

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО

В.о. завідувача кафедри

Олександр КОВАЛЬ

«___»_____2026 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці» спеціальності 121

Інженерія програмного забезпечення

на тему: «Інтелектуальна система прийняття рішень при виявленні
несанкціонованого доступу до хмарного сховища даних на основі машинного
навчання»

Виконала:

студентка IV курсу, групи ТВ-з21

Гуцило Софія Романівна

(прізвище, ім'я, по батькові)

(підпис)

Керівник:

Доцент Шуклін Г. В.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Рецензент:

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цій дипломній
роботі немає запозичень із праць
інших авторів без відповідних
посилань.

Студентка _____

(підпис)

Київ – 2026

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Навчально-науковий інститут атомної та теплової енергетики Кафедра
інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри
Олександр КОВАЛЬ

« » 2026р.

ЗАВДАННЯ

на дипломну роботу студентці

Гуцило Софії Романівні

(прізвище, ім'я, по батькові)

1. Тема роботи: Інтелектуальна система прийняття рішень при виявленні
несанкціонованого доступу до хмарного сховища даних на основі машинного
навчання

керівник роботи Шуклін Герман Вікторович, доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» травня 2026 р.
№ 1993-с

2. Строк подання студентом роботи «10» червня 2026 р.

3. Вихідні дані до роботи: Наукові та нормативні джерела з питань хмарної безпеки,
журнали подій доступу до хмарного сховища, CSV-набори даних, методи
машинного навчання, засоби Python, FastAPI, PostgreSQL, Docker Compose, scikit-
learn і TensorFlow

4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): Аналіз предметної області та загроз несанкціонованого доступу до
хмарних сховищ; проєктування архітектури; реалізація програмного продукту
CloudGuard ML; тестування та експериментальна перевірка розробленої системи

5. Перелік ілюстративного матеріалу: Архітектура захисту хмарного сховища;
функціональна архітектура інтелектуальної системи; компонентна архітектура
CloudGuard ML;

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	31.10.2025	
2	Дослідження предметної області	01.11.2025 – 31.11.2025	
3	Дослідження існуючих рішень	01.12.2025 – 31.12.2025	
4	Постановка вимог до проєктування системи	01.01.2026 – 31.01.2026	
5	Розробка програмного продукту	01.02.2026 – 03.05.2026	
6	Тестування	04.05.2026 – 10.05.2026	
7	Захист програмного продукту	11.05.2026 – 15.05.2026	
8	Оформлення дипломної роботи	18.05.2026 – 31.05.2026	
9	Передзахист	01.06.2026 – 05.06.2026	
10	Захист	15.06.2026 – 19.06.2026	

Студентка

(підпис)

Софія ГУЦИЛО

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Герман ШУКЛІН

(ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 78 сторінок, 13 рисунків, 18 таблиць, 4 додатки та 34 посилання.

Метою роботи є розроблення інтелектуальної системи прийняття рішень при виявленні несанкціонованого доступу до хмарного сховища даних на основі методів машинного навчання.

У дипломній роботі проаналізовано хмарні сховища даних як об'єкт захисту, розглянуто основні загрози безпеці хмарного середовища, сценарії несанкціонованого доступу, традиційні та інтелектуальні підходи до виявлення підозрілої активності. Обґрунтовано доцільність використання поведінкової аналітики, методів машинного навчання, ризик-скорингу та механізмів підтримки прийняття рішень.

У межах роботи спроектовано архітектуру інтелектуальної системи CloudGuard ML, визначено структуру вхідних даних, набір поведінкових ознак, модель оцінювання ризику та логіку формування рекомендацій щодо реагування. Реалізовано веборієнтований програмний продукт на основі Python, FastAPI, PostgreSQL, Docker Compose, pandas, scikit-learn і TensorFlow. Система забезпечує імпорт CSV-журналів доступу, попередню обробку даних, формування ознак, застосування моделей Random Forest, Isolation Forest і TensorFlow Autoencoder, оцінювання risk score, збереження результатів аналізу та візуалізацію показників через вебінтерфейс.

Практичне значення одержаних результатів полягає у створенні програмного прототипу інтелектуальної системи, який може бути використаний як основа для подальшого розвитку засобів моніторингу доступу до хмарних сховищ, виявлення

аномальної активності, оцінювання ризику та підтримки реагування на інциденти інформаційної безпеки.

Ключові слова: хмарне сховище даних; несанкціонований доступ; інформаційна безпека; машинне навчання; поведінкова аналітика; аномалія; risk score; Random Forest; Isolation Forest; Autoencoder; FastAPI; PostgreSQL.

ABSTRACT

Structure and scope of the thesis. The work contains 78 pages, 13 figures, 18 tables, 4 appendices and 34 references.

The purpose of the work is to develop an intelligent decision-making system for detecting unauthorized access to cloud data storage based on machine learning methods.

The thesis analyzes cloud data storage as a security object, considers the main threats to cloud environments, unauthorized access scenarios, and traditional and intelligent approaches to detecting suspicious activity. The relevance of using behavioral analytics, machine learning methods, risk scoring and decision support mechanisms for detecting anomalous actions of users and services in cloud storage is substantiated.

As part of the work, the architecture of the CloudGuard ML intelligent system was designed. The structure of input data, a set of behavioral features, a risk assessment model and the logic for generating response recommendations were defined. A web-oriented software product was implemented using Python, FastAPI, PostgreSQL, Docker Compose, pandas, scikit-learn and TensorFlow. The system supports importing CSV access logs, preprocessing data, generating features, applying Random Forest, Isolation Forest and TensorFlow Autoencoder models, calculating risk score, storing analysis results and visualizing indicators through a web interface.

The practical value of the obtained results lies in the creation of a software prototype of an intelligent system that can be used as a basis for further development of tools for monitoring access to cloud storage, detecting anomalous activity, assessing risk and supporting response to information security incidents.

Keywords: cloud data storage; unauthorized access; information security; machine learning; behavioral analytics; anomaly; risk score; Random Forest; Isolation Forest; Autoencoder; FastAPI; PostgreSQL.

ЗМІСТ

	Стор.
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	10
ВСТУП.....	11
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	14
1.1 Хмарні сховища даних як об’єкт захисту	14
1.2 Загрози безпеці хмарних сховищ та сценарії несанкціонованого доступу.....	15
1.3 Традиційні підходи до виявлення несанкціонованого доступу в хмарному середовищі	16
1.4 Інтелектуальні методи, машинне навчання та UEBA у задачах виявлення аномалій.....	18
1.5 Порівняльний аналіз існуючих рішень і наукових підходів.....	19
1.6 Обґрунтування актуальності теми, мета, об’єкт, предмет і завдання дослідження	20
Висновки до розділу 1	21
2 МЕТОДИ ТА МОДЕЛІ ПОБУДОВИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ.....	22
2.1 Формалізація задачі виявлення несанкціонованого доступу до хмарного сховища	22
2.2 Архітектура інтелектуальної системи прийняття рішень	23
2.3 Модель подій доступу та структура вхідних даних	24
2.4 Формування поведінкових ознак користувачів і сервісів.....	25
2.5 Вибір і обґрунтування алгоритмів машинного навчання	27
2.6 Модель оцінювання ризику та механізм прийняття рішень щодо реагування	29
Висновки до розділу 2	30

3 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ	31
3.1 Загальна характеристика розробленого програмного продукту	31
3.2 Функціональні та нефункціональні вимоги до системи	33
3.3 Сценарії використання системи та ролі користувачів	34
3.4 Загальна архітектура програмної реалізації	35
3.5 Організація серверної частини системи на основі FastAPI	36
3.6 Організація бази даних та збереження результатів аналізу	38
3.7 Реалізація модуля імпорту та попередньої обробки CSV-даних	40
3.8 Реалізація модуля формування поведінкових ознак	41
3.9 Реалізація модуля машинного навчання та виявлення аномалій	43
3.10 Реалізація модуля оцінювання ризику та формування рекомендацій	44
3.11 Реалізація вебінтерфейсу користувача та адміністратора	45
3.12 Контейнеризоване розгортання системи за допомогою Docker Compose	47
Висновки до розділу 3	49
4 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОЗРОБЛЕНОЇ СИСТЕМИ	50
4.1 Методика тестування програмного продукту	50
4.2 Підготовка тестових наборів даних і сценаріїв несанкціонованого доступу	52
4.3 Функціональне тестування основних модулів системи	54
4.4 Перевірка коректності імпорту, обробки та збереження даних	57
4.5 Тестування роботи моделей машинного навчання	59
4.6 Аналіз показників accuracy, precision, recall та F1-score	61
4.7 Аналіз матриці помилок і результатів перехресної перевірки	62

4.8 Порівняння rule-based baseline, Random Forest, Isolation Forest і TensorFlow Autoencoder	64
4.9 Аналіз результатів risk-score та рекомендацій щодо реагування.....	66
4.10 Оцінювання практичної ефективності розробленої системи	68
Висновки до розділу 4	70
ВИСНОВКИ.....	72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	76
ДОДАТОК А Лістинг розробленої систем.....	80
ДОДАТОК Б Презентація.....	94

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовне позначення	Пояснення
API	Application Programming Interface, програмний інтерфейс прикладного програмування, що забезпечує взаємодію між програмними компонентами або сервісами
Autoencoder	Нейромережева модель автоенкодера, яка використовується для виявлення аномалій за помилкою реконструкції даних
CSV	Comma-Separated Values, текстовий формат подання табличних даних, у якому значення розділяються комами або іншими розділювачами
Docker	Інструмент для опису та запуску багатоконтейнерного середовища розробленої системи
Compose	
FastAPI	Вебфреймворк Python, використаний для реалізації серверної частини розробленої системи
IAM	Identity and Access Management, система керування ідентифікацією, автентифікацією та правами доступу користувачів
IDS	Intrusion Detection System, система виявлення вторгнень
IPS	Intrusion Prevention System, система запобігання вторгненням
Isolation Forest	Алгоритм виявлення аномалій, що базується на принципі ізоляції нетипових спостережень
MFA	Multi-Factor Authentication, багатофакторна автентифікація користувача
ML	Machine Learning, машинне навчання, сукупність методів побудови моделей, здатних виявляти закономірності у даних

ВСТУП

В умовах цифрової трансформації підприємств, державних установ і сервісних організацій хмарні технології стали одним із ключових елементів інформаційної інфраструктури. Хмарні сховища даних забезпечують масштабованість, доступність, централізоване зберігання інформації, інтеграцію з корпоративними сервісами та можливість організації спільної роботи з даними незалежно від фізичного розташування користувачів. Водночас перенесення критично важливої, службової, конфіденційної та персональної інформації до хмарного середовища підвищує вимоги до контролю доступу, моніторингу подій і своєчасного виявлення потенційних інцидентів інформаційної безпеки [1–5].

Актуальність теми зумовлена тим, що несанкціонований доступ до хмарних сховищ даних дедалі частіше реалізується не шляхом прямого порушення роботи інфраструктури, а через використання скомпрометованих облікових записів, токенів, API-ключів або легітимних каналів доступу. У таких умовах дії порушника можуть зовні виглядати як штатна активність користувача: вхід до системи, перегляд об'єктів, завантаження файлів, зміна політик доступу або створення публічних посилань. Традиційні механізми безпеки, зокрема рольові моделі доступу, статичні правила, сигнатурні засоби виявлення та ручний аналіз журналів, не завжди здатні своєчасно виявити приховані поведінкові аномалії, особливо за умов великого обсягу подій і динамічної зміни хмарного середовища [2, 3, 6, 10].

Одним із перспективних напрямів розв'язання цієї проблеми є застосування методів машинного навчання та поведінкової аналітики. Такі методи дають змогу аналізувати не лише факт виконання окремої операції, а й ширший контекст доступу: роль користувача, географічне розташування, тип пристрою, спосіб автентифікації, час виконання запиту, кількість невдалих спроб входу, обсяг переданих даних, інтенсивність звернень і критичність операції. На основі цих даних можна формувати поведінкові профілі користувачів і сервісів, виявляти

відхилення від нормального шаблону активності та оцінювати ризик події [10, 16, 17].

Метою дипломної роботи є розроблення інтелектуальної системи прийняття рішень при виявленні несанкціонованого доступу до хмарного сховища даних на основі методів машинного навчання.

Об'єктом дослідження є процеси доступу користувачів і сервісів до хмарного сховища даних у корпоративному інформаційному середовищі.

Предметом дослідження є методи, моделі та програмні засоби виявлення аномальної активності, оцінювання ризику й підтримки прийняття рішень щодо реагування на потенційно несанкціонований доступ до хмарного сховища даних.

Для досягнення поставленої мети необхідно вирішити такі завдання: проаналізувати хмарні сховища даних як об'єкт захисту та визначити основні сценарії несанкціонованого доступу; дослідити традиційні й інтелектуальні підходи до виявлення підозрілої активності в хмарному середовищі; формалізувати задачу виявлення несанкціонованого доступу як задачу аналізу подій доступу та оцінювання рівня ризику; розробити архітектуру інтелектуальної системи прийняття рішень; визначити структуру вхідних даних і набір поведінкових ознак користувачів та сервісів; обґрунтувати вибір алгоритмів машинного навчання для класифікації та виявлення аномалій; реалізувати програмну систему CloudGuard ML з вебінтерфейсом, модулем імпорту CSV-даних, блоком машинного навчання, risk-score-механізмом і журналюванням результатів; провести функціональне тестування та експериментальну перевірку розробленої системи; оцінити ефективність моделей за показниками accuracy, precision, recall, F1-score, матрицею помилок і результатами перехресної перевірки.

У роботі використано такі методи дослідження та розроблення: аналіз наукових, нормативних і технічних джерел з питань хмарної безпеки, контролю доступу та машинного навчання; методи системного аналізу для визначення вимог і структури програмної системи; методи моделювання даних для побудови структури подій доступу; методи інженерії ознак для формування поведінкових

характеристик користувачів і сервісів; методи машинного навчання, зокрема rule-based baseline, Random Forest, Isolation Forest і TensorFlow Autoencoder; методи програмної інженерії для реалізації веборієнтованої системи; методи тестування програмного забезпечення та експериментального оцінювання якості моделей.

Практичне значення отриманих результатів полягає у створенні програмної системи CloudGuard ML, яка забезпечує імпорт і попередню обробку журналів доступу, формування поведінкових ознак, застосування моделей машинного навчання, оцінювання рівня ризику, формування рекомендацій щодо реагування та візуалізацію результатів через вебінтерфейс. Розроблена система може бути використана як прототип модуля моніторингу доступу та підтримки реагування на інциденти для організацій, що працюють із хмарними сховищами даних і потребують оперативного виявлення підозрілої активності.

Дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У першому розділі розглянуто предметну область, загрози безпеці хмарних сховищ, традиційні та інтелектуальні підходи до виявлення несанкціонованого доступу. У другому розділі подано формалізацію задачі, архітектуру інтелектуальної системи, модель подій доступу, поведінкові ознаки, обґрунтування алгоритмів машинного навчання та модель оцінювання ризику. У третьому розділі описано проєктування і програмну реалізацію системи CloudGuard ML, її серверну частину, базу даних, модулі обробки даних, машинного навчання, risk-score та вебінтерфейс. У четвертому розділі проведено тестування та експериментальну перевірку системи, проаналізовано якість моделей, матрицю помилок, результати перехресної перевірки, рекомендації щодо реагування, практичну ефективність і напрями подальшого розвитку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Хмарні сховища даних як об'єкт захисту

Хмарні сховища даних упродовж останнього десятиліття перетворилися на базовий компонент цифрової інфраструктури підприємств, державних установ та сервісних організацій. На практиці це означає перенесення значної частини інформаційних активів до публічних, приватних або гібридних хмарних середовищ, у межах яких забезпечуються зберігання, резервування, оброблення та спільний доступ до даних. Відповідно до рекомендацій NIST, під час використання публічної хмари організація передає частину функцій управління зовнішньому провайдеру, але зберігає відповідальність за захист власних даних, облікових записів, політик доступу та контроль ризиків [1].

Для цілей цієї роботи під хмарним сховищем даних доцільно розуміти не лише репозиторій файлів або об'єктне сховище, а цілісне середовище, яке охоплює самі дані, метадані, журнали подій, сервіси автентифікації, механізми авторизації, API доступу, токени, інтеграції із зовнішніми застосунками та інструменти адміністрування. Отже, об'єктом захисту виступають як власне інформаційні ресурси, так і контекст доступу до них: хто, коли, звідки, з якого пристрою та за яких умов виконував операції читання, завантаження, модифікації або видалення [1, 2].

З позиції інформаційної безпеки хмарне сховище даних є привабливою цілью для зловмисника з кількох причин. По-перше, у ньому концентруються великі обсяги конфіденційної, службової та комерційно важливої інформації. По-друге, атака на механізми доступу до такого сховища може забезпечити масштабований доступ одразу до значної кількості ресурсів. По-третє, несанкціонований доступ у хмарі нерідко маскується під легітимну активність, що ускладнює виявлення інцидентів виключно статичними правилами [1, 3].

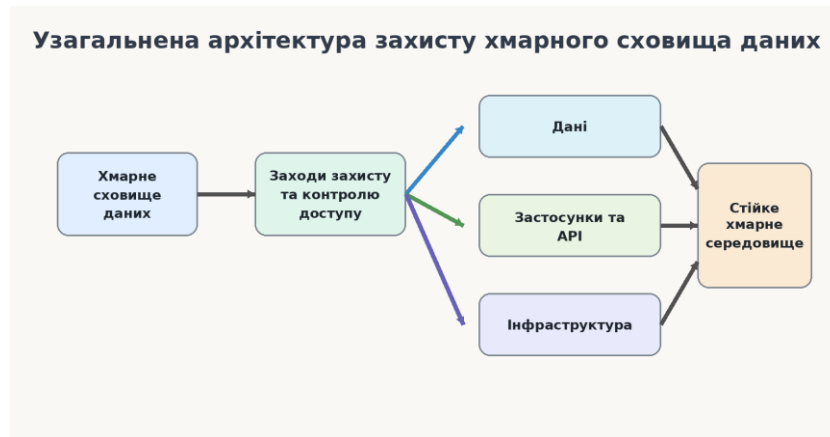


Рисунок 1.1 – Узагальнена архітектура захисту хмарного сховища даних

Узагальнена архітектура захисту хмарного середовища, наведена на рисунку 1.1, демонструє, що механізми безпеки мають бути спрямовані одночасно на кілька рівнів: захист даних, захист застосунків і API, а також захист інфраструктурних компонентів. Для систем, що працюють із хмарними сховищами, це означає необхідність поєднання політик контролю доступу, криптографічних засобів, журналювання подій, засобів реагування на інциденти та аналітичних інструментів виявлення аномалій [1–3].

1.2 Загрози безпеці хмарних сховищ та сценарії несанкціонованого доступу

За матеріалами Cloud Security Alliance, до найбільш поширених та небезпечних проблем належать помилки конфігурації, недосконалість ідентифікації та керування доступом (IAM), незахищені або надмірно відкриті API, компрометація облікових даних, зловживання привілеями й ризики ланцюга постачання [3]. Подібні загрози для хмарних сервісів систематизує також ENISA, наголошуючи на необхідності належного управління ризиками, безперервного моніторингу та контролю інцидентів [4, 5].

Типовим сценарієм несанкціонованого доступу є використання викрадених або підібраних облікових даних. У такому випадку порушник не обов'язково ламає інфраструктуру безпосередньо: достатньо отримати пароль, токен або ключ доступу до API. Після цього зловмисник може входити до системи під виглядом штатного користувача, переглядати вміст сховища, змінювати політики доступу, створювати нові публічні посилання або виконувати масове завантаження даних. У журналах подій така активність часто має вигляд послідовності формально допустимих операцій, однак її контекст і частотні характеристики відхиляються від типового профілю поведінки [1, 3].

Для корпоративного середовища значну небезпеку становлять також внутрішні загрози. Легітимний користувач або адміністратор може свідомо чи несвідомо виконувати дії, що призводять до витоку або модифікації даних. На відміну від зовнішнього нападника, внутрішній користувач уже має авторизований доступ і часто діє в межах наданих повноважень, тому ідентифікація шкідливої активності потребує аналізу аномальних шаблонів поведінки: нетипового часу доступу, нового географічного регіону, незвичного пристрою, аномального обсягу завантаження або серії невдалих спроб автентифікації [3, 5].

1.3 Традиційні підходи до виявлення несанкціонованого доступу в хмарному середовищі

До традиційних засобів забезпечення безпеки хмарного сховища належать механізми ідентифікації та автентифікації, ролева або атрибутна модель доступу, мережеві обмеження, фільтрація IP-адрес, списки контролю доступу, журнали аудиту, засоби шифрування, а також сигнатурні IDS/IPS і класичні SIEM-рішення. Їхнє головне призначення полягає у встановленні базових бар'єрів безпеки, фіксації подій та реагуванні на відомі сценарії атак [1, 2, 5].

У хмарному середовищі традиційні засоби особливо часто зіштовхуються з трьома обмеженнями. Перше – висока динамічність інфраструктури: ресурси створюються, масштабуються і змінюються швидше, ніж адміністратор встигає коригувати правила. Друге – зміщення атак із технічного рівня на поведінковий: порушник дедалі частіше використовує валідні облікові дані. Третє – великий обсяг подій, які необхідно аналізувати в реальному часі, що перевищує можливості ручної експертизи [3, 6].

Таблиця 1.1 – Порівняння традиційних та інтелектуальних підходів до виявлення несанкціонованого доступу

Підхід	Переваги	Основні обмеження	Доцільність для задачі
ACL / RBAC / ABAC	Чітко регламентують права доступу; добре інтегруються з IAM	Не виявляють поведінкові аномалії; залежать від якості налаштування	Потрібні як базовий рівень, але не вирішують задачу самостійно
IP-фільтрація, MFA, мережеві обмеження	Зменшують ризик компрометації; підвищують бар'єр для зловмисника	Обходяться при компрометації токенів або довірених пристроїв	Потрібні як превентивні засоби
Сигнатурні IDS/IPS	Ефективні проти відомих атак; зрозумілі правила спрацювання	Погано виявляють нові, повільні та комбіновані сценарії	Обмежено придатні для теми
UEBA та ML-моделі	Виявляють відхилення від норми; адаптуються до змін поведінки; підтримують ризик-скоринг	Потребують якісних даних, налаштування ознак і контролю false positives	Найбільш придатні для інтелектуальної системи

Дані таблиці 1.1 свідчать, що традиційні засоби безпеки формують необхідний фундамент захисту, однак у задачі виявлення прихованого або поведінково замаскованого несанкціонованого доступу вирішальне значення мають інтелектуальні моделі, здатні працювати з історичним контекстом, профілями користувачів і багатовимірними ознаками [2, 3, 6].

1.4 Інтелектуальні методи, машинне навчання та UEBA у задачах виявлення аномалій

Інтелектуальні підходи до виявлення несанкціонованого доступу орієнтовані на аналіз не лише окремої події, а й поведінкових закономірностей у часі. Концепція UEBA (User and Entity Behavior Analytics) передбачає побудову профілів нормальної активності користувачів, сервісних облікових записів і технічних сутностей, а потім – виявлення значущих відхилень від цих профілів.[2, 9, 10].

Для реалізації UEBA та ML-підходів використовуються структуровані та напівструктуровані дані: логи доступу до сховища, події автентифікації, журналювання операцій читання та модифікації, інформація про геолокацію, user-agent, тип пристрою, час виконання запиту, ознаки мережевої активності, серії невдалих входів, обсяг переданих даних тощо. [6–8].

В літературі описано кілька груп моделей, придатних для цієї задачі. До supervised-підходів належать логістична регресія, метод опорних векторів, дерева рішень, Random Forest, градієнтний бустинг та нейронні мережі. Вони ефективні за наявності розміченого набору даних і дозволяють безпосередньо класифікувати події як нормальні чи аномальні. До unsupervised та semi-supervised підходів відносять кластеризацію, One-Class моделі, Isolation Forest та автоенкодера, які особливо корисні тоді, коли аномальні приклади рідкісні або погано розмічені [6–10].

Наукові огляди останніх років підкреслюють, що найкращі результати дають не ізольовані алгоритми, а комбіновані схеми, де ML-модель поєднується з

правилами доменної логіки, ризик-скорингом і механізмами пояснення рішень. Саме гібридна логіка покладена в основу практичної реалізації, розробленої в межах цієї дипломної роботи: система використовує структуровані логи, модуль попередньої обробки, виділення ознак, ML-компонент для виявлення аномалій та окремий блок оцінювання ризику, який формує рекомендацію щодо реагування [6–10].

1.5 Порівняльний аналіз існуючих рішень і наукових підходів

Аналіз доступних джерел дозволяє виділити чотири основні групи підходів до виявлення несанкціонованого доступу у хмарних середовищах. Перша група – класичні політики доступу та IAM-механізми. Друга – сигнатурні IDS/IPS і SIEM-підходи. Третя – UEBA-рішення, орієнтовані на поведінковий аналіз. Четверта – ML- та DL-моделі, які виконують класифікацію або аномалійний аналіз на основі набору ознак [2, 3, 6–10].

З практичної точки зору класичні IAM-засоби залишаються обов’язковими, однак їхня функція переважно превентивна: вони обмежують доступ, але не завжди здатні вчасно виявити атипову поведінку користувача з валідними правами. Сигнатурні системи зручні для експлуатації та добре інтегруються в процес моніторингу, однак залежать від наявності відомого шаблону загрози. Наукові праці, присвячені аномалійному виявленню в хмарі, навпаки, демонструють переваги аналізу поведінки, однак часто стикаються з проблемами якості датасетів, інтерпретованості моделей та високого рівня false positives [6–8].

Окремий інтерес становлять підходи, які поєднують класифікацію подій, аномалійний аналіз та ризик-скоринг. У таких системах результат моделі не є фінальним рішенням: він виступає одним із факторів у механізмі прийняття рішень, де додатково враховуються критичність операції, роль суб’єкта, наявність MFA, новий контекст входу, інтенсивність звернень і наслідки потенційного інциденту.

Для задачі, що розглядається в цій роботі, саме гібридний підхід є найбільш доцільним, оскільки він поєднує адаптивність ML із керованістю та зрозумілістю правил реагування [2, 3, 9, 10].

1.6 Обґрунтування актуальності теми, мета, об'єкт, предмет і завдання дослідження

Актуальність теми зумовлена тим, що в умовах широкого впровадження хмарних технологій дедалі більше організацій зберігають у хмарних сховищах таємну, конфіденційну та службову інформацію. Водночас зростає і кількість загроз, пов'язаних з несанкціонованим доступом до таких ресурсів. Традиційні засоби захисту – фільтрація IP-адрес, статичні правила доступу, сигнатурні механізми – не завжди забезпечують своєчасне виявлення комбінованих, розподілених або раніше невідомих атак. [2, 3, 5, 10].

Метою дипломної роботи є розробка інтелектуальної системи прийняття рішень для виявлення несанкціонованого доступу до хмарного сховища даних із використанням методів машинного навчання.

Об'єктом дослідження є процеси доступу користувачів та сервісів до хмарного сховища даних у корпоративному середовищі. Предметом дослідження є методи та моделі машинного навчання, поведінкової аналітики та інтелектуальної підтримки прийняття рішень.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання: проаналізувати існуючі підходи до виявлення несанкціонованого доступу в хмарних сховищах; дослідити алгоритми машинного навчання, придатні для виявлення потенційно шкідливої активності; зібрати та підготувати датасет для навчання моделі; побудувати модель машинного навчання для виявлення аномалій у поведінці користувачів; реалізувати систему, що інтегрує модель із механізмом

прийняття рішень; провести тестування та оцінювання ефективності системи; підготувати документацію розробленого програмного забезпечення.

Практичне значення роботи полягає у створенні програмної системи, яка може застосовуватися як модуль платформи інформаційної безпеки для організацій, що використовують хмарні сховища для роботи з критичною або чутливою інформацією.

Висновки до розділу 1

У першому розділі досліджено хмарні сховища даних як об'єкт захисту та показано, що в сучасних умовах вони виступають не лише місцем зберігання інформації, а комплексним середовищем взаємодії даних, сервісів, API, механізмів автентифікації та адміністративних засобів.

Проаналізовано основні загрози безпеці хмарних сховищ, зокрема компрометацію облікових даних, помилки конфігурації, внутрішні загрози, зловживання API та комбіновані сценарії атак. Встановлено, що значна частина сучасних інцидентів маскується під легітимну активність користувачів.

Показано обмеження традиційних засобів контролю доступу та сигнатурних систем виявлення атак. Обґрунтовано доцільність використання інтелектуальних підходів, заснованих на машинному навчанні та UEBA, які дозволяють будувати профілі нормальної поведінки та виявляти відхилення у реальному часі.

За результатами порівняльного аналізу існуючих рішень і наукових підходів встановлено, що найбільш перспективним є гібридний підхід, який поєднує збір і попередню обробку подій доступу, виділення ознак, ML-аналіз, оцінювання ризику та формування рішення щодо реагування. Саме такий підхід покладено в основу подальшого проектування і програмної реалізації системи.

2 МЕТОДИ ТА МОДЕЛІ ПОБУДОВИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ

2.1 Формалізація задачі виявлення несанкціонованого доступу до хмарного сховища

У межах роботи задача виявлення несанкціонованого доступу до хмарного сховища формалізується як задача інтелектуального аналізу подій доступу користувачів та сервісів до ресурсів хмарного середовища з подальшим віднесенням кожної події або сукупності подій до одного з двох базових станів: нормальна активність або потенційно небезпечна активність. На відміну від класичних механізмів контролю доступу, що працюють переважно за наперед визначеними правилами, у запропонованій системі рішення формується на основі поєднання ознак поведінки суб'єкта, контексту доступу та вихідних оцінок моделей машинного навчання [1–3, 11].

Нехай $E = \{e_1, e_2, \dots, e_n\}$ – множина подій доступу до хмарного сховища, де кожна подія e_i описується кортежем атрибутів: ідентифікатором користувача або сервісу, часовою міткою, IP-адресою, геолокацією, типом пристрою, способом автентифікації, видом операції над даними, обсягом переданих даних, наявністю попередніх невдалих спроб входу та іншими контекстними параметрами. Для кожної події необхідно визначити функцію прийняття рішень $D(e_i)$, результатом якої є рівень ризику та рекомендована дія реагування: журналювання, сповіщення, вимога повторної автентифікації або блокування [3, 6, 11].

У запропонованій реалізації передбачено два режими аналізу. У supervised-режимі використовується розмічений датасет подій доступу, на якому навчається модель класифікації. У unsupervised або semi-supervised режимі використовується аномалійна модель, що формує оцінку відхилення без жорсткої залежності від повної розмітки даних. Такий підхід забезпечує гнучкість системи й дає змогу

застосовувати її як до синтетичних, так і до реальних журналів доступу, де кількість розмічених інцидентів зазвичай є обмеженою [6, 7, 12, 13].

2.2 Архітектура інтелектуальної системи прийняття рішень

Архітектура інтелектуальної системи прийняття рішень сформована відповідно до принципу послідовного аналітичного конвеєра та охоплює такі функціональні рівні: рівень збору даних, рівень попередньої обробки, рівень формування ознак, рівень моделювання й аналітики, рівень оцінювання ризику та рівень реалізації рішень щодо реагування. Така побудова відповідає сучасним підходам UEBA та дозволяє послідовно перетворювати сирі події доступу на керовані рішення інформаційної безпеки [2, 6, 10, 17].

У кінцевій практичній реалізації, розробленій у межах дипломної роботи, система реалізована як вебзастосунок на Python із серверною логікою на FastAPI, модулем імпорту CSV-даних, блоком машинного навчання, механізмом серверного збереження результатів аналізу та інтерфейсом для фахівця з безпеки або адміністратора. На рівні інтерфейсу користувач має окремі сторінки огляду, поведінки, аномалій, ризику й реагування, інцидентів, оцінювання, джерел даних та документації, що забезпечує структуровану роботу з результатами аналізу.

Функціональна схема системи наведена на рисунку 2.1. Вона демонструє, що вхідними даними є журнали доступу, події автентифікації, а також контекстна інформація про геолокацію та агент доступу. Після попередньої обробки та виділення ознак ці дані надходять до ML-моделей класифікації або аномалійного аналізу. Результат моделі використовується для ризик-скорингу, на основі якого формується рішення щодо реагування. Така схема повністю узгоджується з логікою кінцевої практичної реалізації системи, у якій рішення не зводиться лише до бінарної класифікації, а включає механізм підтримки дій оператора [11–13].

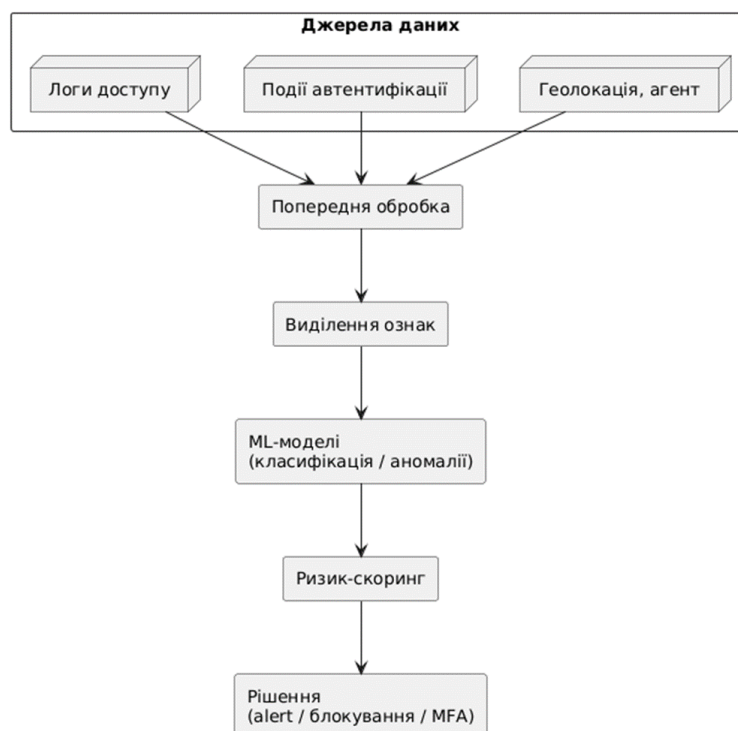


Рисунок 2.1 – Функціональна архітектура інтелектуальної системи прийняття рішень щодо виявлення несанкціонованого доступу

Застосування модульної архітектури має кілька переваг. По-перше, кожен компонент може розвиватися незалежно. По-друге, зберігається можливість інтеграції з зовнішніми джерелами подій або SIEM-платформами. По-третє, архітектура полегшує документування та тестування [5, 11].

2.3 Модель подій доступу та структура вхідних даних

Основою для роботи системи є модель подій доступу до хмарного сховища. Кожна подія описує завершену або частково завершену дію суб'єкта у хмарному середовищі. До таких дій належать вхід до системи, запит на отримання списку об'єктів, завантаження файлів, їх модифікація, видалення, зміна політик доступу або створення нових публічних посилань[7, 12–15].

У практичній реалізації використовується CSV-формат з уніфікованим набором полів. Структуру вхідних даних подано в таблиці 2.1.

Таблиця 2.1 – Структура вхідних даних для аналізу подій доступу

Поле	Призначення	Тип даних
timestamp	Час фіксації події доступу	datetime
user_id	Ідентифікатор користувача або сервісу	string
role	Роль суб'єкта доступу	category
country	Країна або регіон джерела запиту	category
device_id / agent	Тип або ідентифікатор пристрою, user-agent	string
auth_method	Спосіб автентифікації	category
mfa_used	Ознака використання багатофакторної автентифікації	binary
operation_type	Тип операції над даними або політиками	Category
bytes_transferred	Обсяг переданих даних	Numeric
request_count_window	Інтенсивність звернень за часовий інтервал	numeric
failed_attempts	Кількість невдалих спроб входу	numeric
new_device / new_geo	Ознаки нового контексту доступу	binary
label	Розмітка класу для supervised-аналізу	binary / optional

Модель подій доступу повинна забезпечувати сумісність як з реальними журналами доступу, так і з синтетичними наборами даних, створеними для тестування системи. Частина полів може бути факультативною, а механізм попередньої обробки містить процедури перевірки повноти, очищення, кодування категоріальних значень та приведення часових характеристик до єдиного формату. Така уніфікація є необхідною передумовою коректного формування поведінкових ознак і подальшого навчання моделей [6, 7, 17].

2.4 Формування поведінкових ознак користувачів і сервісів

Найважливішим етапом підготовки даних для машинного навчання є формування ознак, які відображають поведінковий профіль користувача або

сервісу. На відміну від сирих подій журналу, ознаки мають компактно й інформативно описувати характер доступу: його частоту, часову структуру, нетиповість контексту, критичність операції та історію попередніх подій. Для задачі виявлення несанкціонованого доступу саме поведінкові ознаки формують основу для UEBA-підходу [10, 16, 17].

У розробленій системі виділяються як статичні, так і динамічні ознаки. До статичних належать роль користувача, тип пристрою, спосіб автентифікації, використання MFA та базові атрибути події. До динамічних – кількість звернень за часовий інтервал, кількість невдалих входів перед успішним доступом, поява нового географічного контексту, робота в нетиповий час, нетиповий обсяг завантаження, повторюваність небезпечних операцій або послідовність аномальних дій. Система також враховує похідні ознаки, отримані шляхом агрегації кількох подій у межах короткого часового вікна. Основні ознаки наведено в таблиці 2.2.

Таблиця 2.2 – Основні поведінкові ознаки користувачів і сервісів

Група ознак	Приклади ознак	Аналітичне призначення
Часові	hour_of_day, day_of_week, after_hours	Виявлення нетипової активності у часі
Контекстні	new_country, new_device, unknown_agent	Виявлення нового або підозрілого контексту
Інтенсивнісні	request_count_10m, bytes_transferred	Виявлення сплесків активності й масового доступу
Автентифікаційні	failed_attempts, auth_method, mfa_used	Оцінювання компрометації облікового запису
Операційні	operation_type, risky_operation, policy_change	Аналіз критичності виконаних дій
Агреговані	user_risk_history, anomaly_burst	Оцінювання поведінки в часовому вікні

Практична реалізація системи передбачає формування ознак під час кожного циклу аналізу. Такий підхід дає змогу виконувати однакову обробку як для

тестових датасетів, так і для нових файлів, що імпортуються користувачем через вебінтерфейс.

2.5 Вибір і обґрунтування алгоритмів машинного навчання

Вибір алгоритмів має ґрунтуватися на кількох критеріях: здатність працювати з табличними даними та змішаними ознаками, стійкість до шуму, інтерпретованість, придатність до задач з дисбалансом класів і можливість використання як у supervised-, так і в anomaly-режимі. У зв'язку з цим у роботі обрано комбінований підхід, що поєднує класифікаційну модель Random Forest та модель аномалійного аналізу Isolation Forest [12, 13].

Алгоритм Random Forest є ансамблевим методом, який формує множину дерев рішень та використовує їх сукупне голосування для класифікації об'єктів. Його перевагами для даної задачі є здатність працювати з неоднорідними ознаками, відносна стійкість до перенавчання, підтримка оцінювання важливості ознак і добрі результати на табличних датасетах з помірною кількістю параметрів [12]. У межах практичної реалізації саме цей алгоритм використовується як основна supervised-модель, коли датасет містить цільову розмітку подій.

Алгоритм Isolation Forest, навпаки, орієнтований на виявлення відхилень без повної залежності від розмічених аномалій. Його ідея полягає в тому, що аномальні спостереження ізолюються в деревоподібній структурі швидше за нормальні, а тому можуть бути ідентифіковані за середньою довжиною шляху ізоляції [13].

Додатково для оцінювання ефективності системи використовується rule-based baseline – набір базових евристичних правил, які враховують критичні операції, нову геолокацію, значну кількість невдалих спроб входу та відсутність MFA. Порівняння з таким baseline потрібне не лише для технічного аналізу, а й для демонстрації переваг інтелектуальної моделі над суто правилорічним підходом, що є

важливим з точки зору обґрунтування доцільності використання машинного навчання [3, 10, 11].

Таблиця 2.3 – Порівняння підходів, використаних у системі

Алгоритм / підхід	Призначення	Переваги	Обмеження
Rule-based baseline	Первинна евристична оцінка ризику на основі заданих правил	Простота, прозорість, контрольованість, зручність для порівняння з ML-моделями	Низька адаптивність до нових сценаріїв, залежність від вручну заданих правил
Random Forest	Класифікація нормальних і потенційно аномальних подій за наявності розміченого датасету	Добре працює з табличними даними, стійкий до шуму, дозволяє оцінювати важливість ознак	Потребує розмічених даних, якість залежить від репрезентативності навчальної вибірки
Isolation Forest	Виявлення відхилень у режимі неповної або відсутньої розмітки	Придатний для anomaly-режиму, може працювати з рідкісними інцидентами, не потребує повної розмітки	Менша інтерпретованість порівняно з правилами, потребує налаштування порогу аномальності
TensorFlow Autoencoder	Нейромережеве виявлення нетипових подій за помилкою реконструкції	Дає змогу виявляти складні нелінійні відхилення, підходить для глибокого аналізу поведінкових патернів	Потребує нормалізації даних, більш складний у налаштуванні, менш прозорий для пояснення рішень

Отже, у межах роботи обґрунтовано доцільність використання гібридної моделі аналізу подій доступу. Для сценаріїв із наявною розміткою використовується supervised-класифікація на основі Random Forest, що забезпечує класифікацію подій на нормальні та потенційно аномальні. Для недостатньо

розмічених наборів даних застосовується Isolation Forest, який дає змогу виявляти відхилення без повної попередньої розмітки. Додатково у практичній реалізації використано TensorFlow Autoencoder як неймережевий підхід до виявлення нетипових подій за помилкою реконструкції. PyTorch розглянуто як альтернативний фреймворк для побудови неймережевих моделей, однак у фінальній реалізації обрано TensorFlow, щоб уникнути дублювання функціональності та зосередити неймережевий компонент в одному технологічному стеку.

2.6 Модель оцінювання ризику та механізм прийняття рішень щодо реагування

Результат роботи ML-моделі не є остаточним управлінським рішенням. Для перетворення технічної оцінки на дію системи безпеки застосовується модель оцінювання ризику. У найзагальнішому вигляді інтегральний ризик події визначається як комбінація трьох компонентів: оцінки моделі машинного навчання, оцінки правил доменної логіки та оцінки контекстної критичності події. У спрощеному вигляді це можна подати співвідношенням:

$$RiskScore = w^1 \cdot SML + w^2 \cdot SRules + w^3 \cdot SContext, \quad (2.1)$$

де SML – оцінка аномальності або ймовірність належності події до небезпечного класу, $SRules$ – результат правилкової перевірки, $SContext$ – додаткова контекстна оцінка критичності дії, а w_1, w_2, w_3 – вагові коефіцієнти. [10, 11, 16].

Механізм прийняття рішень у системі побудований за багаторівневою шкалою ризику. Низький рівень ризику означає лише журналювання події. Середній рівень передбачає сповіщення аналітика або адміністратора. Високий рівень може ініціювати додаткову перевірку особи, зокрема вимогу повторної автентифікації або MFA. Критичний рівень ризику асоціюється з блокуванням сесії, тимчасовим відключенням токена або фіксацією інциденту [11].

Таблиця 2.4 – Шкала оцінювання ризику та дії реагування

Діапазон RiskScore	Рівень ризику	Рекомендована дія
0–30	Низький	Журналювання події без активного втручання
31–60	Середній	Alert аналітику або адміністратору
61–80	Високий	Примусова повторна автентифікація / MFA
81–100	Критичний	Блокування сесії або токена, реєстрація інциденту

На рівні програмної реалізації блок оцінювання ризику формує не лише числове значення, а й пояснювальний опис причин спрацювання.

Висновки до розділу 2

Сформовано модель подій доступу та визначено структуру вхідних даних, придатну для опрацювання засобами Python, Pandas і Scikit-learn. Обґрунтовано: доцільність використання поведінкових ознак, що враховують часовий, контекстний, автентифікаційний та операційний аспекти активності користувачів і сервісів; вибір гібридного підходу до машинного навчання, у якому поєднуються supervised-модель Random Forest, аномалійна модель Isolation Forest та rule-based baseline. Запропоновано модель інтегрального ризик-скорингу та визначено механізм переходу від технічного результату моделі до операційних рішень щодо реагування.

3 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

3.1 Загальна характеристика розробленого програмного продукту

У межах дипломної роботи розроблено веборієнтовану інтелектуальну систему CloudGuard ML, призначену для аналізу подій доступу до хмарного сховища даних, виявлення потенційно небезпечної активності та підтримки прийняття рішень щодо реагування на інциденти. Система не обмежується статистичним переглядом журналів доступу, а реалізує повний цикл опрацювання подій: імпорт CSV-джерела, нормалізацію даних, формування поведінкових ознак, застосування моделей машинного навчання, оцінювання ризику, збереження результатів і подання аналітичної інформації через вебінтерфейс.

Розроблений програмний продукт орієнтовано на сценарії, у яких несанкціонований доступ може маскуватися під легітимну активність користувача або сервісного облікового запису. Саме тому в системі враховуються не лише формальні атрибути операції, а й контекст доступу: роль суб'єкта, тип пристрою, країна, ознака нового пристрою або нового географічного контексту, використання MFA, кількість невдалих спроб входу, обсяг переданих даних, інтенсивність запитів за часове вікно, вік токена та тип операції над об'єктом.

Кінцева реалізація побудована на Python. Для серверної частини використано FastAPI, оскільки цей фреймворк призначений для створення API на основі стандартних Python type hints і підтримує продуктивну розробку вебзастосунків [18]. Для шаблонізації інтерфейсу застосовано Jinja2, що дає змогу формувати HTML-сторінки на основі шаблонів і переданих із серверної частини даних [26]. Збереження користувачів, історії запусків і метаданих результатів виконано у PostgreSQL, який є об'єктно-реляційною системою керування базами даних [20].

У таблиці 3.1 наведено основні функціональні можливості реалізованої системи.

Таблиця 3.1 – Основні функціональні можливості системи CloudGuard ML

Функціональний блок	Реалізована можливість	Практичне призначення
Авторизація	Вхід користувача, реєстрація аналітика, автоматичне створення адміністратора, розмежування ролей admin і analyst	Обмеження доступу до аналітичної панелі та адміністративних функцій
Джерела даних	Вибір підготовленого CSV-набору або завантаження власного CSV-файлу	Гнучке тестування системи на різних сценаріях доступу
Попередня обробка	Перевірка обов'язкових колонок, нормалізація назв, типів і часових атрибутів	Підготовка журналів доступу до єдиного формату аналізу
Машинне навчання	Застосування Random Forest, Isolation Forest і TensorFlow Autoencoder	Виявлення аномальної поведінки у supervised та anomaly-режимах
Risk engine	Обчислення risk score, рівня ризику, пояснення причин і рекомендованої дії	Підтримка прийняття рішень щодо реагування
Візуалізація	Окремі сторінки огляду, поведінки, аномалій, ризику, інцидентів та оцінювання	Надання аналітику структурованої картини стану доступу

З таблиці 3.1 видно, що розроблена система охоплює не один ізольований алгоритм, а комплексну програмну реалізацію з повним циклом роботи з даними. Такий підхід відповідає поставленій меті дипломної роботи, оскільки дозволяє не лише виявляти аномалії, а й формувати операційно придатні рекомендації для адміністратора або фахівця з інформаційної безпеки.

3.2 Функціональні та нефункціональні вимоги до системи

Перед розробленням програмної реалізації визначено вимоги, які впливають із предметної області та результатів аналізу, виконаного в попередніх розділах.

У таблиці 3.2 наведено узагальнений перелік вимог до розробленої інтелектуальної системи.

Таблиця 3.2 – Вимоги до програмної системи

Тип вимоги	Вимога	Реалізація у системі
Функціональна	Система повинна імпортувати журнали доступу у форматі CSV	Реалізовано сторінку джерел даних і завантаження файлів через /sources/upload
Функціональна	Система повинна перевіряти структуру вхідного файлу	Модуль feature_engineering перевіряє наявність обов'язкових колонок
Функціональна	Система повинна формувати поведінкові ознаки	Реалізовано 20 ознак, зокрема role_privilege, operation_risk, bytes_log, unusual_hour
Функціональна	Система повинна виявляти аномальні або підозрілі події	Застосовано Random Forest, Isolation Forest та Autoencoder
Функціональна	Система повинна формувати рекомендацію щодо реагування	Risk engine повертає рівень ризику та дію: журналювання, alert, MFA або блокування
Нефункціональна	Система повинна зберігати результати запусків	Результати та метрики зберігаються у PostgreSQL і CSV-експорті
Нефункціональна	Система повинна бути відтворюваною під час запуску	Підтримано Docker Compose з окремими сервісами web, db і adminer
Нефункціональна	Система повинна мати зрозумілий інтерфейс	Інтерфейс поділено на тематичні сторінки за аспектами аналізу

Сформульовані вимоги визначили структуру програмного продукту. Основними акцентами стали відтворюваність запуску, модульність аналітичного

конвеєра, можливість роботи з різними наборами даних та пояснюваність результатів для користувача.

3.3 Сценарії використання системи та ролі користувачів

Розроблена система передбачає дві основні ролі користувачів: адміністратора та аналітика. Адміністратор має доступ до панелі адміністрування, може переглядати користувачів і журнал запусків, а також створювати нових користувачів. Аналітик працює з джерелами даних, запускає аналіз, переглядає результати, вивчає поведінкові профілі, аномалії, ризикові події та метрики якості моделей.

Основний сценарій використання складається з кількох послідовних кроків. Користувач входить у систему, вибирає одне з підготовлених джерел або завантажує власний CSV-файл, запускає аналіз, після чого система автоматично виконує підготовку даних, навчання або застосування моделей, оцінювання ризику та збереження результатів. Далі користувач переходить між сторінками аналітичного інтерфейсу.

У таблиці 3.3 наведено основні ролі та доступні для них функції.

Таблиця 3.3 – Ролі користувачів і доступні функції

Роль	Доступні функції	Обмеження
Адміністратор	Вхід у систему, запуск аналізу, перегляд результатів, доступ до панелі адміністрування, перегляд користувачів, створення нових користувачів, перегляд повної історії запусків	Не передбачено редагування моделей без зміни програмного коду
Аналітик	Вхід у систему, вибір або завантаження CSV-джерела, запуск аналізу, перегляд сторінок огляду, поведінки, аномалій, ризику, інцидентів, оцінювання та джерел даних	Немає доступу до адміністративної панелі та створення користувачів

Розмежування ролей реалізовано на рівні серверної логіки. Для адміністративної сторінки перевіряється роль активного користувача, а за відсутності прав виконується перенаправлення на головну сторінку. Це спрощує модель доступу та водночас забезпечує базову ізоляцію адміністративних функцій.

3.4 Загальна архітектура програмної реалізації

Архітектура системи побудована за модульним принципом. Серверна частина координує роботу з користувацькими запитами, джерелами даних, моделями машинного навчання, механізмом оцінювання ризику, базою даних і шаблонами вебінтерфейсу. Така структура дає змогу ізолювати різні аспекти системи та спростити подальше розширення програмного продукту.

На рисунку 3.1 наведено загальну компонентну архітектуру розробленої системи.

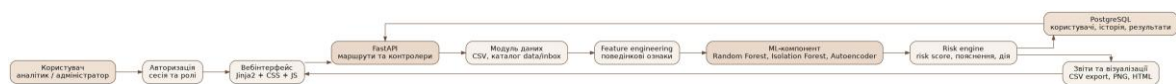


Рисунок 3.1 – Компонентна архітектура системи CloudGuard ML

Як видно з рисунка 3.1, центральним елементом системи є серверний застосунок FastAPI. Він приймає запити від вебінтерфейсу, викликає модулі роботи з даними, ініціює аналітичний конвеєр, зберігає результати у PostgreSQL та повертає користувачеві структуровані сторінки аналізу. База даних використовується не для зберігання всього обсягу журналів доступу, а для збереження користувачів, історії запусків і метаданих активних результатів. Деталізований результат аналізу додатково експортується у CSV-файл.

Технологічний стек системи подано у таблиці 3.4.

Таблиця 3.4 – Технологічний стек програмної реалізації

Компонент	Технологія	Призначення
Мова програмування	Python 3.11	Реалізація серверної логіки, аналітичних модулів і ML-компонентів
Вебфреймворк	FastAPI	Обробка HTTP-запитів, маршрутизація, API та інтеграція з шаблонами [18]
ASGI-сервер	Uvicorn	Запуск FastAPI-застосунку у контейнері
Шаблонізатор	Jinja2	Формування HTML-сторінок вебінтерфейсу [26]
База даних	PostgreSQL	Збереження користувачів, історії запусків і метаданих результатів [20]
Доступ до БД	SQLAlchemy, psycopg2	Створення підключення, виконання SQL-запитів і транзакцій [19]
Обробка даних	pandas, numpy	Читання CSV, нормалізація, перетворення даних і побудова ознак [22]
Машинне навчання	scikit-learn, TensorFlow CPU	Класифікація, anomaly detection, Autoencoder і метрики якості [23, 24]
Розгортання	Docker Compose	Відтворюваний запуск вебсервісу, БД і Adminer [21]

Вибір стеку обґрунтовано тим, що FastAPI забезпечує зручну побудову серверної частини, pandas і scikit-learn є придатними для табличних ML-задач, PostgreSQL забезпечує надійне збереження даних, а Docker Compose спрощує повторюване розгортання багатокomпонентного застосунку [18, 20–23].

3.5 Організація серверної частини системи на основі FastAPI

Серверна частина реалізована у файлі app.py. Вона відповідає за ініціалізацію застосунку, створення необхідних каталогів, підключення статичних файлів, роботу з шаблонами, обробку маршрутів і запуск аналітичного конвеєра. Під час старту застосунку виконується bootstrap-процедура: створюються каталоги data, data/inbox, data/exports, models і static/reports, ініціалізується база даних,

генеруються тестові набори даних і створюється початковий адміністратор, якщо у базі ще немає користувачів.

FastAPI використовується як основний вебфреймворк. Його застосування є доцільним для цієї роботи, оскільки система має поєднувати HTML-інтерфейс, обробку форм, завантаження файлів, API-запити та маршрути для отримання результатів. [18].

У таблиці 3.5 наведено основні маршрути серверної частини.

Таблиця 3.5 – Основні маршрути вебзастосунку

Маршрут	Метод	Призначення
/	GET	Відображення головної панелі та запуску аналізу
/login	GET / POST	Відображення форми входу та автентифікація користувача
/register	GET / POST	Реєстрація користувача з роллю analyst
/sources	GET	Перегляд доступних CSV-джерел і вимог до схеми
/sources/upload	POST	Завантаження користувацького CSV-файлу
/analyze	POST	Запуск повного циклу аналізу вибраного або завантаженого джерела
/analysis/overview	GET	Підсумкова картина активного результату
/analysis/behavior	GET	Аналіз поведінкових профілів
/analysis/anomalies	GET	Перегляд аномалій, тригерів і ризикових користувачів
/analysis/risk	GET	Матриця ризику та рекомендовані дії
/analysis/incidents	GET	Таблиця підозрілих подій
/analysis/evaluation	GET	Оцінювання моделей і метрики якості
/admin	GET	Адміністративна панель для користувачів і журналу запусків
/api/summary	GET	JSON-представлення активного підсумку аналізу

Маршрут `/analyze` є центральним для практичної роботи системи. Він визначає джерело даних, викликає завантаження та нормалізацію подій, ініціює навчання або застосування моделей, запускає блок `risk engine`, зберігає результат у CSV, записує історію запуску та робить поточний результат активним у сесії користувача.

3.6 Організація бази даних та збереження результатів аналізу

Для збереження службових даних використано PostgreSQL. Офіційна документація визначає PostgreSQL як потужну об'єктно-реляційну систему керування базами даних, що використовує та розширює SQL [20]. У межах розробленої системи база даних не перевантажується повними журналами доступу, а зберігає саме ті сутності, які потрібні для керування користувачами, історією запусків і відновленням активного результату.

Для підключення до бази використано SQLAlchemy. У документації SQLAlchemy ORM описано як інструмент для роботи з високорівневими SQL-конструкціями та автоматизованою персистентністю Python-об'єктів [19]. У розробленому застосунку використано підхід з `create_engine`, транзакційними блоками `begin` та виконанням SQL-запитів через `text`, що є достатнім для невеликої прикладної системи.

На рисунку 3.2 наведено модель даних серверної частини.

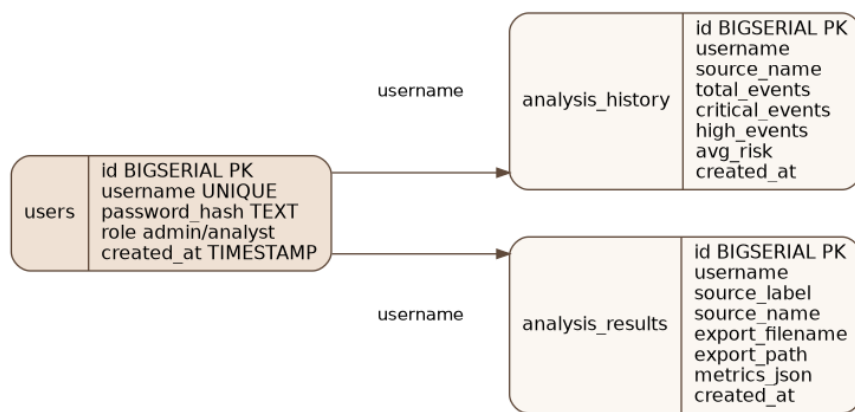


Рисунок 3.2 – Модель даних серверної частини системи

Як показано на рисунку 3.2, база даних містить три основні таблиці. Таблиця `users` зберігає облікові записи, хеші паролів і ролі користувачів. Таблиця `analysis_history` фіксує коротку історію запусків: користувача, джерело даних, кількість подій, кількість критичних і високих подій, середній ризик і дату створення запису. Таблиця `analysis_results` містить метадані активних результатів, шлях до CSV-експорту та JSON із метриками моделі.

Структуру основних таблиць наведено в таблиці 3.6.

Таблиця 3.6 – Структура таблиць бази даних

Таблиця	Основні поля	Призначення
<code>users</code>	<code>id</code> , <code>username</code> , <code>password_hash</code> , <code>role</code> , <code>created_at</code>	Збереження користувачів, ролей і даних автентифікації
<code>analysis_history</code>	<code>id</code> , <code>username</code> , <code>source_name</code> , <code>total_events</code> , <code>critical_events</code> , <code>high_events</code> , <code>avg_risk</code> , <code>created_at</code>	Формування журналу запусків аналізу
<code>analysis_results</code>	<code>id</code> , <code>username</code> , <code>source_label</code> , <code>source_name</code> , <code>export_filename</code> , <code>export_path</code> , <code>metrics_json</code> , <code>created_at</code>	Збереження активних результатів, метрик і посилань на CSV-експорт

Паролі користувачів не зберігаються у відкритому вигляді. Для них формується хеш із сіллю на основі PBKDF2-HMAC-SHA256. Під час автентифікації новий хеш порівнюється з наявним значенням через `hmac.compare_digest`, що знижує ризик некоректного порівняння рядків. Така реалізація є базовою, але достатньою для навчального прототипу, у якому основна увага приділена ML-аналізу подій доступу.

3.7 Реалізація модуля імпорту та попередньої обробки CSV-даних

Вхідними даними системи є CSV-файли з журналами доступу. Для роботи з табличними даними використано `pandas`. Відповідно до офіційної документації, функція `read_csv` призначена для читання CSV-файлів у структуру `DataFrame`, що є зручним форматом для подальшої обробки та аналізу [22].

Модуль `feature_engineering.py` містить функції `load_events` і `normalize_input`. Перша функція читає CSV-джерело, а друга приводить його до уніфікованого представлення. На початку всі назви колонок переводяться до нижнього регістру і очищуються від зайвих пробілів. Після цього перевіряється наявність обов'язкового набору полів. Якщо хоча б однієї обов'язкової колонки немає, система зупиняє імпорт і повертає користувачеві повідомлення про помилку.

Обов'язкові та необов'язкові поля вхідного CSV-файлу наведено в таблиці 3.7.

Таблиця 3.7 – Поля вхідного CSV-файлу

Тип поля	Назви полів	Призначення
Обов'язкові	timestamp, user_id, role, country, device_type, device_trust, operation, mfa_used, bytes_transferred, failed_attempts, is_new_device, is_new_country, session_duration_sec, requests_last_10m	Базові атрибути події доступу, необхідні для аналізу поведінки
Необов'язкові	source_provider, cloud_region, bucket_name, object_key, api_endpoint, auth_method, service_account, token_age_min, user_agent	Додатковий cloud-контекст для точнішого аналізу хмарного середовища
Цільова мітка	label	Використовується для supervised-оцінювання, якщо у наборі є обидва класи

Якщо необов'язкові cloud-поля відсутні, система автоматично додає значення за замовчуванням: local для провайдера, local-bucket для сховища, unknown-api для API-методу, unknown-client для user-agent тощо. Такий підхід дозволяє аналізувати як розширені набори, що імітують AWS S3, Azure Blob Storage або Google Cloud Storage, так і спрощені CSV-файли з базовими подіями доступу.

3.8 Реалізація модуля формування поведінкових ознак

Після нормалізації даних система переходить до формування ознак. Цей етап є критичним, оскільки ML-модель не працює безпосередньо з описовими рядками журналу, а потребує числового представлення поведінкових, часових, контекстних і операційних характеристик. У розробленому модулі виділено 20 основних ознак, які утворюють матрицю для моделей машинного навчання.

На рисунку 3.3 наведено послідовність обробки даних від CSV-файлу до збереження результату аналізу.



Рисунок 3.3 – Послідовність обробки даних у системі CloudGuard ML

На рисунку 3.3 показано, що формування ознак виконується після перевірки структури вхідного файлу та нормалізації даних. У модулі `feature_engineering.py` реалізовано мапінг ролей до рівня привілеїв, операцій до рівня ризику, типу довіри до пристрою до числового коефіцієнта, а також додаткові ознаки.

У таблиці 3.8 наведено приклади ознак, які використовуються в системі.

Таблиця 3.8 – Поведінкові ознаки, сформовані у програмній системі

Ознака	Спосіб формування	Аналітичне значення
role_privilege	Мапінг ролей viewer, analyst, manager, admin у шкалу 1–4	Оцінює рівень повноважень суб'єкта доступу
operation_risk	Мапінг типів операцій login, download, upload, delete, change_policy у числові ризику	Враховує критичність дії над даними або політиками
device_trust_score	Оцінка trusted, standard, unknown	Відображає довіру до пристрою
mfa_missing	1 - mfa_used	Фіксує відсутність багатофакторної автентифікації

Продовження таблиці 3.8

Ознака	Спосіб формування	Аналітичне значення
bytes_log	log1p(bytes_transferred)	Зменшує вплив надвеликих значень обсягу переданих даних
unusual_hour	Ознака доступу у 00:00–05:00 або 23:00	Виявляє нетиповий час активності
activity_intensity	requests_last_10m з урахуванням обсягу переданих даних	Оцінює сплески активності
access_context_risk	Комбінація нового пристрою, нової країни, MFA і довіри до пристрою	Оцінює ризиковість контексту входу

Завдяки такому набору ознак система враховує не тільки факт виконання певної операції, а й контекст її появи. Наприклад, завантаження файлу саме по собі може бути нормальним, однак завантаження великого обсягу даних у нетиповий час з нового пристрою та без MFA підвищує підсумковий рівень ризику.

3.9 Реалізація модуля машинного навчання та виявлення аномалій

ML-компонент реалізовано у модулі modeling.py. У ньому поєднано supervised-класифікацію, безнаглядне виявлення аномалій і нейромережевий підхід на основі автоенкодера.

Для supervised-режиму використано RandomForestClassifier. Згідно з документацією scikit-learn, Random Forest є ансамблевим метаоцінювачем, який будує множину дерев рішень на підвибірках і використовує усереднення для підвищення точності та контролю перенавчання [23]. У розробленій системі модель налаштовано з `n_estimators=180`, `max_depth=11`, `min_samples_leaf=2` і `class_weight="balanced"`, що враховує можливий дисбаланс нормальних і аномальних подій.

Для anomaly-режиму використано Isolation Forest. Цей підхід застосовується тоді, коли мітки класів відсутні або не є достатньо надійними. Додатково у системі

реалізовано TensorFlow Autoencoder. Офіційний навчальний матеріал TensorFlow подає автоенкодер як модель, придатну, зокрема, для задач anomaly detection [24].

Компоненти ML-модуля наведено в таблиці 3.9.

Таблиця 3.9 – Компоненти модуля машинного навчання

Компонент	Режим роботи	Результат
Random Forest	Supervised, якщо у CSV є колонка label з обома класами	Ймовірність належності події до небезпечного класу
Isolation Forest	Unsupervised / anomaly detection	Anomaly score для кожної події
TensorFlow Autoencoder	Нейромережеве anomaly detection за реконструкцією	Deep anomaly score на основі помилки реконструкції
Rule-based baseline	Евристична перевірка доменної логіки	Rule score для порівняння та підсилення risk engine
StratifiedKFold	Перехресна перевірка supervised-моделі	Середні значення accuracy, precision, recall і F1

У режимі supervised система розділяє дані на навчальну й тестову вибірки із застосуванням stratify, навчає Random Forest, обчислює hold-out метрики, формує матрицю помилок, виконує перехресну перевірку та порівнює результати з rule-based baseline, Isolation Forest і TensorFlow Autoencoder.

3.10 Реалізація модуля оцінювання ризику та формування рекомендацій

Модуль оцінювання ризику реалізовано у файлі risk_engine.py. Його призначення полягає у перетворенні технічних результатів ML-компонента на зрозумілий для користувача рівень ризику та рекомендовану дію реагування. Це важливо, оскільки саме по собі значення ймовірності або anomaly score не є управлінським рішенням.

У системі формується кілька проміжних оцінок: `ml_probability`, `anomaly_score`, `deep_anomaly_score`, `hybrid_anomaly_score` і `rule_score`. Підсумковий `risk_score` обчислюється як зважена комбінація ML-оцінки, гібридної аномальності та правилкової оцінки. Після цього подія отримує один із рівнів ризику: низький, середній, високий або критичний.

У таблиці 3.10 наведено шкалу прийняття рішень, реалізовану у системі.

Таблиця 3.10 – Шкала ризику та рекомендовані дії реагування

Діапазон <code>risk_score</code>	Рівень ризик	Рекомендована дія
0–39,99	Низький	Лише журналювання події без активного втручання
40–64,99	Середній	Створити сповіщення, посилити моніторинг і перевірити контекст доступу
65–84,99	Високий	Виконати примусову повторну автентифікацію та сповістити адміністратора
85–100	Критичний	Тимчасово заблокувати сесію, відкликати токен і зафіксувати інцидент

Окрім числової оцінки, система формує текстове пояснення причин спрацювання. До таких причин належать новий географічний контекст, новий пристрій, доступ без MFA, серія невдалих спроб входу, ризикована операція, висока інтенсивність запитів, великий обсяг переданих даних, тривалий вік токена або ризикова дія сервісного облікового запису. Це підвищує прозорість рішення та дає змогу аналітику швидше перевірити контекст події.

3.11 Реалізація вебінтерфейсу користувача та адміністратора

Вебінтерфейс реалізовано на основі HTML-шаблонів Jinja2, CSS-стилів і невеликого JavaScript-модуля. Jinja використовується як шаблонізатор, що приймає дані з серверної частини та формує HTML-представлення сторінок [26]. Інтерфейс

виконано у світлій бежево-коричневій палітрі, що узгоджується з наданими скриншотами та забезпечує візуальну цілісність аналітичної панелі.

На рисунку 3.4 наведено сторінку авторизації системи.

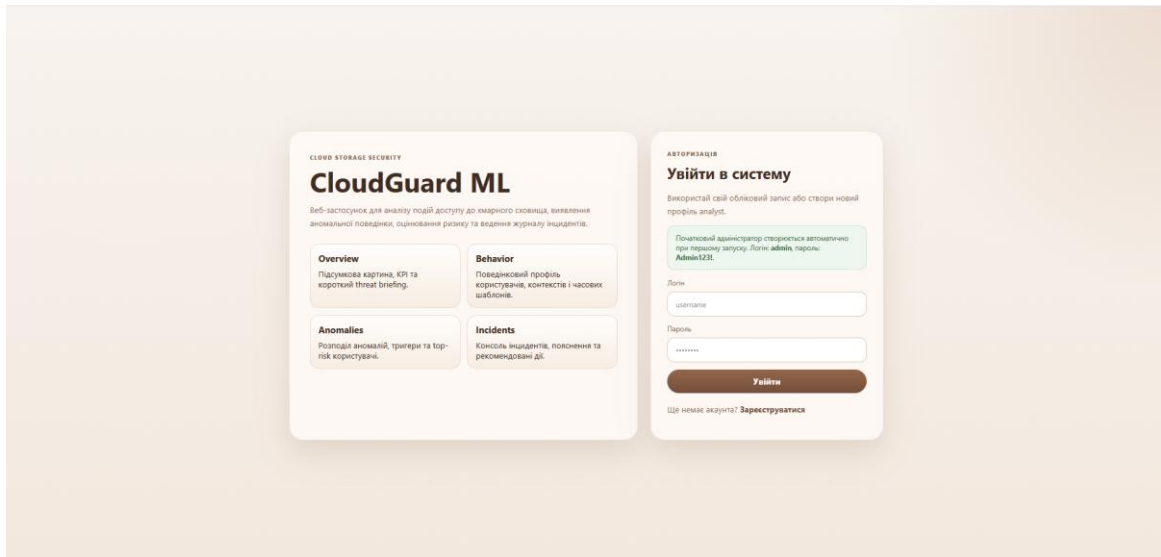


Рисунок 3.4 – Сторінка авторизації системи CloudGuard ML

Сторінка авторизації містить короткий опис призначення системи, блок основних функціональних напрямів і форму входу. Для першого запуску передбачено автоматичне створення адміністратора з обліковими даними, які задаються через змінні середовища. Після успішного входу ім'я користувача зберігається у сесії, а інтерфейс переходить до головної панелі.

На рисунку 3.5 наведено головну панель після виконання аналізу джерела credential_abuse.csv.

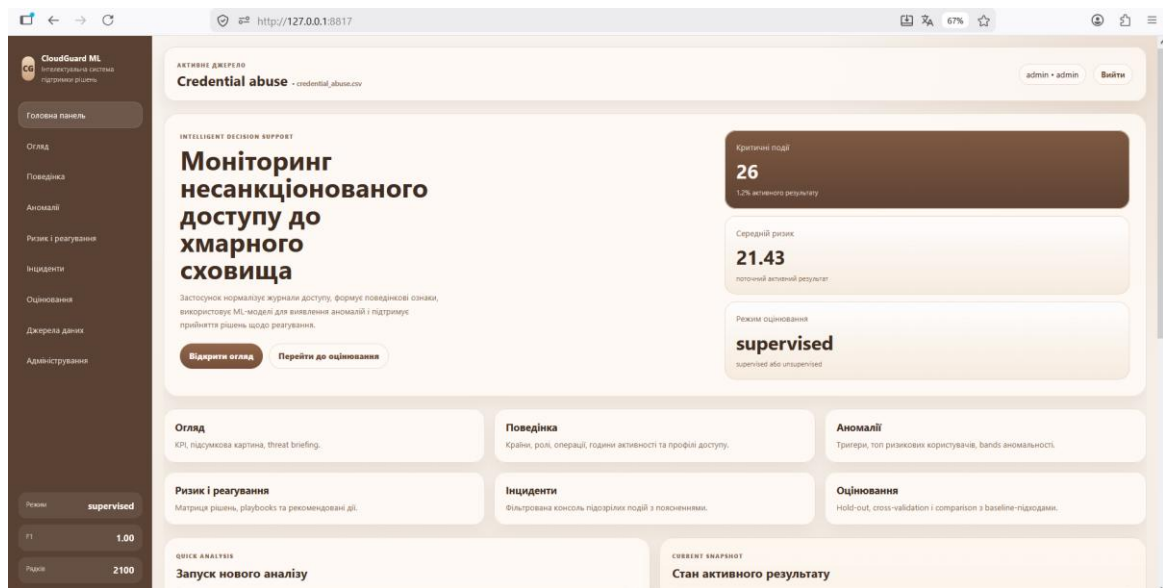


Рисунок 3.5 – Головна панель системи після аналізу джерела credential_abuse.csv

Головна панель відображає активне джерело, роль поточного користувача, режим оцінювання, F1-метрику, кількість рядків, ключові показники й переходи до окремих аспектів аналізу.

3.12 Контейнеризоване розгортання системи за допомогою Docker Compose

Для забезпечення відтворюваного запуску розроблену систему контейнеризовано. Docker Compose використовується для визначення та запуску багатоконтейнерних застосунків через єдиний YAML-файл; офіційна документація підкреслює, що Compose дає змогу керувати сервісами, мережами й томами конфігураційно [21].

У файлі docker-compose.yml визначено три сервіси: cloudguard-db, cloudguard-web і cloudguard-adminer. Перший сервіс запускає PostgreSQL 16-alpine, другий збирає вебзастосунок з Dockerfile і запускає Uvicorn, третій надає вебінтерфейс Adminer для адміністрування бази даних. Порт вебзастосунку проброшено як 8817:8000, PostgreSQL – як 55432:5432, Adminer – як 18081:8080.

На рисунку 3.6 наведено схему контейнеризованого розгортання системи.

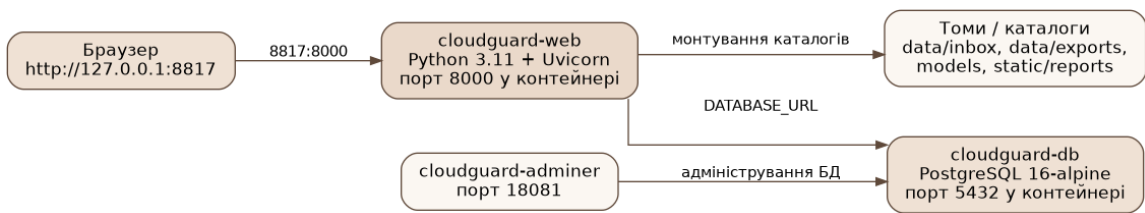


Рисунок 3.6 – Схема контейнеризованого розгортання системи CloudGuard ML

Як видно з рисунка 3.6, вебзастосунок взаємодіє з PostgreSQL через змінну середовища DATABASE_URL. Для збереження стану використовуються Docker volumes і змонтовані каталоги: data/inbox для вхідних CSV-файлів, data/exports для результатів аналізу, models для збережених моделей і static/reports для графічних або HTML-звітів. Така організація дозволяє перезапускати контейнери без втрати робочих даних.

Параметри розгортання наведено в таблиці 3.12.

Таблиця 3.12 – Параметри контейнеризованого розгортання

Сервіс	Образ / збірка	Порт	Призначення
cloudguard-web	Локальна збірка з Dockerfile на основі python:3.11-slim	8817:8000	Запуск вебзастосунку FastAPI через Uvicorn
cloudguard-db	postgres:16-alpine	55432:5432	Збереження користувачів, історії запусків і результатів
cloudguard-adminer	adminer:4	18081:8080	Перегляд і адміністрування PostgreSQL

Рекомендований запуск виконується командою `docker compose up --build`. Після старту система доступна за адресою `http://127.0.0.1:8817/login`. Наявність Docker Compose-конфігурації важлива для дипломної роботи, оскільки вона демонструє не лише локальну працездатність коду, а й можливість відтвореного розгортання системи в ізольованому середовищі.

Висновки до розділу 3

У третьому розділі розроблено та описано програмну реалізацію інтелектуальної системи CloudGuard ML для виявлення несанкціонованого доступу до хмарного сховища даних. Визначено функціональні й нефункціональні вимоги, ролі користувачів, сценарії використання та загальну архітектуру системи.

Реалізовано серверну частину на основі FastAPI, модуль авторизації, роботу з PostgreSQL, імпорт CSV-файлів, нормалізацію даних, формування поведінкових ознак, застосування Random Forest, Isolation Forest і TensorFlow Autoencoder, а також risk engine для формування рівня ризику, пояснення причин спрацювання і рекомендованої дії реагування.

Описано вебінтерфейс користувача та адміністратора, наведено сторінку авторизації, головну панель, структуру тематичних сторінок аналізу й схему контейнеризованого розгортання через Docker Compose. Результати розділу підтверджують, що створена система є не лише демонстраційною ML-моделлю, а завершеним веборієнтованим програмним продуктом, придатним для подальшого тестування, розширення та включення до повної дипломної роботи.

4 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Методика тестування програмного продукту

Тестування розробленої системи CloudGuard ML виконувалось з метою перевірки її відповідності функціональним вимогам, коректності оброблення вхідних даних, стабільності взаємодії серверної частини з базою даних, правильності роботи моделей машинного навчання та придатності інтерфейсу для підтримки прийняття рішень під час реагування на події несанкціонованого доступу. Оскільки система поєднує вебзастосунок, модуль імпорту CSV-даних, блок формування ознак, ML-компоненти, ризик-скоринг, журнал інцидентів і адміністративну частину, методика тестування була побудована як багаторівнева перевірка всього аналітичного конвеєра.

Під час планування перевірки враховувались підходи, характерні для тестування вебзастосунків і сервісів: перевірка маршрутизації, автентифікації, авторизації, оброблення форм, роботи з файлами, коректності API-взаємодії та збереження стану. Для автоматизованих перевірок доцільним є використання pytest, оскільки цей фреймворк підтримує компактні тестові функції, фікстури та масштабування тестового набору для складних застосунків [27]. Для перевірки ASGI-застосунку без запуску зовнішнього HTTP-сервера може застосовуватися HTTPX ASGI Transport, що дає змогу надсилати тестові запити безпосередньо до FastAPI-додатка [28]. Безпекові аспекти тестування узгоджуються з логікою OWASP Web Security Testing Guide, де вебзастосунки розглядаються через тестування конфігурації, ідентифікації, автентифікації, авторизації, керування сесіями, введення даних та API [31].

Загальну послідовність тестування розробленої системи подано на рисунку 4.1. Вона передбачає підготовку контрольованих CSV-наборів, запуск аналізу,

перевірку імпорту й формування ознак, оцінювання моделей, аналіз risk-score, функціональну перевірку сторінок інтерфейсу та фіксацію результатів у журналі запусків.

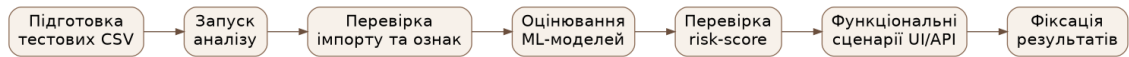


Рисунок 4.1 – Узагальнена схема методики тестування системи

Методика тестування охоплювала кілька взаємопов'язаних напрямів, що наведені в таблиці 4.1. Такий поділ дозволив окремо оцінити роботу прикладної логіки, аналітичних компонентів та візуальної частини системи.

Таблиця 4.1 – Основні напрями тестування програмного продукту

Напрямок тестування	Мета перевірки	Основні критерії успішності
Функціональне тестування	Перевірка доступності сторінок, форм, фільтрів і сценаріїв користувача	Сценарій завершується без помилок, сторінки відображають коректний стан системи
Тестування імпорту даних	Перевірка завантаження CSV-файлів і валідації структури даних	Файл приймається лише за наявності обов'язкових колонок, помилки коректно повідомляються користувачу
Тестування обробки ознак	Перевірка нормалізації, кодування категорій і розрахунку поведінкових параметрів	Кількість рядків зберігається, ознаки мають допустимі типи й діапазони
Тестування ML-моделей	Оцінювання Random Forest, Isolation Forest, Autoencoder і rule-based baseline	Метрики accuracy, precision, recall і F1 розраховуються без помилок та відображаються в інтерфейсі
Тестування risk-score	Перевірка перетворення оцінок моделей у рівень ризику й дію реагування	Критичні події отримують пріоритетні рекомендації, низькоризикові події лише журналюються
Тестування збереження результатів	Перевірка фіксації активного результату, експорту та журналу запусків	Результати не втрачаються між сторінками, журнал містить історію запусків

Таким чином, тестування було спрямоване не лише на підтвердження працездатності окремих сторінок, а й на перевірку повного циклу: від отримання події доступу до формування управлінського рішення для адміністратора або SOC-аналітика.

4.2 Підготовка тестових наборів даних і сценаріїв несанкціонованого доступу

Експериментальна перевірка виконувалась на підготовлених CSV-наборах, що входять до складу програмного продукту. Набори даних моделюють як нормальну поведінку користувачів, так і типові сценарії несанкціонованого доступу: компрометацію облікових даних, внутрішню ексфільтрацію, зміну політик доступу, комбінований сплеск атак, а також імітацію подій для AWS S3, Azure Blob Storage і Google Cloud Storage.

Кожний запис у тестовому наборі описує одну подію доступу до хмарного сховища. Мінімальна структура охоплює часову мітку, ідентифікатор користувача, роль, країну, тип пристрою, рівень довіри до пристрою, тип операції, факт використання MFA, обсяг переданих даних, кількість невдалих спроб входу, ознаки нового пристрою й нового географічного контексту, тривалість сесії, інтенсивність запитів за останні 10 хвилин, годину доби, ознаку вихідного дня та цільову мітку label. Для хмарно-орієнтованих наборів додатково використовуються поля source_provider, cloud_region, bucket_name, object_key, api_endpoint, auth_method, service_account, token_age_min і user_agent.

У таблиці 4.2 наведено склад тестових наборів, які використовувались для експериментальної перевірки. Вони забезпечують достатню різноманітність поведінкових профілів і дають змогу перевірити систему як на простих, так і на комбінованих сценаріях ризику.

Таблиця 4.2 – Тестові набори даних для експериментальної перевірки

Назва набору даних	Кількість подій	Призначення набору
normal_behavior.csv	2100	Базовий контрольний набір нормальної активності користувачів
credential_abuse.csv	2100	Сценарій використання скомпрометованих облікових даних
insider_exfiltration.csv	2100	Сценарій внутрішнього витоку або масового завантаження даних
policy_tampering.csv	2100	Сценарій несанкціонованої зміни політик доступу
mixed_attack_burst.csv	2600	Комбінований сценарій різких сплесків активності
aws_s3_enterprise.csv	2400	Імітація подій доступу до AWS S3
azure_blob_finance.csv	2400	Імітація подій доступу до Azure Blob Storage
gcp_bucket_research.csv	2400	Імітація подій доступу до Google Cloud Storage

Для основної демонстрації було використано активний набір `credential_abuse.csv`. Він містить 2100 подій, з яких 462 події позначені як підозрілі, а 1638 – як нормальні. Отже, частка позитивного класу становить 22,0 %, що дає змогу оцінити якість виявлення аномалій в умовах помірного дисбалансу класів. Після запуску аналізу система відображає активне джерело, кількість рядків, режим оцінювання та узагальнені результати на головній панелі.

На рисунку 4.2 наведено сторінку огляду активного результату. Вона відображає підсумкову кількість подій, кількість критичних і високоризикових випадків, середній ризик, розподіл рівнів ризику та ключові контрольні індикатори.

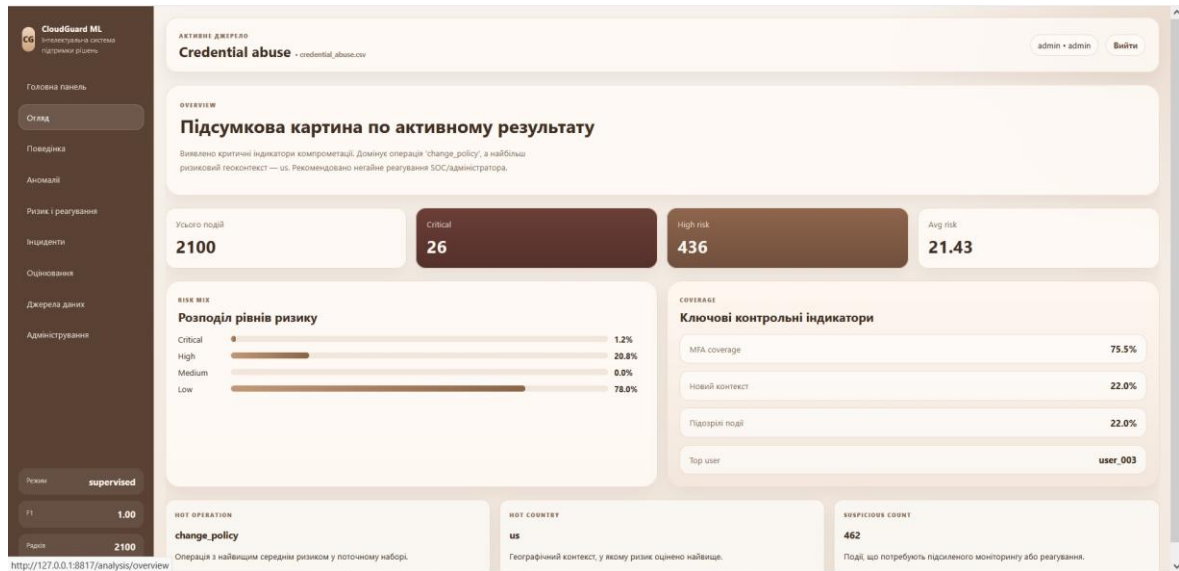


Рисунок 4.2 – Сторінка огляду активного результату аналізу

Наявність контрольованих сценаріїв важлива для повторюваності експерименту. Завдяки цьому один і той самий набір може бути використаний для перевірки імпорту, побудови ознак, оцінювання моделей, формування рекомендацій та перевірки інтерфейсних сторінок.

4.3 Функціональне тестування основних модулів системи

Функціональне тестування було спрямоване на перевірку того, що основні модулі CloudGuard ML виконують свої прикладні функції в очікуваній послідовності. Перевірялись сценарії входу до системи, вибору активного CSV-джерела, запуску аналізу, перегляду огляду, поведінкового профілю, аномалій, ризикових рішень, інцидентів, оцінювання моделей, джерел даних і адміністративної панелі.

На відміну від ізолюваного тестування окремої функції, функціональна перевірка охоплювала взаємодію кількох модулів: вебмаршруту, шаблону сторінки, серверного стану активного результату, об'єкта користувача та даних, збережених у PostgreSQL. Такий підхід дозволяє перевірити не лише факт відкриття сторінки, а й правильність її наповнення після виконання аналізу.

Ключові функціональні сценарії та очікувані результати наведено в таблиці 4.3.

Таблиця 4.3 – Результати функціонального тестування основних модулів

Модуль	Перевірений сценарій	Очікуваний результат	Фактичний результат
Автентифікація	Вхід адміністратора за початковим обліковим записом	Користувач потрапляє до головної панелі	Сценарій виконано успішно
Головна панель	Відображення активного джерела та KPI після аналізу	Показано назву CSV, кількість подій, F1 і режим оцінювання	Сценарій виконано успішно
Огляд	Перегляд підсумкової картини активного результату	Відображено кількість critical/high/avg risk та контрольні індикатори	Сценарій виконано успішно
Поведінка	Аналіз країн, операцій, ролей, пристроїв і годин активності	Сторінка показує групування за поведінковими ознаками	Сценарій виконано успішно
Аномалії	Перегляд розподілу anomaly score, тригерів і ризикових користувачів	Виведено bands, triggers і top anomalous users	Сценарій виконано успішно
Ризик і реагування	Перевірка матриці рішень і рекомендованих дій	Події розподілені за playbook реагування	Сценарій виконано успішно

Продовження таблиці 4.3

Модуль	Перевірений сценарій	Очікуваний результат	Фактичний результат
Інциденти	Фільтрація таблиці підозрілих подій за рівнем ризику	Таблиця показує user, country, operation, risk, level, reasons і action	Сценарій виконано успішно
Оцінювання	Перегляд порівняння моделей і матриці помилок	Метрики та графіки відображаються коректно	Сценарій виконано успішно
Адміністрування	Перегляд користувачів і журналу запусків	Адміністратор бачить поточні акаунти та історію аналізів	Сценарій виконано успішно

На рисунку 4.3 наведено приклад сторінки поведінкового профілю. Вона підтверджує, що система коректно агрегує активність за країнами, типами операцій, ролями, пристроями та часовими інтервалами.

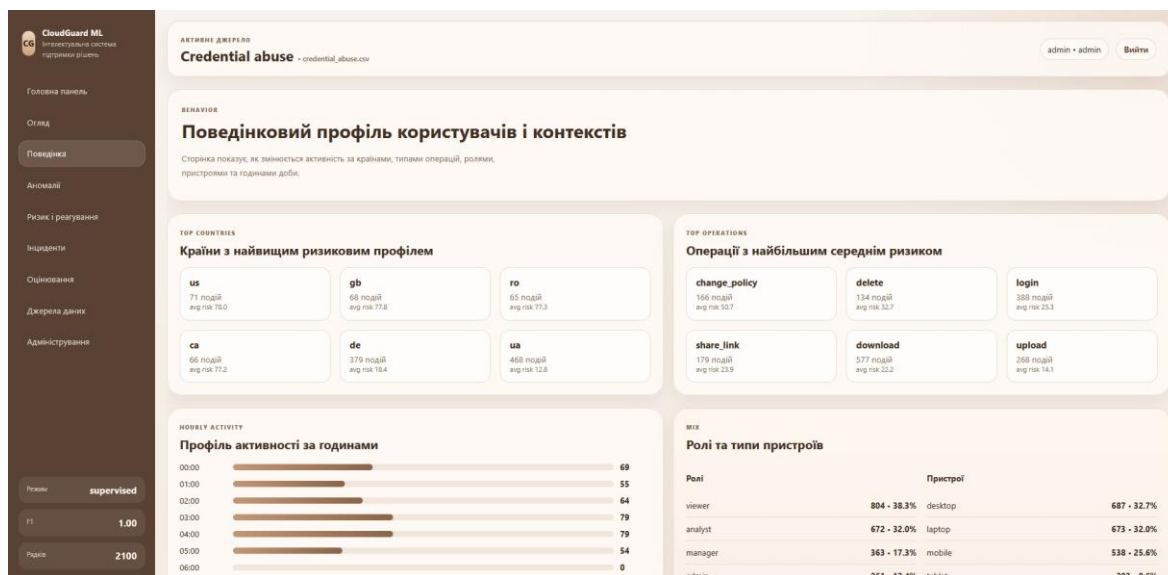


Рисунок 4.3 – Сторінка поведінкового профілю користувачів і контекстів

Функціональна перевірка показала, що основні сторінки не є ізольованими візуальними макетами: вони працюють з одним активним результатом аналізу,

використовують однакову логіку групування та дають змогу переходити від загальної картини до деталізації конкретних інцидентів.

4.4 Перевірка коректності імпорту, обробки та збереження даних

Окремим етапом перевірено коректність імпорту CSV-файлів, попередньої обробки даних і збереження результатів аналізу. Цей етап є критичним, оскільки помилки в структурі вхідних даних безпосередньо впливають на якість формування ознак і подальші результати ML-моделей.

Під час імпорту система перевіряє наявність обов'язкових колонок, читає табличні дані, приводить числові й категоріальні поля до потрібного формату, створює додаткові ознаки та передає матрицю ознак до моделей. Після завершення аналізу активний результат зберігається на серверному боці та стає доступним на сторінках огляду, поведінки, аномалій, ризику, інцидентів і оцінювання. Адміністративна панель додатково фіксує журнал запусків, що дозволяє перевірити факт виконання аналізу й кількісні характеристики результату.

У таблиці 4.4 наведено основні перевірки, виконані для етапу імпорту та збереження.

Таблиця 4.4 – Перевірка імпорту, обробки та збереження даних

Етап перевірки	Контрольна дія	Очікуваний результат	Результат
Читання CSV	Вибір credential_abuse.csv як активного джерела	Система читає 2100 рядків без втрати структури	Виконано
Валідація колонок	Перевірка наявності timestamp, user_id, role, operation, mfa_used, label тощо	Обов'язкові поля розпізнаються, додаткові поля не порушують обробку	Виконано

Продовження таблиці 4.3

Етап перевірки	Контрольна дія	Очікуваний результат	Результат
Нормалізація даних	Приведення числових і категоріальних ознак	Дані придатні для формування feature matrix	Виконано
Формування ознак	Розрахунок activity_intensity, unusual_hour, access_context_risk та інших ознак	Ознаки передаються до ML-компонента	Виконано
Збереження результату	Перехід між сторінками після запуску аналізу	Активний результат не втрачається	Виконано
Журнал запусків	Перевірка адміністративної панелі	Журнал містить користувача, джерело, кількість подій, critical/high/avg risk і час	Виконано

На рисунку 4.4 наведено адміністративну панель із журналом запусків. Наявність записів у журналі підтверджує, що результати аналізу не лише відображаються в інтерфейсі, а й зберігаються як керований стан системи.

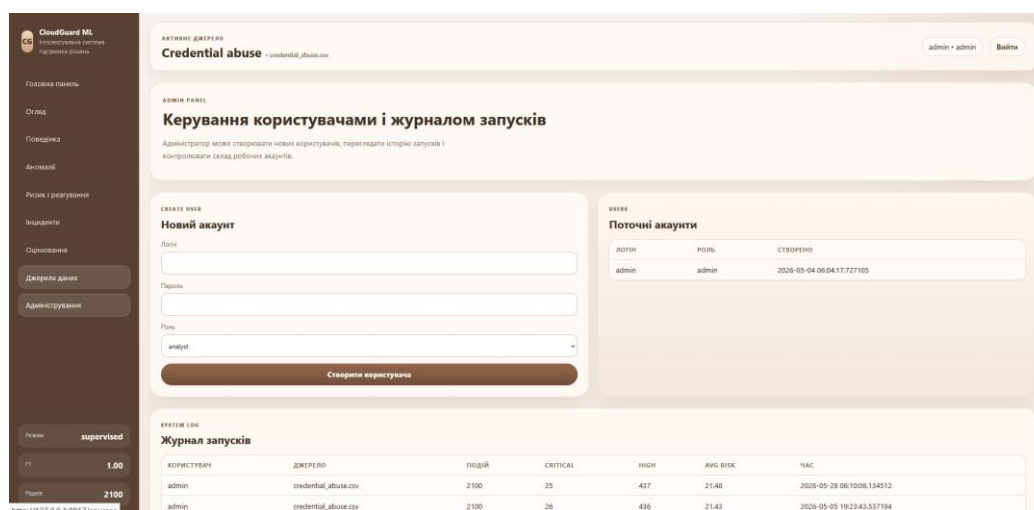


Рисунок 4.4 – Адміністративна панель із журналом запусків аналізу

За результатами перевірки встановлено, що імпорт і збереження даних виконуються коректно для контрольних CSV-наборів. Система зберігає кількісні параметри запуску, що є важливим для відтворення експериментів і подальшого аудиту результатів.

4.5 Тестування роботи моделей машинного навчання

Тестування ML-компонента передбачало перевірку чотирьох підходів: rule-based baseline, Random Forest classifier, Isolation Forest baseline і TensorFlow Autoencoder. Така структура перевірки дає змогу порівняти просту правилу логіку, supervised-класифікацію, безнаглядне виявлення аномалій та нейромережевий підхід за помилкою реконструкції.

Для supervised-режиму використано наявну колонку label, яка позначає нормальні та потенційно небезпечні події. Основна модель Random Forest навчалась на матриці поведінкових ознак, після чого її якість оцінювалася на hold-out вибірці. Для перевірки стійкості результату застосовано StratifiedKFold, який зберігає частку класів у кожному фолді та є придатним для задач із бінарною або мультикласовою класифікацією [30]. Метрики classification performance розраховувались засобами sklearn.metrics, які призначені для кількісного оцінювання якості прогнозів [29].

Безнаглядний підхід Isolation Forest використовувався як baseline для виявлення відхилень без повної залежності від розмітки. TensorFlow Autoencoder застосовувався як нейромережева модель, що навчається відтворювати нормальний поведінковий профіль, а аномалії визначає через підвищену помилку реконструкції. Такий підхід відповідає загальній ідеї autoencoder-моделей, що розглядаються в документації TensorFlow також для задач anomaly detection [32].

На рисунку 4.5 подано логіку переходу від окремих оцінок моделей і правил до інтегрального risk-score та рекомендованої дії реагування.

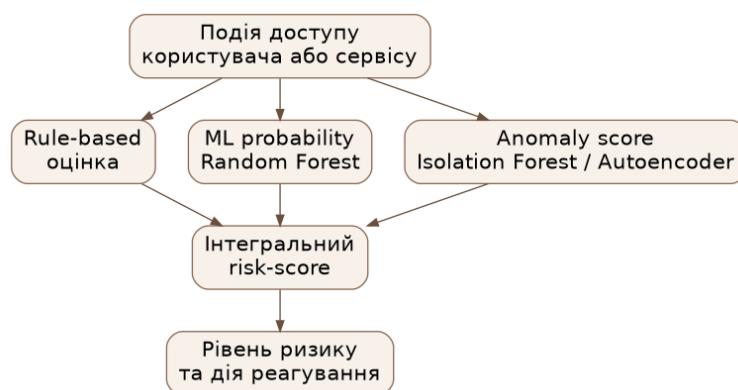


Рисунок 4.5 – Логіка формування risk-score на основі правил, ML-моделі та anomaly score

У таблиці 4.5 наведено роль кожного підходу в експериментальній перевірці.

Таблиця 4.5 – Призначення підходів, використаних під час тестування ML-компонента

Підхід	Тип аналізу	Роль у системі	Очікувана перевага
Rule-based baseline	Евристичний	Контрольний орієнтир без ML	Прозорість і пояснюваність правил
Random Forest classifier	Supervised	Основна модель класифікації подій	Висока якість на табличних даних і підтримка важливості ознак
Isolation Forest baseline	Unsupervised / anomaly	Виявлення відхилень без повної залежності від label	Придатність для сценаріїв з рідкісними інцидентами
TensorFlow Autoencoder	Neural anomaly detection	Виявлення нетипових подій за помилкою реконструкції	Здатність фіксувати складні нелінійні відхилення

Перевірка показала, що всі чотири підходи коректно інтегровані в систему оцінювання, а результати порівняння відображаються на сторінці evaluation. Це

дозволяє оцінити не лише фінальний прогноз, а й відносну ефективність альтернативних методів.

4.6 Аналіз показників accuracy, precision, recall та F1-score

Якість класифікації оцінювалась за показниками accuracy, precision, recall та F1-score. Accuracy відображає частку правильно класифікованих подій серед усіх проаналізованих записів. Precision показує, яка частка подій, позначених системою як аномальні, справді належить до класу ризикових. Recall характеризує здатність системи знаходити всі реальні ризикові події. F1-score використовується як узагальнений баланс між precision і recall, що особливо важливо для задач інформаційної безпеки, де одночасно небажані як пропущені інциденти, так і надмірні хибні спрацювання [29].

Для активного набору credential_abuse.csv система сформувала порівняння чотирьох підходів. Результати подано в таблиці 4.6.

Таблиця 4.6 – Порівняння моделей за метриками класифікації

Підхід	Accuracy	Precision	Recall	F1-score
Rules-only baseline	0.9848	1.0000	0.9304	0.9640
IsolationForest baseline	0.9505	0.8938	0.8783	0.8860
TensorFlow Autoencoder	0.9505	0.8156	1.0000	0.8984
RandomForest classifier	1.0000	1.0000	1.0000	1.0000

Дані таблиці 4.6 свідчать, що rule-based baseline забезпечує високу precision, однак має нижчий recall порівняно з Random Forest і TensorFlow Autoencoder. Це означає, що жорсткі правила добре відсікають очевидні ризикові події, але можуть не охоплювати частину менш типових сценаріїв. Isolation Forest має нижчі показники, що очікувано для безнаглядного baseline, який не використовує повну

інформацію про мітки класів. TensorFlow Autoencoder досягає recall 1,0000, тобто не пропускає аномальні приклади в контрольній вибірці, але має нижчу precision, що вказує на більшу схильність до хибнопозитивних спрацювань. Найвищі результати показала модель Random Forest, для якої всі чотири метрики дорівнюють 1,0000.

На рисунку 4.6 наведено графічне порівняння підходів за accuracy та F1-score. Візуалізація побудована за результатами експериментального запуску й демонструє перевагу supervised-класифікації на підготовленому датасеті.

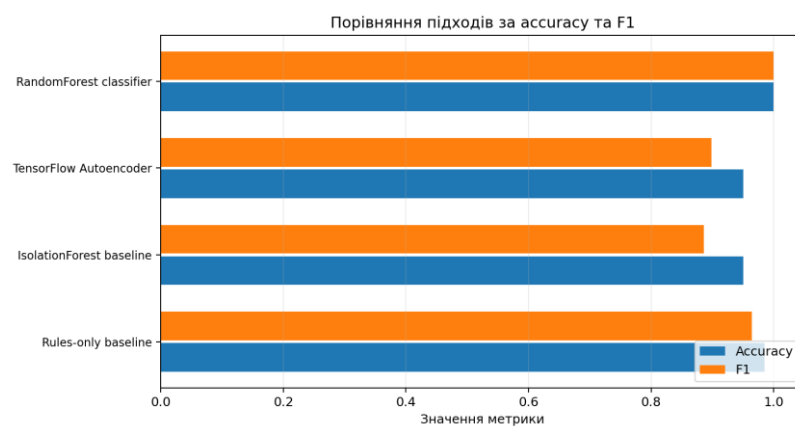


Рисунок 4.6 – Порівняння підходів за accuracy та F1-score

Водночас надто високі значення метрик для Random Forest необхідно інтерпретувати обережно. Вони підтверджують якісну відокремлюваність класів у синтетичному наборі `credential_abuse.csv`, але не гарантують аналогічної точності на реальних журналах доступу з шумом, неповною розміткою та складнішими поведінковими патернами.

4.7 Аналіз матриці помилок і результатів перехресної перевірки

Для деталізації результатів Random Forest було проаналізовано матрицю помилок. Вона дозволяє відокремити правильні класифікації нормальних подій, правильні класифікації аномалій, хибнопозитивні та хибнонегативні випадки. У

задачі інформаційної безпеки особливо небезпечними є false negative, оскільки вони означають пропущені потенційні інциденти.

Таблиця 4.7 – Матриця помилок Random Forest для hold-out вибірки

Показник	Значення	Інтерпретація
TN	410	Нормальні події, правильно визначені як нормальні
FP	0	Нормальні події, помилково визначені як аномальні
FN	0	Аномальні події, помилково пропущені системою
TP	115	Аномальні події, правильно визначені як аномальні

За результатами hold-out перевірки отримано 410 true negative і 115 true positive випадків, тоді як false positive і false negative дорівнюють нулю. Такий результат пояснює значення accuracy, precision, recall і F1-score на рівні 1,0000 для Random Forest. Графічне представлення матриці наведено на рисунку 4.7.

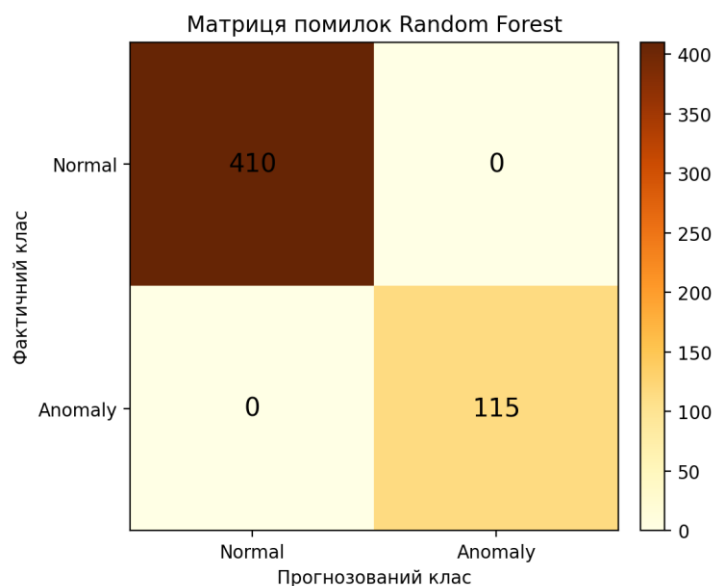


Рисунок 4.7 – Матриця помилок Random Forest для контрольної вибірки

Додатково стабільність результату перевірено методом StratifiedKFold. Цей підхід формує кілька навчально-тестових розбиттів із збереженням пропорцій класів у кожному фолді, що робить оцінювання стійкішим для задач з дисбалансом [30]. Результати перехресної перевірки наведено в таблиці 4.8.

Таблиця 4.8 – Результати перехресної перевірки Random Forest

Показник	Середнє значення	Стандартне відхилення
Accuracy	1.0000	0.0000
Precision	1.0000	0.0000
Recall	1.0000	0.0000
F1-score	1.0000	0.0000

Отримані значення перехресної перевірки підтверджують стабільність моделі на підготовленому наборі. Нульове стандартне відхилення у межах експерименту свідчить про те, що розділення класів у синтетичних даних є чітким. Саме тому в обмеженнях роботи окремо зазначено необхідність перевірки системи на реальних журналах доступу або на змішаних наборах із менш вираженими ознаками атак.

4.8 Порівняння rule-based baseline, Random Forest, Isolation Forest і TensorFlow Autoencoder

Порівняння чотирьох підходів дозволяє оцінити не лише абсолютні метрики, а й практичну доцільність кожного методу у складі інтелектуальної системи. Rule-based baseline є прозорим і легко пояснюваним, однак недостатньо адаптивним. Random Forest демонструє найкращу якість на розміченому наборі даних. Isolation Forest корисний як fallback-механізм у випадках, коли розмітка відсутня або недостатня. TensorFlow Autoencoder може використовуватись для виявлення складніших поведінкових відхилень, проте потребує обережного налаштування порога реконструкції.

Перевага гібридного підходу полягає в тому, що система не залежить від одного алгоритму. У supervised-режимі основний акцент робиться на Random Forest, тоді як anomaly score та rule score залишаються додатковими сигналами для

risk-score. У разі неповної розмітки або переходу до реальних журналів доступу важливішими стають Isolation Forest і Autoencoder, оскільки вони можуть працювати з відхиленнями без повної попередньої розмітки.

У таблиці 4.9 наведено аналітичне порівняння підходів з погляду практичного використання.

Таблиця 4.9 – Практичне порівняння підходів до виявлення аномалій

Критерій	Rule-based baseline	Random Forest	Isolation Forest	TensorFlow Autoencoder
Потреба у розмітці	Не потребує	Потребує label	Не потребує повної розмітки	Бажано мати профіль нормальної поведінки
Пояснюваність	Висока	Середня, через feature importance	Нижча	Нижча
Якість на активному наборі	Висока, але нижчий recall	Найвища	Помірна	Високий recall, нижча precision
Стійкість до нових сценаріїв	Обмежена	Залежить від навчальної вибірки	Вища для невідомих відхилень	Потенційно висока для складних патернів
Доцільність у системі	Базовий контроль і пояснення	Основна модель у supervised-режимі	Baseline для anomaly-режиму	Додатковий нейромережевий сигнал

На сторінці оцінювання системи відображаються порівняльна таблиця підходів, матриця помилок, статус TensorFlow Autoencoder, поріг реконструкції, графіки розподілу ризику та важливість ознак. Приклад цієї сторінки наведено на рисунку 4.8.

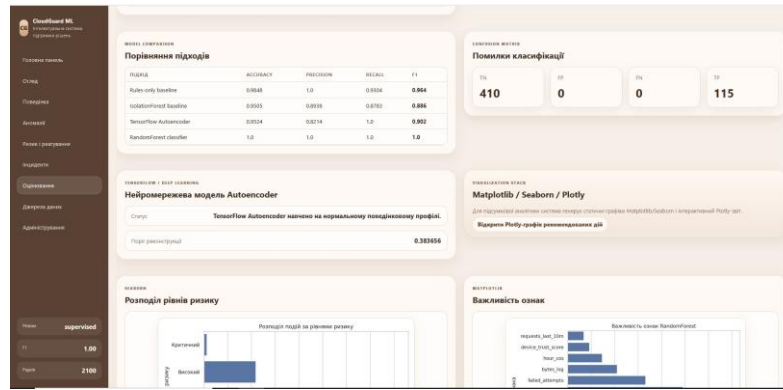


Рисунок 4.8 – Сторінка оцінювання моделей і результатів класифікації

Аналіз важливості ознак Random Forest показав, що найбільший внесок у рішення моделі мають `access_context_risk`, `is_new_country`, `is_new_device`, `unusual_hour` та `activity_intensity`.

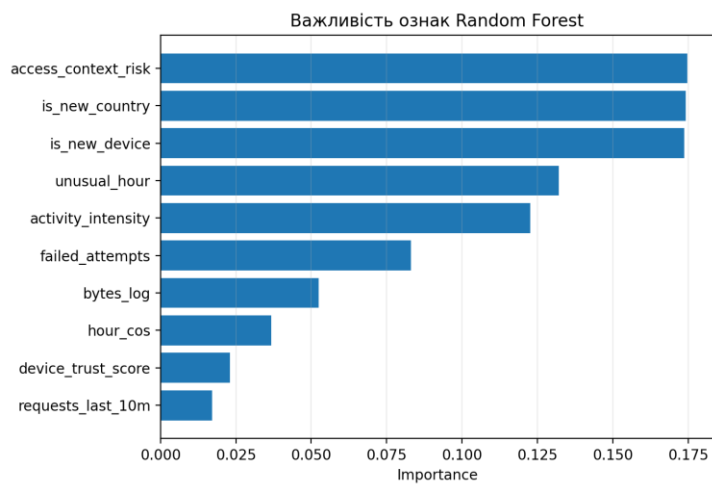


Рисунок 4.9 – Важливість ознак Random Forest у межах експерименту

4.9 Аналіз результатів risk-score та рекомендацій щодо реагування

Risk-score у розробленій системі виконує роль проміжного шару між технічним результатом ML-моделі та операційним рішенням щодо реагування. Він поєднує ймовірність аномальності, `anomaly score`, `deep anomaly score` та `rule score`.

На основі інтегральної оцінки система призначає рівень ризику й формує рекомендовану дію.

Для активного результату `credential_abuse.csv` система зафіксувала переважання низькоризикових подій, однак також виявила суттєву кількість подій високого ризику та окремі критичні інциденти. Розподіл за рівнями ризику наведено на рисунку 4.10.

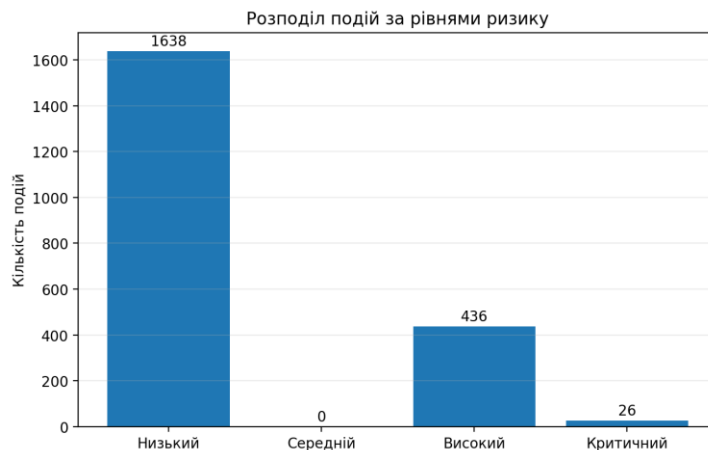


Рисунок 4.10 – Розподіл подій за рівнями ризику в активному результаті

Сторінка `risk & response`, наведена на рисунку 4.11, показує `playbook` реагування. Низький ризик передбачає журналювання, середній – створення сповіщення, високий – повторну автентифікацію або додаткову перевірку контексту, а критичний – блокування сесії, відкликання токена та фіксацію інциденту в SIEM.

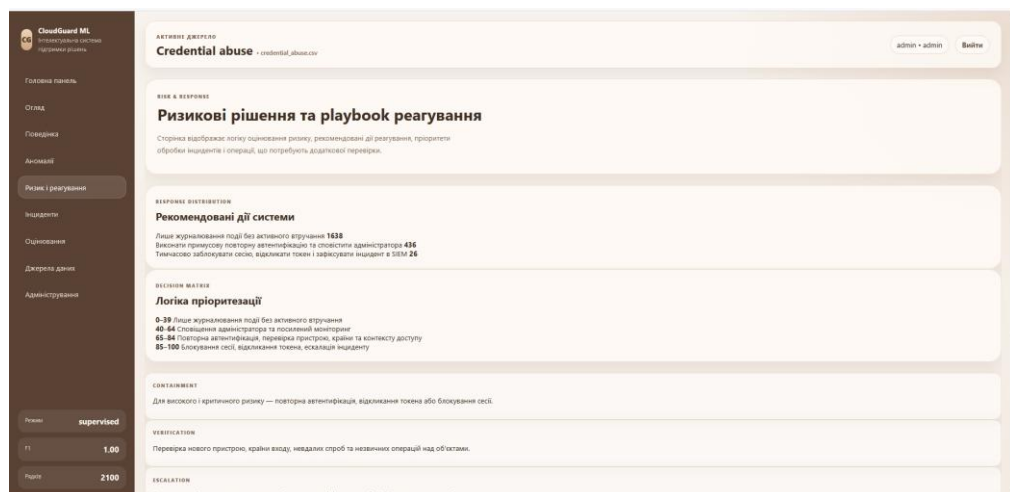
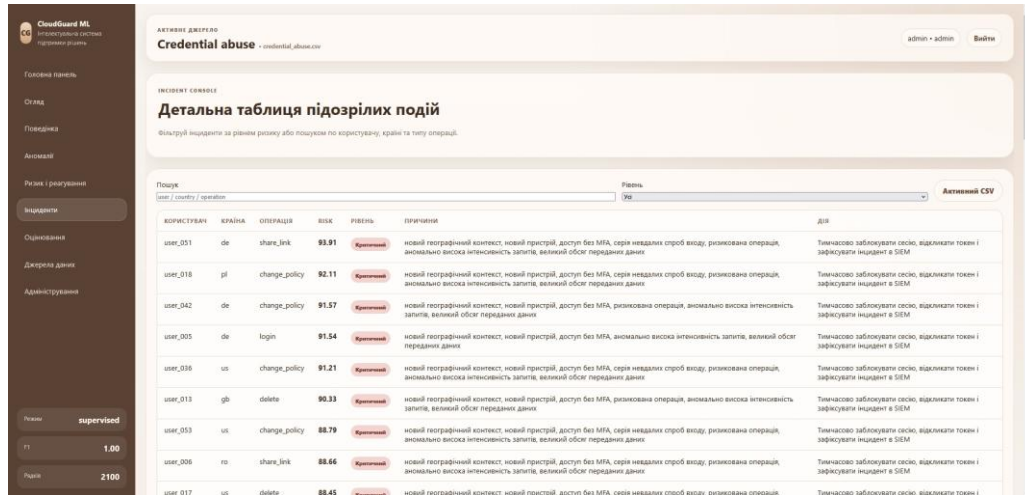


Рисунок 4.11 – Сторінка ризикових рішень і playbook реагування

Деталізація підозрілих подій доступна на сторінці інцидентів. Вона містить користувача, країну, операцію, числовий risk-score, рівень ризику, пояснення причин спрацювання та рекомендовану дію. Приклад наведено на рисунку 4.12.



АКТИВНЕ ДЖЕРЕЛО
Credential abuse - credential_abuse.csv

ІНЦИДЕНТ СЕРВІС

Детальна таблиця підозрілих подій
Фільтруй інциденти за рівнем ризику або пошуком по користувачу, країні та типу операції.

Поиск: user / country / operation Рівень: High Активний CSV

КОРИСТУВАЧ	КРАЇНА	ОПЕРАЦІЯ	ВОСК	РІВЕНЬ	ПІСЬМОВИЙ	ДІЯ
user_051	de	share_link	88.91	Високий	новий географічний контекст, новий пристрій, доступ без MFA, серія невдалих спроб входу, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_018	pl	change_policy	82.11	Високий	новий географічний контекст, новий пристрій, доступ без MFA, серія невдалих спроб входу, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_042	de	change_policy	91.57	Високий	новий географічний контекст, новий пристрій, доступ без MFA, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_005	de	login	91.54	Високий	новий географічний контекст, новий пристрій, доступ без MFA, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_036	us	change_policy	91.21	Високий	новий географічний контекст, новий пристрій, доступ без MFA, серія невдалих спроб входу, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_013	gb	delete	90.33	Високий	новий географічний контекст, новий пристрій, доступ без MFA, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_053	us	change_policy	88.79	Високий	новий географічний контекст, новий пристрій, доступ без MFA, серія невдалих спроб входу, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_006	pl	share_link	88.66	Високий	новий географічний контекст, новий пристрій, доступ без MFA, серія невдалих спроб входу, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM
user_017	us	delete	88.45	Високий	новий географічний контекст, новий пристрій, доступ без MFA, серія невдалих спроб входу, ризикова операція, аномально висока інтенсивність запитів, великий обсяг переданих даних	Тимчасово заблокувати сесію, відкрити токен і зареєструвати інцидент в SEM

Рисунок 4.12 – Детальна таблиця підозрілих подій і рекомендованих дій

Наявність пояснювального поля є важливою для практичного використання системи. Адміністратор або SOC-аналітик бачить не лише числовий бал, а й причини спрацювання: новий географічний контекст, новий пристрій, доступ без MFA, серію невдалих спроб входу, ризиковану операцію, аномально високу інтенсивність запитів або великий обсяг переданих даних. Це підвищує прозорість системи й дає змогу швидше приймати рішення щодо реагування.

4.10 Оцінювання практичної ефективності розробленої системи

Практична ефективність CloudGuard ML оцінювалась не лише за метриками машинного навчання, а й за здатністю системи виконувати повний прикладний сценарій моніторингу доступу до хмарного сховища. Розроблений продукт забезпечує завантаження або вибір CSV-джерела, автоматичну підготовку ознак, застосування кількох моделей, формування risk-score, пояснення причин

спрацювання, відображення інцидентів, побудову аналітичних графіків і ведення журналу запусків.

У таблиці 4.10 наведено оцінювання практичної ефективності системи за основними критеріями.

Таблиця 4.10 – Оцінювання практичної ефективності системи

Критерій	Результат перевірки	Практичне значення
Повнота аналітичного циклу	Реалізовано цикл від CSV-імпорту до risk-score і рекомендації	Система придатна для демонстрації повного процесу підтримки прийняття рішень
Підтримка кількох сценаріїв	Передбачено набори normal_behavior, credential_abuse, insider_exfiltration, policy_tampering, mixed_attack_burst та cloud-набори	Можна перевіряти різні типи несанкціонованого доступу
Якість supervised-моделі	Random Forest показав accuracy, precision, recall і F1 на рівні 1,0000 на контрольній вибірці	На підготовленому наборі модель чітко відокремлює аномальні події
Пояснюваність результату	Формуються explanation, trigger counts і feature importance	Аналітик бачить причини ризикового рішення
Підтримка реагування	Система генерує playbook-дії для різних рівнів ризику	Результат може використовуватися як основа для SOC-процесу
Візуальна аналітика	Використано таблиці, KPI, розподіли, матрицю помилок, графіки важливості ознак	Користувач швидко переходить від загальної картини до деталей
Відтворюваність запусків	Адміністративний журнал фіксує джерело, користувача, кількість подій і час	Забезпечується аудит експериментів і повторна перевірка

Результати експериментальної перевірки підтверджують, що розроблена система відповідає поставленій меті: вона дозволяє виявляти потенційно

несанкціонований доступ до хмарного сховища, оцінювати ризик подій і формувати рекомендації щодо реагування. Використання Matplotlib і Plotly для візуалізації підтримує як статичні графіки для звітності, так і інтерактивні подання для аналізу результатів [33, 34].

Висновки до розділу 4

У четвертому розділі проведено тестування та експериментальну перевірку розробленої системи CloudGuard ML. Сформовано методику тестування, яка охоплює функціональну перевірку вебінтерфейсу, імпорт і попередню обробку CSV-даних, збереження результатів, оцінювання ML-моделей, аналіз risk-score та перевірку рекомендацій щодо реагування.

Підготовлено й використано кілька тестових наборів даних, що моделюють нормальну поведінку, компрометацію облікових даних, внутрішню ексфільтрацію, зміну політик доступу, комбіновані атаки та події для хмарних середовищ AWS, Azure і GCP. Основний експеримент виконано на наборі `credential_abuse.csv`, який містить 2100 подій і 22,0 % ризикових записів.

Проведено функціональне тестування основних модулів системи: автентифікації, головної панелі, огляду, поведінкового аналізу, аномалій, ризику й реагування, інцидентів, оцінювання, джерел даних та адміністративної панелі. Підтверджено, що система зберігає активний результат аналізу, коректно відображає агреговані показники та фіксує запуск у журналі адміністратора.

Виконано порівняння rule-based baseline, Random Forest, Isolation Forest і TensorFlow Autoencoder. Найкращий результат на контрольному наборі показала модель Random Forest, для якої accuracy, precision, recall і F1-score становили 1,0000. За результатами матриці помилок отримано 410 true negative, 115 true positive, 0 false positive і 0 false negative. Перехресна перевірка StratifiedKFold підтвердила стабільність результату на підготовленому синтетичному наборі.

Проаналізовано результати risk-score і рекомендації щодо реагування. Показано, що система не обмежується бінарною класифікацією, а формує практичні дії: журналювання, посилений моніторинг, повторну автентифікацію, блокування сесії, відкликання токена та фіксацію інциденту. Визначено також обмеження системи, пов'язані з синтетичним характером даних, потребою в реальних журналах доступу та необхідністю подальшої інтеграції з хмарними провайдерами й SIEM/SOAR-платформами.

ВИСНОВКИ

У дипломній роботі досягнуто поставленої мети – розроблено інтелектуальну систему прийняття рішень при виявленні несанкціонованого доступу до хмарного сховища даних на основі методів машинного навчання. Реалізована система CloudGuard ML забезпечує повний цикл аналізу подій доступу: імпорт журналів у форматі CSV, перевірку структури даних, попередню обробку, формування поведінкових ознак, застосування моделей машинного навчання, обчислення рівня ризику, формування рекомендацій щодо реагування та збереження результатів аналізу.

Проаналізовано хмарні сховища даних як об'єкт захисту та встановлено, що основна складність виявлення несанкціонованого доступу полягає не лише у фіксації факту входу до системи, а у визначенні нетипової поведінки користувача або сервісу. Показано, що порушник може використовувати легітимні облікові дані, токени або дозволені канали доступу, тому класичні механізми контролю доступу, статичні правила та ручний аналіз журналів не забезпечують достатньої ефективності для своєчасного виявлення прихованих поведінкових аномалій.

Досліджено традиційні та інтелектуальні підходи до виявлення підозрілої активності в хмарному середовищі. Визначено, що для розв'язання поставленої задачі доцільним є поєднання поведінкової аналітики, машинного навчання та ризик-скорингу. Такий підхід дає змогу враховувати не одну окрему подію, а сукупність контекстних ознак: роль користувача, країну доступу, тип пристрою, спосіб автентифікації, використання багатофакторної автентифікації, кількість невдалих спроб входу, інтенсивність запитів, обсяг переданих даних і критичність операції.

Сформовано модель подій доступу до хмарного сховища та визначено набір поведінкових ознак, придатних для машинного аналізу. До моделі включено часові, контекстні, автентифікаційні, операційні та інтенсивнісні характеристики. Така

структура дала змогу перейти від простого перегляду журналів до формалізованого оцінювання ризику кожної події доступу.

Спроектовано архітектуру інтелектуальної системи CloudGuard ML. Архітектуру побудовано за модульним принципом, що забезпечує розділення функцій між вебінтерфейсом, серверною частиною, модулем імпорту даних, модулем попередньої обробки, модулем формування ознак, блоком машинного навчання, механізмом оцінювання ризику та базою даних. Обраний підхід спрощує тестування, супровід і подальше розширення системи.

Реалізовано програмний продукт CloudGuard ML як веборієнтовану систему з розмежуванням ролей адміністратора та аналітика. Для реалізації використано Python, FastAPI, PostgreSQL, Docker Compose, pandas, scikit-learn і TensorFlow. Система підтримує вибір підготовленого набору даних, завантаження власного CSV-файлу, запуск аналізу, перегляд огляду активного результату, поведінкових профілів, аномалій, інцидентів, оцінювання моделей і журналу запусків.

Обґрунтовано та реалізовано гібридний підхід до виявлення підозрілої активності. У системі використано rule-based baseline як контрольний правилочий підхід, Random Forest як основну модель для розмічених даних, Isolation Forest для пошуку аномалій за умов неповної розмітки та TensorFlow Autoencoder для виявлення нетипових подій за помилкою реконструкції. Застосування кількох підходів дало змогу порівняти їхню ефективність і підвищити гнучкість системи для різних сценаріїв аналізу.

Розроблено механізм оцінювання ризику RiskScore, який перетворює результат машинного аналізу на практичну рекомендацію щодо реагування. Для кожної події визначається рівень ризику: низький, середній, високий або критичний. Залежно від рівня ризику система формує рекомендовану дію: журналювання події, сповіщення адміністратора, повторна автентифікація, блокування сесії або фіксація інциденту для подальшого розслідування. Завдяки цьому система виконує не лише функцію класифікації, а й функцію підтримки прийняття рішень.

Проведено функціональне тестування основних модулів системи та експериментальну перевірку на тестовому наборі даних. За результатами аналізу опрацьовано 2100 подій доступу. Система визначила 25–26 критичних подій, 436–437 подій високого ризику, середній рівень ризику близько 21 та домінування низькоризикових подій у загальному розподілі. Встановлено, що найризиковішими сценаріями стали зміна політик доступу, новий географічний контекст, новий пристрій, доступ без багатфакторної автентифікації, серія невдалих спроб входу та підвищений обсяг переданих даних.

Оцінено якість роботи моделей машинного навчання. На тестовому наборі Random Forest досяг значень accuracy, precision, recall і F1-score на рівні 1,00. Матриця помилок для цієї моделі містила 410 правильних визначень нормальних подій, 115 правильних визначень аномальних подій, 0 хибнопозитивних і 0 хибнонегативних спрацювань. Для порівняння, rule-based baseline отримав F1-score 0,964, Isolation Forest – 0,886, TensorFlow Autoencoder – 0,902. Отримані результати підтвердили найвищу ефективність Random Forest на підготовленому розміченому тестовому наборі та практичну доцільність використання гібридного підходу.

Встановлено, що розроблена система відповідає поставленим функціональним вимогам: виконує імпорт даних, перевіряє структуру вхідного файлу, формує поведінкові ознаки, застосовує моделі машинного навчання, обчислює ризик, формує рекомендації, зберігає результати та надає їх користувачу через інформаційну панель. Нефункціональні вимоги також реалізовано: система має модульну структуру, підтримує контейнеризоване розгортання, зберігає історію запусків і забезпечує відтворюваність результатів аналізу.

Практичне значення отриманих результатів полягає у створенні програмного прототипу, який може використовуватися як основа для побудови модуля моніторингу доступу до хмарних сховищ. Розроблена система дає змогу автоматизувати аналіз журналів доступу, виявляти ризикові події, пояснювати причини спрацювання, ранжувати інциденти за рівнем небезпеки та підтримувати адміністратора або фахівця з інформаційної безпеки під час вибору дії реагування.

Подальший розвиток роботи доцільно спрямувати на інтеграцію з реальними хмарними API, розширення набору поведінкових ознак, додавання механізмів сповіщення, підтримку журналу розслідувань, налаштування політик ризику для різних організацій, розширення ролей користувачів і перевірку системи на більших наборах реальних журналів доступу. Такі напрями дадуть змогу підвищити прикладну цінність системи та наблизити її до використання в реальному середовищі моніторингу інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jansen W., Grance T. Guidelines on Security and Privacy in Public Cloud Computing. NIST Special Publication 800-144. Gaithersburg: National Institute of Standards and Technology, 2011. URL: <https://csrc.nist.gov/pubs/sp/800/144/final> (дата звернення: 24.04.2026).
2. Rose S., Borchert O., Mitchell S., Connelly S. Zero Trust Architecture. NIST Special Publication 800-207. Gaithersburg: National Institute of Standards and Technology, 2020. URL: <https://csrc.nist.gov/pubs/sp/800/207/final> (дата звернення: 24.04.2026).
3. Cloud Security Alliance. Top Threats to Cloud Computing 2025. URL: <https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2025> (дата звернення: 24.04.2026).
4. European Union Agency for Cybersecurity (ENISA). Cloud Computing: Benefits, Risks and Recommendations for Information Security. URL: <https://www.enisa.europa.eu/publications/cloud-computing-benefits-risks-and-recommendations-for-information-security> (дата звернення: 24.04.2026).
5. Hagemann T., Dauer M., Fietz A., Kiesel R., Kintz M., Vater T. A Systematic Review on Anomaly Detection for Cloud Computing Environments. Proceedings of the 2020 3rd Artificial Intelligence and Cloud Computing Conference. 2021. URL: <https://dl.acm.org/doi/10.1145/3442536.3442550> (дата звернення: 25.04.2026).
6. Al-Ghuwairi A.-R., Sharrab Y., Al-Fraihat D. et al. Intrusion detection in cloud computing based on time series anomalies utilizing machine learning. Journal of Cloud Computing. 2023. Vol. 12. Article 127. URL: <https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00491-x> (дата звернення: 25.04.2026).
7. Hozouri A., Mirzaei A., Effatparvar M. A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges. Discover Artificial Intelligence. 2025. URL:

<https://link.springer.com/article/10.1007/s44163-025-00578-1> (дата звернення: 26.04.2026).

8. Brandao P. R., et al. Exploring the Role of Artificial Intelligence in Detecting and Mitigating Advanced Persistent Threats: A UEBA Perspective. Computers. 2025. URL: <https://www.mdpi.com/2073-431X/14/7/245> (дата звернення: 26.04.2026).

9. Nelson A., et al. Computer Security Incident Handling Guide. NIST Special Publication 800-61 Revision 3. 2025. URL: <https://csrc.nist.gov/projects/incident-response> (дата звернення: 26.04.2026).

10. Breiman L. Random Forests. Machine Learning. 2001. Vol. 45, No. 1. P. 5–32. URL: <https://link.springer.com/article/10.1023/A:1010933404324> (дата звернення: 26.04.2026).

11. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest. Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. 2008. P. 413–422. URL: <https://doi.org/10.1109/ICDM.2008.17> (дата звернення: 26.04.2026).

12. Scikit-learn developers. StratifiedKFold. Scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (дата звернення: 26.04.2026).

13. Scikit-learn developers. Metrics and scoring: quantifying the quality of predictions. Scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата звернення: 26.04.2026).

14. Al-Mazrawe A. Anomaly Detection in Cloud Network: A Review. BIO Web of Conferences. 2024. URL: https://www.bioconferences.org/articles/bioconf/abs/2024/16/bioconf_iscku2024_00019/bioconf_iscku2024_00019.html (дата звернення: 26.04.2026).

15. Fuentes J., et al. Cybersecurity threat detection based on a UEBA framework using Deep Autoencoders. AIMS Mathematics. 2025. URL: <https://www.aimspress.com/article/doi/10.3934/math.20251043> (дата звернення: 26.04.2026).

16. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey. ACM Computing Surveys. 2009. Vol. 41, No. 3. Article 15. 58 p. DOI: 10.1145/1541880.1541882.
17. Goldstein M., Uchida S. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PLOS ONE. 2016. Vol. 11, No. 4. Article e0152173. DOI: 10.1371/journal.pone.0152173.
18. FastAPI. FastAPI documentation. URL: <https://fastapi.tiangolo.com/> (дата звернення: 28.05.2026).
19. SQLAlchemy. SQLAlchemy ORM Documentation. URL: <https://docs.sqlalchemy.org/orm/> (дата звернення: 28.05.2026).
20. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 28.05.2026).
21. Docker Inc. Docker Compose Documentation. URL: <https://docs.docker.com/compose/> (дата звернення: 28.05.2026).
22. pandas. pandas.read_csv documentation. URL: https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html (дата звернення: 28.05.2026).
23. scikit-learn developers. scikit-learn User Guide and API Reference. URL: <https://scikit-learn.org/stable/> (дата звернення: 28.05.2026).
24. TensorFlow. Intro to Autoencoders. URL: <https://www.tensorflow.org/tutorials/generative/autoencoder> (дата звернення: 28.05.2026).
25. Pydantic. Pydantic Documentation. URL: <https://pydantic.dev/> (дата звернення: 28.05.2026).
26. Jinja. Jinja Documentation. URL: <https://jinja.palletsprojects.com/> (дата звернення: 28.05.2026).
27. pytest documentation [Electronic resource]. URL: <https://docs.pytest.org/> (date of access: 28.05.2026).

28. HTTPX. Transports. ASGI Transport [Electronic resource]. URL: <https://www.python-httpx.org/advanced/transports/> (date of access: 28.05.2026).
29. Scikit-learn. Metrics and scoring: quantifying the quality of predictions [Electronic resource]. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (date of access: 28.05.2026).
30. Scikit-learn. StratifiedKFold [Electronic resource]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (date of access: 28.05.2026).
31. OWASP Foundation. Web Security Testing Guide [Electronic resource]. URL: <https://owasp.org/www-project-web-security-testing-guide/> (date of access: 28.05.2026).
32. TensorFlow. Intro to Autoencoders [Electronic resource]. URL: <https://www.tensorflow.org/tutorials/generative/autoencoder> (date of access: 28.05.2026).
33. Plotly. Plotly Python Graphing Library [Electronic resource]. URL: <https://plotly.com/python/> (date of access: 28.05.2026).
34. Matplotlib documentation [Electronic resource]. URL: <https://matplotlib.org/stable/index.html> (date of access: 28.05.2026).

ДОДАТОК А Лістинг розробленої систем

```
from __future__ import annotations

from contextlib import asynccontextmanager
from datetime import datetime
from io import BytesIO
import os
from pathlib import Path
from typing import Any, Dict, List, Optional
from uuid import uuid4

import pandas as pd
from fastapi import FastAPI, File, Form, Request, UploadFile
from fastapi.responses import FileResponse, HTMLResponse, JSONResponse, RedirectResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from starlette.middleware.sessions import SessionMiddleware

from src.auth import (
    authenticate_user,
    count_users,
    create_initial_admin,
    create_user,
    get_analysis_result,
    get_latest_analysis_result,
    get_user,
    init_database,
    list_history,
    list_users,
    log_analysis_run,
    save_analysis_result,
)
from src.data_generator import SCENARIO_LIBRARY, ensure_test_datasets
from src.cloud_simulator import PROVIDERS as CLOUD_SCENARIOS, generate_cloud_environment_datasets
from src.visualization import generate_report_assets
from src.feature_engineering import FEATURE_COLUMNS, load_events
from src.modeling import ensure_models, predict_probabilities
from src.risk_engine import build_threat_briefing, build_trigger_counts, finalize_decisions

BASE_DIR = Path(__file__).resolve().parent
DATA_DIR = BASE_DIR / "data"
INPUT_DIR = DATA_DIR / "inbox"
EXPORTS_DIR = DATA_DIR / "exports"
MODEL_DIR = BASE_DIR / "models"
STATIC_DIR = BASE_DIR / "static"
REPORTS_DIR = STATIC_DIR / "reports"
TEMPLATES_DIR = BASE_DIR / "templates"
DB_PATH = os.getenv("DATABASE_URL",
    "postgresql+psycpg2://cloudguard:cloudguard_pass@127.0.0.1:55432/cloudguard_ml")
SECRET_KEY = "cloud-guard-ml-final-secret-key"
DEFAULT_ADMIN_USERNAME = os.getenv("CLOUDGUARD_ADMIN_USERNAME", "admin")
DEFAULT_ADMIN_PASSWORD = os.getenv("CLOUDGUARD_ADMIN_PASSWORD", "Admin123!")
```

```

templates = Jinja2Templates(directory=str(TEMPLATES_DIR))
DEFAULT_METRICS = {"accuracy": 0.0, "precision": 0.0, "recall": 0.0, "f1": 0.0, "mode": "pending"}

def build_source_catalog() -> List[Dict[str, str]]:
    catalog: List[Dict[str, str]] = []
    for path in sorted(INPUT_DIR.glob("*.csv")):
        meta = SCENARIO_LIBRARY.get(path.stem, CLOUD_SCENARIOS.get(path.stem, {}))
        catalog.append({
            "slug": path.stem,
            "title": str(meta.get("title", path.stem.replace("_", " ").title())),
            "description": str(meta.get("description", "Імпортований CSV-файл із локального каталогу data/inbox.")),
            "path": str(path),
            "filename": path.name,
        })
    return catalog

def empty_summary() -> Dict[str, Any]:
    return {
        "total_events": 0,
        "avg_risk": 0.0,
        "critical_events": 0,
        "high_events": 0,
        "medium_events": 0,
        "low_events": 0,
        "critical_pct": 0.0,
        "high_pct": 0.0,
        "medium_pct": 0.0,
        "low_pct": 0.0,
        "mfa_pct": 0.0,
        "new_context_pct": 0.0,
        "suspicious_count": 0,
        "suspicious_pct": 0.0,
        "top_operation": "-",
        "top_country": "-",
        "top_user": "-",
        "briefing": "Поки що немає активного результату аналізу. Завантаж CSV-файл або вибери джерело з каталогу data/inbox.",
        "triggers": [],
        "suspicious_rows": [],
    }

def empty_evaluation() -> Dict[str, Any]:
    return {
        "mode": "pending",
        "accuracy": 0.0,
        "precision": 0.0,
        "recall": 0.0,
        "f1": 0.0,
        "holdout": {"accuracy": 0.0, "precision": 0.0, "recall": 0.0, "f1": 0.0},
        "cv": {"enabled": False, "splits": 0, "accuracy_mean": 0.0, "accuracy_std": 0.0, "precision_mean": 0.0, "precision_std": 0.0, "recall_mean": 0.0, "recall_std": 0.0, "f1_mean": 0.0, "f1_std": 0.0},
        "confusion_matrix": {"tn": 0, "fp": 0, "fn": 0, "tp": 0},
        "comparison": [],
        "feature_importances": [],
    }

```

```

    "dataset_profile": {"rows": 0, "features": len(FEATURE_COLUMNS), "has_labels": False, "positive_rate": 0.0,
    "class_balance": "Немає даних"},
    "tensorflow": {"enabled": False, "status": "Очікується запуск аналізу"},
    "tensorflow_runtime": {"enabled": False, "status": "Очікується запуск аналізу"},
    "visuals": {},
    "notes": ["Оцінювання моделі з'явиться після виконання аналізу на датасеті."],
}

```

```

def summarize_results(df: pd.DataFrame) -> Dict[str, Any]:

```

```

    if df.empty:
        return empty_summary()

    total = max(int(len(df)), 1)
    risk_counts = {
        "critical": int((df["risk_level"] == "Критичний").sum()),
        "high": int((df["risk_level"] == "Високий").sum()),
        "medium": int((df["risk_level"] == "Середній").sum()),
        "low": int((df["risk_level"] == "Низький").sum()),
    }
    suspicious = df[df["risk_score"] >= 40].copy()
    top_operation = str(df.groupby("operation")["risk_score"].mean().sort_values(ascending=False).index[0])
    top_country = str(df.groupby("country")["risk_score"].mean().sort_values(ascending=False).index[0])
    top_user = str(df.groupby("user_id")["risk_score"].mean().sort_values(ascending=False).index[0])
    return {
        "total_events": int(len(df)),
        "avg_risk": round(float(df["risk_score"].mean()), 2),
        "critical_events": risk_counts["critical"],
        "high_events": risk_counts["high"],
        "medium_events": risk_counts["medium"],
        "low_events": risk_counts["low"],
        "critical_pct": round(100 * risk_counts["critical"] / total, 1),
        "high_pct": round(100 * risk_counts["high"] / total, 1),
        "medium_pct": round(100 * risk_counts["medium"] / total, 1),
        "low_pct": round(100 * risk_counts["low"] / total, 1),
        "mfa_pct": round(100 * float(pd.to_numeric(df.get("mfa_used", 0), errors="coerce").fillna(0).mean()), 1),
        "new_context_pct": round(100 * float(((df.get("is_new_country", 0) == 1) | (df.get("is_new_device", 0) ==
1)).mean()), 1),
        "suspicious_count": int(len(suspicious)),
        "suspicious_pct": round(100 * len(suspicious) / total, 1),
        "top_operation": top_operation,
        "top_country": top_country,
        "top_user": top_user,
        "briefing": build_threat_briefing(df),
        "triggers": build_trigger_counts(suspicious.head(80)) if not suspicious.empty else [],
        "suspicious_rows": suspicious.head(120).to_dict(orient="records"),
    }

```

```

def top_table(df: pd.DataFrame, key: str, label_name: str, count_name: str = "events", limit: int = 6) -> List[Dict[str,
Any]]:

```

```

    if df.empty or key not in df.columns:
        return []
    grouped = (
        df.groupby(key)
        .agg(events=(key, "size"), avg_risk=("risk_score", "mean"), critical=("risk_level", lambda s: int((s ==
"Критичний").sum())))

```

```

.reset_index()
.sort_values(["avg_risk", "events"], ascending=[False, False])
.head(limit)
)
results: List[Dict[str, Any]] = []
for _, row in grouped.iterrows():
    results.append({
        label_name: str(row[key]),
        count_name: int(row["events"]),
        "avg_risk": round(float(row["avg_risk"]), 1),
        "critical": int(row["critical"]),
    })
return results

def _distribution(df: pd.DataFrame, key: str, label_name: str, limit: int = 8) -> List[Dict[str, Any]]:
    if df.empty or key not in df.columns:
        return []
    counts = df[key].astype(str).value_counts().head(limit)
    total = max(int(counts.sum()), 1)
    return [{"label_name": idx, "count": int(val), "pct": round(100 * val / total, 1)} for idx, val in counts.items()]

def _hour_profile(df: pd.DataFrame) -> List[Dict[str, Any]]:
    if df.empty:
        return []
    hours = pd.to_numeric(df.get("hour", 0), errors="coerce").fillna(0).astype(int)
    counts = hours.value_counts().reindex(range(24), fill_value=0)
    max_val = max(int(counts.max()), 1)
    result = []
    for hour, value in counts.items():
        result.append({"hour": f"{hour:02d}:00", "count": int(value), "width": round(100 * int(value) / max_val, 1)})
    return result

def empty_analysis_context() -> Dict[str, Any]:
    return {
        "has_data": False,
        "summary": empty_summary(),
        "top_users": [],
        "top_operations": [],
        "top_countries": [],
        "role_mix": [],
        "device_mix": [],
        "hour_profile": [],
        "actions": [],
        "anomaly_bands": [
            {"label": "0.00–0.24", "count": 0, "width": 0},
            {"label": "0.25–0.49", "count": 0, "width": 0},
            {"label": "0.50–0.74", "count": 0, "width": 0},
            {"label": "0.75–1.00", "count": 0, "width": 0},
        ],
        "findings": ["Активний результат відсутній. Система готова до завантаження робочого CSV-файлу."],
        "playbooks": [
            {"title": "Containment", "text": "Після появи ризикових подій система сформує рекомендовані кроки стримування."},
            {"title": "Verification", "text": "Після завантаження даних стануть доступними контекстні пояснення й

```

```

перевірки."},
    {"title": "Escalation", "text": "Для високого ризику буде сформовано журнал інцидентів і подальші дії."},
],
"recent_incidents": [],
}

def build_analysis_context(df: Optional[pd.DataFrame]) -> Dict[str, Any]:
    if df is None or df.empty:
        return empty_analysis_context()

    summary = summarize_results(df)
    actions = _distribution(df, "recommended_action", "action", limit=5)
    top_users = _top_table(df[df["risk_score"] >= 40], "user_id", "label", limit=6)
    top_operations = _top_table(df, "operation", "label", limit=6)
    top_countries = _top_table(df, "country", "label", limit=6)
    top_roles = _distribution(df, "role", "label", limit=6)
    top_devices = _distribution(df, "device_type", "label", limit=6)
    anomalies = pd.to_numeric(df.get("anomaly_score", 0), errors="coerce").fillna(0)
    anomaly_bands = [
        {"label": "0.00–0.24", "count": int(((anomalies >= 0.00) & (anomalies < 0.25)).sum())},
        {"label": "0.25–0.49", "count": int(((anomalies >= 0.25) & (anomalies < 0.50)).sum())},
        {"label": "0.50–0.74", "count": int(((anomalies >= 0.50) & (anomalies < 0.75)).sum())},
        {"label": "0.75–1.00", "count": int((anomalies >= 0.75).sum())},
    ]
    max_band = max([item["count"] for item in anomaly_bands] + [1])
    for item in anomaly_bands:
        item["width"] = round(100 * item["count"] / max_band, 1)

    findings: List[str] = []
    if summary["critical_events"] > 0:
        findings.append("У наборі є критичні події, що вимагають негайного реагування та додаткової перевірки сесій.")
    if summary["new_context_pct"] >= 20:
        findings.append("Висока частка нових географічних контекстів або пристроїв свідчить про нестабільний профіль доступу.")
    if summary["mfa_pct"] < 70:
        findings.append("Частка використання MFA нижча за бажаний рівень, тому ризик компрометації облікових даних зростає.")
    if not findings:
        findings.append("Поведінковий профіль виглядає відносно стабільним, а виявлені події мають помірний ризик.")

    playbooks = [
        {"title": "Containment", "text": "Для високого і критичного ризику – повторна автентифікація, відкликання токена або блокування сесії."},
        {"title": "Verification", "text": "Перевірка нового пристрою, країни входу, невдалих спроб та незвичних операцій над об'єктами."},
        {"title": "Escalation", "text": "Якщо аномалія повторюється, передати інцидент адміністратору або SOC для ручного розслідування."},
    ]

    return {
        "has_data": True,
        "summary": summary,
        "top_users": top_users,
        "top_operations": top_operations,
        "top_countries": top_countries,
    }

```

```

    "role_mix": top_roles,
    "device_mix": top_devices,
    "hour_profile": _hour_profile(df),
    "actions": actions,
    "anomaly_bands": anomaly_bands,
    "findings": findings,
    "playbooks": playbooks,
    "recent_incidents": summary["suspicious_rows"][:50],
}

def bootstrap(app_obj: FastAPI) -> None:
    DATA_DIR.mkdir(parents=True, exist_ok=True)
    INPUT_DIR.mkdir(parents=True, exist_ok=True)
    EXPORTS_DIR.mkdir(parents=True, exist_ok=True)
    MODEL_DIR.mkdir(parents=True, exist_ok=True)
    REPORTS_DIR.mkdir(parents=True, exist_ok=True)
    init_database(DB_PATH)
    ensure_test_datasets(INPUT_DIR)
    generate_cloud_environment_datasets(INPUT_DIR)
    if count_users(DB_PATH) == 0:
        create_initial_admin(DB_PATH, DEFAULT_ADMIN_USERNAME, DEFAULT_ADMIN_PASSWORD)
    app_obj.state.metrics = empty_evaluation()
    app_obj.state.source_catalog = build_source_catalog()

@asynccontextmanager
async def lifespan(app_obj: FastAPI):
    bootstrap(app_obj)
    yield

app = FastAPI(title="CloudGuard ML", lifespan=lifespan)
app.add_middleware(SessionMiddleware, secret_key=SECRET_KEY, same_site="lax", https_only=False)
app.mount("/static", StaticFiles(directory=STATIC_DIR), name="static")

def needs_initial_setup() -> bool:
    return False

def get_current_user(request: Request) -> Optional[Dict[str, Any]]:
    username = request.session.get("username")
    if not username:
        return None
    return get_user(DB_PATH, username)

def require_user(request: Request) -> Optional[RedirectResponse]:
    if needs_initial_setup():
        return RedirectResponse(url="/setup", status_code=303)
    if not get_current_user(request):
        return RedirectResponse(url="/login", status_code=303)
    return None

def _metrics_from_record(record: Optional[Dict[str, Any]]) -> Dict[str, Any]:

```

```

if not record:
    return DEFAULT_METRICS.copy()
metrics = record.get("metrics") or {}
return {
    "accuracy": float(metrics.get("accuracy", 0.0) or 0.0),
    "precision": float(metrics.get("precision", 0.0) or 0.0),
    "recall": float(metrics.get("recall", 0.0) or 0.0),
    "f1": float(metrics.get("f1", 0.0) or 0.0),
    "mode": str(metrics.get("mode", "pending") or "pending"),
}

def get_active_result_record(request: Request) -> Optional[Dict[str, Any]]:
    user = get_current_user(request)
    if not user:
        return None

    result_id = request.session.get("active_result_id")
    if result_id is not None:
        try:
            record = get_analysis_result(DB_PATH, int(result_id), username=user["username"])
        except (TypeError, ValueError):
            record = None
        if record:
            export_path = Path(record["export_path"])
            if export_path.exists():
                return record

    record = get_latest_analysis_result(DB_PATH, username=user["username"])
    if not record:
        return None

    export_path = Path(record["export_path"])
    if not export_path.exists():
        return None

    request.session["active_result_id"] = int(record["id"])
    return record

def get_active_result_path(request: Request) -> Optional[Path]:
    record = get_active_result_record(request)
    if not record:
        return None
    export_path = Path(record["export_path"])
    return export_path if export_path.exists() else None

def get_active_source(request: Request) -> Dict[str, str]:
    record = get_active_result_record(request)
    if not record:
        return {
            "label": "Активне джерело не вибрано",
            "name": "Очікується завантаження CSV",
        }
    return {
        "label": str(record.get("source_label") or "Активне джерело не вибрано"),
    }

```

```

    "name": str(record.get("source_name") or "Очікується завантаження CSV"),
}

def get_active_evaluation(request: Request) -> Dict[str, Any]:
    record = get_active_result_record(request)
    if not record:
        return empty_evaluation()
    evaluation = record.get("metrics") or {}
    if not isinstance(evaluation, dict):
        return empty_evaluation()
    base = empty_evaluation()
    for key, value in evaluation.items():
        base[key] = value
    return base

def pop_flash(request: Request) -> Dict[str, Optional[str]]:
    return {
        "flash_success": request.session.pop("flash_success", None),
        "flash_error": request.session.pop("flash_error", None),
    }

def build_page_context(request: Request, page_key: str, page_title: str) -> Dict[str, Any]:
    user = get_current_user(request)
    assert user is not None
    result_record = get_active_result_record(request)
    result_path = Path(result_record["export_path"]) if result_record else None
    result_df = pd.read_csv(result_path) if result_path and result_path.exists() else None
    analysis = build_analysis_context(result_df)
    source = get_active_source(request)
    evaluation = get_active_evaluation(request)
    return {
        "request": request,
        "title": page_title,
        "page_key": page_key,
        "user": user,
        "analysis": analysis,
        "source": source,
        "evaluation": evaluation,
        "active_export_name": result_record["export_filename"] if result_record else None,
        "metrics": _metrics_from_record(result_record),
        "sources": app.state.source_catalog,
        "history": list_history(DB_PATH, None if user["role"] == "admin" else user["username"], limit=8),
        **pop_flash(request),
    }

def render_page(request: Request, template_name: str, page_key: str, page_title: str, extra: Optional[Dict[str, Any]] = None) -> HTMLResponse:
    context = build_page_context(request, page_key, page_title)
    if extra:
        context.update(extra)
    return templates.TemplateResponse(template_name, context)

```

```

@app.get("/", response_class=HTMLResponse)
def dashboard(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "dashboard.html", "dashboard", "CloudGuard ML")

@app.get("/analysis/overview", response_class=HTMLResponse)
def analysis_overview(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "aspect_overview.html", "overview", "Analysis Overview")

@app.get("/analysis/behavior", response_class=HTMLResponse)
def analysis_behavior(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "aspect_behavior.html", "behavior", "Behavior Profile")

@app.get("/analysis/anomalies", response_class=HTMLResponse)
def analysis_anomalies(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "aspect_anomalies.html", "anomalies", "Anomaly Detection")

@app.get("/analysis/risk", response_class=HTMLResponse)
def analysis_risk(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "aspect_risk.html", "risk", "Risk & Response")

@app.get("/analysis/incidents", response_class=HTMLResponse)
def analysis_incidents(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "aspect_incidents.html", "incidents", "Incident Console")

@app.get("/analysis/evaluation", response_class=HTMLResponse)
def analysis_evaluation(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "aspect_evaluation.html", "evaluation", "Оцінювання моделі")

@app.get("/sources", response_class=HTMLResponse)

```

```

def sources_page(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    return render_page(request, "sources.html", "sources", "Data Sources")

@app.get("/setup", response_class=HTMLResponse)
def setup_page(request: Request) -> HTMLResponse:
    request.session["flash_success"] = "Початковий адміністратор створюється автоматично при першому запуску системи."
    return RedirectResponse(url="/login", status_code=303)

@app.post("/setup", response_class=HTMLResponse)
def setup_submit(request: Request, username: str = Form(...), password: str = Form(...), password_confirm: str = Form(...)) -> HTMLResponse:
    request.session["flash_success"] = "Початковий адміністратор уже створюється автоматично. Використай сторінку входу."
    return RedirectResponse(url="/login", status_code=303)

@app.get("/login", response_class=HTMLResponse)
def login_page(request: Request) -> HTMLResponse:
    if needs_initial_setup():
        return RedirectResponse(url="/setup", status_code=303)
    if get_current_user(request):
        return RedirectResponse(url="/", status_code=303)
    return templates.TemplateResponse("login.html", {"request": request, "title": "Login", "auth_error": None, "auth_success": request.session.pop("flash_success", None), "default_admin_username": DEFAULT_ADMIN_USERNAME, "default_admin_password": DEFAULT_ADMIN_PASSWORD})

@app.post("/login", response_class=HTMLResponse)
def login_submit(request: Request, username: str = Form(...), password: str = Form(...)) -> HTMLResponse:
    user = authenticate_user(DB_PATH, username, password)
    if not user:
        return templates.TemplateResponse("login.html", {"request": request, "title": "Login", "auth_error": "Невірний логін або пароль.", "auth_success": request.session.pop("flash_success", None), "default_admin_username": DEFAULT_ADMIN_USERNAME, "default_admin_password": DEFAULT_ADMIN_PASSWORD}, status_code=400)
    request.session["username"] = user["username"]
    return RedirectResponse(url="/", status_code=303)

@app.get("/register", response_class=HTMLResponse)
def register_page(request: Request) -> HTMLResponse:
    if needs_initial_setup():
        return RedirectResponse(url="/setup", status_code=303)
    if get_current_user(request):
        return RedirectResponse(url="/", status_code=303)
    return templates.TemplateResponse("register.html", {"request": request, "title": "Register", "auth_error": None, "auth_success": None})

@app.post("/register", response_class=HTMLResponse)
def register_submit(request: Request, username: str = Form(...), password: str = Form(...), password_confirm: str = Form(...)) -> HTMLResponse:

```

```

    if password != password_confirm:
        return templates.TemplateResponse("register.html", {"request": request, "title": "Register", "auth_error": "Паролі не збігаються.", "auth_success": None}, status_code=400)
    ok, message = create_user(DB_PATH, username, password, role="analyst")
    return templates.TemplateResponse(
        "register.html",
        {"request": request, "title": "Register", "auth_error": None if ok else message, "auth_success": message if ok else None},
        status_code=200 if ok else 400,
    )

@app.get("/logout")
def logout(request: Request) -> RedirectResponse:
    request.session.clear()
    return RedirectResponse(url="/login", status_code=303)

@app.post("/sources/upload")
async def upload_source(request: Request, file: UploadFile = File(...)) -> RedirectResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    try:
        if not file.filename or not file.filename.lower().endswith(".csv"):
            raise ValueError("Можна завантажувати лише CSV-файли.")
        safe_name = Path(file.filename).name
        target = INPUT_DIR / safe_name
        content = await file.read()
        temp_buffer = BytesIO(content)
        load_events(temp_buffer)
        target.write_bytes(content)
        app.state.source_catalog = build_source_catalog()
        request.session["flash_success"] = f"Файл {safe_name} успішно завантажено до каталогу data/inbox."
    except Exception as exc:
        request.session["flash_error"] = f"Не вдалося імпортувати файл: {exc}"
    return RedirectResponse(url="/sources", status_code=303)

@app.get("/sources/{filename}")
def download_source(request: Request, filename: str) -> FileResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    target = INPUT_DIR / filename
    if target.exists():
        return FileResponse(target, filename=target.name, media_type="text/csv")
    return FileResponse(str(BASE_DIR / "README.md"), filename="README.md", media_type="text/plain")

@app.get("/exports/{filename}")
def download_export(request: Request, filename: str) -> FileResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    target = EXPORTS_DIR / filename
    if not target.exists():

```

```

result = get_active_result_path(request)
if result is None:
    return FileResponse(str(BASE_DIR / "README.md"), filename="README.md", media_type="text/plain")
target = result
return FileResponse(target, filename=target.name, media_type="text/csv")

@app.post("/analyze", response_class=HTMLResponse)
async def analyze(request: Request, dataset_mode: str = Form("existing"), selected_source: str = Form(""), file:
Optional[UploadFile] = File(None)) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    user = get_current_user(request)
    assert user is not None
    try:
        app.state.source_catalog = build_source_catalog()
        source_map = {item["slug"]: Path(item["path"]) for item in app.state.source_catalog}
        source_titles = {item["slug"]: item["title"] for item in app.state.source_catalog}

        if dataset_mode == "upload":
            if file is None or not file.filename:
                raise ValueError("Оберіть CSV-файл для аналізу.")
            content = await file.read()
            source_df = load_events(BytesIO(content))
            source_name = file.filename
            source_label = "Завантажений CSV"
        else:
            if not selected_source or selected_source not in source_map:
                raise ValueError("Оберіть джерело з каталогу data/inbox або завантаж власний CSV.")
            target_path = source_map[selected_source]
            source_df = load_events(target_path)
            source_name = target_path.name
            source_label = source_titles.get(selected_source, "Локальне джерело")

        trained = ensure_models(source_df, MODEL_DIR, force_retrain=True)
        predicted = predict_probabilities(source_df, trained)
        decided = finalize_decisions(predicted)
        export_filename = "analysis_{user}_{stamp}_{uid}.csv".format(
            user=user["username"],
            stamp=datetime.now().strftime("%Y%m%d_%H%M%S"),
            uid=uuid4().hex[:6],
        )
        export_path = EXPORTS_DIR / export_filename
        decided.to_csv(export_path, index=False)
        summary = summarize_results(decided)
        log_analysis_run(
            DB_PATH,
            username=user["username"],
            source_name=source_name,
            total_events=summary["total_events"],
            critical_events=summary["critical_events"],
            high_events=summary["high_events"],
            avg_risk=summary["avg_risk"],
        )
        evaluation = dict(trained.metrics)
        evaluation["comparison"] = trained.comparison

```

```

evaluation["feature_importances"] = trained.feature_importances
evaluation["dataset_profile"] = trained.dataset_profile
report_uid = uuid4().hex[:10]
evaluation["visuals"] = generate_report_assets(decided, evaluation, REPORTS_DIR, report_uid)
result_id = save_analysis_result(
    DB_PATH,
    username=user["username"],
    source_label=source_label,
    source_name=source_name,
    export_filename=export_filename,
    export_path=export_path,
    metrics=evaluation,
)
request.session["active_result_id"] = int(result_id)
app.state.metrics = evaluation
request.session["flash_success"] = f"Аналіз для джерела {source_name} завершено успішно."
return RedirectResponse(url="/analysis/overview", status_code=303)
except Exception as exc:
    return render_page(request, "dashboard.html", "dashboard", "CloudGuard ML", extra={"error": str(exc)})

@app.get("/admin", response_class=HTMLResponse)
def admin_panel(request: Request) -> HTMLResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    user = get_current_user(request)
    assert user is not None
    if user["role"] != "admin":
        return RedirectResponse(url="/", status_code=303)
    context = build_page_context(request, "admin", "Admin Panel")
    context.update({
        "users": list_users(DB_PATH),
        "history_full": list_history(DB_PATH, None, limit=30),
    })
    return templates.TemplateResponse("admin.html", context)

@app.post("/admin/create-user")
def admin_create_user(request: Request, username: str = Form(...), password: str = Form(...), role: str = Form("analyst")) -> RedirectResponse:
    redirect = require_user(request)
    if redirect:
        return redirect
    user = get_current_user(request)
    assert user is not None
    if user["role"] != "admin":
        return RedirectResponse(url="/", status_code=303)
    create_user(DB_PATH, username, password, role=role)
    return RedirectResponse(url="/admin", status_code=303)

@app.get("/api/health")
def api_health() -> JSONResponse:
    return JSONResponse({"status": "ok", "service": "CloudGuard ML", "time": datetime.utcnow().isoformat() + "Z"})

```

```

@app.get("/api/summary")
def api_summary(request: Request) -> JSONResponse:
    redirect = require_user(request)
    if redirect:
        return JSONResponse({"error": "authentication required"}, status_code=401)
    record = get_active_result_record(request)
    result_path = Path(record["export_path"]) if record else None
    result_df = pd.read_csv(result_path) if result_path and result_path.exists() else None
    summary = summarize_results(result_df) if result_df is not None else empty_summary()
    return JSONResponse({"source": get_active_source(request), "summary": summary, "evaluation":
get_active_evaluation(request)})

if __name__ == "__main__":
    import uvicorn

    uvicorn.run("app:app", host=os.getenv("APP_HOST", "127.0.0.1"), port=int(os.getenv("APP_PORT", "8817")),
reload=False)

```

ДОДАТОК Б Презентація



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ИНТЕЛЕКТУАЛЬНА СИСТЕМА ПРИЙНЯТТЯ РІШЕНЬ ПРИ ВИЯВЛЕННІ НЕСАНКЦІОНОВАНОГО ДОСТУПУ ДО ХМАРНОГО СХОВИЩА ДАНИХ НА ОСНОВІ МАШИННОГО НАВЧАННЯ

виконала: студентка групи ТВ-321 Гуцило Софія Романівна
керівник: Шуклін Герман Вікторович

2026

Актуальність теми

- Хмарні сховища концентрують критичні корпоративні дані та журнали доступу
- Несанкціонований доступ часто маскується під дії легітимного користувача
- Статичні правила і ручний аналіз не виявляють усі поведінкові відхилення
- Потрібна система, що аналізує контекст події та оцінює ризик дії

У роботі зроблено акцент не на разовій перевірці доступу, а на аналізі поведінки користувача в контексті подій.

Проблема

Порушник може мати справжній обліковий запис, токен або ключ доступу, тому формально його дії виглядають дозволеними.

Рішення

Поеднати поведінкові ознаки, моделі машинного навчання та числову оцінку ризику події.

Мета, об'єкт і предмет дослідження. Постановка задачі

Мета

Розробити інтелектуальну систему прийняття рішень для виявлення несанкціонованого доступу до хмарного сховища.

Об'єкт

Процеси доступу користувачів і сервісів до хмарного сховища даних у корпоративному середовищі.

Предмет

Методи машинного навчання, поведінкової аналітики, оцінювання ризику та підтримки реагування.

Постановка задачі

- Проаналізувати загрози хмарним сховищам і обмеження традиційних засобів
- Сформувати модель подій доступу та поведінкові ознаки
- Обґрунтувати вибір алгоритмів машинного навчання
- Спроектувати архітектуру та реалізувати cloudguard ML
- Провести тестування, порівняти підходи та оцінити якість рішень

Аналіз предметної області

Традиційні засоби

IAM, MFA, SIEM, IDS/IPS та статичні правила створюють базовий захист, але не завжди виявляють поведінкову аномалію.

Основне обмеження

Якщо порушник діє через легітимний обліковий запис, події можуть бути формально дозволеними.

Висновок аналізу

Для роботи потрібен гібридний підхід: поведінкові ознаки, моделі виявлення аномалій і оцінювання ризику.

Правиловий підхід

Контрольний рівень для перевірки явних ознак ризику: новий пристрій, географія, MFA, небезпечна операція.

Random Forest

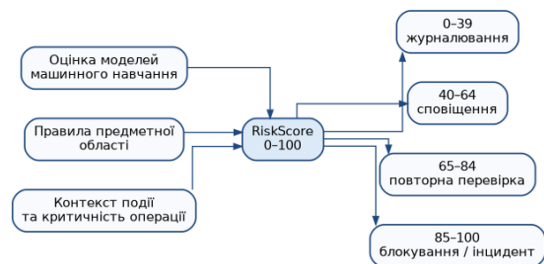
Основна модель для розмічених даних: добре працює з табличними ознаками та дає оцінку важливості факторів.

Isolation Forest

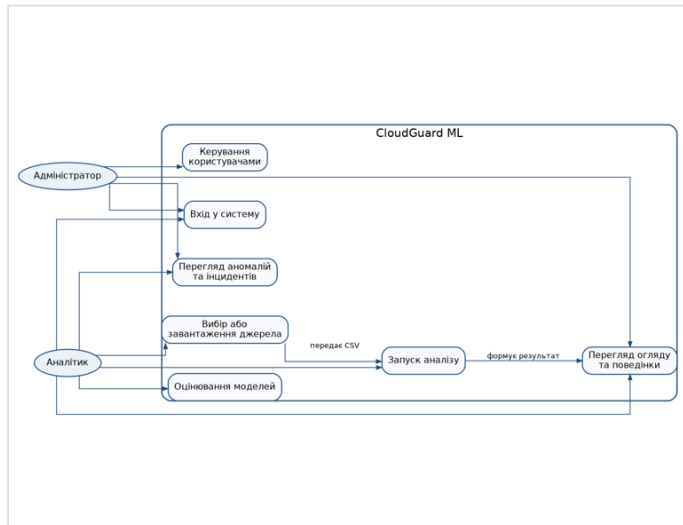
Пошук аномалій у випадках, коли повна розмітка подій відсутня або неповна.

Autoencoder

Нейромережева модель виявляє нетипові події за помилкою відновлення даних.



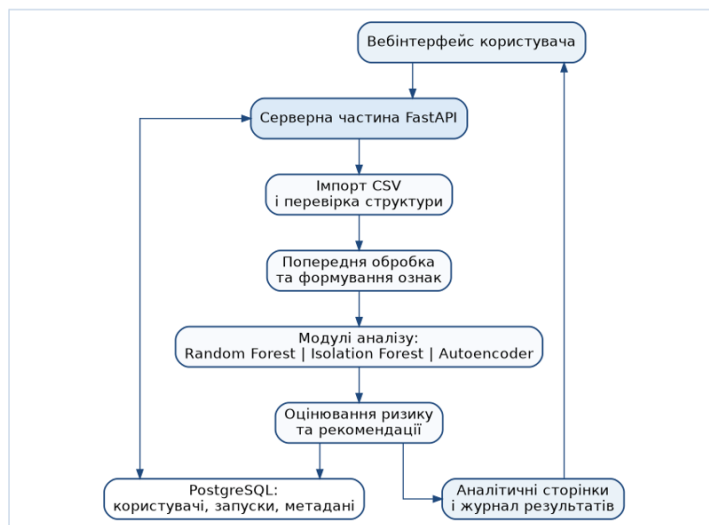
Проектування системи



- Аналітик запускає аналіз і працює з результатами
- Адміністратор додатково керує користувачами
- Центральний сценарій – від вибору джерела до перегляду інцидентів
- Розмежування ролей зменшує ризик неконтрольованих дій у системі

Дві ролі залишено навмисно: цього достатньо для програмної реалізації та не ускладнює модель доступу.

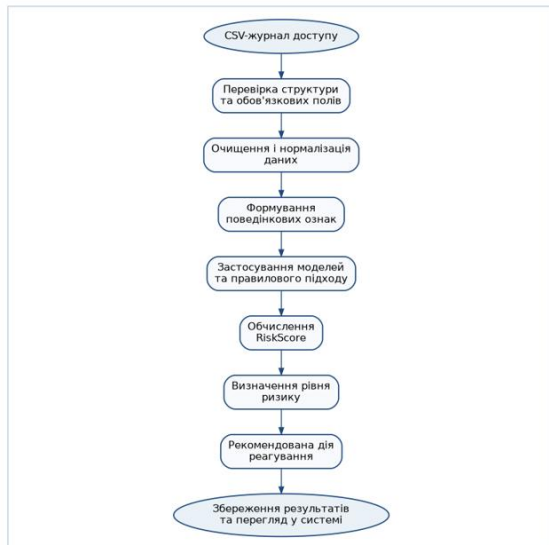
Архітектура системи



- Модульна структура спрощує тестування та супровід
- Обробка даних відокремлена від вебінтерфейсу
- Моделі можна змінювати без перероблення всієї системи
- PostgreSQL зберігає користувачів, запуски та метадані результатів

Архітектуру побудовано як послідовність незалежних компонентів, тому систему можна розширювати новими джерелами даних і моделями.

Алгоритм роботи системи

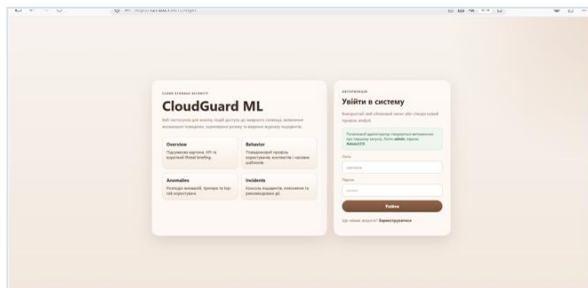


- Вхідним об'єктом є csv-журнал подій доступу
- Система перевіряє структуру, очищує й уніфікує дані
- З подій формуються поведінкові ознаки користувача та контексту
- Після аналізу система видає не лише клас події, а й рекомендовану дію

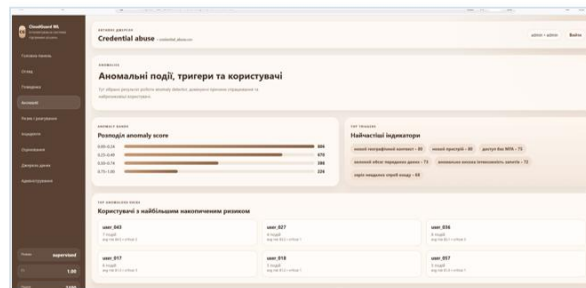
Засоби розробки



Реалізація системи CloudGuard ML



Сторінка авторизації системи CloudGuard ML



Головна панель системи після аналізу джерела

- Вебзастосунок підтримує ролі admin і analyst
- Користувач може вибрати підготовлений набір або завантажити csv-файл
- Результати зберігаються та доступні на окремих аналітичних сторінках
- Інтерфейс побудований навколо огляду, поведінки, аномалій, інцидентів та оцінювання

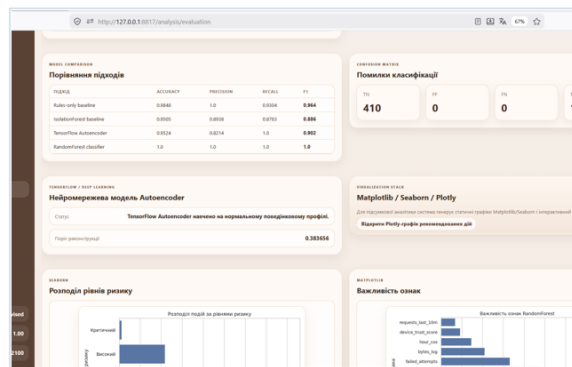
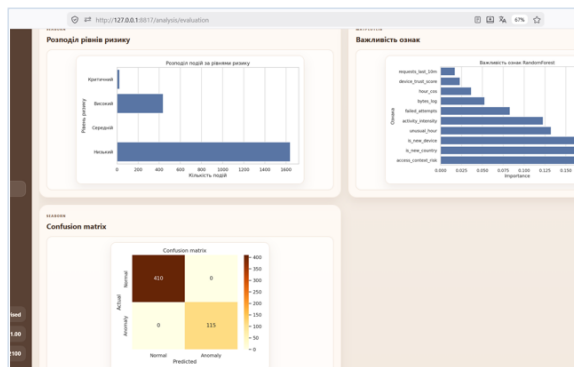
Тестування та експериментальна перевірка

2100 подій
перевірено в активному
наборі

25–26 критичних
подій з найвищим ризиком

436–437 високих
потребують посиленого
контролю

F1 = 1.00
для Random Forest на
тестовому наборі



Практична цінність. Висновки

- Система виявляє ризикові події та пояснює причини спрацювання
- Адміністратор отримує рекомендовану дію реагування
- Зберігається історія запусків і активних результатів аналізу
- Контейнерне розгортання спрощує відтворення середовища
- Прототип можна розвивати як модуль моніторингу доступу

Висновки:

- Проаналізовано загрози хмарним сховищам та обмеження традиційних засобів
- Спроектовано архітектуру інтелектуальної системи cloudguard ML
- Реалізовано імпорт даних, формування ознак, моделі аналізу та оцінювання ризику
- Проведено тестування та підтверджено працездатність системи
- Визначено розвиток: інтеграція з реальними хмарними API, розширення ознак, сповіщення та журнал розслідувань

Мета роботи досягнута: створено програмний прототип, який перетворює журнали доступу на керовані рішення для фахівця з безпеки.



Дякую за увагу

Готова відповісти на запитання