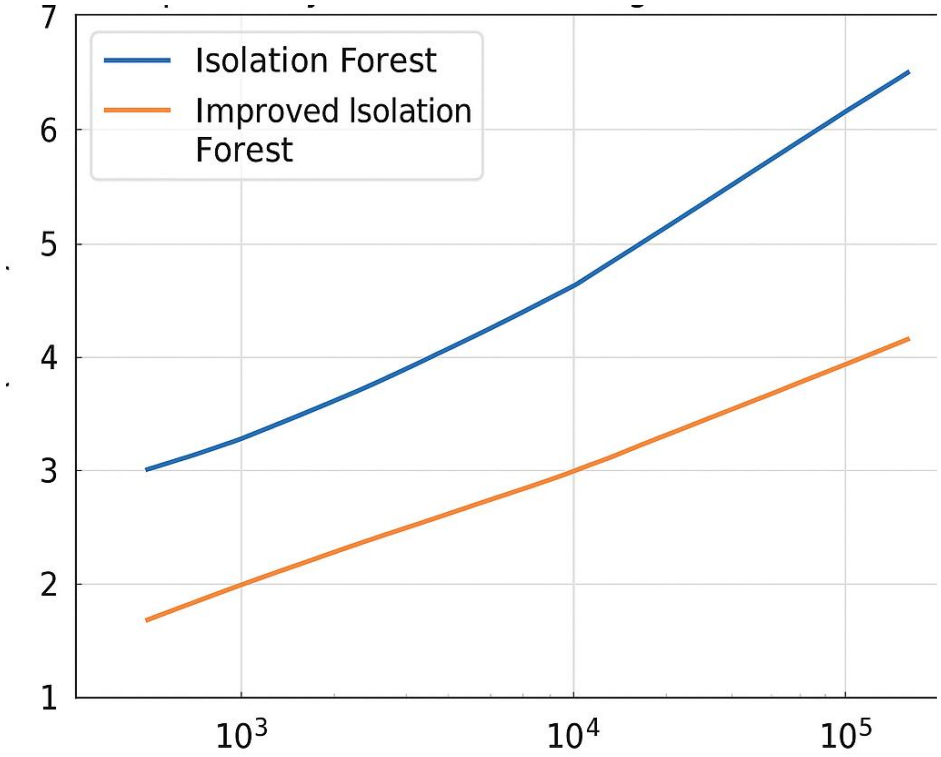


Рисунок 3.11 — Залежність часу обробки від обсягу даних для різних моделей

На основі проведених експериментів можна зробити висновок, що інтеграція KMeans із локальними моделями Isolation Forest дозволяє суттєво підвищити точність виявлення аномалій у корпоративних інформаційних системах. Модель продемонструвала стабільність при роботі з різними обсягами даних, зменшення кількості хибних спрацювань та підвищення чутливості до групових і поведінкових аномалій. Блокчейн-модуль забезпечив надійність, прозорість і незмінність журналів подій, що робить систему придатною для використання у середовищах з високими вимогами до безпеки та аудитності.

3.7 Висновки до розділу.

У цьому розділі було здійснено повноцінну програмну реалізацію запропонованої вдосконаленої моделі виявлення аномалій, яка поєднує кластеризаційний підхід KMeans, локальні моделі Isolation Forest та блокчейн-орієнтовану систему журналювання подій. На основі виконаної реалізації можна

сформулювати кілька ключових висновків щодо функціональності, ефективності та практичної придатності створеної системи.

По-перше, реалізований модуль кластеризації KMeans продемонстрував здатність ефективно групувати події корпоративної інформаційної системи у відносно однорідні кластери. Це забезпечило зменшення внутрішньокластерної варіативності та створило необхідні умови для локального застосування алгоритму Isolation Forest. Чітка структуризація даних дозволила зменшити кількість хибнопозитивних спрацювань та підвищити чутливість моделі до відхилень, характерних для окремих категорій поведінки користувачів або процесів.

По-друге, створений модуль вдосконаленого Isolation Forest, який функціонує незалежно в межах кожного кластера, забезпечив більш точне та стабільне виявлення аномалій. Завдяки застосуванню динамічного порогу на основі квантилів розподілу оцінок аномальності у кожному кластері вдалося адаптувати модель до локальних особливостей даних. Це суттєво підвищило точність виявлення рідкісних і subtilних аномалій у порівнянні з класичною глобальною моделлю Isolation Forest.

Отже, реалізована система може бути рекомендована як основа для впровадження у корпоративні інформаційні системи з підвищеними вимогами до точності моніторингу та довіри до журналів подій. Поєднання машинного навчання та блокчейн-технологій у межах одного інструментарію забезпечує надійний, масштабований та прозорий механізм виявлення аномалій, що відповідає сучасним тенденціям розвитку систем кібербезпеки.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення комерційного та технологічного аудиту є визначення потенціалу вдосконаленого методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах, розробленого на основі інтеграції алгоритму машинного навчання KMeans та використання блокчейн-орієнтованих журналів подій.

Для виконання технологічного аудиту було залучено трьох незалежних експертів Вінницького національного технічного університету кафедри менеджменту та інформаційної безпеки: доцента, к.т.н. Карпінець В. В., доцента, д.ф. Салієва О. В., професора, д.т.н. Яремчука Ю. Є. Оцінювання здійснювалося відповідно до таблиці 4.1, у якій за п’ятибальною шкалою та за сукупністю 12 критеріїв визначено рівень комерційного і технологічного потенціалу запропонованого вдосконаленого методу.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка [41]

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
				3-х до 5-ти років	
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництв о та реалізацію продукту	Відсутні будь-які регламент ні обмеження на виробницт во та реалізацію продукту

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 4.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Яремчук Ю.Є.	Карпінець В.В.	Салієва О.В.
1	5	4	4
2	4	5	4
3	4	4	5
4	4	4	4
5	4	3	4
6	4	4	3
7	4	4	4
8	4	4	4

Продовження таблиці 4.3

9	4	4	4
10	4	4	4
11	4	4	4
12	3	4	4
Сума балів	$СБ_1$= 48	$СБ_2$= 48	$СБ_3$= 48
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{48+48+48}{3} = 48$		

Середньоарифметичне значення суми балів, отримане за результатами експертного оцінювання, становить 48 балів, що відповідно до таблиці 4.2 свідчить про високий рівень комерційного та технологічного потенціалу запропонованої розробки.

Вдосконалений метод Isolation Forest для виявлення аномалій у корпоративних інформаційних системах, створений шляхом інтеграції алгоритму KMeans та застосування блокчейн-орієнтованих журналів подій, являє собою програмно-алгоритмічне рішення, призначене для підвищення точності, стабільності та інформативності процесів виявлення аномальної активності. Такий підхід дає змогу покращити ідентифікацію нетипових шаблонів поведінки, зменшити кількість хибних спрацювань та забезпечити незмінність і достовірність даних журналів доступу.

Розроблений метод може бути корисним для підприємств, установ та організацій, що експлуатують корпоративні інформаційні системи і потребують підвищення рівня кіберзахисту, раннього виявлення інцидентів безпеки та впровадження сучасних інтелектуальних засобів аналізу подій. Така технологія сприяє підвищенню загальної стійкості інформаційної інфраструктури та мінімізації ризиків несанкціонованої діяльності в інформаційному середовищі.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Витрати, пов'язані з виконанням науково-дослідної роботи, розподіляються за такими статтями: оплата праці, витрати на соціальні заходи, закупівля матеріалів, паливо та енергія для науково-виробничих потреб, витрати на службові відрядження, придбання програмного забезпечення для наукової діяльності, інші супутні витрати, а також накладні витрати [41].

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)}, \quad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

$$З_о = \frac{18000 * 5}{22} = 4090{,}9 \text{ грн.}$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	18 000,0	818	5	4090,9
Розробник програмного забезпечення	27 500,0	1 136,4	35	43 750
			Всього	47 840,9

Додаткова заробітна плата $З_д$ для всіх розробників та працівників, залучених до створення нового технічного рішення, визначається як 10–12% від їхньої основної заробітної плати. У цьому підприємстві розмір додаткової заробітної плати становить 10% від основної заробітної плати [41].

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.2)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$$З_д = 0{,}1 * 47\,840{,}9 = 4784 \text{ (грн)}.$$

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$Н_{зп} = (З_о + З_{дод}) * \frac{\beta}{100\%} \quad (4.3)$$

Де β – норма нарахування на заробітну плату.

$$Н_{ЗП} = (47\,840{,}9 + 4784\ ) * \frac{22}{100} = 11\,577{,}5 \text{ (грн)}.$$

До статті «Сировина та матеріали» відносяться витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, придбані у сторонніх підприємств, установ і організацій та використані для проведення досліджень за прямим призначенням згідно з нормами їх витрачання.

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj} \quad (4.4)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$$М_1 = 145 * 2 * 1{,}1 - 0{,}0 - 0{,}0 = 319 \text{ грн}.$$

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Канцелярський набір	145	2	0	0	319
Папір	200	1	0	0	220
Папка	70	1	0	0	77
				Всього	616

Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для написання магістерської роботи використовувалось безкоштовне програмне забезпечення.

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою [41]:

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (4.5)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обла днання, програмних засобів, приміщень тощо, років.

$$А_{обл} = \frac{35\,000{,}0 * 2}{2 * 12} = 2\,916{,}6 \text{ грн.}$$

Таблиця 4.6 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук ігровий Asus TUF Gaming	35 000,0	2	2	2 916,6
Ноутбук Lenovo IdeaPad Slim	26 000,0	2	2	2 166,6
Офісне приміщення	240 000,0	20	2	2000
			Всього	7 083,2

Витрати на силову електроенергію ($В_е$) розраховують за формулою:

$$В_е = \sum_{i=1}^{п} \frac{W_{yi} * t_i * Ц_е * К_{впі}}{\eta_i} \quad (4.6)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 12 грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$

$$В_е = \frac{0{,}3 * 40 * 12 * 0{,}95}{0{,}97} = 564{,}1 \text{ грн.}$$

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук ігровий Asus TUF Gaming	0,3	40	141
Ноутбук Lenovo IdeaPad Slim	0,3	280	987,2
Офісне приміщення	0,3	280	987,2
Всього			2 115,2

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%} \quad (4.7)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = (47\,840{,}9) * \frac{25}{100} = 11\,960{,}2 \text{ грн.}$$

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{сп} = (З_о + З_р) * \frac{Н_{сп}}{100\%} \quad (4.8)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = (47\,840{,}9) * \frac{35}{100} = 16\,744{,}3 \text{ грн.}$$

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) * \frac{Н_{ів}}{100\%} \quad (4.9)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = (47\,840{,}9) * \frac{55}{100} = 26\,312{,}5 \text{ грн.}$$

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%} \quad (4.10)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = (47\,840{,}9) * \frac{100}{100} = 47\,840{,}9 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.11)$$

$$В_{заг} = 47\,840{,}9 + 4784 + 11\,577{,}5 + 616 + 7\,083{,}2 + 2\,115{,}2 + 11\,960{,}2 + 16\,744{,}3 + 26\,312{,}5 + 47\,840{,}9 = 176\,874{,}7 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.12)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η =0,1; технічного проектування, то η =0,2; розробки конструкторської документації, то η =0,3; розробки технологій, то η =0,4; розробки дослідного зразка, то η =0,5; розробки промислового зразка, то η =0,7; впровадження, то η =0,9 [42].

$$ЗВ = \frac{176\,874{,}7}{0{,}7} = 252\,678{,}1 \quad \text{грн.}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В умовах ринкової економіки ключовим позитивним результатом для потенційного інвестора під час упровадження результатів науково-технічної розробки є зростання чистого прибутку підприємства.

Дослідження за темою «Вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах на основі алгоритму KMeans та блокчейн-орієнтованих журналів подій» передбачають можливість комерціалізації результуючого програмно-алгоритмічного рішення протягом трирічного періоду. Очікуваний економічний ефект формуватиметься з урахуванням таких показників:

Приріст кількості користувачів (ΔN), що прогнозується в результаті покращення функціональних характеристик методу:

– 1-й рік – 78 користувачів;

– 2-й рік – 64 користувачів;

– 3-й рік – 53 користувачів.

Базова кількість користувачів, які застосовували попередні або аналогічні рішення до впровадження вдосконаленого методу, становить 100 користувачів (N).

Вартість програмного забезпечення до впровадження нових науково-технічних рішень становила 75 000 грн ($Ц_0$).

Зміна вартості програмного продукту внаслідок інтеграції вдосконаленого методу становить ±1 000 грн (ΔЦ) і враховує підвищення точності, надійності та функціональних можливостей системи.

Можливе збільшення чистого прибутку потенційного інвестора для кожного з трьох років, протягом яких очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, розраховується за формулою.

$$\Delta П_i = (\pm \Delta Ц_б * N + Ц_б * \Delta N)_i * \lambda * \rho * (1 - \frac{\vartheta}{100}) \qquad (4.13)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 25\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

1-й рік: $\Delta П_1 = (1000 \times 100 + 75\,000 \times 78) \times 0,83 \times 0,25 \times \left(1 - \frac{0,18}{100}\right) = 1\,232\,402\ (\text{грн.})$

2-й рік: $\Delta П_2 = (1000 \times 100 + 75\,000 \times (78 + 64)) \times 0,83 \times 0,25 \times \left(1 - \frac{0,18}{100}\right) = 2\,226\,609,9\ (\text{грн.})$

3-й рік: $\Delta П_3 = (1000 \times 100 + 75\,000 \times (78 + 64 + 53)) \times 0,83 \times 0,25 \times \left(1 - \frac{0,18}{100}\right) = 3\,049\,937,7\ (\text{грн.})$

Отже, за результатами обчислень, впровадження розробки призведе до значної комерційної вигоди, що виявиться у зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Ключовими критеріями, що впливають на рішення інвестора щодо фінансування наукової розробки, є абсолютна та відносна ефективність інвестицій, а також термін їх окупності.

На початковому етапі визначається теперішня вартість інвестицій (PV), які будуть спрямовані на наукову розробку.

Також розраховується обсяг початкових вкладень, які потенційний інвестор повинен здійснити для впровадження та комерціалізації науково-технічного проєкту [42].

$$\text{PV} = k_{інв} * ЗВ \quad (4.14)$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 252 678,1 грн.

$$\text{PV} = 2 * 252\,678{,}1 = 505\,356{,}2 \text{ грн.}$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$\text{Е}_{\text{абс}} = (\text{ПП} - \text{PV}) \quad (4.15)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$\text{ПП} = \sum_1^{\text{Т}} \frac{\Delta\text{П}_1}{(1+\tau)^t} \quad (4.16)$$

де ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$\text{ПП} = \frac{1\,232\,402}{(1+0{,}2)^1} + \frac{2\,226\,609{,}9}{(1+0{,}2)^2} + \frac{3\,049\,937{,}7}{(1+0{,}2)^3} = 4\,338\,268{,}8 \text{ грн.}$$

$$\text{Е}_{\text{абс}} = 4\,338\,268{,}8 - 505\,356{,}2 = 3\,832\,912{,}6 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, встановлено, що проведення наукових досліджень для розробки програмного продукту та його подальше впровадження принесуть прибуток. Це підтверджує доцільність проведення досліджень [42].

Внутрішня економічна дохідність інвестицій $Е_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (4.17)$$

де $Е_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$Т_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$Е_в = \sqrt[3]{1 + \frac{3\,832\,912{,}6}{505\,356{,}2}} - 1 = 1$$

Порівняємо Ев з мінімальною (бар'єрною) ставкою дисконтування τ_{min}, яка визначає мінімальну дохідність, нижче якої інвестиції не будуть здійснюватися.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ_{min} визначається за формулою:

$$\tau_{min} = d + f \quad (4.18)$$

d – середньозважена ставка за депозитними операціями в комерційних банка;

f – показник, що характеризує ризикованість вкладень; *f* = 0,4.

d = 0,2.

$$\tau_{min} = 0{,}2 + 0{,}4 = 0{,}6$$

Оскільки $E_в = 100\% > \tau_{min} = 60\%$, то у інвестора є потенційна зацікавленість у фінансуванні даної наукової розробки.

Далі розраховуємо період окупності інвестицій $Т_ок$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки [42]:

$$T_{ок} = \frac{1}{E_в} \quad (4.19)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{1} = 1$$

Якщо $T_{ок}$ < 3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.5 Висновки до розділу

Проведено комплексну оцінку комерційного та технологічного потенціалу вдосконаленого методу Isolation Forest, розробленого з використанням алгоритму KMeans та блокчейн-орієнтованих журналів подій. Запропоноване рішення забезпечує підвищення точності й стабільності виявлення аномалій, зменшення кількості хибних спрацювань та підвищення рівня захищеності корпоративних інформаційних систем завдяки використанню незмінних та достовірних журналів подій.

Оцінювання свідчить, що витрати на науково-дослідні роботи, розробку та впровадження вдосконаленого методу є економічно доцільними. Інвестиції, необхідні для реалізації проєкту, здатні забезпечити швидке повернення вкладених ресурсів за рахунок зростання попиту на інтелектуальні засоби виявлення аномалій та підвищення рівня кіберзахисту підприємств.

Розрахунки підтверджують високий економічний ефект від комерціалізації вдосконаленого методу в умовах ринку. Використання розробленого рішення дає змогу підприємству отримати суттєве збільшення чистого прибутку за рахунок розширення кола користувачів, підвищення конкурентоспроможності продукту та його значної цінності для корпоративних систем безпеки.

Отже, проєкт має високий потенціал упровадження, відповідає сучасним вимогам до систем аналізу подій безпеки та є перспективним з позиції подальшої комерціалізації.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено, теоретично обґрунтовано та програмно реалізовано вдосконалений підхід до виявлення аномалій у корпоративних інформаційних системах на основі комбінування кластеризаційного методу KMeans, алгоритму Isolation Forest та блокчейн-орієнтованого журналювання подій. Запропонований метод вирішує критичні проблеми сучасних систем моніторингу безпеки, пов’язані з високою варіативністю поведінкових шаблонів, великою кількістю хибнопозитивних спрацювань та недостатньою доказовою надійністю журналів подій.

Проведений аналіз сучасних загроз корпоративних систем продемонстрував необхідність застосування адаптивних інтелектуальних методів, здатних реагувати на складні та динамічні аномалії, що не піддаються класичним правилам сигнатурного аналізу. Огляд існуючих моделей виявлення аномалій підтвердив обмеженість суто глобальних алгоритмів, зокрема традиційного Isolation Forest, у контексті високогетерогенних даних корпоративного середовища.

Запропонована у роботі комбінована модель усуває зазначені недоліки шляхом поетапної обробки подій: спочатку KMeans структурує дані у поведінково однорідні кластери, а потім у межах кожного кластера працює локальна модель Isolation Forest з адаптивним порогом аномальності. Такий підхід дає змогу значно точніше виявляти відхилення, релевантні саме для конкретної групи користувачів чи процесів, та мінімізувати кількість хибних спрацювань.

Програмна реалізація запропонованого підходу включає кластеризатор, модуль локального Isolation Forest, конвеєр інтеграції та блокчейн-журнал. Усі компоненти були протестовані на синтетичних і реалістичних даних, що підтвердило працездатність системи. Результати експериментів засвідчили підвищення показників F1-score, Recall та Precision порівняно з класичними моделями, а також зменшення латентності обробки. Блокчейн-модуль успішно пройшов перевірку цілісності, що підтвердило його коректність і практичну придатність.

Отримані результати підтверджують ефективність та інноваційність запропонованої системи. Вони можуть бути використані як основа для впровадження у промислові рішення, а також як база для подальшого наукового розвитку — зокрема, розширення моделі підтримкою потокових даних, застосування глибоких нейронних мереж для поведінкової аналітики або інтеграції міжорганізаційних блокчейн-ланцюгів для колективного обміну інформацією про загрози.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. — MIT Press, 2023. — 801 p.

2. Stallings W. Data and Computer Communications. — 11th ed. — Pearson, 2023. — 912 p.

3. Bishop C. M. Pattern Recognition and Machine Learning. — Springer, 2022. — 804 p.

4. Kshetri N. Blockchain and Cybersecurity. — MIT Press, 2022. — 328 p.

5. Provos N., Honeyman P. The Future of Security Analytics. — ACM Computing Surveys, 2022.

6. Isolation Forest Revisited: Advanced Applications in Large Scale Systems. — IEEE Security & Privacy, 2021.

7. Lars K., Meyer T. Machine Learning for Cybersecurity: Challenges and Opportunities. — Springer, 2021.

8. Лі В. Основи криптографії з відкритим ключем. — К. : Наш Формат, 2021. — 320 с.

9. Aggarwal C. Outlier Analysis. — Springer, 2020. — 464 p.

10. Zhao Y. et al. PyOD: A Python Toolbox for Scalable Outlier Detection. — Journal of Machine Learning Research, 2020.

11. Blockchain-Based Security for Distributed Systems. — MDPI Sensors, 2020.

12. Hands-On Blockchain with Python. — Packt Publishing, 2020. — 358 p.

13. Kaur H., Bhatia S. Efficient Anomaly Detection Using KMeans and Isolation Forest. — IEEE Access, 2019.

14. Ересь А. Криптографічні методи захисту інформації. — К. : Ліра-К, 2019. — 276 с.

15. Федотов А. А. Веб-програмування. Серверна частина. — СПб. : Питер, 2019. — 448 с.

16. Sousa P. Blockchain for Cybersecurity and Privacy. — Artech House, 2019. — 340 p.

17. ISO/IEC 27001:2013. Information security management systems — Requirements.

18. Python Software Foundation. Python 3.12 Documentation [Електронний ресурс]. — Режим доступу: https://docs.python.org/3/

19. Tanenbaum A. S., Wetherall D. Computer Networks. — 6th ed. — Pearson, 2019.

20. Shah K., Patel M. Machine Learning Algorithms for Intrusion Detection. — Elsevier, 2019.

21. Grama A. Introduction to Parallel Computing. — Pearson, 2019.

22. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. — IETF, 2017.

23. Столяренко О. М. Захист інформації в комп’ютерних системах. — К. : КНЕУ, 2017. — 312 с.

24. Шнайєр Б. Прикладна криптографія: протоколи, алгоритми. — К. : Вільямс, 2016. — 784 с.

25. Таненбаум Е. Архітектура комп'ютерних мереж. — К. : Вільямс, 2015. — 1024 с.

26. Bishop M. Computer Security: Art and Science. — 2nd ed. — Addison–Wesley, 2014.

27. Войцеховський В. Основи інформаційної безпеки. — К. : Кондор, 2014. — 240 с.

28. Stallings W. Cryptography and Network Security. — 5th ed. — Pearson, 2014.

29. RFC 6455: The WebSocket Protocol — IETF, 2011.

30. Грінстайн М., Спатсчек Р. Протоколи Інтернету. — К. : Діалектика, 2010. — 432 с.

31. Provos N., Honeyman P. Intrusion Detection Systems: A Survey. — ACM Computing Surveys, 2009.

32. RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. — IETF, 2008.

33. Kaufman C., Perlman R., Speciner M. Network Security: Private Communication in a Public World. — Prentice Hall, 2002.

34. ДСТУ ISO/IEC 18033-3:2017. Криптографічні алгоритми. Блокові шифри.

35. ДСТУ 3396.2-2014. Захист інформації. Технічний захист. Основні положення.

36. Menezes A., Van Oorschot P., Vanstone S. Handbook of Applied Cryptography. — CRC Press, 1996.

37. FIPS 197: Advanced Encryption Standard (AES). — NIST, 2001.

38. Merkle R. A Digital Signature Based on a Conventional Encryption Function. — CRYPTO'87.

39. Diffie W., Hellman M. New Directions in Cryptography. — IEEE Transactions on Information Theory, 1976.

40. Shannon C. Communication Theory of Secrecy Systems. — Bell System Technical Journal, 1949.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор

_____ Юрій ЯРЕМЧУК

"24" Вересня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

Вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах на основі метода машинного навчання KMeans та блокчейн-орієнтованих журналів подій

08-72.МКР.021.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент каф. МБІС

_____ Грицак А.В.

Вінниця – 2025 р.

1. Найменування та область застосування

Програмний засіб dдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах на основі метода машинного навчання KMeans та блокчейн-орієнтованих журналів подій

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №96 від 20. 03. 2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах

3.2 Призначення: розроблений програмний засіб виконує виявлення аномалій у корпоративних інформаційних системах

4. Джерела розробки

4.1. Ахрамович В. М. Ідентифікація й аутентифікація, керування доступом // Сучасний захист інформації. – 2016. №4.– С. 47-51.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

4.4. ZakariaOmar, ZangooeiToomaj, MohdAfiziMohdShukran. Enhancing Mixing Recognition-Based and Recall-Based Approach in Graphical Password Scheme. IJACT, Vol. 4, No. 15, pp. 189-197, 2012.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми		
2	Аналіз предметної області обраної теми		
3	Апробація отриманих результатів		
4	Розробка алгоритму роботи		
5	Написання магістерської роботи на основі розробленої теми		
6	Розробка економічної частини		
7	Передзахист магістерської кваліфікаційної роботи		
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи		
9	Захист магістерської кваліфікаційної роботи		

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Шкробот Є.С.

Додаток Б. Лістинг програми

```
def score_event(
    self,
    event: dict[str, Any],
    cluster_idx: int,
) -> Tuple[float, bool]:
    if cluster_idx not in self._cluster_models:
        raise ValueError(f"Для кластера {cluster_idx} модель не навчена")
    model_info = self._cluster_models[cluster_idx]
    model = model_info.isolation_forest
    threshold = model_info.threshold
    feature_vector = build_feature_vector(event).reshape(1, -1)
    raw_score = model.decision_function(feature_vector)[0]
    anomaly_score = -float(raw_score)  # чим більше, тим гірше
    is_anomaly = anomaly_score >= threshold
    return anomaly_score, is_anomaly
def save(self, path: str) -> None:
    payload = {
        "config": self.config.__dict__,
        "cluster_models": {
            cluster_idx: {
                "model": model_info.isolation_forest,
                "threshold": model_info.threshold,
            }
            for cluster_idx, model_info in self._cluster_models.items()
        },
    }
    joblib.dump(payload, path)
@classmethod
def load(cls, path: str) -> "ClusterIsolationForest":
    payload = joblib.load(path)
    config = IsolationForestConfig(**payload["config"])
    instance = cls(config=config)
    restored_models = {}
    for cluster_idx, data in payload["cluster_models"].items():
        restored_models[int(cluster_idx)] = ClusterIsolationModel(
            isolation_forest=data["model"],
            threshold=float(data["threshold"]),
        )
```

```
        instance._cluster_models = restored_models
        return instance
def score_event_with_pipeline(
    event: dict[str, Any],
    kmeans_model_path: str,
    isolation_model_path: str,
) -> tuple[int, float, bool]:
    clusterer = KMeansClusterer.load(kmeans_model_path)
    isolation_module = ClusterIsolationForest.load(isolation_model_path)
    cluster_idx = clusterer.predict_cluster(event)
    anomaly_score, is_anomaly = isolation_module.score_event(event, cluster_idx)
    return cluster_idx, anomaly_score, is_anomaly
from dataclasses import dataclass
from typing import Optional, Tuple
@dataclass
class AnomalyPipelineConfig:
    kmeans: KMeansConfig
    isolation: IsolationForestConfig
class AnomalyDetectionPipeline:
    def __init__(self, config: AnomalyPipelineConfig):
        self.config = config
        self._kmeans: Optional[KMeansClusterer] = None
        self._isolation: Optional[ClusterIsolationForest] = None
    @property
    def is_trained(self) -> bool:
        return self._kmeans is not None and self._isolation is not None
    def fit(self, events: list[dict[str, Any]]) -> None:
        if not events:
            raise ValueError("Список подій для навчання порожній")
        self._kmeans = KMeansClusterer(config=self.config.kmeans)
        self._kmeans.fit(events)
        feature_matrix = np.vstack([build_feature_vector(e) for e in events])
        feature_matrix_scaled = self._kmeans._scaler.transform(feature_matrix)
        cluster_labels = self._kmeans._model.predict(feature_matrix_scaled)
        self._isolation = ClusterIsolationForest(config=self.config.isolation)
        self._isolation.fit(events=events, cluster_labels=list(cluster_labels))
    def score_event(self, event: dict[str, Any]) -> Tuple[int, float, bool]:
        if not self.is_trained:
            raise RuntimeError("Комбінована модель не навчена")
```

```
    cluster_idx = self._kmeans.predict_cluster(event)
    anomaly_score, is_anomaly = self._isolation.score_event(event, cluster_idx)

    return cluster_idx, anomaly_score, is_anomaly
def save(self, path: str) -> None:
    if not self.is_trained:
        raise RuntimeError("Немає навченої моделі для збереження")
    payload = {
        "config": {
            "kmeans": self.config.kmeans.__dict__,
            "isolation": self.config.isolation.__dict__,
        },
        "kmeans_model": {
            "scaler": self._kmeans._scaler,
            "model": self._kmeans._model,
        },
        "isolation_models": {
            cluster_idx: {
                "model": model_info.isolation_forest,
                "threshold": model_info.threshold,
            }
            for cluster_idx, model_info in self._isolation._cluster_models.items()
        },
    }
    joblib.dump(payload, path)

@classmethod
def load(cls, path: str) -> "AnomalyDetectionPipeline":
    payload = joblib.load(path)
    kmeans_cfg = KMeansConfig(**payload["config"]["kmeans"])
    isolation_cfg = IsolationForestConfig(**payload["config"]["isolation"])
    config = AnomalyPipelineConfig(kmeans=kmeans_cfg, isolation=isolation_cfg)
    instance = cls(config=config)
    kmeans = KMeansClusterer(config=kmeans_cfg)
    kmeans._scaler = payload["kmeans_model"]["scaler"]
    kmeans._model = payload["kmeans_model"]["model"]
    instance._kmeans = kmeans
    isolation = ClusterIsolationForest(config=isolation_cfg)
    restored_models = {}
```

```
        for cluster_idx, data in payload["isolation_models"].items():
            restored_models[int(cluster_idx)] = ClusterIsolationModel(
                isolation_forest=data["model"],
                threshold=float(data["threshold"]),
            )
        isolation._cluster_models = restored_models
        instance._isolation = isolation

        return instance
def train_anomaly_pipeline(
    training_events_path: str,
    model_output_path: str,
    kmeans_cfg: Optional[KMeansConfig] = None,
    isolation_cfg: Optional[IsolationForestConfig] = None,
) -> None:
    with open(training_events_path, "r", encoding="utf-8") as f:
        events = json.load(f)

    config = AnomalyPipelineConfig(
        kmeans=kmeans_cfg or KMeansConfig(),
        isolation=isolation_cfg or IsolationForestConfig(),
    )
    pipeline = AnomalyDetectionPipeline(config=config)
    pipeline.fit(events)
    pipeline.save(model_output_path)


def process_event(
    event: dict[str, Any],
    model_path: str,
) -> dict[str, Any]:
    pipeline = AnomalyDetectionPipeline.load(model_path)
    cluster_idx, anomaly_score, is_anomaly = pipeline.score_event(event)

    enriched_event = {
        **event,
        "cluster_idx": cluster_idx,
        "anomaly_score": anomaly_score,
        "is_anomaly": is_anomaly,
```

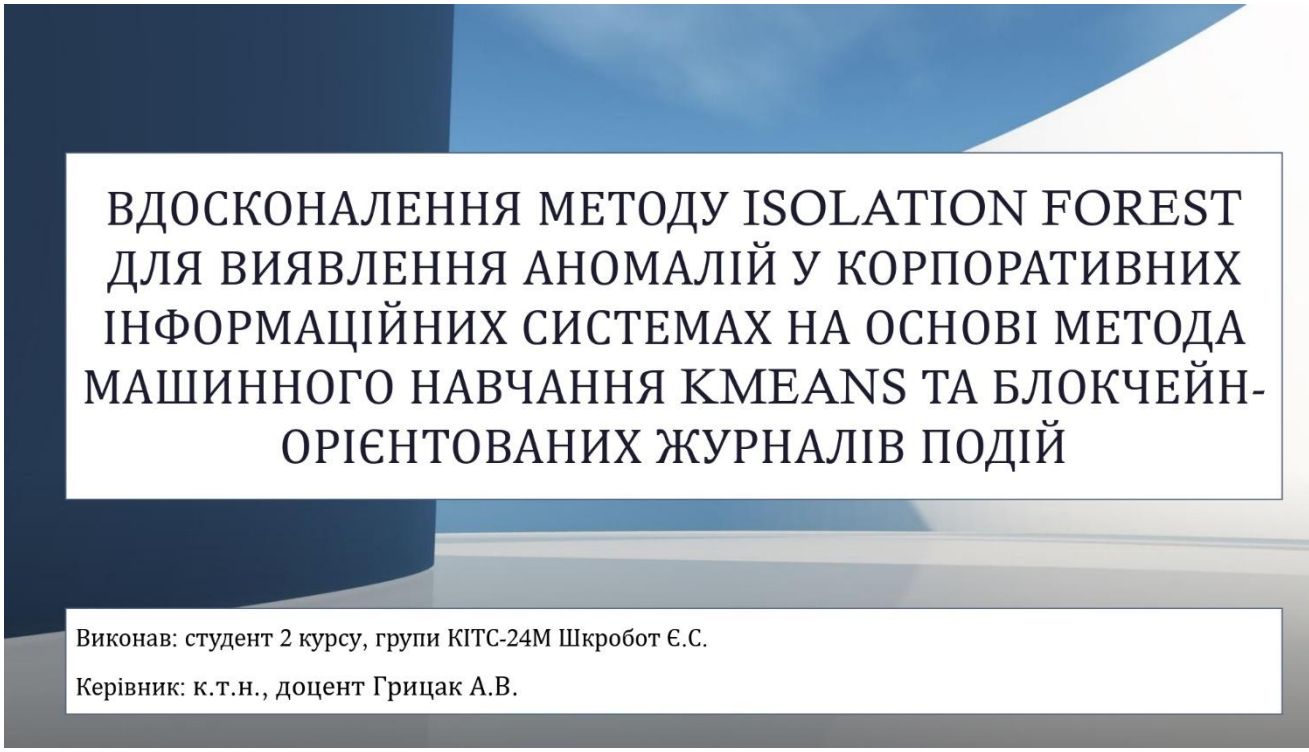
```
    }
    return enriched_event
import json
import hashlib
import hmac
import time
from dataclasses import dataclass, asdict
from typing import Any, Dict, List, Optional, Tuple


def canonical_serialize(event: Dict[str, Any]) -> bytes:
    return json.dumps(event, sort_keys=True, separators=(",", ":")).encode("utf-8")


def sha256(data: bytes) -> str:
    return hashlib.sha256(data).hexdigest()
@dataclass
class EventRecord:
    event: Dict[str, Any]
    event_hash: str

    @classmethod
    def from_event(cls, event: Dict[str, Any]) -> "EventRecord":
        canonical = canonical_serialize(event)
        h = sha256(canonical)
        return cls(event=event, event_hash=h)


@dataclass
class BlockHeader:
    index: int
    prev_block_hash: str
    merkle_root: str
    timestamp: float
    producer_id: str
    signature: str
def merkle_root_hash(hashes: List[str]) -> str:
    if not hashes:
        return sha256(b"")
```

```
    current_level = [h.encode("utf-8") for h in hashes]

    while len(current_level) > 1:
        next_level: List[bytes] = []
        for i in range(0, len(current_level), 2):
            left = current_level[i]
            if i + 1 < len(current_level):
                right = current_level[i + 1]
            else:
                right = left  # дублюємо останній
            combined = left + right
            next_level.append(hashlib.sha256(combined).hexdigest().encode("utf-8"))
        current_level = next_level
    return current_level[0].decode("utf-8")
@dataclass
class Block:
    header: BlockHeader
    records: List[EventRecord]


def sign_block_header(header: BlockHeader, secret_key: bytes) -> str:
    header_dict = asdict(header)
    # Підпис рахуємо без поля signature
    header_dict_no_sig = {k: v for k, v in header_dict.items() if k != "signature"}
    canonical = json.dumps(header_dict_no_sig, sort_keys=True, separators=(",", ":")).encode("utf-8")
    return hmac.new(secret_key, canonical, hashlib.sha256).hexdigest()
class BlockchainLog:
    def __init__(self, producer_id: str, secret_key: bytes):
        self.producer_id = producer_id
        self.secret_key = secret_key
        self.blocks: List[Block] = []
        self._pending_events: List[EventRecord] = []
    def add_event(self, event: Dict[str, Any]) -> EventRecord:
        record = EventRecord.from_event(event)
        self._pending_events.append(record)
        return record
```

```
def create_block(self) -> Optional[Block]:
    if not self._pending_events:
        return None

    index = len(self.blocks)
    prev_hash = self.blocks[-1].header.merkle_root if self.blocks else "GENESIS"

    record_hashes = [r.event_hash for r in self._pending_events]
    root = merkle_root_hash(record_hashes)
    ts = time.time()

    header = BlockHeader(
        index=index,
        prev_block_hash=prev_hash,
        merkle_root=root,
        timestamp=ts,
        producer_id=self.producer_id,
        signature="",  # тимчасово порожнє, додамо після підпису
    )

    signature = sign_block_header(header, self.secret_key)
    header.signature = signature

    block = Block(header=header, records=list(self._pending_events))

    self._pending_events.clear()
    self.blocks.append(block)

    return block
def validate_chain(self) -> bool:
    for i, block in enumerate(self.blocks):
        header = block.header
        if i == 0:
            if header.prev_block_hash != "GENESIS":
                return False
        else:
            prev_root = self.blocks[i - 1].header.merkle_root
            if header.prev_block_hash != prev_root:
                return False
```

```
    record_hashes = [r.event_hash for r in block.records]
    expected_root = merkle_root_hash(record_hashes)
    if expected_root != header.merkle_root:
        return False
    expected_sig = sign_block_header(
        BlockHeader(
            index=header.index,
            prev_block_hash=header.prev_block_hash,
            merkle_root=header.merkle_root,
            timestamp=header.timestamp,
            producer_id=header.producer_id,
            signature="",  # підпис не включається в дані для підпису
        ),
        self.secret_key,
    )
    if expected_sig != header.signature:
        return False

return True
```

Додаток В. Ілюстративний матеріал



ВСТУП

Сучасні корпоративні інформаційні системи генерують величезні обсяги різнорідних даних і функціонують у складних динамічних умовах. Традиційні сигнатурні методи захисту вже не справляються з виявленням нових типів атак, які маскуються під нормальну активність та формують складні поведінкові патерни. Це створює потребу у використанні інтелектуальних моделей, здатних адаптивно аналізувати поведінку системи та виявляти приховані аномалії.

Водночас критично важливим стає питання довіри до журналів подій. Звичайні логи можуть бути змінені або видалені, що унеможливлює достовірне розслідування інцидентів. Тому застосування блокчейн-технологій відкриває можливість забезпечити незмінність, прозорість та доказовість даних безпеки.

Робота присвячена створенню комплексної системи виявлення аномалій, яка поєднує кластеризацію KMeans, локальні моделі Isolation Forest та блокчейн-орієнтоване журналювання подій. Такий підхід забезпечує вищу точність детекції, зниження хибнопозитивних спрацювань і підвищення довіри до результатів моніторингу.

АКТУАЛЬНІСТЬ ТА МЕТА

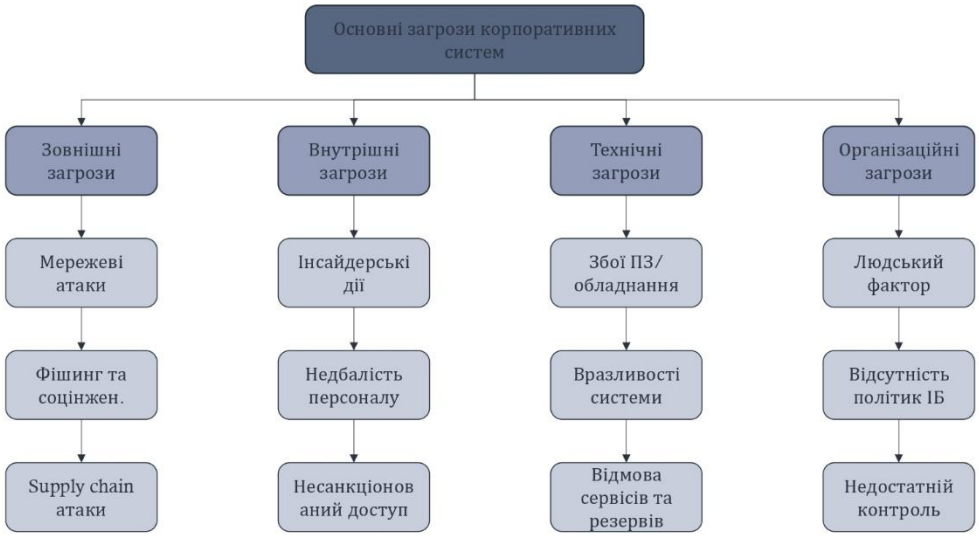
МЕТА

Створити вдосконалену систему виявлення аномалій у корпоративних інформаційних системах шляхом комбінування методів KMeans та Isolation Forest із використанням блокчейн-орієнтованого журналювання подій для забезпечення високої точності детекції та цілісності логів.

АКТУАЛЬНІСТЬ

- Різке зростання обсягів і складності корпоративних даних.
- Недостатня ефективність сигнатурних та глобальних моделей виявлення аномалій.
- Потреба у незмінних, захищених від маніпуляцій журналах безпеки.
- Необхідність зменшення хибнопозитивних спрацювань у системах моніторингу.

Основні загрози корпоративних систем



Методи виявлення аномалій

Схема роботи методу Isolation Forest

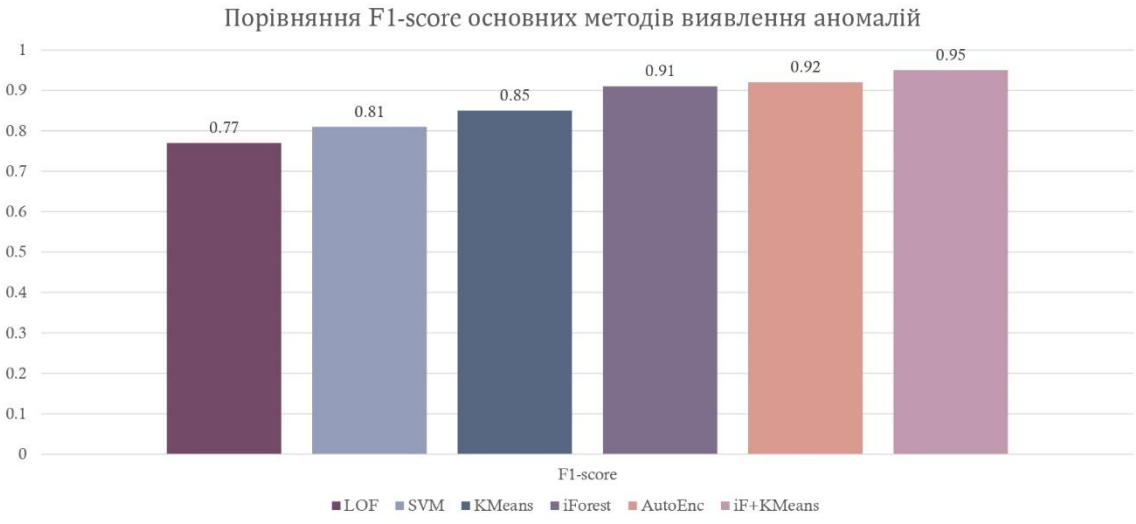
Принципова схема роботи алгоритму KMeans

АНАЛІЗ СУЧАСНИХ МЕТОДІВ І МОДЕЛЕЙ ВИЯВЛЕННЯ АНОМАЛІЙ У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

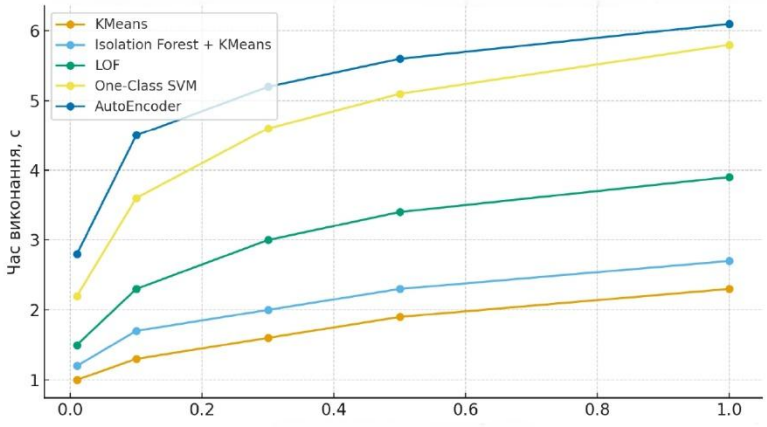
ПОРІВНЯЛЬНІ РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМІВ ВИЯВЛЕННЯ АНОМАЛІЙ

№	Метод	Precision	Recall	F1-score	TNR	t, сек	Особливості
1	Isolation Forest	0.94	0.88	0.91	0.96	1.8	Стабільна точність, низькі витрати пам'яті
2	KMeans (кластерна відстань)	0.89	0.81	0.85	0.92	1.3	Висока швидкодія, чутливість до вибору k
3	One-Class SVM	0.91	0.74	0.81	0.94	4.1	Висока обчислювальна складність, хороша узагальнюваність
4	LOF (Local Outlier Factor)	0.87	0.69	0.77	0.89	2.7	Ефективний для локальних аномалій, нестабільний при шумі
5	AutoEncoder	0.95	0.90	0.92	0.97	5.4	Висока точність, але потребує навчання
6	Isolation Forest + KMeans	0.97	0.94	0.95	0.98	2.0	Гібридна модель: оптимальний баланс точності й швидкодії

Порівняння F1-score основних методів виявлення аномалій



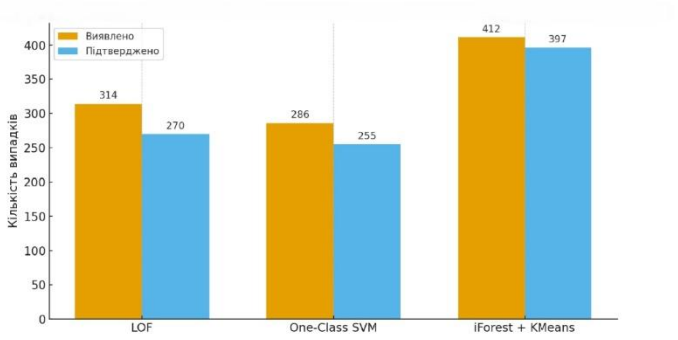
ЗАЛЕЖНІСТЬ ЧАСУ ВИКОНАННЯ ВІД ОБСЯГУ ДАНИХ



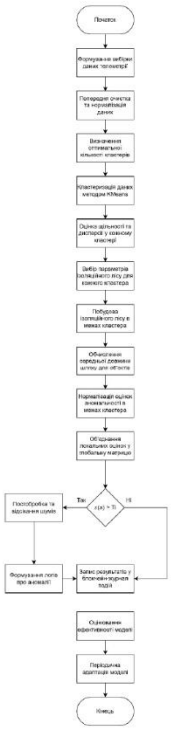
РОЗРАХУНОК ІНТЕГРАЛЬНОГО ПОКАЗНИКА ЕФЕКТИВНОСТІ АЛГОРИТМІВ ВИЯВЛЕННЯ АНОМАЛІЙ

Метод	F1-score	TNR	t, сек	t_{norm}	E
Isolation Forest	0.91	0.96	1.8	1.38	1.35
KMeans	0.85	0.92	1.3	1.00	1.35
One-Class SVM	0.81	0.94	4.1	3.15	0.97
AutoEncoder	0.92	0.97	5.4	4.15	0.95
iForest + KMeans	0.95	0.98	2.0	1.54	1.52

ПОРІВНЯННЯ ФАКТИЧНИХ ТА ВИЯВЛЕНИХ АНОМАЛІЙ У ЖУРНАЛАХ ДОСТУПУ

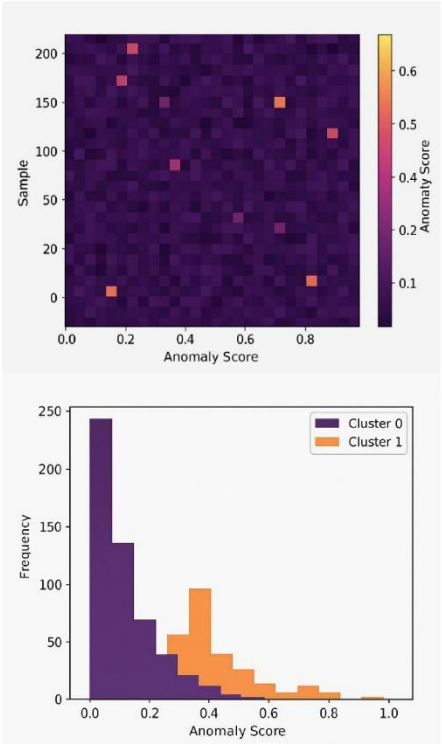


РОЗРОБЛЕНИЙ КОМБІНОВАНИЙ АЛГОРИТМ ISOLATION FOREST – KMEANS



РОЗРОБЛЕНИЙ АЛГОРИТМ ФОРМУВАННЯ ТА ВАЛІДАЦІЇ БЛОКЧЕЙН-ЗАПИСІВ ПОДІЙ

ТЕСТУВАННЯ СИСТЕМИ



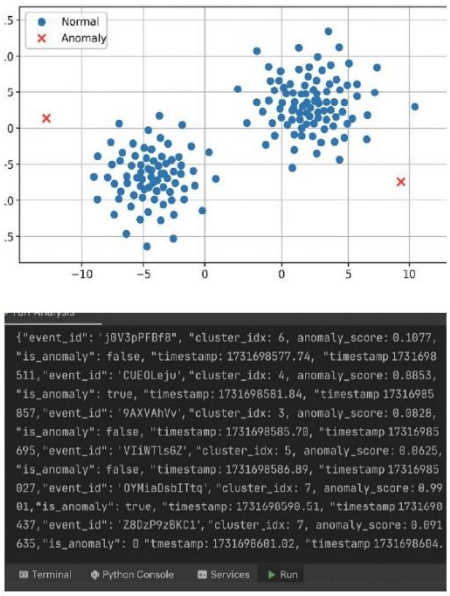
Візуалізація кластерів після навчання алгоритму KMeans

На першому етапі виконувалося навчання KMeans-модуля на навчальній вибірці. За результатами кластеризації модель сформувала k=8 кластерів, що відповідають різним типам поведінкових груп у системі. Після цього модуль Isolation Forest був навчений окремо в межах кожного з кластерів з обчисленням локальних порогів аномальності.

Розподіл оцінок аномальності та вибір порогу для кластерів

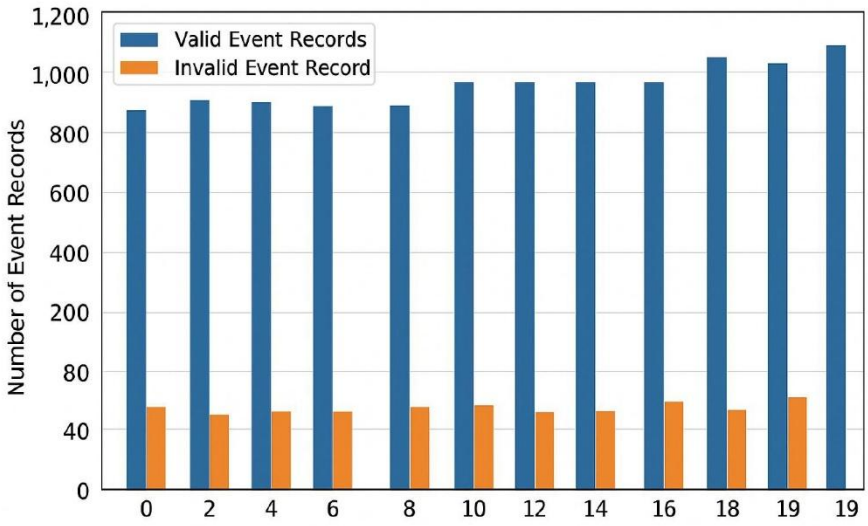
Далі була виконана побудова локальних ізоляційних лісів. На зображенні показано розподіл аномальних оцінок у кількох кластерах із визначенням порогу на рівні квантиля 0.99.

На цьому етапі тестові події пропускалися через комбіновану модель. Для кожної події модель спочатку визначала кластер, а потім — її аномальність у межах цього кластера.

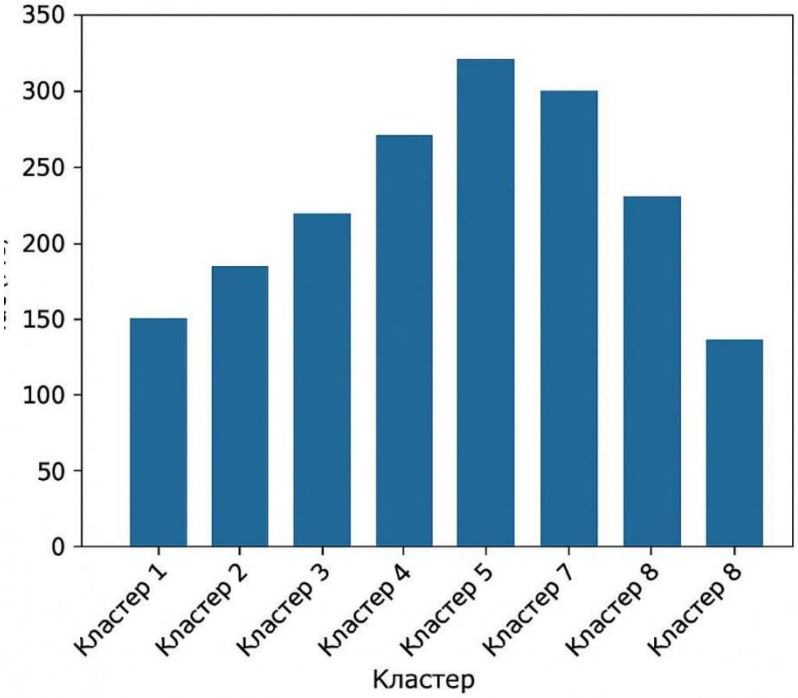


Результати класифікації подій у PyCharm

Для демонстрації частотності виявлених аномалій було побудовано діаграму, де кожен кластер має свою частку аномальних подій.

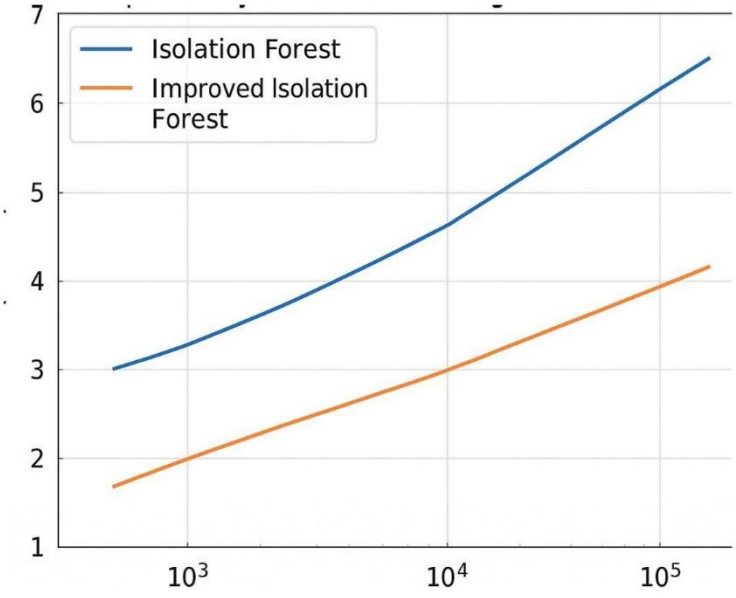


РЕЗУЛЬТАТ УСПІШНОЇ ПЕРЕВІРКИ ЦІЛІСНОСТІ БЛОКЧЕЙН-ЛАНЦЮГА



ПОРІВНЯННЯ F1-SCORE КЛАСИЧНОГО ТА ВДОСКОНАЛЕНОГО ISOLATION FOREST

ЗАЛЕЖНІСТЬ ЧАСУ ОБРОБКИ ВІД ОБСЯГУ ДАНИХ ДЛЯ РІЗНИХ МОДЕЛЕЙ

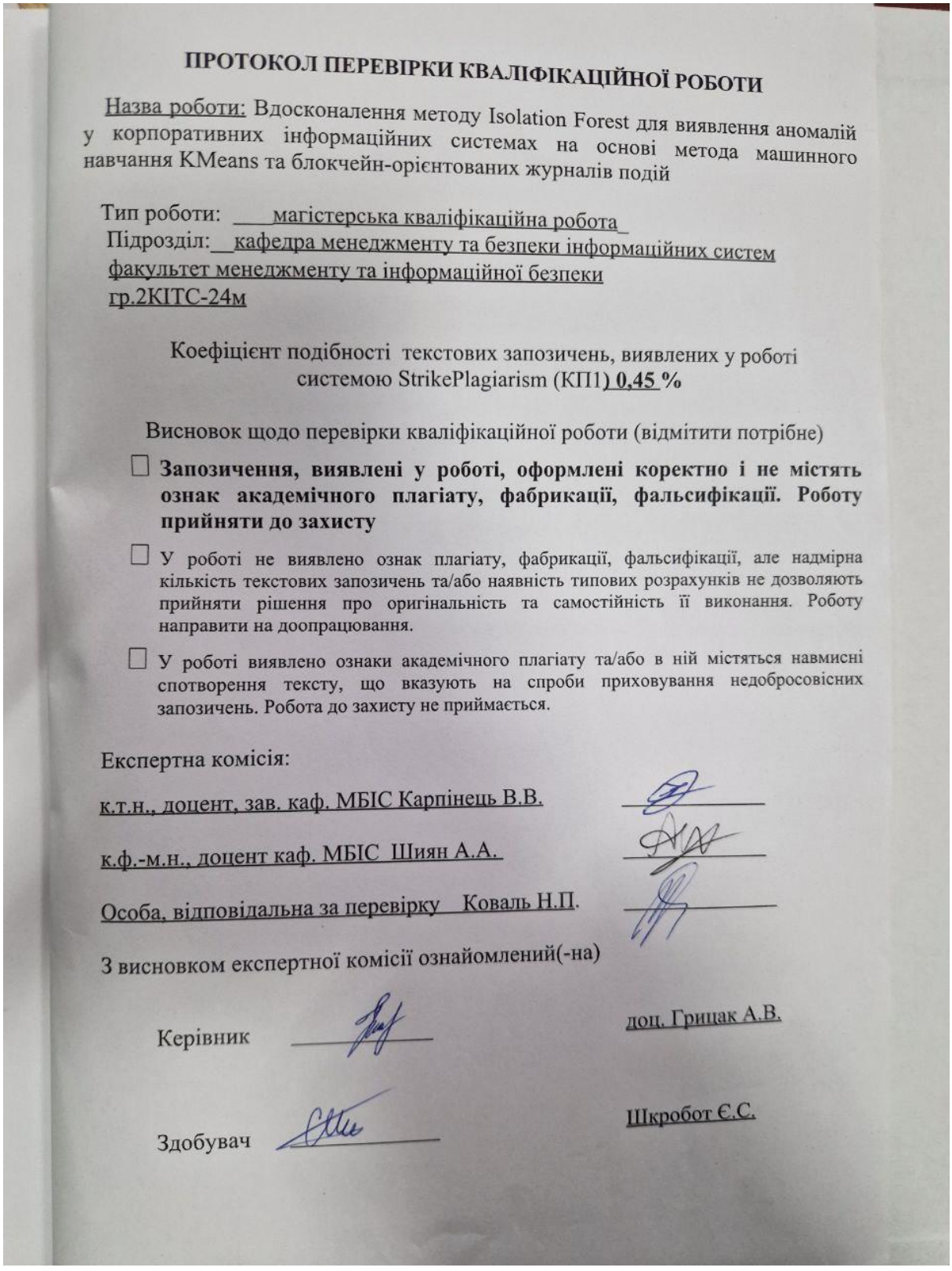


ВИСНОВОК

- У роботі розроблено комбіновану модель виявлення аномалій, яка значно підвищує точність та знижує кількість хибних спрацювань завдяки поєднанню KMeans та вдосконаленого Isolation Forest. Додатково створено блокчейн-модуль журналювання подій, який забезпечує незмінність, надійність і доказовість логів безпеки. Результати експериментів підтвердили ефективність запропонованої системи та її придатність до впровадження у високонавантажені корпоративні середовища.

ДЯКУЮ ЗА УВАГУ!

Додаток Г. Протокол перевірки на антиплагіат



ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Вдосконалення методу Isolation Forest для виявлення аномалій у корпоративних інформаційних системах на основі метода машинного навчання KMeans та блокчейн-орієнтованих журналів подій

Тип роботи: магістерська кваліфікаційна робота
Підрозділ: кафедра менеджменту та безпеки інформаційних систем
факультет менеджменту та інформаційної безпеки
гр.2КІТС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) **0,45 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту**

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В. _____

к.ф.-м.н., доцент каф. МБІС Шиян А.А. _____

Особа, відповідальна за перевірку Коваль Н.П. _____

З висновком експертної комісії ознайомлений(-на)

Керівник _____ доц. Грицак А.В.

Здобувач _____ Шкробот Є.С.