Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра оптимального керування та економічної кібернетики

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Пошук аномалій в даних моніторингу IT систем»

«Searching for anomalies in IT systems monitoring data»

Виконав: здобувач денної форми навчання
спеціальності 113 Прикладна математика
Освітня програма «Прикладна математика»
Циркун Олег Вікторович

Керівник: ст. викл. Платонов В. В. _____
Рецензент: доц. Яровий А. Т.

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від _____ 2023 р.
Завідувач кафедри
_____ _____

Захищено на засіданні ЕК № _____
Протокол № ___ від _____ 2023 р.
Оцінка _____ / _____ / _____
Голова ЕК
_____ _____

Одеса — 2025 р.

ЗМІСТ

ВСТУП

У сучасному світі інформаційних технологій стабільна робота IT-систем має вирішальне значення для функціонування як приватного бізнесу, так і державних установ. З розвитком цифрових технологій обсяг даних, які генеруються різними елементами IT-інфраструктури, зростає в геометричній прогресії. Ці дані охоплюють журнали подій (логи), мережевий трафік, показники продуктивності серверів і додатків, дані про навантаження на ресурси тощо. Моніторинг таких даних дозволяє оперативно реагувати на відхилення від нормальної роботи системи, виявляти потенційні загрози, попереджати збої та здійснювати стратегічне планування ресурсів.

Однак, одним з основних викликів у процесі моніторингу є виявлення аномалій — незвичних або неочікуваних змін у даних, які можуть свідчити про порушення нормального функціонування IT-системи. Такі аномалії можуть бути ознаками як зовнішніх атак, наприклад, DDoS-атак чи спроб несанкціонованого доступу, так і внутрішніх проблем, таких як помилки у конфігурації, збій апаратного забезпечення або перевантаження серверів. З огляду на велику кількість інформації, що генерується системами в реальному часі, традиційні методи аналізу часто виявляються неефективними або надто повільними.

На практиці аналіз логів серверів є одним із найбільш поширених підходів до виявлення аномалій. У логах можуть міститися повідомлення про збої, попередження про нестандартну поведінку систем або записи про підозрілу активність користувачів. Ручна обробка таких даних є малоефективною, оскільки обсяг логів може становити десятки гігабайтів на день, особливо у великих організаціях. Автоматизація процесу виявлення аномалій дозволяє зменшити час реагування на інциденти та підвищити ефективність роботи технічного персоналу.

Іншим прикладом є моніторинг мережевого трафіку, в рамках якого аналізуються обсяги переданих даних, кількість підключень, частота звернень до певних сервісів тощо. Аномалії в трафіку можуть свідчити про початок DDoS-атаки, проникнення у мережу чи витік даних. Своєчасне виявлення таких загроз дозволяє запобігти значним збиткам і забезпечити

безперервність бізнес-процесів.

Також не менш важливою є задача аналізу продуктивності додатків. Наприклад, різке зростання часу відгуку або аномальне використання пам'яті може свідчити про помилки в коді, неефективні алгоритми або проблеми з масштабуванням. Виявлення таких аномалій на ранніх етапах дозволяє запобігти деградації продуктивності та покращити якість обслуговування кінцевих користувачів.

Таким чином, пошук аномалій у даних моніторингу IT-систем є ключовим напрямом сучасної IT-аналітики, який поєднує елементи статистики, машинного навчання, математичного моделювання та кібербезпеки. Актуальність теми дослідження зумовлена низкою ключових чинників. По-перше, це необхідність підвищення безпеки IT-середовища, оскільки аномалії часто є першими індикаторами шкідливої активності. По-друге, актуальність обумовлюється потребою в забезпеченні стабільної роботи систем: своєчасне виявлення потенційних проблем дозволяє мінімізувати ризики простоїв та втрати даних. Крім того, важливою є оптимізація використання ресурсів — аналіз аномальних навантажень допомагає виявити неефективності в роботі інфраструктури. Нарешті, постійно зростають вимоги до продуктивності IT-систем, що потребує безперервного моніторингу ключових показників і оперативного реагування на відхилення від норми.

У зв'язку з цим, дана дипломна робота присвячена дослідженню методів виявлення аномалій у даних моніторингу IT-систем, а також розробці та реалізації підходів, які дозволяють підвищити точність і швидкість виявлення таких аномалій в умовах реального навантаження.

Метою роботи є розробка ефективного інструменту або методології для автоматичного виявлення аномальних подій у великому потоці моніторингових даних. Для досягнення цієї мети необхідно проаналізувати існуючі підходи до пошуку аномалій, вибрати найбільш релевантні для обраного типу даних, реалізувати обраний метод і провести тестування на реальних або змодельованих даних.

Очікувані результати дослідження мають прикладне значення для практичної діяльності IT-фахівців, зокрема для підвищення ефективності роботи систем моніторингу, оптимізації процесів підтримки та обслуговува-

ння інфраструктури, а також забезпечення високого рівня інформаційної безпеки.

РОЗДІЛ 1

ПОСТАНОВКА ЗАДАЧІ ТА ЗБІР ДАНИХ

1.1 Актуальність теми та мета дослідження

У сучасному цифровому світі інформаційні технології стали основою функціонування майже всіх сфер діяльності: від бізнесу до державного управління. Безперебійна робота IT-систем є критично важливою для забезпечення надійності, безпеки та ефективності цифрової інфраструктури. З цієї причини моніторинг IT-систем — процес спостереження за їхнім станом у режимі реального часу — набуває особливого значення.

З розвитком хмарних обчислень, мікросервісної архітектури та розподілених систем обсяги моніторингових даних стрімко зростають. У такому середовищі ручне виявлення відхилень і збоїв стає малоефективним або зовсім неможливим. Аномалії в моніторингових даних можуть вказувати на потенційні збої, несанкціонований доступ, перевантаження або інші критичні інциденти, тому завчасне їх виявлення дозволяє оперативно реагувати на проблеми, зменшити втрати та підвищити загальну стійкість IT-інфраструктури.

У зв'язку з цим актуальним є дослідження методів автоматизованого виявлення аномалій у даних моніторингу IT-систем із застосуванням сучасних підходів до обробки даних, машинного навчання та статистичного аналізу.

Мета дослідження полягає у розробці, реалізації та дослідженні методів виявлення аномалій у даних моніторингу IT-систем з метою підвищення ефективності виявлення проблем, забезпечення стабільності та безпеки цифрової інфраструктури.

Для досягнення цієї мети ставляться наступні задачі:

- проаналізувати джерела та особливості моніторингових даних IT-систем;
- дослідити сучасні методи виявлення аномалій, зокрема статистичні,

машинного навчання та на основі нейронних мереж;
- реалізувати обрані алгоритми для обробки реальних або синтетичних даних;
- провести експериментальну оцінку ефективності підходів;
- сформулювати рекомендації щодо застосування методів виявлення аномалій у практиці IT-моніторингу.

1.2 Постановка задачі

Сучасні IT-системи генерують великі обсяги даних у процесі своєї роботи: журнали подій (логи), метрики використання ресурсів, телеметричні дані, тощо. Ці дані використовуються для моніторингу системного стану, аналізу продуктивності та забезпечення безпеки. Однак серед великої кількості нормальних записів можуть з'являтися аномалії — нетипові, рідкісні або підозрілі значення, які можуть вказувати на потенційні проблеми або збої.

Завдання автоматичного виявлення аномалій полягає у виявленні таких значень або послідовностей значень без попередньо заданих міток. Це завдання ускладнюється високою розмірністю даних, варіативністю нормальної поведінки, наявністю сезонності, шумів і нерегулярностей.

Формулювання задачі можна подати наступним чином:

Нехай задано множину даних моніторингу:

$$X = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^d$$

де x_i — це вектор спостережень або метрик за певний момент часу i, а d — кількість параметрів, що моніторяться.

Потрібно побудувати модель або набір алгоритмів $f : \mathbb{R}^d \to \{-1,1\}$, яка для кожного спостереження x_i визначає, чи є воно аномальним:

$$f(x_i) = \begin{cases} 1, & \text{якщо } x_i \text{ є нормальним.} \\ -1, & \text{якщо } x_i \text{ є аномалією,} \end{cases}$$

Ціль полягає у:

- мінімізації помилок першого роду (false positives) — нормальні дані помилково визначено як аномальні;
- мінімізації помилок другого роду (false negatives) — аномальні дані не були виявлені;
- забезпеченні масштабованості та ефективності алгоритму на великих обсягах даних;
- адаптації до специфіки конкретного типу моніторингових даних (логи, метрики, мережевий трафік тощо).

Таким чином, задача пошуку аномалій у моніторингових даних IT-систем є прикладом задачі безнаглядного навчання або статистичного контролю, де важливо знайти відхилення від нормальної поведінки системи без попереднього знання про можливі типи аномалій.

1.3 Архітектура систем моніторингу IT-інфраструктури

Система моніторингу IT-інфраструктури — це комплекс апаратних і програмних засобів, які забезпечують спостереження за станом компонентів інформаційної системи (серверів, мережевого обладнання, додатків, сервісів тощо) у режимі реального часу. Основна мета таких систем — забезпечення стабільної, безпечної та ефективної роботи інфраструктури за рахунок раннього виявлення відхилень і потенційних збоїв.

Типова архітектура системи моніторингу

Узагальнена архітектура системи моніторингу зазвичай складається з наступних компонентів:

1) **Агенти збору даних** — програми, встановлені на вузлах інфраструктури, які збирають системні метрики (навантаження CPU, використання памяті, диска, мережевий трафік), логи та інші дані.
2) **Системи агрегації та передачі даних** — відповідальні за збір

даних з агентів, їх попередню обробку (наприклад, нормалізацію, фільтрацію) та передачу до центрального сховища. Прикладами таких систем є *Fluentd*, *Logstash*, *Kafka*.

3) **Сховище даних** — бази даних, оптимізовані для зберігання часових рядів (*InfluxDB*, *Prometheus*, *Elasticsearch*) або логів. Це дозволяє ефективно здійснювати ретроспективний аналіз та побудову запитів.
4) **Модуль обробки та аналізу** — містить алгоритми виявлення аномалій, візуалізації, машинного навчання. Може працювати в реальному часі або в режимі пакетної обробки. У цьому модулі реалізуються основні алгоритмічні підходи до виявлення аномалій.
5) **Інтерфейс користувача** — дозволяє адміністраторам та операторам відстежувати стан систем, переглядати метрики та логи, налаштовувати сповіщення. Прикладами є *Grafana*, *Kibana*, *Zabbix UI*.
6) **Система сповіщення** — автоматично повідомляє користувачів про критичні події (через email, Slack, Telegram тощо).

Особливості побудови сучасних систем моніторингу

Сучасні системи моніторингу повинні задовольняти наступні вимоги:

- **Масштабованість** — здатність обробляти великі обсяги даних у реальному часі.
- **Надійність** — стійкість до втрати даних і збоїв у компонентів системи.
- **Гнучкість та модульність** — можливість додавання нових джерел даних або алгоритмів виявлення без значної перебудови системи.
- **Безпека** — захист моніторингових даних від несанкціонованого доступу та їх цілісність.
- **Інтеграція з іншими сервісами** — можливість взаємодії з CI/CD, системами автоматизації, хмарними сервісами.

Таким чином, архітектура системи моніторингу є фундаментом для ефективного виявлення аномалій, збору даних та підтримки стабільності IT-інфраструктури.

1.4 Існуючі підходи до виявлення аномалій

У сфері аналізу даних моніторингу IT-систем виявлення аномалій є ключовою задачею, що дозволяє оперативно ідентифікувати потенційні збої, кібератаки або проблеми продуктивності. Існує низка підходів до вирішення цієї задачі, які умовно можна поділити на статистичні, кластерні, нейромережеві та засновані на експертних правилах.

До найпростіших і водночас інтерпретованих методів належать статистичні підходи. Наприклад, метод Z-оцінки (*Z-score*) оцінює, наскільки значення відхиляється від середнього у одиницях стандартного відхилення. Значення, що виходять за межі певного порогу (наприклад, $|z| > 3$), вважаються потенційними аномаліями. Іншим прикладом є ковзне середнє (*Moving Average*), яке дозволяє згладити часовий ряд і виявити різкі коливання, що можуть свідчити про порушення у роботі системи.

Широко використовуються методи, засновані на кластеризації. Зокрема, алгоритм K-Means дозволяє визначити точки, які мають велику відстань до центру кластера, а отже можуть бути аномальними. Метод DBSCAN, що базується на густоті розміщення точок, дозволяє ідентифікувати віддалені об'єкти, які не належать до жодного кластера, як потенційні відхилення. Такі методи добре працюють у випадках, коли структура даних не є лінійною або містить ізольовані групи.

У більш складних сценаріях, особливо при роботі з часовими рядами або багатовимірними даними, використовуються нейромережеві підходи. Наприклад, автоенкодери (*Autoencoders*) — це мережі, які навчаються відтворювати вхідні дані. Якщо мережа не здатна точно реконструювати певний фрагмент, це може вказувати на аномалію. У випадку часових залежностей застосовуються рекурентні нейронні мережі (RNN), які дозволяють моделювати послідовності і виявляти порушення в закономірностях.

Ще один підхід — системи на основі правил. Вони використовують заздалегідь визначені порогові значення або логіку. Наприклад, якщо навантаження на процесор перевищує 95% протягом 5 хвилин, система може вважати це аномальною ситуацією. Подібні методи легко реалізуються, проте є негнучкими та вимагають постійного оновлення в міру зміни поведінки

системи.

Таким чином, вибір методу виявлення аномалій залежить від природи даних, вимог до точності, обчислювальних ресурсів та необхідного рівня інтерпретованості. У багатьох випадках доцільним є комбінування кількох підходів для досягнення більшої надійності та узагальнюваності моделі.

Висновок

Кожен з описаних підходів має свої переваги та недоліки, і вибір методу значною мірою залежить від типу даних, вимог до точності, швидкості обробки та можливості масштабування. У межах цієї роботи було протестовано кілька методів, зокрема Isolation forest та автоенкодер, з метою порівняння їх ефективності на реальних та синтетичних наборах даних.

РОЗДІЛ 2

ПОБУДОВА ТА ЗАСТОСУВАННЯ МОДЕЛЕЙ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ

2.1 Підготовка даних до моделювання

Коректна підготовка даних є критично важливою складовою процесу побудови моделей виявлення аномалій. Вона включає збирання, очищення, нормалізацію, формування ознак, а також інші етапи, що забезпечують якісне навчання моделей. У цій роботі було використано три основні джерела даних: штучно згенеровані (синтетичні) дані, відкритий набір з ресурсу `Kaggle`, а також реальні дані моніторингу IT-систем, надані науковим керівником.

2.1.1 Синтетичні дані

З метою початкового тестування алгоритмів виявлення аномалій та вивчення їх поведінки в контрольованих умовах, було створено авторський синтетичний набір даних. Такий підхід дозволив повністю контролювати характер та локалізацію аномалій у вибірці, оскільки вони були навмисно закладені вручну. Завдяки наявності точних міток (лейблів) аномальних і нормальних спостережень, стало можливим об'єктивно оцінити якість навчання моделей, зокрема їх точність, повноту та F1-метрику.

Структура даних була представлена у вигляді табличного формату `CSV` з наступними колонками:

- timestamp — позначка часу для кожного вимірювання,
- CPU usage — завантаження центрального процесора,
- RAM usage — обсяг використаної оперативної пам'яті,
- Disk usage — активність жорсткого диска,
- Network traffic — інтенсивність мережевого трафіку,
- is-anomaly — бінарна мітка аномальності.

Кожен рядок таблиці відображає стан IT-системи в конкретний момент часу, а саме в межах однієї хвилини. Усього датасет охоплює 24-годинний період з інтервалом у одну хвилину між записами, що становить загалом 1440 спостережень. Аномальні події було змодельовано шляхом різких змін у навантаженні одного або кількох компонентів, що дозволило перевірити чутливість моделей до різних типів відхилень. Для досягнення варіативності та наближеності до реальних умов було сформовано комплексну систему генерації аномалій, яка включає вставки типових сценаріїв: insert-cpu-spike (раптове зростання навантаження на процесор), insert-memory-leak (витік пам'яті з накопиченням використання RAM), insert-network-burst (стрибок мережевого трафіку), insert-disk-saturation (перевантаження дискової системи), а також insert-composite-anomaly — комбінація кількох одночасних аномалій. Такий підхід дозволив емулювати широкий спектр потенційних інцидентів у IT-інфраструктурі та забезпечив більш реалістичні умови для тестування алгоритмів.

На відміну від ранніх версій штучно згенерованих наборів, які виявились надто спрощеними й містили значний рівень випадковості, оновлений синтетичний датасет дозволив створити кероване середовище для базового тестування моделей та перевірки їх візуальної поведінки.

Незважаючи на обмежену реалістичність у порівнянні з реальними операційними даними, синтетичний набір відіграв важливу роль як інструмент попереднього налагодження моделей, забезпечуючи можливість аналізу результатів у добре відомому середовищі з повністю контрольованими умовами. Це стало особливо корисним на етапі підготовки до переходу до роботи з відкритими або реальними даними, такими як обраний набір з платформи `Kaggle`.

2.1.2 Набір даних з Kaggle

Для перевірки моделей на реальних умовно-реалістичних даних було використано набір *System Resources*[1], один із відкритих наборів із платформи `Kaggle`, який містить інформацію про використання ресурсів одної

[1] `https://www.kaggle.com/datasets/omnamahshivai/dataset-system-resources-cpu-ram-disk-network/data`

комп'ютерної системи (віртуальної машини), зокрема:

- index — позначка часу для кожного вимірювання.
- CPU usage — завантаження центрального процесора,
- RAM usage — обсяг використаної оперативної пам'яті,
- Disk usage — активність жорсткого диска,
- Network traffic — інтенсивність мережевого трафіку.

Дані подані у вигляді часового ряду з часовими мітками у вигляді таблиці з розширенням `csv` (comma-separated values), зібраними з певною періодичністю. Цей набір має структуру, подібну до попередньо згенерованого синтетичного датасету: кожен запис відповідає окремому часовому відрізку з фіксованим інтервалом у 1 хвилину, що дозволяє здійснювати порівняльний аналіз між різними джерелами. Усього набір охоплює приблизно 17 годин спостережень, що забезпечує достатній обсяг для виявлення трендів та потенційних аномалій.

Важливо зазначити, що дані не містять лейблів аномальності, тобто не позначено явно, які саме моменти є відхиленнями. Це робить набір псевдо-реальним — він не є повністю реальним, але й не синтетичним, імітуючи поведінку реальної системи без надання точних відповідей. Така природа даних дозволяє оцінювати ефективність моделей в умовах, наближених до реальних, але вимагає застосування непрямого аналізу результатів моделювання.

2.1.3 Реальні дані моніторингу

Найціннішим джерелом були реальні дані моніторингу IT-системи, отримані від наукового керівника. Ці дані охоплювали метрики роботи серверів, навантаження на мережу, використання ресурсів (CPU, RAM, диски), тощо. Дані представляють набір записів метрик різноманітних віртуальних машин які працюють в одному кластері системи оркестрації (Kubernetes), що значно збільшує вимірність даних, так як кількість метрик множиться на кількість окремих мікросервісів (віртуальних машин) на яких розгоорнуті ті чи інші частини IT-системи. Попередньо дані отримані у форматі JSON (JavaScript Object Notation) і потребують подальшої обробки.

Приклад необроблених даних знаходиться у додатку А. Так само як і попередній набір, тут міститься інформація про використання ресурсів:

- timestamp — позначка часу для кожного вимірювання,
- cpu usage percent — завантаження процесора у %,
- memory usage in bytes — споживання памяті,
- network usage received bandwidth in bytes — мережеву активність на отримання,
- network usage transmit bandwidth in bytes — мережеву активність на передачу.

Цей набір даних теж було переведено до зведеної таблиці типу `csv`, що зберігає структуру, аналогічну попереднім джерелам: записи формуються з інтервалом у 1 хвилину, але цього разу охоплюється період у 10 годин, що забезпечує 600 спостережень. Дані були попередньо оброблені та зведені у табличний вигляд, що значно спрощує їх подальший аналіз та уніфікує структуру.

Однак, ключовою відмінністю цього варіанту є значне збільшення розмірності простору ознак. Замість агрегованих метрик для всієї системи, як у попередніх прикладах, тут фіксується поведінка 19 окремих навантажень (workloads) у кластері Kubernetes, кожне з яких характеризується чотирма основними метриками: використання CPU, памяті, диску та мережі. У результаті формується 76 ознак (19×4), що створює багатовимірний простір, набагато більш реалістичний та складний, типово для виробничих середовищ. Такий набір даних дозволяє моделювати та оцінювати роботу алгоритмів в умовах підвищеної складності, ближчих до реальних сценаріїв експлуатації.

Висновки щодо використаних джерел даних

З огляду на особливості кожного джерела даних, у процесі дослідження було реалізовано поетапну стратегію їхнього використання, яка дозволила всебічно перевірити ефективність обраних моделей.

На початковому етапі було створено та використано синтетичний набір даних, що відіграв ключову роль у формуванні та первинній оцінці

моделей. Завдяки повному контролю над структурою даних та наявності точних міток аномалій, стало можливим візуалізувати поведінку алгоритмів у контрольованому середовищі, де заздалегідь відомо місцезнаходження аномальних точок. Це дозволило детально проаналізувати якість реконструкції, чутливість до різних сценаріїв аномалій, а також оцінити точність виявлення відхилень до застосування моделей на реальних даних. Попри те, що обмежена складність і штучна природа синтетичних даних знижують їхню придатність для остаточного висновку, вони забезпечили надійне середовище для попередньої валідації підходів.

Другим етапом стало використання умовно-реального набору з платформи `Kaggle`, що містить телеметричні дані однієї віртуальної машини з часовими мітками. Цей набір забезпечує низьку вимірність і просту структуру, що дозволило ефективно протестувати здатність моделей до виявлення аномалій в обмежених, але ближчих до практики умовах. Наявність еталонних лейблів також надала можливість кількісно оцінити результати виявлення.

Заключний етап дослідження передбачав застосування моделей до реального набору даних моніторингу продуктивного Kubernetes-кластера, що охоплює 19 робочих навантажень із 76 метриками. Таке ускладнення дозволило перевірити масштабованість підходу, його здатність до узагальнення у багатовимірних просторах та ефективність у реальних умовах експлуатації IT-систем.

Таким чином, побудовано послідовну триетапну схему: первинна перевірка моделей на контрольованому синтетичному наборі; тестування на компактному умовно-реальному наборі; фінальна апробація на складних багатовимірних реальних даних.

Такий підхід забезпечив надійне обґрунтування доцільності використання розглянутих алгоритмів у реальних сценаріях моніторингу та виявлення аномалій.

2.1.4 Попередня обробка та нормалізація

Після збору даних наступним ключовим етапом стало їхнє попереднє опрацювання. Мета цього етапу полягала у приведенні даних до уніфікованого вигляду, зменшенні впливу шумів, відновленні відсутніх значень та підготовці до моделювання.

Оскільки всі метрики мають різні масштаби (наприклад, використання CPU у відсотках, використання памяті у байтах, мережевий трафік у байтах/сек), було здійснено нормалізацію значень кожної ознаки до інтервалу [0,1]. Для цього використано метод мін-макс нормалізації:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Цей підхід дозволив уникнути домінування окремих метрик під час навчання моделей, зокрема у методах, чутливих до масштабу ознак.

Аналіз показав наявність пропущених або пошкоджених записів. Було обрано комбінований підхід до обробки таких ситуацій:

- повністю або частково пошкоджені рядки (тобто, такі що містили значну кількість пропущених значень) були видалені з датасету як непридатні для аналізу;
- окремі відсутні значення в межах рядка (1–2 пропуски) були замінені на середнє значення відповідного атрибута по колонці;

Такий підхід дозволив зберегти основний обєм даних без значних спотворень, при цьому забезпечуючи цілісність структури таблиці.

Особливу увагу приділено обробці реальних даних з кластеру `Kubernetes`, які надійшли у форматі JSON. Ці дані були трансформовані у табличну форму шляхом парсингу вкладених структур і приведення до стандартного формату:

- кожен рядок відповідає певному моменту часу (timestamp);
- ложна колонка відповідає конкретній метриці певного піднавантаження (workload) у системі. Наприклад: `workload1_cpu`, `workload1_ram`, `workload2_cpu`, `workload2_ram`, ...

Загалом було сформовано таблицю з 76 ознаками (19 workloads × 4 метрики), яка представляє стан IT-системи у кожен момент часу. Це дозволило використовувати її як вхід до моделей машинного навчання, орієнтованих на аналіз часових рядів та виявлення аномалій у високовимірному середовищі.

Для забезпечення єдності під час навчання та тестування моделей, обидва основні набори даних (з Kaggle та реальний з Kubernetes) було приведено до уніфікованого формату: таблиця зі структурою *"timestamp + N ознак"*. Це дозволило застосовувати однакові алгоритмічні процедури без додаткового адаптаційного коду для кожного набору окремо.

2.2 Модель Isolation Forest

Одним із ключових етапів дослідження було обрання методу, здатного ефективно працювати з умовно-реальними та реальними часовими рядами метрик IT-систем. Після початкових експериментів з простими статистичними підходами було вирішено зосередитись на моделі **Isolation Forest**, як базовій моделі для виявлення аномалій без потреби у розмічених даних.

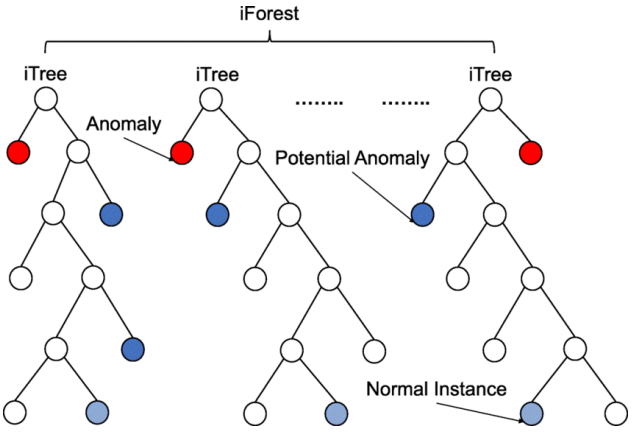

Рис. 2.1. Загальний приклад структури ізоляційного лісу

Причини вибору моделі Модель Isolation Forest була обрана з кількох причин, які безпосередньо відповідають вимогам поточного дослідження.

Відсутність потреби у мітках — модель працює у повністю ненаглядному режимі (unsupervised), що є критично важливим для задач з реальними даними, де аномалії не марковані.

Паралелізація — як і Random Forest, Isolation Forest будує кожне дерево незалежно, що дозволяє ефективно використовувати багатоядерні системи або розподілені обчислення. Це значно зменшує час навчання навіть на великих датасетах.

Масштабованість — модель здатна обробляти великі обсяги високовимірних даних. У дослідженні вона успішно працювала з наборами даних, що включають до 76 ознак (наприклад, метрики з Kubernetes-моніторингу).

Ефективність обчислень — попри використання ансамблю дерев, Isolation Forest є обчислювально легким у порівнянні з багатьма складними моделями, зокрема глибокими нейронними мережами. Це дозволяє застосовувати її в реальному часі або на ресурсно обмежених системах.

Стійкість до пропусків та викидів — як і більшість деревоподібних методів, Isolation Forest толерує відсутні значення та аномальні точки в навчальних даних, що особливо важливо для роботи з "живими"телеметричними даними.

Інтерпретованість — модель повертає скалярний “anomaly score” для кожного об’єкта, що дозволяє легко візуалізувати результати й аналізувати поведінку системи.

Таким чином, Isolation Forest використовується як первинна модель для перевірки гіпотези про виявлення аномалій на різних рівнях складності даних — від Kaggle-датасету до власних моніторингових даних з продуктивного середовища.

2.2.1 Підбір гіперпараметрів

Модель Isolation Forest має низку ключових гіперпараметрів, які значною мірою визначають її поведінку під час виявлення аномалій. До основних параметрів належать:

- n_estimators — кількість дерев у моделі. Збільшення цього параметра підвищує стабільність і точність, але збільшує час навчання;
- max_samples — кількість підвибірок для побудови кожного дерева. Якщо встановлено як число, воно задає фіксовану кількість випадко-

вих точок для кожного дерева. Якщо як частку (наприклад, 0.5), тоді береться відповідний відсоток від усієї вибірки;

- contamination — очікувана частка аномальних точок у даних. Цей параметр безпосередньо впливає на поріг, за яким модель класифікує об'єкти як аномалії. Занадто низьке або високе значення може призвести до недостатнього або надмірного виявлення аномалій;
- max_features — частка ознак, які випадково вибираються для кожного дерева. Зменшення цього параметра підвищує варіативність дерев, що може допомогти уникнути перенавчання, але водночас знижує точність кожного окремого дерева.

	n_estimators	max_samples	contamination	anomaly_ratio	avg_anomaly_score
0	50	auto	0.01	0.010204	0.243219
3	50	0.6	0.01	0.010204	0.271718
6	50	0.8	0.01	0.010204	0.269273
9	100	auto	0.01	0.010204	0.246322
12	100	0.6	0.01	0.010204	0.268414
15	100	0.8	0.01	0.010204	0.268918
21	200	0.6	0.01	0.010204	0.265502
18	200	auto	0.01	0.010204	0.241481
24	200	0.8	0.01	0.010204	0.259752
13	100	0.6	0.05	0.051020	0.077393
4	50	0.6	0.05	0.051020	0.067861
22	200	0.6	0.05	0.051020	0.076367
19	200	auto	0.05	0.051020	0.076882
10	100	auto	0.05	0.051020	0.082125
7	50	0.8	0.05	0.051020	0.058125
1	50	auto	0.05	0.051020	0.073147
25	200	0.8	0.05	0.051020	0.071078
16	100	0.8	0.05	0.051020	0.074187
14	100	0.6	0.10	0.100340	0.045777
5	50	0.6	0.10	0.100340	0.042912
2	50	auto	0.10	0.100340	0.046775
11	100	auto	0.10	0.100340	0.049663
8	50	0.8	0.10	0.100340	0.036739
17	100	0.8	0.10	0.100340	0.042449
23	200	0.6	0.10	0.100340	0.043739
20	200	auto	0.10	0.100340	0.048318
26	200	0.8	0.10	0.100340	0.041023

Рис. 2.2. Результати експериментів з підбору гіперпараметрів

У ході експериментів було проведено систематичне варіювання гіперпараметрів з метою аналізу чутливості моделі до їх значень. Виявилося, що залежно від налаштувань параметрів:

- кількість виявлених аномалій змінювалась майже у 10 разів;
- середнє значення anomaly score для виявлених точок коливалось майже у 6 разів.

Ці результати свідчать про те, що поведінка Isolation Forest є надзвичайно чутливою до гіперпараметрів і потребує уважного налаштування відповідно до специфіки даних.

Оскільки модель застосовується до реальних або умовно-реальних метрик IT-систем, було прийнято рішення обрати більш "нейтральну" конфігурацію, орієнтовану на стабільність, швидкодію та зниження ризику хибних спрацювань.Це рішення було ухвалене з огляду на кілька ключових міркувань. По-перше, більшу частину часу IT-система функціонує в нормальному режимі, тому моделі виявлення аномалій не повинні реагувати на незначні відхилення або шум у даних. Надмірна чутливість моделі могла б призводити до великої кількості хибнопозитивних сповіщень, що, у свою чергу, знижує практичну цінність системи моніторингу та ускладнює прийняття рішень. Крім того, критично важливим фактором є швидкодія, оскільки система повинна масштабуватись на великі обсяги телеметричних даних і працювати в реальному часі без затримок.

Зрештою, було обрано компромісне значення гіперпараметрів, які дозволяють зберегти розумну точність виявлення аномалій без втрати обчислювальної ефективності та з урахуванням реального профілю роботи IT-систем. Кількість дерев у моделі - 100, кількість підвибірок для побудови кожного дерева - 1, очікувана частка аномальних точок у даних - 0.02.

2.2.2 Результати моделі

Першим етапом практичного дослідження стала перевірка поведінки моделі Isolation Forest на синтетичному наборі даних, який було спеціально створено для тестування в контрольованому середовищі. Однією з ключових переваг цього підходу є повна обізнаність про розташування аномальних точок, оскільки вони були вручну вставлені під час генерації даних. Завдяки цьому можна об'єктивно оцінити якість навчання моделі за допомогою класичних метрик класифікації.

Крім того, структурованість даних та контроль над параметрами дозволили виконати додаткову оцінку за допомогою метрик кластеризації, що відображають внутрішню структуру отриманого поділу простору ознак на

нормальні та аномальні області. Це особливо важливо для моделі Isolation Forest, яка працює без використання міток під час навчання (unsupervised learning), тому аналіз як кластерної структури, так і відповідності з реальними аномаліями дозволяє краще оцінити її потенціал.

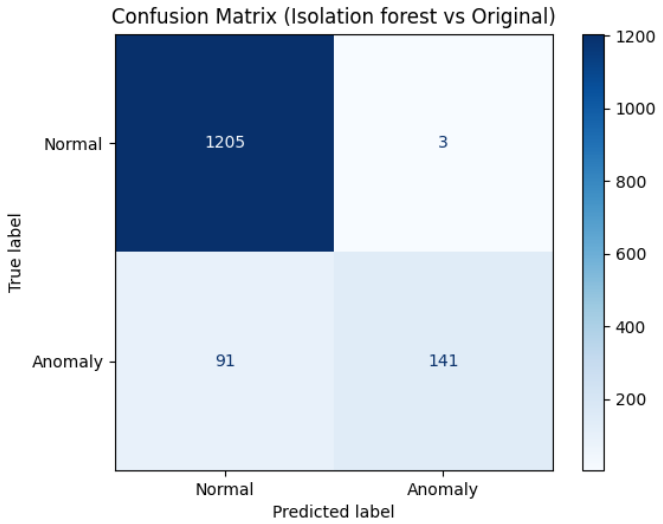

Рис. 2.3. Матриця конфузії (синтетичні дані)

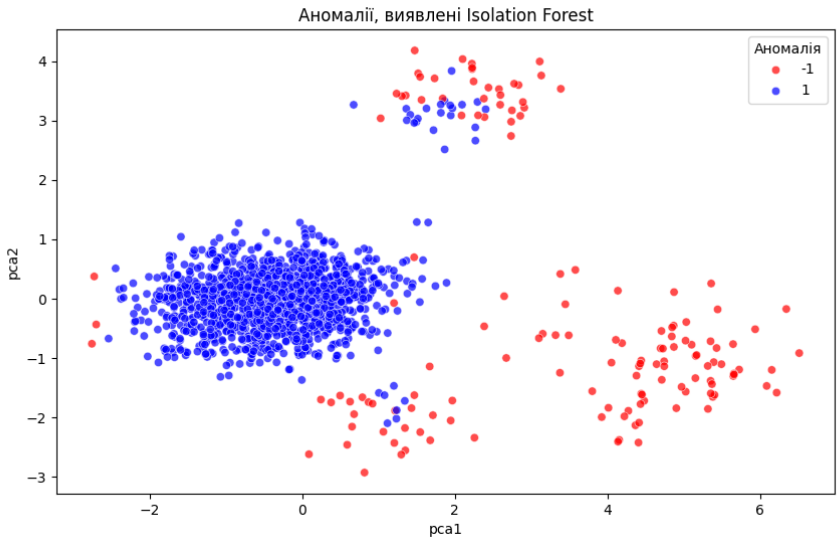

Рис. 2.4. Результат розподілення у двовимірному просторі (синтетичні дані)

У цьому підрозділі наведено результати застосування моделі Isolation Forest до 24-годинного синтетичного датасету з інтервалом у 1 хвилину, що містить 1440 записів. Модель продемонструвала високі значення точності (accuracy) та прецизійності (precision), однак recall залишився дещо нижчим, що вказує на часткову втрату аномальних точок. Це може бути пов'язано з

тим, що модель схильна до консервативного визначення аномалій, віддаючи перевагу впевненим випадкам, ігноруючи менш виражені порушення.

Оцінка за кластерними метриками, такими як Silhouette Score та індекс Девіса-Болдіна, також свідчить про загалом задовільну структурну роздільність між нормальними та аномальними спостереженнями. Однак з огляду на слабке значення recall можна припустити, що в умовах складніших або нелінійних взаємозв'язків між ознаками моделі Isolation Forest бракує глибини представлення, що обмежує її здатність охоплювати весь спектр відхилень. Це свідчить про потенційну доцільність використання більш гнучких моделей, здатних виявляти тонші й комплексні залежності у високовимірнісних просторах. Детальні числові показники наведено нижче.

Метрики класифікації (Isolation Forest vs Реальні мітки)

- Accuracy: 0.935
- Precision: 0.979
- Recall: 0.608
- F1 Score: 0.750

Оцінка кластеризації (Isolation Forest синтетичні дані)

- Silhouette Score: 0.612 (вище — краще, макс 1, мін -1)
- Davies-Bouldin Index: 1.194 (нижче — краще, мін 0, 0.5-1.5 гарне відокремлення, >2 погана кластеизація)

Наступним було використання даних з Kaggle, оскільки в даному випадку відсутні мітки (тобто не відомо наперед, які точки є аномальними), задача виявлення аномалій має ускладнений характер. Проте, результати роботи алгоритму Isolation Forest дозволяють проаналізувати поведінку системи за допомогою оцінки аномальності (anomaly score) кожного зразка.

Візуалізація аномалій у вигляді графіка зміни оцінки у часі демонструє чітке виявлення потенційно аномальних періодів — це видно за різким падінням значення anomaly score нижче встановленого порогу. Це дозволяє зробити попередні висновки про можливі проблеми у відповідні моменти.

Для наочного розуміння структури даних та перевірки здатності моделі до розділення нормальних та аномальних точок, було застосовано метод головних компонент (PCA) зі зниженням розмірності до двох вимірів.

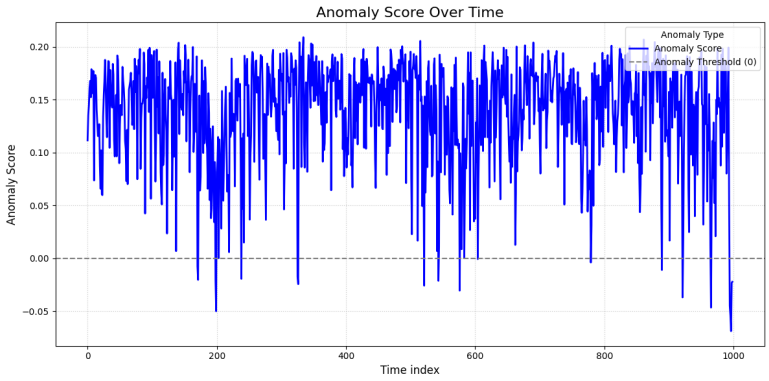

Рис. 2.5. anomaly_score протягом усього датасету (Kaggle)

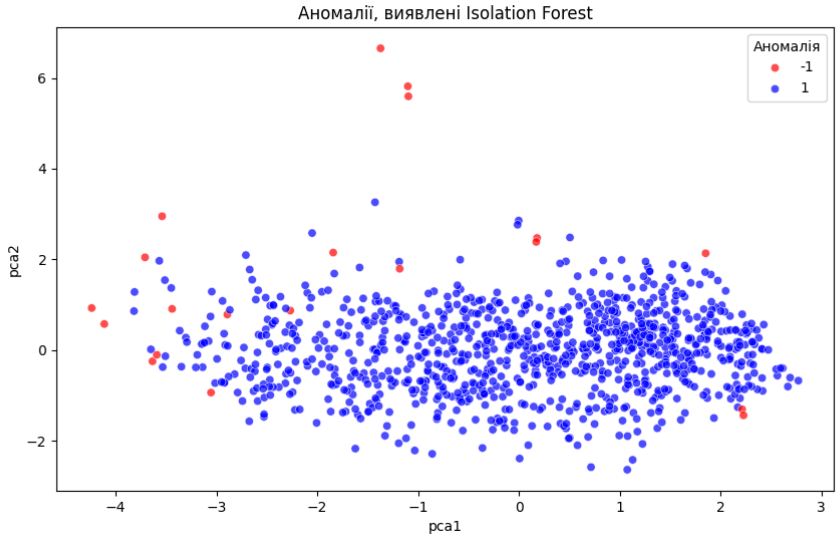

Рис. 2.6. Результат розподілення у двовимірному просторі (Kaggle)

На отриманому scatter plot видно, що аномальні точки (червоні) у більшості своїй чітко відокремлюються від основної маси нормальних (синіх), що свідчить про ефективність моделі у виявленні відхилень.

Реальний датасает демонструє навіть кращі результати попри те, що він являється більш вимірним (ознак більше у 19 разів). Можемо спостерігати відносно стабільну роботу системи з деякими сильними викидами, які перевищують поріг аномальності.

Даний графік підтверджує завдяки PCA, що наші дані мають певний патерн, який є доволі помітний, навіть на зменшеному просторі. Також варто зазначити гарну сегментацію на нормальні та аномальні дані.

Оцінка кластеризації (Isolation Forest реальні дані)

- Silhouette Score: 0.712 (вище — краще, макс 1)

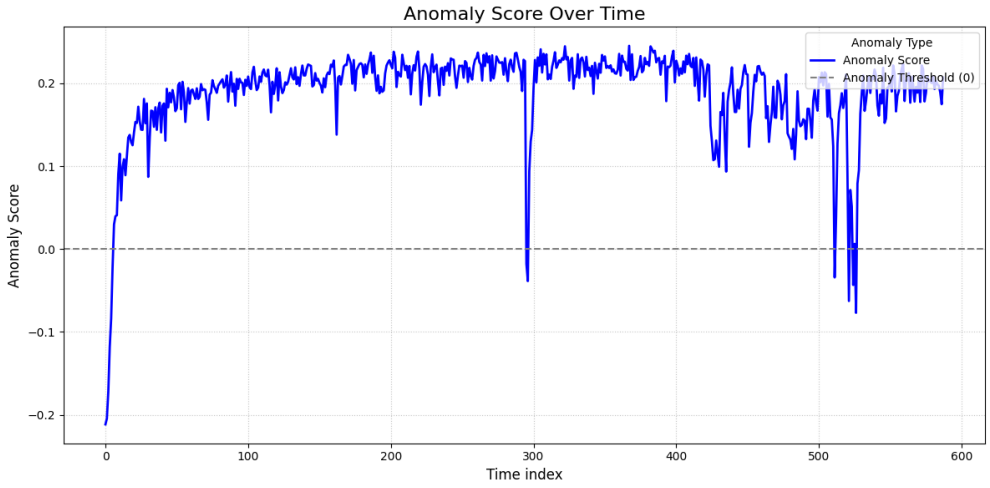

Рис. 2.7. anomaly_score протягом усього датасету (real data)

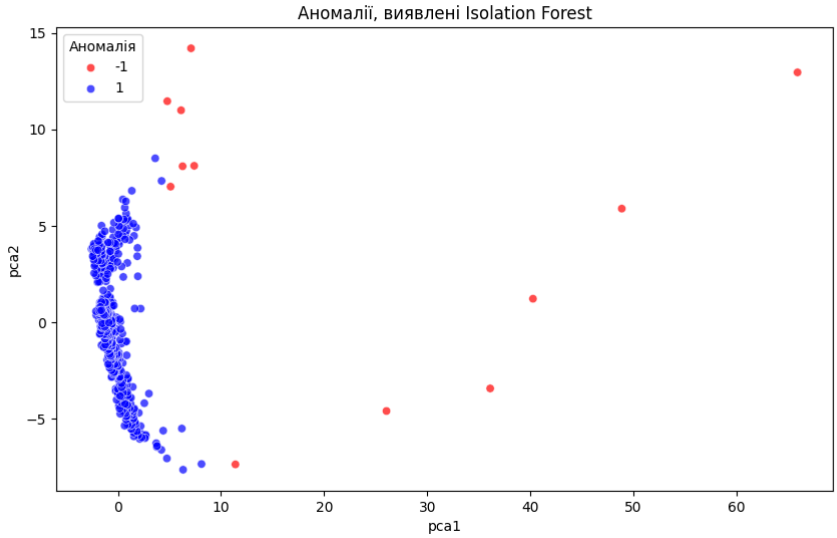

Рис. 2.8. Результат розподілення у двовимірному просторі (real data)

- Davies-Bouldin Index: 1.294 (нижче — краще, мін 0)

З урахуванням швидкості роботи алгоритму Isolation Forest та його здатності виявляти структуру в даних без потреби в мітках, отримані результати можна вважати достатньо переконливими для задачі попереднього моніторингу.

Додаткове дослідження виявлених аномальних векторів підтверджує практичну значущість спрацювань: у більшості з них спостерігалося підвищене використання ресурсів, зокрема CPU та мережевої активності (network usage) в обох внаборах даних. Це підтверджує, що модель дійсно реагує на нетипову поведінку системи, а не на випадкові флуктуації даних, а та-

кож підходить для різних випадків і навіть показує себе краще на більш комплексному та багатовимірному реальному прикладі.

2.3 Модель Autoencoder

Попередній підхід на основі алгоритму Isolation Forest продемонстрував ефективність у виявленні потенційно аномальних зразків, особливо завдяки своїй простоті та швидкості. Проте, зважаючи на те, що ми працюємо з комплексною розподіленою системою (особливо у випадку реального датасету), у якій можуть існувати неочевидні, нелінійні залежності між характеристиками різних віртуальних машин, доцільним кроком є використання більш гнучкого підходу — автоенкодера.

Це тип нейронної мережі, навченої реконструювати вхідні дані. Його архітектура зазвичай складається з двох частин:

- енкодер стискає (кодує) вхідні дані у вектор меншої розмірності — латентний простір;
- декодер відновлює початкові дані з цього стислого представлення.

У процесі навчання мережа вчиться вловлювати найважливіші закономірності у нормальних даних. Якщо на вхід подати аномальний зразок, автоенкодер не зможе його точно реконструювати, і виникне велика похибка відновлення. Це й є сигнал про аномалію.

Автоенкодер є перспективною моделлю для нашого випадку через низку важливих властивостей. По-перше, він здатен працювати з нелінійними залежностями між ознаками, які часто спостерігаються у реальних системах моніторингу віртуалізації. Це дає змогу виявляти аномалії, що базуються на складних взаємозв'язках, які складно вловити лінійними моделями. По-друге, автоенкодери належать до класу самонавчальних моделей (self-supervised), що не потребують міток для тренування. Це особливо важливо в умовах відсутності чітко визначених прикладів аномалій. Крім того, архітектура автоенкодера є гнучкою й легко адаптується під специфіку даних — зокрема, вона дозволяє працювати з багатовимірними метриками з різних віртуальних машин і впроваджувати регуляризацію для покра-

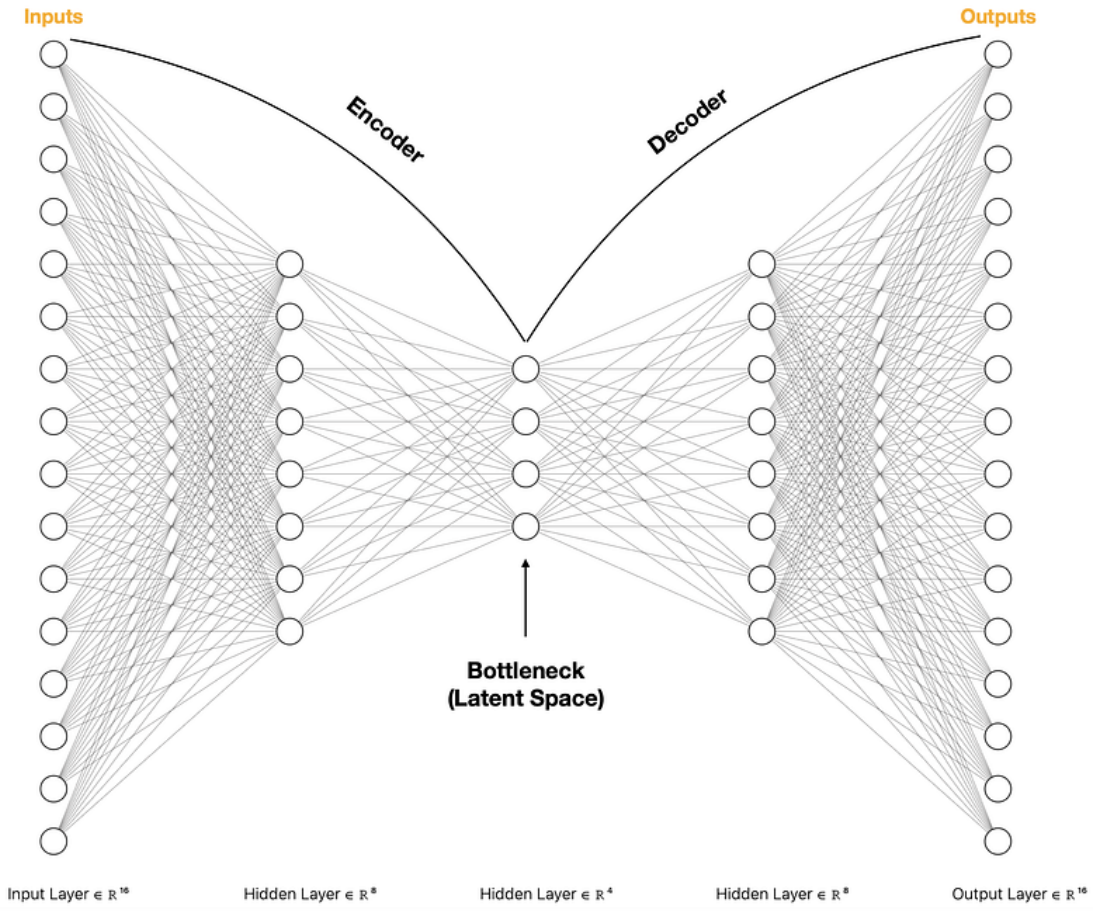

Рис. 2.9. Загальний приклад структури автоенкодера

щення узагальнення. Ще однією перевагою є можливість використовувати латентний простір моделі для зниження розмірності й візуального аналізу даних, що подібно до PCA, але з урахуванням нелінійної структури вхідних ознак. У рамках цього дослідження автоенкодер було навчено на нормальних даних, після чого для кожного вхідного вектора обчислювалась похибка реконструкції, наприклад, за допомогою середньоквадратичного відхилення.

2.3.1 Структура моделі та підбір гіперпараметрів

Для задачі виявлення аномалій у багатовимірних часових рядах було реалізовано оптимізований варіант автоенкодера — нейронної мережі, яка навчається відновлювати вхідні дані, стискаючи їх у латентний простір. Основна ідея полягає в тому, що модель, навчившись реконструювати нормальні дані, демонструє вищу похибку на аномальних зразках, що і використовується для детекції.

Реалізована архітектура є симетричною: вона складається з енкодера та декодера. Енкодер зменшує розмірність вхідного простору до вузького латентного представлення (ботлнеку) розміром 16, проходячи через проміжний шар з 48 нейронів. У декодері виконується зворотне перетворення — відновлення початкового простору з латентного. Як функція активації використовується ReLU, а також додано Dropout-шари з ймовірністю 0.2 для зменшення переобучення. Це дозволяє моделі зберігати узагальнюючу здатність при роботі з шумними даними.

Під час навчання як функцію втрат використовували середньоквадратичну помилку (MSE), яка вимірює різницю між вхідним вектором x і його реконструкцією $\hat{x}$.

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

Оптимізація здійснювалась за допомогою алгоритму Adam з навчальною швидкістю 0.01 і регуляризацією L2 (weight_decay=1e-5), що також сприяє зниженню ризику перенавчання.

Навчання проводилося протягом 50 епох, що забезпечило достатню стабілізацію втрат. Після навчання модель використовувалася для обчислення похибок реконструкції як на тренувальній вибірці, так і на повному наборі даних. Для виявлення аномалій використовувався статистичний підхід: було визначено поріг як середнє значення помилок на тренувальній вибірці плюс три стандартні відхилення ($\mu + 3\sigma$). Зразки з помилкою, що перевищує цей поріг, позначались як аномальні.

Загалом, підбір гіперпараметрів був спрямований на досягнення балансу між складністю моделі та її здатністю узагальнювати залежності в даних. Архітектура була навмисно зменшена за обсягом, щоб зберегти високу швидкодію, при цьому зберігаючи достатню потужність для розпізнавання нелінійних закономірностей. Dropout і L2-регуляризація допомогли покращити стійкість до перенавчання, що критично важливо в умовах відсутності міток класів.

2.3.2 Результати моделі

У наступному експериментальному етапі було здійснено навчання та оцінювання автоенкодера на тому ж синтетичному наборі даних, що і для моделі Isolation Forest. Завдяки наявності ручно вбудованих аномалій, а також відомих міток (label) для кожного спостереження, стало можливим провести пряме порівняння ефективності обох підходів за ключовими метриками класифікації та кластеризації.

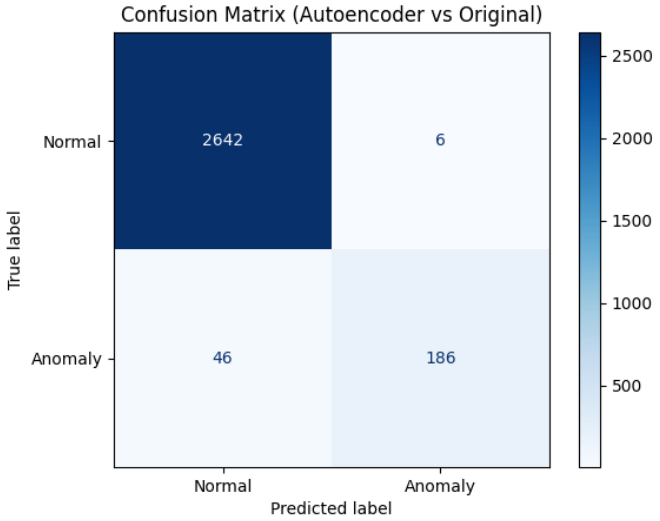

Рис. 2.10. Матриця конфузії (синтетичні дані)

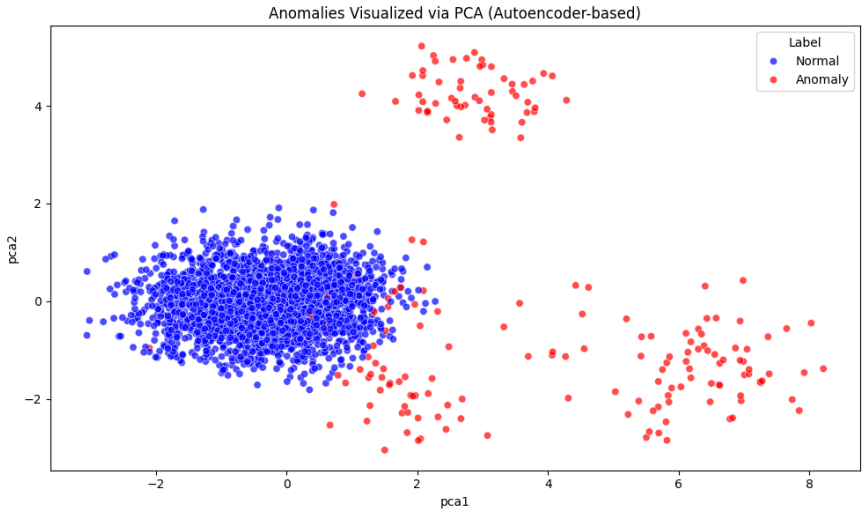

Рис. 2.11. Результат розподілення у двовимірному просторі (синтетичні дані)

За результатами видно суттєве покращення по всіх основних показниках. Значення точності (accuracy) підвищилось до 0.982, а також значно

зросли recall (0.802) та F1-міра (0.878), що свідчить про кращу здатність моделі виявляти як явні, так і менш виражені відхилення від нормальної поведінки. Також помітно зросли значення кластерних метрик — Silhouette Score досяг 0.6895, а індекс Девіса-Болдіна знизився до 1.041, що вказує на чіткіше структурне відділення нормальних і аномальних точок у просторі ознак.

Метрики класифікації (Autoencoder vs Реальні мітки)

- Accuracy: 0.982
- Precision: 0.969
- Recall: 0.802
- F1 Score: 0.878

Оцінка кластеризації (Autoencoder синтетичні дані)

- Silhouette Score: 0.690 (вище — краще, макс 1)
- Davies-Bouldin Index: 1.041 (нижче — краще, мін 0)

Це покращення підтверджує переваги автоенкодера в умовах контрольованого середовища, де модель має змогу навчатися на відносно чистих нормальних даних. Високі результати особливо підкреслюють здатність нейронної мережі моделювати складні, нелінійні залежності між параметрами системи, що є критично важливим для задачі виявлення аномалій у реальних інфраструктурах.

Наступними плодами побудови та налаштування моделі можна вважати пристойні показники loss під час навчання, що свідчить про добре навчання моделі.

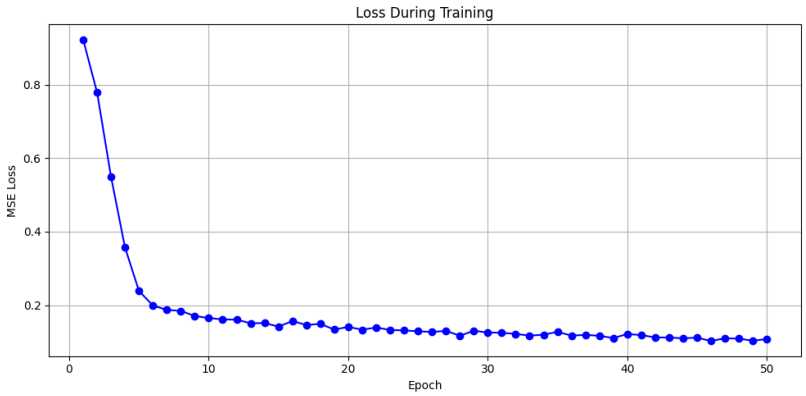

Рис. 2.12. Графік функції втрат протягом навчання

Далі перейдемо до результатів, досліди проводились так само, як і у випадку з Isolation forest спочатку на даних з Kaggle, потім на реальному датасеті. Тренуваня моделі проходили на умовно нормальних даних. Натспуні результати будуть подані відповідно до попередної моделі - спочатку графік реконструкційної помилки (аналог оцінки аномалії) відносно часової мітки, та 2D візуалізація розділення даних за допомогою методу головних компонент для зниження розмірності простору та його відображення.

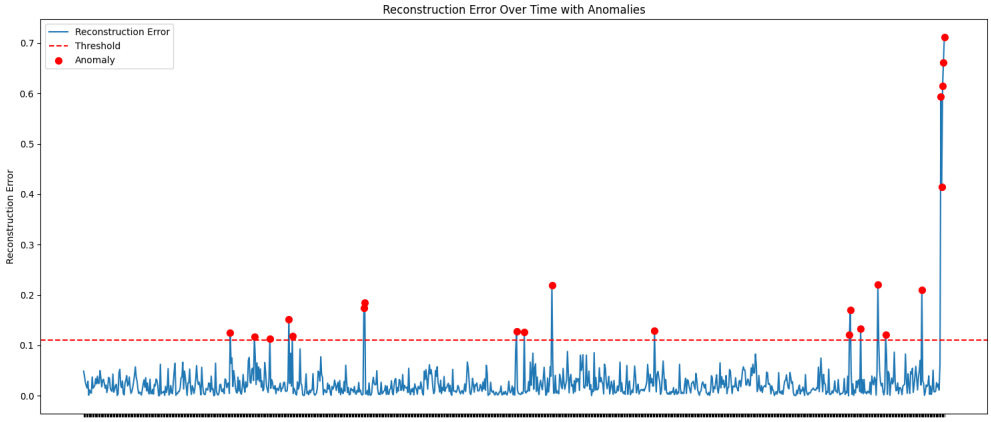

Рис. 2.13. Графік помилки реконструкції відносн часу (Kaggle)

Графік результатів демонструє схожий патерн виявлення аномалій, як і у випадку з попередньою моделлю (ізоляційним лісом), незважаючи на суттєві відмінності в підходах та реалізації. Це узгодження між двома незалежними методами підтверджує високу ймовірність того, що виділені точки справді є аномальними. Водночас, крива реконструкційної помилки автоенкодера виявилась менш шумною, що свідчить про здатність моделі глибше захоплювати структуру вхідних даних і виявляти приховані нелінійні зв'язки між ознаками. У більшості випадків обидві моделі фіксують аномалії в схожих околицях, однак автоенкодер виявляє трохи більшу кількість підозрілих зразків, що можна пояснити його здатністю розрізняти більш тонкі відхилення від норми.

Візуалізація розподілу даних у зниженому до двох вимірів просторі за допомогою методу головних компонент (PCA) також підтверджує ефективність автоенкодера у виявленні аномалій. На відповідному скатер-плоті чітко видно, що виявлені аномальні точки здебільшого зосереджені на краях кластерів нормальних даних, тобто в околицях граничних значень. Це узгоджується з очікуваною поведінкою аномалій, які рідко потрапляють

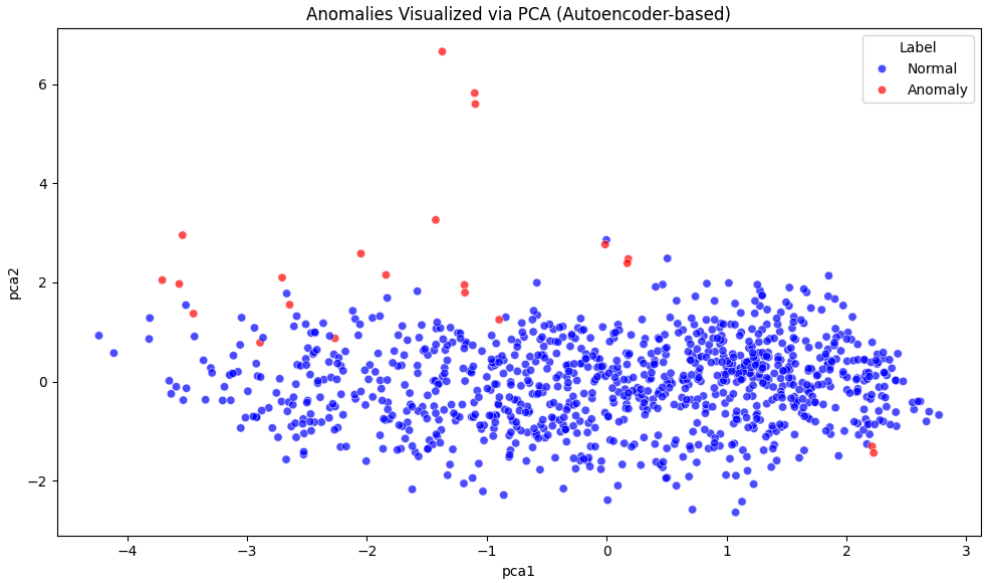

Рис. 2.14. Результат розподілення у двовимірному просторі (Kaggle)

у щільні регіони нормального розподілу. При цьому автоенкодер вловив дещо ширший контекст таких аномальних зон, ніж ізоляційний ліс, що свідчить про його кращу здатність до узагальнення та глибше навчання. Таким чином, результати візуалізації підтримують попередню гіпотезу про переваги автоенкодера у виявленні більш тонких і складних відхилень.

Далі перейдемо до більш комплексного випадку з реальними даними.

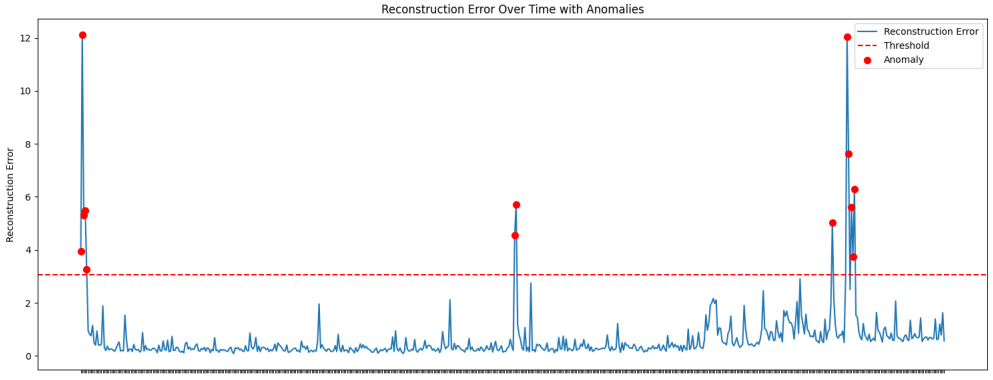

Рис. 2.15. anomaly_score протягом усього датасету (real data)

У випадку з реальним датасетом поведінка автоенкодера залишилася подібною до тієї, що була зафіксована на даних із Kaggle. Графік помилки реконструкції демонструє схожий загальний патерн — основна частина значень зосереджена в межах низької похибки, у той час як поодинокі сплески відповідають потенційним аномаліям. Це узгоджується з очікуваннями після успішного застосування моделі на попередньому наборі даних. Більше

того, автоенкодер знову продемонстрував здатність чітко виокремлювати аномальні точки — їхня кількість виявилася трохи більшою, і вони були виділені з помітно вищою амплітудою, що підтверджує ефективність автоенкодера у виявленні значущих відхилень навіть у складному середовищі реальних систем.

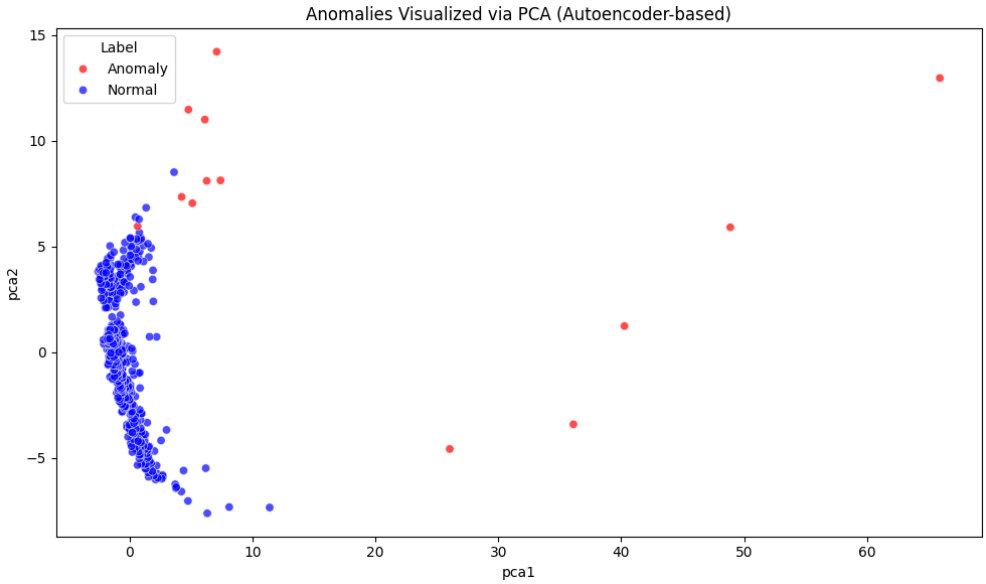

Рис. 2.16. Результат розподілення у двовимірному просторі (real data)

Візуалізація розподілу реальних даних після зниження розмірності за допомогою PCA демонструє наявність одного основного щільного кластера нормальних точок, що чітко відокремлений від менш щільного скупчення аномальних точок. Така структура свідчить про ефективне сегментування даних моделлю: автоенкодер виявляє не лише чітко виражені відхилення, а й більш м'які, граничні випадки. У порівнянні з ізоляційним лісом, кількість виявлених аномалій в прикордонних зонах дещо більша, що вказує на вищу чутливість моделі до прихованих нелінійних залежностей між ознаками, що додатков підтверджується метриками оцінки клстеризації нижче, які поазують покращення відносно попередньої моделі.

Оцінка кластеризації (Autoencoder реальні дані)

- Silhouette Score: 0.785 (вище — краще, макс 1)
- Davies-Bouldin Index: 1.097 (нижче — краще, мін 0)

ВИСНОВКИ

У цій дипломній роботі було проведено порівняльне дослідження двох підходів до задачі виявлення аномалій у даних систем віртуалізації: методу *Isolation Forest* та нейромережевої моделі *автоенкодера*. Обидва методи продемонстрували здатність до успішного виявлення атипових патернів у вхідних даних, зберігаючи узгодженість виявлених результатів як на відкритих тестових вибірках (зокрема, даних з платформи Kaggle / синтетичних), так і на реальних системних метриках.

Зокрема, було показано, що модель Isolation Forest є ефективним інструментом для початкового виявлення аномалій, оскільки вона не потребує попереднього навчання, є інтерпретованою та швидко масштабується на великі обсяги даних. Водночас, автоенкодер — більш складна глибока модель — продемонстрував підвищену чутливість до нелінійних залежностей між ознаками, що дало змогу виявити більшу кількість потенційних аномалій, особливо в граничних просторах розподілу. Це підтверджує гіпотезу про перевагу нейронних підходів у випадках складних, високовимірних структур даних, характерних для систем моніторингу інфраструктури віртуалізації.

На основі отриманих експериментальних результатів можна зробити висновок, що обидва підходи є життєздатними рішеннями за умови належної підготовки даних та адаптації параметрів до предметної області. Проте, зважаючи на комплементарність властивостей обох методів, доцільно запропонувати гібридний підхід, який дозволяє максимально ефективно використовувати переваги кожного з них.

Запропонований автором пайплайн виявлення аномалій має наступну послідовність:

1) Попереднє сканування даних за допомогою моделі Isolation Forest для швидкої локалізації потенційно аномальних областей без потреби в навчанні.
2) Валідація знайдених аномалій з експертною оцінкою, видалення явно некоректних або зашумлених точок.
3) Навчання автоенкодера на очищеній множині даних, що репрезентує нормальну поведінку системи.

4) Доналаштування моделі на актуальних надходженнях даних та використання її в режимі реального часу для аналізу кожного нового спостереження.

Такий підхід дозволяє забезпечити стійке, адаптивне та масштабоване виявлення аномалій, що критично важливо для сучасних розподілених систем. Крім того, важливою перевагою є те, що автоенкодер підтримує донавчання на нових даних без повної перебудови моделі, на відміну від Isolation Forest, що забезпечує гнучкість та сталість у змінних умовах.

Таким чином, об'єднання швидкодії і простоти традиційного алгоритму з глибиною аналізу та адаптивністю нейронної моделі створює передумови для формування *ефективної інтегрованої системи виявлення аномалій*, здатної до роботи як у потоковому, так і в офлайн-режимі, з урахуванням специфіки предметної галузі.

СПИСОК ЛІТЕРАТУРИ

1. Пірсон К. Про прямі та площини найкращого наближення до систем точок у просторі / К. Пірсон // Philosophical Magazine. — 1901. — Т.2, №11. — С.559–572.
2. Джолліфф І. Т. Головні компоненти / І. Т. Джолліфф. — М.: Мир, 1986. — 487 с.
3. Хінтон Г. Е., Салахутдінов Р. Р. Зниження розмірності даних за допомогою нейронних мереж / Г. Е. Хінтон, Р. Р. Салахутдінов // Science. — 2006. — Т.313, №5786. — С.504–507.
4. Гудфеллоу І., Бенджіо Й., Курвіль А. Глибоке навчання / І. Гудфеллоу, Й. Бенджіо, А. Курвіль. — М.: Вільямс, 2018. — 736 с.
5. Лю Ф. Т., Тінг К. М., Чжоу Ч. Х. Ізоляційний ліс / Ф. Т. Лю, К. М. Тінг, Ч. Х. Чжоу // IEEE ICDM. — 2008. — С.413–422.
6. Чандола В., Банерджі А., Кумар В. Огляд методів виявлення аномалій: підходи та додатки / В. Чандола, А. Банерджі, В. Кумар // ACM Computing Surveys. — 2009. — Т.41, №3. — Стаття 15. — 58 с.
7. Аггарвал Ч. Ч. Виявлення аномалій: сучасні методи / Ч. Ч. Аггарвал. — Нью-Йорк: Springer, 2017. — 380 с.
8. Пан Г., Шао Л., Лі Л., Джанг Ч. Глибоке виявлення аномалій: огляд / Г. Пан та ін. // ACM Computing Surveys. — 2021. — Т.54, №2. — Стаття 38. — 38 с.
9. Хан Дж., Камбер М., Пей Дж. Інтелектуальний аналіз даних: концепції та методи / Дж. Хан, М. Камбер, Дж. Пей. — М.: Вільямс, 2012. — 752 с.
10. Гарсія С., Луго А., Родрігес Д. Попередня обробка та нормалізація в машинному навчанні: стан і перспективи / С. Гарсія, А. Луго, Д. Родрігес // Knowledge-Based Systems. — 2015. — Т.74. — С.68–85.
11. Граббс Ф. Е. Виявлення аномальних спостережень / Ф. Е. Граббс // Technometrics. — 1969. — Т.11, №1. — С.1–21.
12. Літтл Р. Д., Рубін Д. Б. Статистичний аналіз пропущених даних / Р. Д. Літтл, Д. Б. Рубін. — Нью-Йорк: Wiley, 2002. — 381 с.

ДОДАТОК А

Приклад того, як виглядають реальні вхідні дані системи мониторігну IT-системи


```
{
  "parameters" : {
    "run" : {
      "datasource" : "prometheus_awspgdev",
      "timerange" : {
        "start" : "2025-03-24T03:39:24.743",
        "end" : "2025-03-28T15:36:31.25"
      },
      "name" : "WALLET2 Metrics",
      "description" : "Some long description",
      "project" : "WALLET2"
    },
    "metadata" : {
      "datasource" : {
        "mysql" : {
          "url" : "172.30.90.63:3306",
          "databaseName" : "jmeter_report",
          "type" : "MY_SQL"
        },
        "prometheus_awspgdev" : {
          "url" : "https://prometheus.awspgdev.gpas.io",
          "type" : "PROMETHEUS"
        }
      }
    },
    "metrics" : [ {
      "name" : "POP.wallet2.cpu.usage.percent",
      "description" : "",
      "type" : "VECTOR",
      "category" : "wallet2.server",
      "datasource" : "prometheus_awspgdev",
      "expr" : "sum(node_namespace_pod_container:container_cpu_usage_seconds_total:sum_rate{cluster=\"\", namespace=\"pop-awspgdev\"}* on(namespace,pod)  group_left(workload,
workload_type) namespace_workload_pod:kube_pod_owner:relabel{cluster=\"\", namespace=\"pop-awspgdev\", workload_type=\"deployment\"}) by (workload, workload_type)\n"
    }, {
      "name" : "POP.wallet2.memory.usage.in.bytes",
      "description" : "",
      "type" : "VECTOR",
      "category" : "wallet2.server",
      "datasource" : "prometheus_awspgdev",
      "expr" : "sum( container_memory_working_set_bytes{cluster=\"\", namespace=\"pop-awspgdev\"} * on(namespace,pod) group_left(workload, workload_type)
namespace_workload_pod:kube_pod_owner:relabel{cluster=\"\", namespace=\"pop-awspgdev\", workload_type=\"deployment\"} ) by (workload, workload_type)\n"
    }, {
      "name" : "POP.wallet2.network.usage.received.bandwidth.in.bytes",
      "description" : "",
      "type" : "VECTOR",
      "category" : "wallet2.server",
      "datasource" : "prometheus_awspgdev",
      "expr" : "(sum(irate(container_network_receive_bytes_total{cluster=\"\", namespace=~\"pop-awspgdev\"}[1m]) * on (namespace,pod) group_left(workload,workload_type)
namespace_workload_pod:kube_pod_owner:relabel{cluster=\"\", namespace=~\"pop-awspgdev\", workload=~\".*\", workload_type=\"deployment\"}) by (workload))\n"
    }, {
      "name" : "POP.wallet2.network.usage.transmit.bandwidth.in.bytes",
      "description" : "",
      "type" : "VECTOR",
      "category" : "wallet2.server",
      "datasource" : "prometheus_awspgdev",
      "expr" : "(sum(irate(container_network_transmit_bytes_total{cluster=\"\", namespace=~\"pop-awspgdev\"}[1m]) * on (namespace,pod) group_left(workload,workload_type)
namespace_workload_pod:kube_pod_owner:relabel{cluster=\"\", namespace=~\"pop-awspgdev\", workload=~\".*\", workload_type=\"deployment\"}) by (workload))"
    } ]
  }
,"data":[{"POP wallet2 cpu usage percent":{"status":"success","data":{"resultType":"matrix","result":[{"metric":{"workload":"iat-worker-
mock","workload_type":"deployment"},"values":[[1743140964.743,"0.10885484365389449"],[1743141024.743,"0.11069807763589402"],[1743141084.743,"0.12026942059140176"],
[1743141144.743,"0.11642976503442566"],[1743141204.743,"0.02106984831100656"],[1743141264.743,"0.01731358240242793"],[1743141324.743,"0.01166074788483012l"],
[1743141384.743,"0.009111406558644956"],[1743141444.743,"0.010417567296378792"],[1743141504.743,"0.010275236237089765"],[1743141564.743,"0.010068744758189646"],
[1743141624.743,"0.010070324517595737"],[1743141684.743,"0.011278033387584215"],[1743141744.743,"0.009910135300103819"],[1743141804.743,"0.009624205924315 3"],
[1743141864.743,"0.009723863743964337"],[1743141924.743,"0.008236471859285409"],[1743141984.743,"0.008201808047591446"],[1743142044.743,"0.008206768217272668"],
[1743142104.743,"0.008347388218474 01"],[1743142164.743,"0.008280205671983166"],[1743142224.743,"0.008474043960992918"],[1743142284.743,"0.008106069093000913"],
[1743142344.743,"0.008372234321299184"],[1743142404.743,"0.008118244368930685"],[1743142464.743,"0.007880678936066 15"],[1743142524.743,"0.007929772054753035"],
[1743142584.743,"0.007470807135889688"],[1743142644.743,"0.007171599678514618 5"],[1743142704.743,"0.007120078645144275"],[1743142764.743,"0.007191176970194385"],
[1743142824.743,"0.007566327421397959"],[1743142884.743,"0.007882225902919301"],[1743142944.743,"0.008231301856348221"],[1743143004.743,"0.008381066281357 34"],
[1743143064.743,"0.008948951259377873"],[1743143124.743,"0.008949569389599789"],[1743143184.743,"0.008760893291140097"],[1743143244.743,"0.008749087755127052"],
[1743143304.743,"0.010102009946500035"],[1743143364.743,"0.009066684920751477"],[1743143424.743,"0.009279564312799629"],[1743143484.743,"0.009437654590442601"],
[1743143544.743,"0.008023070913508961"],[1743143604.743,"0.008093987870364487"],[1743143664.743,"0.008113029775747013"],[1743143724.743,"0.007627484002904824"],
[1743143784.743,"0.007085498908102718"],[1743143844.743,"0.006683929345210164"],[1743143904.743,"0.006469279649875854"],[1743143964.743,"0.006443980316730661"],
[1743144024.743,"0.006652153091484612 5"],[1743144084.743,"0.007282260911792012"],[1743144144.743,"0.007432963486744866"],[1743144204.743,"0.007337803394607973"],
[1743144264.743,"0.007328229229156623"],[1743144324.743,"0.006481665004362782"],[1743144384.743,"0.006359986333754891 5"],[1743144444.743,"0.006020825466356919"],
[1743144504.743,"0.007979544073291702"],[1743144564.743,"0.008193875444328362"],[1743144624.743,"0.008340480429743688"],[1743144684.743,"0.008657005645757127"],
[1743144744.743,"0.008794396395712 83"],[1743144804.743,"0.007375938535247934"],[1743144864.743,"0.007269787883834495"],[1743144924.743,"0.007236543484814983"],
[1743144984.743,"0.007134799470018234"],[1743145044.743,"0.008046692469794909"],[1743145104.743,"0.008678855189574718"],[1743145164.743,"0.014150686001107346"],
[1743145224.743,"0.014091230942398 45"],[1743145284.743,"0.012894269010469181"],[1743145344.743,"0.012593363345676258"],[1743145404.743,"0.006585489387898012"],
[1743145464.743,"0.006520416722151 41"],[1743145524.743,"0.006465854582088098"],[1743145584.743,"0.006791795101617421"],[1743145644.743,"0.006726196358933831"],
[1743145704.743,"0.006724120621241595"],[1743145764.743,"0.006425417243728405"],[1743145824.743,"0.006531558449651997"],[1743145884.743,"0.006165093963411 67"],
[1743145944.743,"0.006110649805487864"],[1743146004.743,"0.006172551796881423"],[1743146064.743,"0.006633199582065118"],[1743146124.743,"0.006659665518381039"],
[1743146184.743,"0.006561004713371465"],[1743146244.743,"0.006525505282991 53"],[1743146304.743,"0.006761340096363539"],[1743146364.743,"0.006261956848484223"],
[1743146424.743,"0.007226199711219856 4"],[1743146484.743,"0.007190920450842846"],[1743146544.743,"0.007144438176269629"],[1743146604.743,"0.007794512954283171 5"],
[1743146664.743,"0.006792348054387269"],[1743146724.743,"0.007379178714704662"],[1743146784.743,"0.007295543831339699"],[1743146844.743,"0.006868946278621165"],
[1743146904.743,"0.006709506322913247"],[1743146964.743,"0.007656343814721618"],[1743147024.743,"0.007533547357984 44"],[1743147084.743,"0.007808613576834345"],
[1743147144.743,"0.008348596537955083"],[1743147204.743,"0.008381742055819803"],[1743147264.743,"0.007506745679564335"],[1743147324.743,"0.007503026706668851"],
[1743147384.743,"0.007520562960248 16"],[1743147444.743,"0.008147869805796914"],[1743147504.743,"0.007531506716486239"],[1743147564.743,"0.007264680868380728"],
[1743147624.743,"0.007079308425169246 5"],[1743147684.743,"0.008040425482702607"],[1743147744.743,"0.008655628205160028"],[1743147804.743,"0.009051037638117602"],
[1743147864.743,"0.009530695222530541"],[1743147924.743,"0.008794148006484061"],[1743147984.743,"0.008130901956164704"],[1743148044.743,"0.007706288821159975 5"],
[1743148104.743,"0.007615508445347124"],[1743148164.743,"0.006727871606094818"],[1743148224.743,"0.006502353807490811"],[1743148284.743,"0.006244298906439967"],
[1743148344.743,"0.006239949049979957 5"],[1743148404.743,"0.006271535645140294"],[1743148464.743,"0.006288202811852802"],[1743148524.743,"0.006505490524516 31"],
[1743148584.743,"0.006733343624389048"],[1743148644.743,"0.006680455431875985"],[1743148704.743,"0.006727813731511215"],[1743148764.743,"0.006792098462245188"],
```


Посиалння на дані з kaggle https://www.kaggle.com/datasets/omnamahshivai/dataset-system-resources-cpu-ram-disk-network/data

Посилання на код використаний для розрахунків та експериментів https://github.com/superolegatron/anomaly-detection