Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель виявлення аномалій в мікросервісних інфраструктурах на основі методу головних компонент

(тема)

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

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

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

Освітня програма
Системне програмування
(повна назва освітньої програми)

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

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Гудзинському Івану Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель виявлення аномалій в мікросервісних інфраструктурах на основі методу головних компонент

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

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

3. Вхідні дані до роботи Набір даних

4. Перелік питань, що потрібно опрацювати у роботі

Що таке мікросервісна архітектура та які її особливості

Які типи аномалій виникають у мікросервісах

Які існують підходи до виявлення аномалій у мікросервісах

Як реалізувати модель PCA на практиці

Як інтегрувати PCA-модель у мікросервісну систему

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 13 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання " 21 " квітня 2025 р.

Здобувач _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 55 с., 9 рис., 11 табл., 1 дод., 23 джерела.

АНОМАЛІЇ, МІКРОСЕРВІСНА АРХІТЕКТУРА, RPCA, ВИЯВЛЕННЯ ВІДХИЛЕНЬ, ВИСОКОВИМІРНІ ДАНІ, ГОЛОВНІ КОМПОНЕНТИ, РОБАСТНИЙ АНАЛІЗ, МОНІТОРИНГ, КІБЕРБЕЗПЕКА, ОБРОБКА ДАНИХ.

Метою кваліфікаційної роботи є розробка моделі виявлення аномалій у мікросервісних інфраструктурах на основі методу головних компонент, зокрема – робастного методу RPCA (Robust Principal Component Analysis), для покращення надійності та безпеки розподілених систем.

У ході виконання кваліфікаційної роботи проаналізовано особливості архітектури мікросервісних систем та проблематику виявлення аномалій у високовимірних даних. Застосовано метод RPCA, що дозволяє розкласти дані на низькорівневу частину (нормальні патерни) та розріджену частину (аномалії), забезпечуючи виявлення відхилень без потреби в попередньому маркуванні. Реалізовано прототип моделі та проведено експериментальну перевірку на тестових даних, що моделюють типові навантаження та збої у мікросервісному середовищі. Результати підтвердили ефективність RPCA у виявленні як одиничних, так і комплексних аномалій, що можуть впливати на працездатність мікросервісів.

ABSTRACT

Master’s thesis: 55 pages, 9 figures, 11 tables, 1 appendices, 23 sources.

ANOMALIES, MICROSERVICE ARCHITECTURE, RPCA, ANOMALY DETECTION, HIGH-DIMENSIONAL DATA, PRINCIPAL COMPONENTS, ROBUST ANALYSIS, MONITORING, CYBERSECURITY, DATA PROCESSING.

The major goal of this thesis is to develop a model for anomaly detection in microservice infrastructures based on the method of principal components, specifically the robust method RPCA (Robust Principal Component Analysis), to enhance the reliability and security of distributed systems. During the course of the work, the features of microservice system architecture and the challenges of detecting anomalies in high-dimensional data were analyzed. The RPCA method was applied, which decomposes the data into a low-rank component (normal patterns) and a sparse component (anomalies), enabling the detection of deviations without the need for prior labeling. A prototype model was implemented and experimentally validated using test data simulating typical loads and failures in a microservice environment. The results confirmed the effectiveness of RPCA in detecting both isolated and complex anomalies that may affect the performance of microservices.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – прикладний програмний інтерфейс (англ., Application Programming Interface)

APM – моніторинг продуктивності застосунків (англ., Application Performance Monitoring)

CPU – центральний процесор (англ., Central Processing Unit)

KPI – ключовий показник ефективності (англ., Key Performance Indicator)

MSA – мікросервісна архітектура (англ., Microservices Architecture)

MSE – середньоквадратична помилка (англ., Mean Squared Error)

PCA – метод головних компонент (англ., Principal Component Analysis)

RAM – оперативна пам’ять (англ., Random Access Memory)

ВСТУП

У сучасних інформаційних системах мікросервісна архітектура стала одним з найпоширеніших підходів до проєктування складних програмних рішень. Вона дозволяє підвищити масштабованість, гнучкість та ефективність обслуговування програмних продуктів завдяки розподілу системи на автономні, незалежно керовані сервіси. Разом з цим, мікросервісні інфраструктури стають усе більш складними, динамічними та чутливими до збоїв. Аномалії, які можуть виникати в роботі окремих сервісів або міжсервісних взаємодіях, часто мають непередбачуваний характер і можуть призводити до критичних відмов системи або деградації її продуктивності.

Завчасне виявлення аномальних поведінкових патернів у мікросервісних середовищах є важливою задачею для забезпечення високої доступності та надійності програмного забезпечення. Проте традиційні методи моніторингу часто не здатні ефективно обробляти великі обсяги багатовимірних і динамічних даних, які генерує розподілена система. У цьому контексті особливу актуальність набувають методи виявлення аномалій, що базуються на математичних підходах до зниження розмірності даних, зокрема – метод головних компонент (PCA).

Метод головних компонент дає змогу виокремити основні напрямки варіації в багатовимірних наборах даних, що дозволяє ефективно зменшувати їхню складність без істотної втрати інформативності. Це особливо важливо для побудови моделей виявлення аномалій, оскільки дозволяє виявляти нетипові відхилення від «нормальної» поведінки системи на основі зниженого простору ознак. Застосування PCA в контексті мікросервісних інфраструктур дозволяє розробити адаптивну модель, здатну аналізувати поведінкові метрики та виявляти потенційні проблеми ще до моменту їх критичного прояву.

Метою цієї роботи є розробка та дослідження моделі виявлення аномалій у мікросервісних архітектурах на основі методу головних компонент, яка б забезпечувала високу точність виявлення при збереженні обчислювальної ефективності. У межах роботи буде проаналізовано існуючі підходи до виявлення аномалій, здійснено збір і попередню обробку даних із мікросервісного середовища, побудовано модель з використанням PCA та оцінено її ефективність на практичних прикладах.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості про виявлення аномалій

Виявлення аномалій у сучасних розподілених системах виявилося нетривіальним завданням. Виявлення відхилень, або аномалій, вивчалося ще в 1887 році [1], і через різноманітну природу даних для аналізу на наявність аномалій досі не існує де-факто методу для виявлення аномалій. Навіть для виявлення аномалій у конкретній області, такій як виявлення аномалій у часових рядах, існує величезна кількість ідей та моделей для виявлення аномалій різного роду. Крім того, дані часових рядів мають такі характеристики, як сезонність, тренди та шум, що ускладнює визначення справжніх аномалій і ще більше ускладнює їх пошук.

Для того, щоб працювати з будь-якою системою, необхідно мати можливість виявляти аномальні умови та можливі першопричини. Зазвичай це називається моніторингом, і це досягається за допомогою інструментів програмного забезпечення та зовнішньої системи, яка збирає та обробляє сигнали, що надходять від системи, яка працює. Нерідко система, що працює, видає сотні різних сигналів за один екземпляр, що робить загальну кількість сигналів, які потрібно відстежувати, дуже великою, іноді на кілька порядків більшою, ніж кількість сигналів, що видаються одним екземпляром системи.

Кінцевою метою моніторингу є правильне виявлення помилкових умов до того, як вони вплинуть на бізнес, надаючи при цьому достатню інформацію для усунення проблеми.

Очевидно, що виявити з цих величезних наборів сигналів, коли система поводиться ненормально, є непростим завданням. Один із способів зробити це – створити оповіщення про інциденти, змодельовані відповідно до складних правил, що застосовуються до випромінюваних сигналів. Це означає, що система моніторингу шукає відомі сценарії збоїв. Невідомі

сценарії відмов зазвичай моделюються після того, як інцидент стався, і був проведений аналіз аномальної події. Таким чином, це ручний процес постфактум. Додатковим підходом є візуальний моніторинг сигналів за допомогою графіків. Однак когнітивне навантаження накладає обмеження на кількість графіків, з якими можна працювати.

Якби ці системи можна було контролювати без необхідності вручну визначати аномалії та сценарії збоїв, це дозволило б системі більш надійно масштабуватися і зробило б величезну кількість сигналів більш керованою. Крім того, якби інформаційні панелі могли визначати пріоритетність контенту, що цікавить, когнітивне навантаження зменшилося б.

1.2 Мікросервісна інфраструктура

Система, що досліджується в цій кваліфікаційній роботі, побудована з використанням мікросервісної архітектури. Використання мікросервісної архітектури дає кілька переваг, які представляють великий інтерес при створенні великомасштабного програмного забезпечення. Деякі з переваг мікросервісної архітектури, визначені Драгоні та ін. [2], є наступними:

- гнучкість. Система здатна йти в ногу з постійно мінливим бізнес-середовищем і може підтримувати всі модифікації, які необхідні для того, щоб організація залишалася конкурентоспроможною на ринку;

- модульність. Система складається з ізольованих компонентів, де кожен компонент робить свій внесок у загальну поведінку системи, замість того, щоб мати один компонент, який пропонує повну функціональність;

- еволюція. Система залишається придатною для обслуговування, постійно розвиваючись і додаючи нові функції.

Як зазначають Драгоні та ін., мікросервісні архітектури дозволяють створювати слабко пов'язані проекти з чітким розподілом прав власності на різні компоненти, тоді як взаємодія між сервісами регулюється за допомогою інтерфейсів API. Таким чином, різні команди можуть нести повну

відповідальність за різні частини системи протягом усього циклу розробки. Великі мікросервісні архітектури, як та, що досліджується в цій роботі, складаються з тисяч окремих мікросервісів. Кожен мікросервіс має багато екземплярів, розгорнутих для того, щоб впоратися з навантаженням або зменшити затримку для кінцевих користувачів через географічні причини.

Побудова великомасштабної архітектури мікросервісів створює додаткову складність. Мікросервіси повинні інтегруватися один з одним і, можливо, з сотнями інших сервісів. Між сервісами можуть існувати залежності, а отже, вони повинні бути організовані в правильних конфігураціях. Збої в роботі одного мікросервісу можуть мати ефект доміно, викликаючи збої в інших частинах системи.

Мікросервісні архітектури та практика DevOps стають стандартом в індустрії програмного забезпечення [3, 4]. Ці дві концепції є ключовими для підтримки швидкої ітерації та розгортання змін у виробничих системах, які обслуговують мільйони та мільярди запитів на секунду від користувачів з усього світу, а отже, повинні бути високомасштабованими та підтримувати безперебійне оновлення.

1.3 Концепція моніторингу мікросервісів

Мікросервіси періодично надсилають дані, які дають уявлення про стан і продуктивність сервісів. Ці сигнали формують часові ряди, які використовуються для моніторингу. До цих показників можуть застосовуватися правила, наприклад, порогові значення, щоб визначити, чи виходять вони за межі визначеної нормальної поведінки. Вони також формують інформаційні панелі графіків для візуального огляду інженерами. Однак аналіз часових рядів є активною сферою досліджень, і протягом багатьох років були запропоновані більш досконалі методи виявлення аномалій. Попит на аналіз даних часових рядів для виявлення інцидентів став настільки необхідним, що перетворився на цілу індустрію з такими

компаніями, як Datadog, Metricly та Unomaly, що пропонують моніторинг як послугу. Тим часом такі веб-гіганти, як Netflix , LinkedIn і Twitter опублікували свої методи виявлення аномалій. Всі ці компанії мають власні моделі виявлення аномалій, а в останні роки багатообіцяючими в цій сфері стали підходи машинного навчання. Одним із прикладів цього є робота дослідницької групи Numenta [6]. Numenta розробила модель нейронної мережі під назвою HTM, яка показала багатообіцяючі результати у виявленні аномалій, перевершивши інші архітектури нейронних мереж, такі як архітектура LSTM. Однак, всі публікації, знайдені на тему HTM, були написані Numenta. Жодного незалежного дослідження не було знайдено.

Також нещодавно алгоритм RPCA привернув увагу до виявлення аномалій. Цей алгоритм лежить в основі бібліотеки виявлення аномалій Netflix [6], однак Netflix є єдиним знайденим джерелом, яке використовує RPCA для пошуку аномалій у часових рядах, і вони не дають жодних вказівок на те, наскільки ефективно він справляється з цим завданням. Історично RPCA в основному використовувався для маніпуляцій із зображеннями, таких як виділення фону та переднього плану, а також для виявлення мережних аномалій.

1.4 Мета та постановка задачі дослідження

Мета цієї роботи – порівняти ефективність використання мереж HTM та RPCA для моніторингу інфраструктури мікросервісу.

Для цього потрібно виконати наступні завдання:

- проаналізувати сучасні підходи до виявлення аномалій у мікросервісних інфраструктурах;
- розглянути теоретичні основи HTM та RPCA;
- збірати та підготувати дані із мікросервісної інфраструктури;
- реалізувати моделі HTM та RPCA для виявлення аномалій;

провести експерименти та оцінити якість моделей.

2 АНАЛІЗ ПІДХОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ

2.1 Виявлення аномалій

Виявлення аномалій – це проблема виявлення в даних закономірностей, які не відповідають нормальній поведінці. Залежно від предметної області ці патерни називають аномаліями, викидами, суперечливими спостереженнями, винятками, аберраціями, несподіванками, особливостями або забрудненнями. Виявлення аномалій має широкі сфери застосування, такі як виявлення вторгнень у кібербезпеці, ненормального стану в охороні здоров'я та виявлення несправностей у критично важливих системах [7].

Виходячи з наведеного вище визначення аномалії, може здатися, що виявити аномалії в системі дуже просто. Простий підхід полягає в тому, щоб визначити область даних, яка відповідає нормальній поведінці, і оголосити будь-які дані, які не відповідають цій області, аномальними. Однак існує багато проблем. Визначення нормальної області, яка охоплює всі можливі варіанти нормальної поведінки, є складним завданням. Наявність мічених даних для навчання/перевірки моделей, що використовуються деякими методами виявлення аномалій та класифікації часових рядів, зазвичай є основною проблемою. Поняття аномалії відрізняється для різних доменів. Наприклад, незначні коливання температури тіла можуть бути серйозною аномалією, тоді як такі ж коливання в іншій галузі, як фондовий ринок, є цілком нормальним явищем.

Таким чином, методи виявлення аномалій з одного домену не обов'язково працюють в інших доменах. У багатьох доменах нормальна поведінка продовжує розвиватися, і поточне поняття нормальної поведінки може бути недостатньо репрезентативним у майбутньому, наприклад, при застосуванні виявлення аномалій до великомасштабної інфраструктури мікросервісів. Характер і поведінка сервісів постійно змінюються - залежні

сервіси додаються і видаляються, ресурси сервісів динамічно масштабуються, а місце розташування сервісу може змінюватися. Всі ці фактори можуть впливати на функціонування сервісу. Це лише деякі з проблем, пов'язаних з виявленням аномальної поведінки.

2.2 Класи аномалій

Аномалії в даних можуть приймати різні форми. Щоб надати єдине поняття різних видів аномалій, наведемо визначення аномалій, запропоноване в роботі [7]. Це визначення поділяє аномалії на три основні класи. Алгоритм виявлення аномалій для часових рядів повинен бути здатним виявляти всі ці класи аномалій.

Точкові аномалії – це точки даних, які відрізняються від нормальних точок даних, більшість досліджених алгоритмів стосуються саме цього типу аномалій. Розглянемо рисунок 2.1, де нормальними даними будуть c_1 і c_2. Дві аномалії можна легко виявити візуально як x_1 і x_2. Це дані, які значно відрізняються від усіх інших даних, і їх часто називають глобальними аномаліями. Що стосується точки x_3, то вона не відповідає поняттю глобальної аномалії. Вона здається схожою на кластер c_2. Однак, дивлячись на локальні точки даних в c_2, ігноруючи глобальний контекст, вважаємо, що x_3 є локальною аномалією. Останній тип аномалії, виявлений на рисунку 2.1, – це точки навколо c_3, так званий мікрокластер. Це всі точки даних, які лежать за межами класифікованих нормальних даних.

Другий клас аномалій – це колективні аномалії. Це клас аномалій, коли окремі точки даних самі по собі не є аномаліями, але їхня поява разом як сукупності є аномальною. Це може бути, наприклад, послідовність дій на веб-сервері представлений в лістингу 2.1

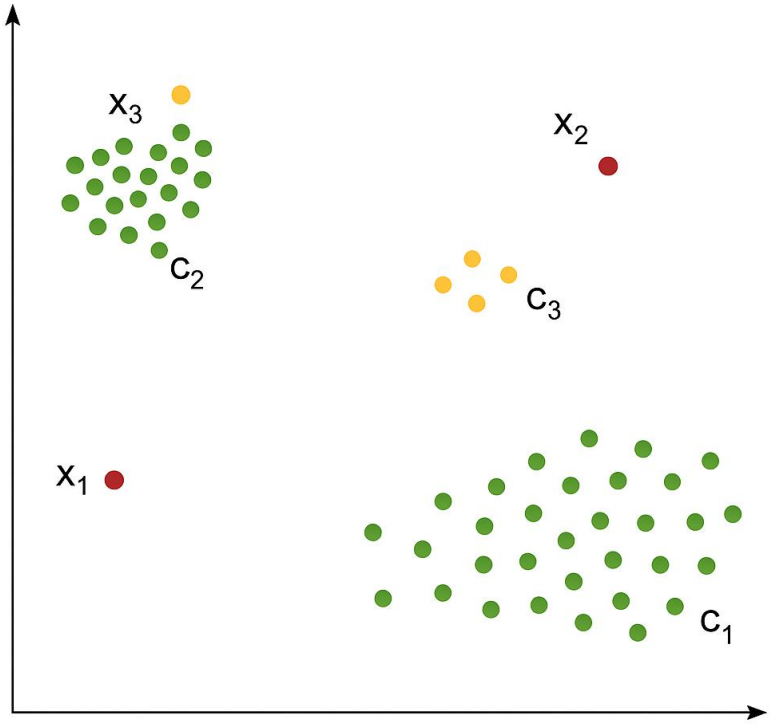

Рисунок 2.1 – Приклад аномалій

Лістинг 2.1 – Послідовність дій на веб-сервері

```
... http-web, buffer-overflow, http-web, http-web, smtpmail,
ftp, http-web, ssh, smtp-mail, http-web, ssh, bufferoverflow,
ftp, http-web, ftp, smtp-mail,http-web...
```

Де кожна окрема подія у виділеній жирним шрифтом послідовності ssh, buffer-overflow, ftp не була б аномальною, проте їх сукупність разом вказує на вторгнення.

Дані часових рядів залежать від контексту і тому мають клас аномалій, які називаються контекстними або часовими аномаліями [8], оскільки кожне значення пов'язане з іншими значеннями в часі. Кожне значення має часовий атрибут, пов'язаний з ним, який визначає його позицію в усьому ряді. Цей атрибут накладає обмеження на те, як можна виявити аномалії. Як приклад, дивіться фіктивне зображення зовнішньої температури протягом року на рисунку 2.2. Температура t_2 є нормальною влітку, проте взимку вона буде аномалією. Це означає, що це не можна вважати простою точковою аномалією, оскільки саме значення не є рідкісним.

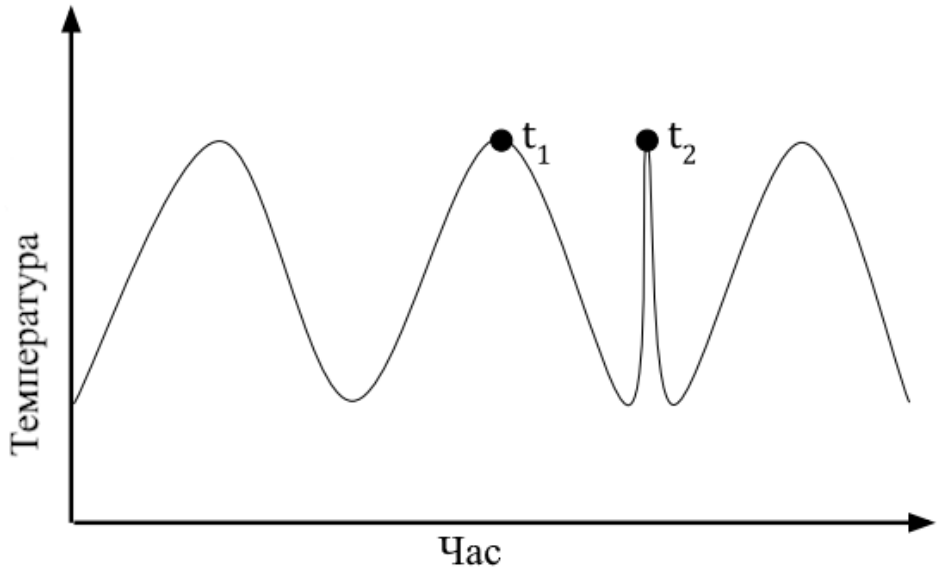

Рисунок 2.2 – Точки t1 і t2 мають однакове значення, але в різні моменти часу

2.3 Класифікація часових рядів

Деякі підходи до виявлення аномалій часових рядів полягають у застосуванні класифікації часових рядів (TSC). Класифікація – це завдання зіставлення вхідного простору з деякою дискретною міткою або класом [9]. Існує багато алгоритмів для класифікації, багато сотень створених спеціально для аналізу даних часових рядів. Однак існують проблеми, пов'язані з TSC. Одна з них полягає в тому, що більшість моделей класифікації використовують підхід керованого навчання [10], а отже, потребують маркованих даних, інша – в необхідності пошуку ознак у даних часових рядів [11]. Існують також проблеми з тим, як проводилися дослідження в цій галузі, оскільки не було універсальної оцінки, і що ще гірше - часто нові опубліковані підходи оцінювалися лише на синтетичних наборах даних з метою демонстрації алгоритму. Спроби виправити це були зроблені, наприклад, архів UCR [12] з маркованими даними часових рядів, а також набір даних, зібраний Гольдштейном та Учидою [13], доступний на Harvard Dataverse. Проте всі оцінки все ще проводяться на одному і тому ж розподілі

тренувань/тестів, а алгоритми класифікації припускають, що патерни, які становлять інтерес, можуть бути правильно ідентифіковані як на етапі навчання, так і під час розгортання.

Виявлення аномалії в часовому ряді можна розглядати як завдання бінарної класифікації. Однак через свою контрольовану природу більшість алгоритмів TSC не зможуть автоматично виявляти аномалії в немічених даних.

Неконтрольовані підходи можуть подолати цей недолік, оскільки вони не покладаються на марковані дані для класифікації. Підходи найближчого сусіда виявилися успішними у виявленні аномалій, а K-NN був визнаний найточнішим методом виявлення аномалій незалежними дослідниками, які використовували різні набори даних. Наприклад, автори в роботі [7] визнали точною міру подібності для часових рядів за допомогою DTW у поєднанні з 1-NN. В роботі [13] використовували K-NN на багатовимірних даних, а не на даних часових рядів. Їхній метод все ще може бути застосований до часових рядів, якщо для вилучення багатовимірних ознак з рядів використовувати попередню обробку. Автори також оцінили інші методи кластеризації для пошуку викидів, вони виявили, що алгоритм CBLOF [14] і модифікований uCBLOF дають хороші результати для глобального виявлення викидів.

Не всі алгоритми класифікації та регресії можуть працювати на необроблених даних часових рядів, для того, щоб бути ефективними, їм потрібно спочатку виділити ознаки, а одним з обмежуючих факторів при прогнозуванні часових рядів є відсутність ознак. Пошук ознак у часовому ряді не є тривіальним завданням. Зазвичай дані часових рядів містять шум і надмірність. Деякі ознаки, наприклад, медіанне значення, будуть стійкими і не будуть сильно залежати від викидів. Інші характеристики, такі як максимальне значення, за своєю суттю є крихкими. Вибір правильних характеристик для часового ряду має вирішальне значення для завдання класифікації. Автори [11] запропонували FRESH, FeatuRe Extraction based on Scalable Hypothesis tests, конвеєр для вилучення ознак з

високомасштабованою продуктивністю, створений для класифікації та регресії. Автори показали, що їхній метод виділення ознак добре працює для бінарної класифікації, і вони припускають, що їхній метод виділення ознак буде точним, якщо його застосувати до задач регресії та багатокласової класифікації.

Процес відбору ознак включав відображення кожного часового ряду в простір ознак, а потім використання статистичних моделей для перевірки гіпотез і збереження лише тих ознак, які, як видається, вказують на релевантність. Таким чином вони зменшили кількість ознак, релевантних для часового ряду, залишивши лише найбільш релевантні.

2.4 Методи скорочення даних для виявлення аномалій

Метою зменшення даних є їхнє компактніше представлення. Існують різні причини для цього, наприклад, коли розмір даних менший, дешевше застосовувати обчислювально дорогі алгоритми, а також покращити продуктивність алгоритмів. Прикладом може бути вилучення ознак. Однак зосередимося на зменшенні обсягу даних як засобі виявлення аномалій.

Спектральні методи виявлення аномалій намагаються апроксимувати дані, фіксуючи основну частину варіабельності даних за рахунок зменшення їх розмірності. Ключове припущення полягає в тому, що дані можуть бути вбудовані в простір меншої розмірності, в якому нормальні та аномальні дані виглядають значно відмінними. Наочний приклад наведено на рисунку 2.3. Найпоширенішим підходом до цього є аналіз головних компонент (Principal Component Analysis, PCA) [15], який розкладає початкову матрицю даних M наступним чином:

$$M = L + N. \tag{2.1}$$

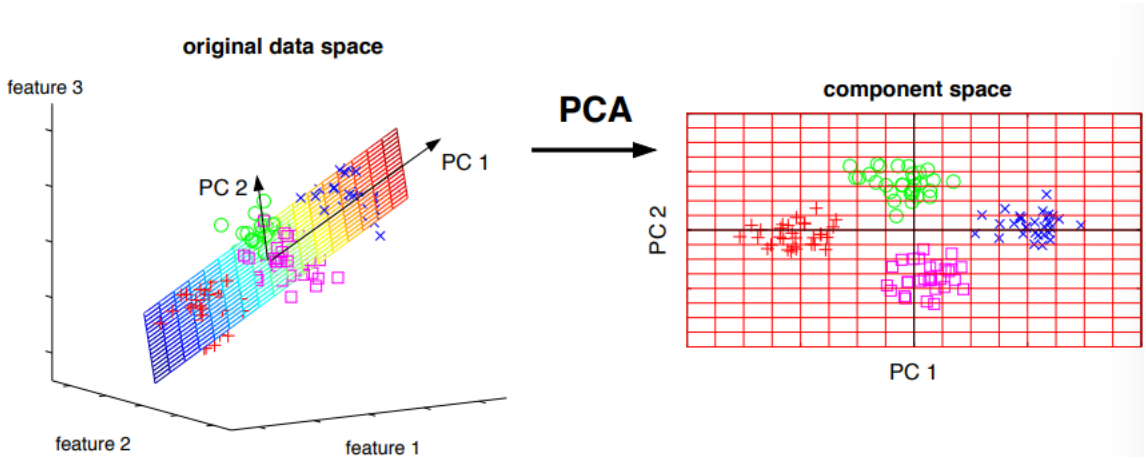

Рисунок 2.3 – Приклад PCA, де дані проектуються в простір меншої розмірності [16]

Де L має низький ранг, а N – мала матриця збурень. Щоб розв'язати цю декомпозицію, PCA шукає найкращу (в сенсі l^2) оцінку рангу k для L, розв'язуючи

$$\begin{aligned} &minimize \ \|M\ L\| \\ &subject\ to\ rank(L) \leq k \end{aligned} . \tag{2.2}$$

Це можна ефективно вирішити за допомогою таких методів, як розкладання за сингулярними значеннями (SVD). Однак, PCA є крихким, коли має справу з сильно пошкодженими даними, що призводить до того, що знайдений L може бути як завгодно далеким від реального L. Пошкоджені дані можуть бути спричинені відмовами датчиків, оклюзіями або просто нерелевантними даними для низькорозмірного представлення .

Більш надійний PCA (RPCA) привернув увагу до виявлення аномалій у часових рядах після того, як компанія Netflix опублікувала статтю про виявлення ними аномалій [17]. Їхній підхід базується на RPCA, використовуючи метод переслідування головної компоненти за змінними напрямками (PCPAD). Подібно до алгоритму PCA вище, початкова матриця M розкладається на дві компоненти:

$$M = L + S, \tag{2.3}$$

де L – низькорангова, а S – розріджена. Фактично S містить вилучені аномалії з вихідних даних. Задача полягає в тому, щоб знайти матрицю низького рангу L і розріджену матрицю S без попереднього знання істинного рангу L або кількості елементів у S. Її можна сформулювати як оптимізаційну задачу на кшталт наступної

$$\begin{matrix} min_{L,S}\ \rho(L) + \lambda\|S\|_0 \\ subject\ to\ |M\ (L + S)| = 0 \end{matrix}, \tag{2.4}$$

де ρ(L) – ранг L, $||S||_0$ – кількість ненульових елементів у S і – константа зв'язку, яка контролює компроміс між низькоранговою матрицею L і розрідженою S. Грубий перебір S і L вимагає перевірки всіх можливих комбінацій аномалій у S і низькоранговій L і був би NP-важким. Кандес та ін. показали в теоремі 1.2 в Robust Principal Component Analysis [15], що насправді можна дати гарантії для знаходження S та L за умови виконання деяких умов:

- обмежений ранг L;
- обмежена розрідженість S;
- вимагається, щоб стовпці L були некогерентними далеко від стандартного базису;
- ненульові записи в S повинні бути рівномірно розподілені.

Припускаючи, що умови виконано, задачу можна переформулювати у вигляді опуклої програми

$$\begin{matrix} min_{L,S}\ \|L\| + \|S\|_1 \\ subject\ to\ |M\ (L + S)| \prec \varepsilon \end{matrix}, \tag{2.5}$$

де $\|L\|$ – сума сингулярних значень в L і $\|S\|_1 = \sum_{ij} |S_{ij}|$. так само, як у (2.12), а ε – набір точкових обмежень на похибки, що використовуються для покращення шуму. Такі типи опуклих задач можна ефективно розв'язувати.

Зауважте, що метою в цьому контексті є не відновлення S і L, а скоріше виявлення аномалій.

2.5 Прогнозування часових рядів

Іншим підходом до виявлення аномалій у часових рядах є прогнозування майбутніх значень. Створивши модель, здатну точно передбачити часовий ряд, можна виявити аномалії, дивлячись на різницю між прогнозом і фактичним значенням. Використання моделей прогнозування для виявлення аномалій, однак, вимагає, щоб модель була точною, оскільки в іншому випадку помилкові спрацьовування можуть переважити переваги детектора.

Останнім часом для прогнозування часових рядів використовують штучні нейронні мережі. Штучні нейронні мережі – це сімейство алгоритмів, відповідальних за основні досягнення в таких галузях науки про дані, як комп'ютерний зір, розпізнавання мови, обробка природної мови та розпізнавання звуку. Прикладом може слугувати сервіс транскрипції голосу Google, який використовує LSTM [18].

Для прогнозування часових рядів рекурентні нейронні мережі (RNN) показали великі перспективи. Особливо багатообіцяючі результати були отримані при використанні моделі довгої короткочасної пам'яті (LSTM). LSTM – це нейромережева модель, яка, як було встановлено, дає хорошу точність прогнозування для даних часових рядів, перевершуючи інші типи рекурентних нейронних мереж завдяки своїй здатності зберігати довготривалу пам'ять про раніше бачені послідовності.

Іншою нейромережною моделлю, яка показала чудові результати, є модель HTM, яка, як було показано, перевершує LSTM у прогнозуванні часових рядів.

Ієрархічна тимчасова пам'ять (HTM) – це тип нейронної мережі, який суттєво відрізняється від традиційних нейронних мереж. Модель HTM базується на обчислювальних принципах неокортексу [19]. Майже всі штучні нейронні мережі моделюють нейрони як "точковий нейрон", де обчислюється єдина зважена сума вхідних даних нейронів [20]. Приклад можна побачити на рисунку 2.4 A. HTM натомість використовує пірамідальну модель нейрона, яка намагається імітувати роботу реального нейрона в мозку, з тисячами синапсів, і навчається, моделюючи зростання нових синапсів. На рисунку 2.4 показано, як HTM (C) порівнюється з біологічним нейроном (B).

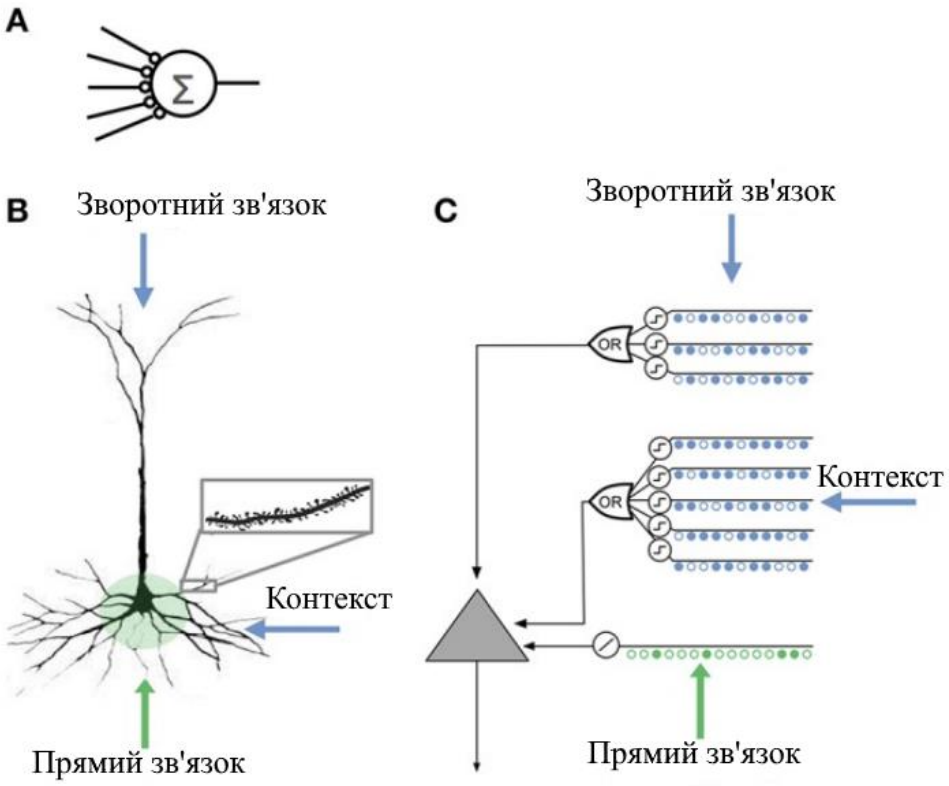

Рисунок 2.4 – Порівняння штучної нейронної мережі (A), біологічного нейрона (B) та нейрона HTM (C)[20]

Унікальність НТМ порівняно з іншими штучними нейронними мережами полягає в їхній здатності безперервно навчатися, робити висновки та запам'ятовувати складні закономірності вищого порядку. Більшість інших нейронних мереж навчаються пакетно, тоді як НТМ-мережі можуть навчатися безперервно. НТМ також стійкий до шуму і здатний до високої вхідної ємності.

Для того, щоб мати можливість емулювати тисячі синапсів, НТМ використовує спеціальне кодування значень, яке називається розрідженим розподіленим представленням (Sparse Distributed Representation, SDR). SDR, що використовуються в НТМ, – це двійкові матриці даних, де кожен біт в SDR представляє нейрон. Як і в мозку, більшість нейронів неактивні. У типовому представленні дані можуть мати 2048 стовпців з 32 нейронами в кожному стовпці, загалом 64 тис. нейронів. Ці SDR використовуються як всередині ШНМ, так і для кодування вхідних даних мережі. Однак, щоб передбачити подібні послідовності даних, внутрішні SDR відрізняються від зовнішніх, щоб зафіксувати ті самі дані в різних послідовностях. На рисунку 2.5 показано спрощений приклад кодування SDR, що складається з 21 стовпчика з 6 клітинками в стовпчику. Ці панелі представляють зріз через один клітинний шар в неокортексі (рисунок 2.5A). На рисунку 2.5B можно бачити дві послідовності, ABCD та XBCY, закодовані у відповідні SDR. Внутрішньо ці послідовності транслюються на різні SDR, оскільки підпослідовність BC міститься в обох послідовностях, існує потреба в різних внутрішніх представленнях. Більше інформації про те, як працює пам'ять послідовностей з тисячами синапсів, можна знайти в [20].

Зверніть увагу, що символи A і B мають однакове кодування до навчання, але різне після навчання.

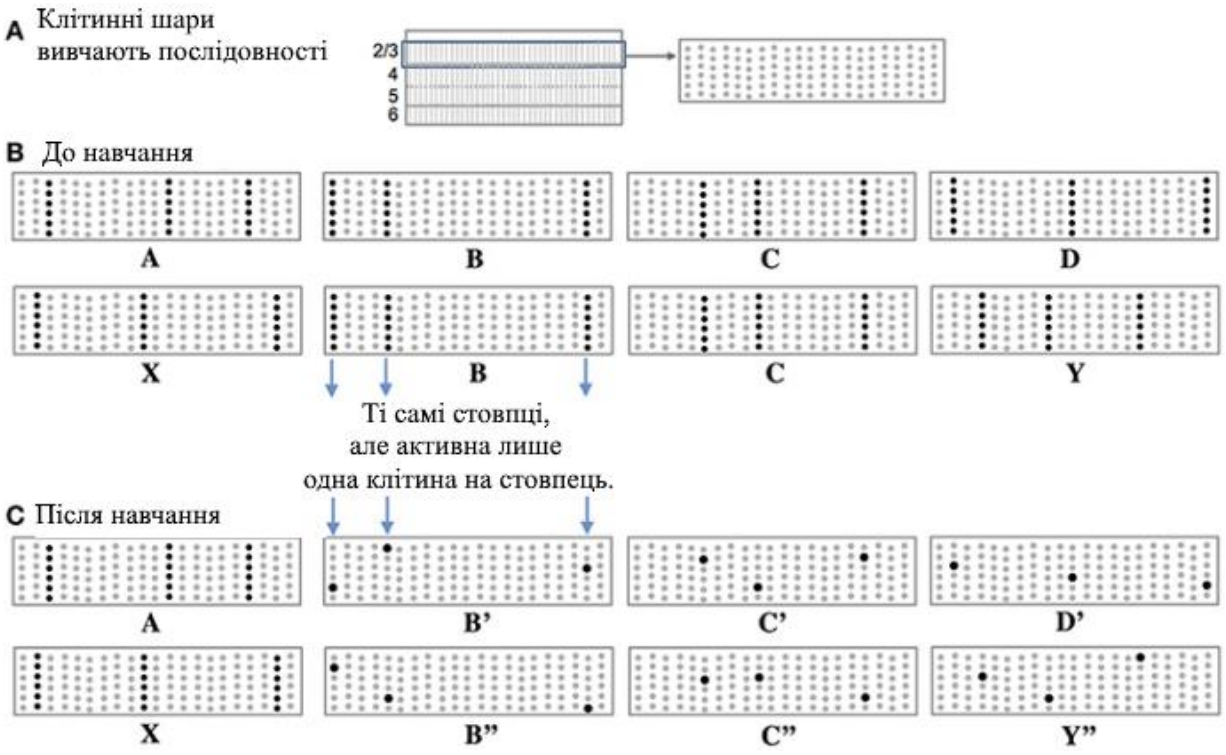

Рисунок 2.5 – SDR до і після вивчення послідовності.

Здатність мережі НТМ обробляти часовы дані та її здатність постійно адаптуватися до нових даних робить її добре придатною для виявлення аномалій часових рядів, де часовий ряд може змінюватися з плином часу. Попередні дослідження показали, що виявлення аномалій у часових рядах на основі НТМ перевершує інші неконтрольовані онлайн-алгоритми, такі як сезонна гібридна модель Twitter на основі ESD та Skyline (статистичний детектор аномалій з відкритим вихідним кодом).

У порівняльному дослідженні НТМ та інших нейромережевих моделей для онлайн-навчання послідовностей з потоковими даними було показано, що НТМ перевершує інші нейронні мережі, включаючи LSTM

НТМ є єдиною ідентифікованою нейронною мережею, здатною до справжнього онлайн-навчання. LSTM показали багатообіцяючі результати у прогнозуванні часових рядів, але їх потрібно навчати партіями, а також перенавчати, якщо характеристики даних змінюються в часовому ряді. Було виявлено, що їхні прогностичні характеристики схожі між собою, якщо використовується відповідне вікно перенавчання, однак творці НТМ

виділяють кілька переваг їхньої архітектури, таких як можливість робити одночасні прогнози, відсутність необхідності перенавчати модель для адаптації до нових даних, стійкість до шуму і відсутність необхідності в налаштуванні гіперпараметрів.

Як згадувалося у вступі до прогнозування часових рядів, моделі прогнозування виявляють аномалії, використовуючи різницю між прогнозом і фактичним значенням. Спосіб використання цієї різниці залежить від моделі прогнозування. Простим способом є використання залишку між прогнозом і реальним значенням. Іншим більш складним підходом для виявлення більш складних аномалій, таких як клас колективних аномалій, може бути врахування кількості послідовних аномалій.

Детектор аномалій на основі HTM, описаний у роботі [8], використовував наступну формулу.

$$S_t = \frac{\pi(x_{t-1}) \cdot a(x_t)}{|a(x_t)|}, \tag{2.6}$$

де S_t – оцінка прогнозу, x_t – поточні вхідні дані, $a(x_t)$ – SDR-кодування вхідних даних, і $\pi(x_{t-1})$ – SDR, що представляє передбачення мережі ШНМ щодо $a(x_t)$. | $a(x_t)$ – скалярна норма $a(x_t)$, тобто загальна кількість одиниць у $a(x_t)$. Рівняння (2.6) буде значенням на інтервалі [0, 1] залежно від того, наскільки прогноз перекриває фактичне значення. Якщо прогноз точно збігається з фактичним значенням, S_t буде 0, якщо прогноз і фактичне значення ортогональні, тобто не мають спільних одиниць, значення буде 1.

Ахмад та ін. також додали те, що вони називають ймовірністю аномалії. Припускаючи, що розподіл помилок прогнозування, визначений у (2.6), відповідає нормальному розподілу, функція ймовірності гауссового хвоста використовується для того, щоб вирішити, чи вважати значення аномалією, чи ні. Оскільки алгоритм було розроблено для роботи в потоковому режимі, обчислюється вибіркове середнє значення μ_t згідно з

(2.7) і дисперсія σ^2_t згідно з (2.8) для останніх W прогнозованих значень помилки

$$\mu_t = \frac{\sum_{i=0}^{i=W-1} S_{t-i}}{W}, \quad (2.7)$$

$$\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1} (S_{t-i} - \mu_t)^2}{W-1}. \quad (2.8)$$

Ймовірність аномалії визначається як доповнення до хвостової ймовірності:

$$L_t = 1 - Q(\frac{\tilde{\mu}_t - \mu_t}{\sigma_t}), \quad (2.9)$$

де

$$\tilde{\mu}_t = \frac{\sum_{i=0}^{i=W'-1} S_{t-i}}{W'}, \quad (2.10)$$

W’ – вікно для розрахунку короткострокової ковзної середньої, де W’ << W. Аномалії ґрунтуються на параметрі °, визначеному користувачем, такому, що:

$$Anomaly\ detectedt = L_t \geq 1 - \varepsilon . \quad (2.11)$$

3 ВИЯВЛЕННЯ АНОМАЛІЙ В МІКРОСЕРВІСНИХ ІНФРАСТРУКТУРАХ

3.1 Опис набору даних

Для оцінки детекторів аномалій використовуються два набори даних. Один набір даних, який називається набором даних NAB, зібрано з NAB [21] і створено Ahamad та іншими дослідниками. Для їхнього тестування виявлення аномалій у реальному часі потокового мовлення. Цей набір даних містить як синтетичні, так і реальні дані, а також мітки часу аномалій. Ці мітки були створені за допомогою керівних принципів маркування, доступних за посиланням [21], і мають високий рівень точності.

Другий набір даних, який називається набором даних певної компанії, де міститься дані, взяті з історичної бази даних метрик анонімної компанії саме для таких досліджент. Випадкова вибірка метрик була зібрана з бази даних. Відомо, що всі показники мають певний історичний інцидент, пов'язаний з ними, щоб збільшити ймовірність присутності аномалій у даних. Кожен зібраний часовий ряд вручну анотується часом аномалій - якщо такі були знайдені. Ці анотації, однак, не є вичерпними, тобто в часових рядах можуть бути аномалії, які не анотовані. Для того, щоб анотувати дані, створено простий інструмент анотування, який дозволяє користувачеві використовувати графічний інтерфейс для взаємодії з часовими рядами та вибору часу аномалій.

Для маркування даних створено графічний інструмент, який дозволяє вручну перевіряти часові ряди, а потім маркувати часові ряди з аномаліями. Щоб збільшити ймовірність того, що часовий ряд має аномалії, використовується інформація про минулі інциденти з припущенням, що інциденти будуть корелювати з аномаліями. Після створення часових міток аномалій вони розгортаються у вікна аномалій. Це пояснюється тим, що аномалія рідко є єдиною точкою, яку можна ідентифікувати, а скоріше

вікном аномальної поведінки. Це проілюстровано на рисунку 3.1, разом з анотацією.

Рисунок 3.1 – Приклад часового ряду з анотацією аномалії

Використання вікон аномалій замість точок також забезпечує врахування ранніх і пізніх виявлень аномалій алгоритмом. Розмір вікна для аномалії було встановлено на рівні 10% від довжини часового ряду, виходячи з розміру вікна NAB-тесту.

3.2 Система бенчмаркінгу детекторів аномалій

Для ефективного порівняння алгоритмів побудовано конвеєр обробки даних. Конвеєр складається з наступних кроків:

- зчитування даних часових рядів у форматі CSV;
- прогнозування аномалій за допомогою детекторів;
- додавання вікон аномалій з позначених міток часу;
- збереження результатів.

Потім створюється окремий механізм підрахунку балів, який зчитує результати та обчислює остаточні результати для кожного з детекторів. Загальний вигляд системи зображено на рисунку 3.2. Слід зазначити, що система не враховує аналіз у реальному часі, і весь часовий ряд для аналізу доступний для детектора на старті.

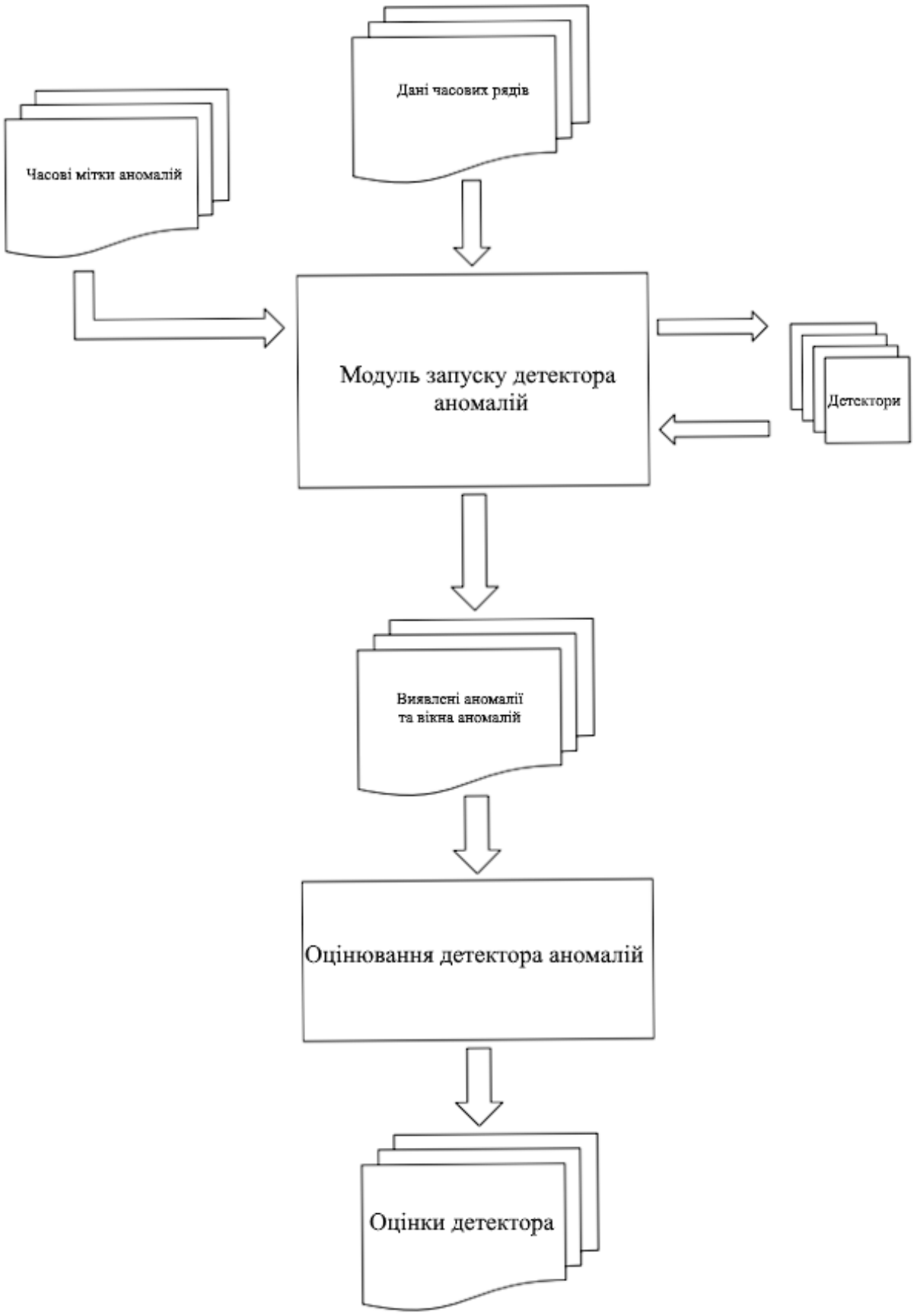

Рисунок 3.2 – Огляд системи

HTM-детектор, як описано в розділі 2, є прогностичною моделлю. Це означає, що вона прогнозує наступне значення в часовому ряді, а аномалії виявляються, дивлячись на те, як це прогнозоване значення відрізняється від фактичного. Нейронна мережа була побудована за допомогою бібліотеки з відкритим кодом nupic, створеної компанією Numenta. Детектор побудований з використанням тих самих параметрів HTM, що і в базовій моделі.

Вхідними даними для HTM є SDR. nupic надає декілька вбудованих кодерів, які можна використовувати для зіставлення звичайних значень з представленнями SDR. Було використано два кодувальники, один для мітки часу і один для значення. Для позначки часу було використано кодер DateEncoder, оскільки, як випливає з назви, це кодер SDR для кодування дат. Було закодовано лише час доби з використанням параметрів width 21 та radius 9.49. Для значення використовувався RandomDistributedScalarEncoder, кодер, створений для скалярних значень, де близькі значення слід вважати не пов'язаними між собою. Після того, як вхідні значення були закодовані, їх було подано до мережі HTM.

Модель приймає закодовані вхідні дані і повертає результат, що містить оцінку аномалії як частку бітів, що збігаються в прогнозованому та отриманому SDR. Цей необроблений результат використовується для обчислення оцінки ймовірності аномалії, що дає оцінку на інтервалі [0, 1]. Результат також відображається на логарифмічну шкалу.

Алгоритм RPCA не є прогностичною моделлю. Алгоритм бере матрицю і обчислює декомпозицію, де один компонент має зменшену розмірність, а інший є розрідженим. Цей розріджений компонент, S0, містить точки даних, які вважаються аномаліями в основних даних. Це також пакетний алгоритм, що означає, що для пошуку аномалій дані потрібно розбити на вікна.

Реалізація RPCA є модифікованою версією алгоритму Netflix Surus з відкритим вихідним кодом, написаним на мові Java. Спочатку реалізований

для запуску в розподіленому кластері Hadoop, він був модифікований для незалежної роботи в звичайному середовищі Java 8.

Дані часового ряду – це вектор значень, а оскільки алгоритм RPCA потребує матриці, вектор має бути перетворений у відповідний формат. Найкращий спосіб зробити це – у кожному стовпчику представити сезонність часового ряду, як це зробили представлено в роботі [15], склавши дані відеокадрів за стовпчиками. Оскільки дані часових рядів можуть не мати сезонності, а всі часові ряди мають різні часові масштаби, виявлення сезонності саме по собі стає проблемою. Було застосовано простий підхід, коли було побудовано дещо збалансовану матрицю розміром M x N шляхом знаходження всіх дільників D довжини часового ряду L, а потім складено матрицю таким чином, щоб M і N мали мінімальну різницю всіх дільників. Звичайно, це не працює, якщо довжина часового ряду є простим числом, однак це можна подолати, зменшивши розмір вікна та ігноруючи найстаріші точки даних.

Значення також були нормалізовані відповідно. Після того, як дані були нормалізовані та перетворені в матрицю, на них було запущено RPCA для вилучення аномальних точок.

Після того, як RPCA розклав вхідну матрицю, розріджений компонент перетворюється назад у вектор. Розріджений вектор має значення аномалій тієї ж величини, що й вихідні дані. Потім ці значення перетворюються на ймовірність того, що точка даних є фактичною аномалією. Використовується ідея, подібна до тієї, що використовується в реалізації HTM. Припускаючи, що відхилення значень від середнього відповідає нормальному розподілу, можемо обчислити ймовірність значення v за допомогою кумулятивної функції розподілу (CDF)

$$probability\ of\ value\ =\ CDF\left(|\,v\,|\right). \tag{3.1}$$

CDF також може бути виражена як обернена до Q-функції

$$Q(v) = 1 \ - \ CDF(v). \tag{3.2}$$

CDF – це ймовірність того, що значення в нормальному розподілі буде менше або дорівнюватиме v. Очевидно, що чим більше значення відхиляється від стандартного відхилення, тим вищою буде ймовірність аномалії. Однак, оскільки результат у (3.1) знаходиться в діапазоні [0,5, 1], він також коригується, щоб бути в бажаному діапазоні [0, 1].

$$anomaly\ score \ = \ 2\big(CDF(|v|) \ - 0.5\big). \tag{3.3}$$

Результат вищенаведеного рівняння потім використовується як оцінка аномалії для детектора RPCA.

3.3 Метрика оцінки

Оцінюючи детектори аномалій, слід враховувати багато критеріїв. Бали за аномалії для кожного детектора визначаються як оцінка аномалії, що перевищує деяке порогове значення. Детектори оцінюються за кількома пороговими значеннями, оскільки вони по-різному обчислюють кількість балів за аномалії і можуть показувати найкращі результати за різних порогових значень. Для детектора і j-те значення класифікується як аномалія, якщо:

$$anomaly\ score_{i,j} \ > \ \varepsilon_i. \tag{3.4}$$

Оптимальне значення ε_i, очевидно, залежить від бажаної чутливості детектора. Менше значення класифікує більше значень як аномалії, але також збільшує ризик хибних спрацьовувань. У реалізації HTM в NAB значення ε_i динамічно встановлювалося для кожного вхідного файлу для оптимізації

оцінки, і значення було приблизно 0,5 виявилося оптимальним для більшості вхідних файлів. Для алгоритму RPCA початкове значення ε_i невідоме, однак правило трьох сигм [22] дає відправну точку для того, які значення можуть бути цікавими.

Використовуючи кожну точку даних як класифікований екземпляр, показники можна обчислити, віднісши точки в межах позначеного вікна до позитивного класу, а всі точки за межами вікна – до негативного класу.

Інший, можливо, більш інтуїтивний підхід полягає в тому, щоб замість цього дивитися на те, чи було виявлено вікно. Інтуїція підказує, що якщо виявлено аномалії всередині вікна, то це означає, що виявили аномалію з анотацією. Таким чином, вікнам аномалій присвоюється позитивний клас. Однак це призводить до проблем з негативним класом. Невідомо, чи точки аномалії за межами вікна дійсно є аномаліями, а через різницю в розмірах вікон незрозуміло, як визначити, чи є область хибнопозитивною. Тому клас від'ємних значень ігнорується.

4 РЕЗУЛЬТАТИ ЕКСПЕРЕМЕНТІВ

4.1 Часові характеристики продуктивності детекторів

Детектори відрізняються за часом виявлення аномалій. Для підвищення продуктивності бенчмаркінгу кожен детектор запускається в окремому процесі ОС паралельно з використанням бібліотеки багатопроцесорної обробки Python. Тестова система працює на комп'ютері з 64 ядрами з тактовою частотою 3,10 ГГц і 32 ГБ оперативної пам'яті. Вимірювання часу виконується за допомогою утиліти UNIX time. sys + user показує загальний процесорний час, використаний усім конвеєром для обох наборів даних, використовуючи лише вказаний детектор.

Таблиця 4.1 – Час обробки конвеєра

Детектор	Реальний час	Системний час	Режим користувача час	система + користувач
HTM	16хв 57.700с	567хв 2.295с	1хв 19.714с	568хв 22.009с
RPCA	0хв 42.537с	16хв 6.158с	1хв 6.765с	17хв 12.923с

4.2 Результати роботи детектора

Кожен детектор аномалій запускається на великій вибірці різних часових рядів. Приклади результатів, що показують вихід кожного детектора на циклічному часовому ряді, можна побачити на рисунках 4.1 і 4.2. Графіки мають три секції. Секція значень показує часові ряди, секція сирих оцінок - це чистий результат роботи детектора аномалій до того, як буде обчислена оцінка аномалії. Нарешті, оцінка аномалії показана в самому нижньому розділі. Позначені вікна аномалій виділено червоним фоном.

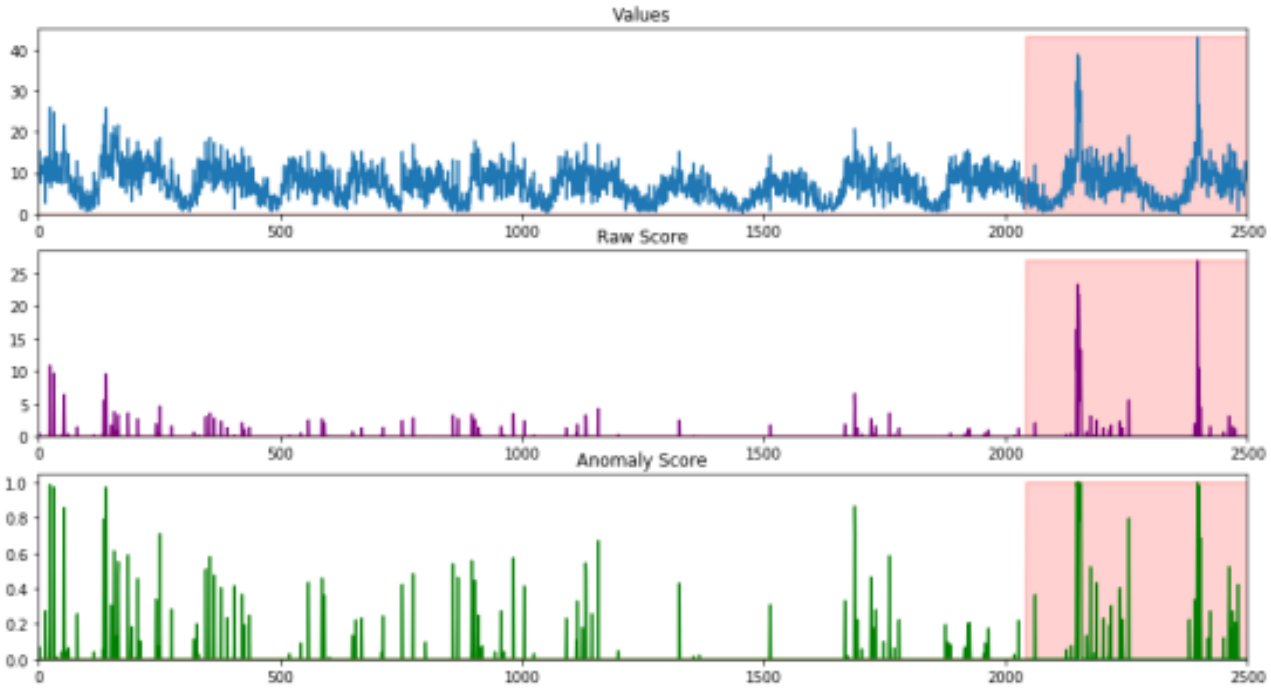

Рисунок 4.1 – Приклад детектора RPCA на циклічному часовому ряді

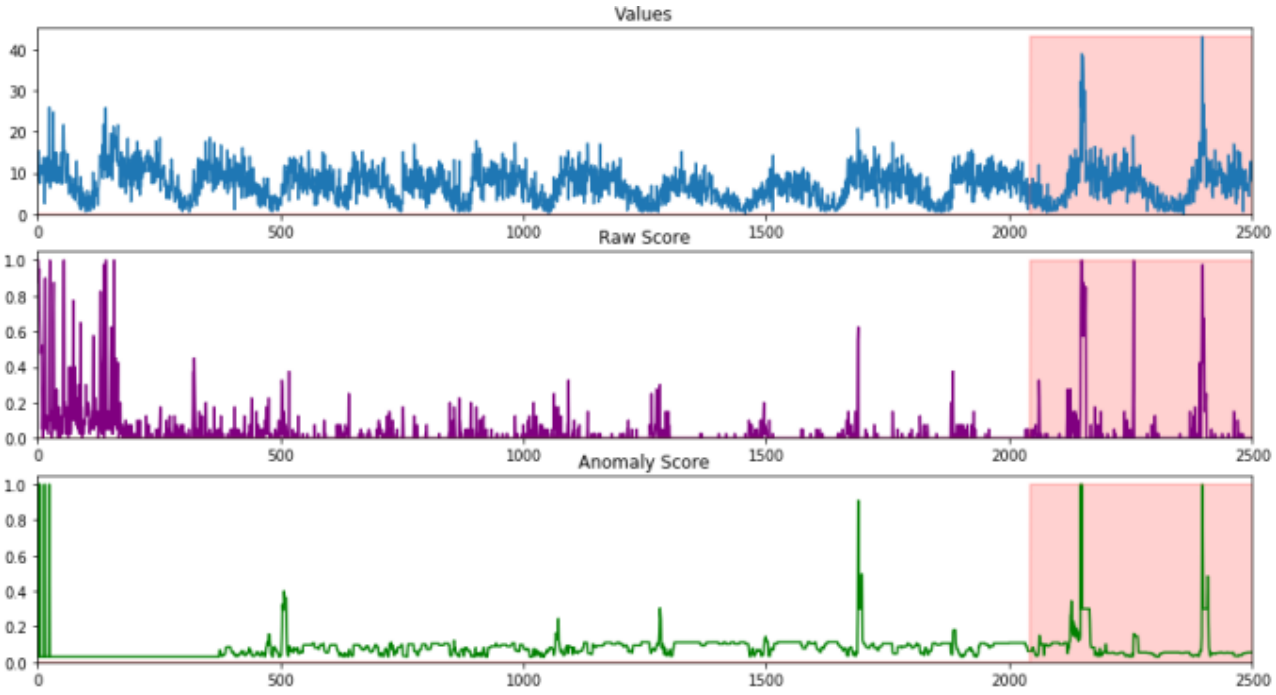

Рисунок 4.2 – Приклад детектора HTM на циклічному виході

Результати оцінки бінарної класифікації для набору даних NAB можна знайти в таблицях 4.2 і 4.3, а для набору даних компаній – в таблицях 4.4 і 4.5. Кожен алгоритм оцінювався при різних порогових значеннях. P_ratio – це співвідношення точок, класифікованих як аномалії, а N_ratio –

співвідношення точок, класифікованих як неаномалії. Як базову лінію було побудовано два детектори. Один детектор і нульовий детектор, і їхні результати можна побачити в таблиці 4.6. Детектор одного сигналу класифікував кожну точку як аномалію, а детектор нульового сигналу робив навпаки і класифікував кожну точку як не аномалію. Це дає еталонні значення, необхідні через незбалансований характер набору даних. Значення 68,27%, 95,45% і 99,73% є трьома значеннями сигми σ, 2 σ і 3 σ.

Таблиця 4.2 – Вимірювання бінарної класифікації на наборі даних NAB за допомогою детектора аномалій HTM для різних порогових значень

ε	F1	Точність (precision)	Точність (accuracy)	P_частка	N_частка
0.0	0.306	0.192	0.192	1.000	0.000
0.1	0.247	0.314	0.708	0.226	0.774
0.2	0.132	0.484	0.802	0.028	0.972
0.3	0.104	0.403	0.805	0.010	0.990
0.4	0.093	0.412	0.807	0.006	0.994
0.5	0.097	0.372	0.807	0.003	0.997
0.6	0.129	0.356	0.807	0.003	0.997
0.6827	0.145	0.337	0.807	0.003	0.997
0.7	0.162	0.331	0.807	0.003	0.997
0.8	0.179	0.317	0.807	0.003	0.997
0.9	0.196	0.310	0.807	0.002	0.998
0.9545	0.195	0.309	0.807	0.002	0.998
0.9973	0.212	0.301	0.807	0.002	0.998

Таблиця 4.3 – Вимірювання бінарної класифікації на наборі даних NAB за допомогою детектора аномалій RPCA для різних порогових значень

ε	F1	Точність (precision)	Точність (accuracy)	P_частка	N_частка
0.0	0.306	0.192	0.192	1.000	0.000
0.1	0.170	0.294	0.653	0.229	0.771
0.2	0.163	0.320	0.676	0.195	0.805
0.3	0.152	0.335	0.718	0.140	0.860
0.4	0.161	0.343	0.752	0.099	0.901
0.5	0.170	0.360	0.760	0.085	0.915
0.6	0.164	0.376	0.771	0.069	0.931
0.6827	0.159	0.394	0.791	0.046	0.954
0.7	0.156	0.395	0.792	0.043	0.957
0.8	0.145	0.415	0.803	0.020	0.980
0.9	0.174	0.527	0.805	0.016	0.984
0.9545	0.129	0.601	0.807	0.014	0.986
0.9973	0.147	0.645	0.807	0.007	0.993

Таблиця 4.4 – Вимірювання бінарної класифікації на наборі даних компанії за допомогою HTM Anomaly Detector для різних порогових значень

ε	F1	Точність (precision)	Точність (accuracy)	P_частка	N_частка
1	2	3	4	5	6
0.0	0.222	0.127	0.127	1.000	0.000
0.1	0.180	0.220	0.565	0.405	0.595
0.2	0.184	0.398	0.855	0.037	0.963
0.3	0.135	0.330	0.865	0.017	0.983
0.4	0.124	0.329	0.868	0.011	0.989
0.5	0.106	0.298	0.869	0.008	0.992
0.6	0.133	0.286	0.869	0.007	0.993

Продовження таблиці 4.4

1	2	3	4	5	6
0.6827	0.132	0.281	0.869	0.007	0.993
0.7	0.132	0.281	0.869	0.007	0.993
0.8	0.151	0.276	0.869	0.007	0.993
0.9	0.180	0.271	0.869	0.007	0.993
0.9545	0.180	0.270	0.869	0.007	0.993
0.9973	0.179	0.265	0.869	0.007	0.993

Таблиця 4.5 – Вимірювання бінарної класифікації на наборі даних компанії за допомогою детектора аномалій RPCA для різних порогових значень

ε	F1	Точність (precision)	Точність (accuracy)	P_частка	N_частка
0.0	0.222	0.127	0.127	1.000	0.000
0.1	0.127	0.328	0.853	0.042	0.958
0.2	0.114	0.357	0.859	0.034	0.966
0.3	0.103	0.377	0.861	0.029	0.971
0.4	0.094	0.390	0.863	0.025	0.975
0.5	0.085	0.412	0.869	0.015	0.985
0.6	0.081	0.440	0.870	0.013	0.987
0.6827	0.078	0.455	0.871	0.012	0.988
0.7	0.077	0.459	0.871	0.011	0.989
0.8	0.074	0.488	0.872	0.010	0.990
0.9	0.059	0.524	0.873	0.008	0.992
0.9545	0.065	0.555	0.874	0.006	0.994
0.9973	0.085	0.608	0.874	0.004	0.996

Результати виявлення аномальних вікон у наборі даних NAB можна побачити в таблицях 4.7 і 4.8, а для набору даних компанії – в таблицях 4.9 і 4.10. Стовпчики total_found та total_missed показують кількість аномальних

вікон, які було позначено як аномалії, а total_tpr – відношення знайдених вікон.

Таблиця 4.6 – Два еталонних детектори, нульовий детектор не оголошує жодне значення аномалією, а одиничний детектор оголошує кожне значення аномалією

Метрика	Null	One
F1	0.103	0.306
Точність (precision)	1.000	0.192
TPR (іст. позитивна)	0.103	1.0
FPR (хибно позитивна)	0.0	1.0
TNR (іст. негативна)	1.0	0.0
Точність (accuracy)	0.808	0.192
P_частка	0.0	1.0
N_частка	1.0	0.0

Таблиця 4.7 – Результат виявлення аномального вікна на наборі даних NAB за допомогою HTM Anomaly Detector для різних порогових значень

ε	Знайдено (total_found)	Пропущено (total_missed)	TPR загальний (total_tpr)
1	2	3	4
0.0	99	0	1.000
0.1	94	5	0.949
0.2	89	10	0.899
0.3	86	13	0.869
0.4	84	15	0.848
0.5	83	16	0.838
0.6	77	22	0.778
0.6827	74	25	0.747

Продовження таблиці 4.7

1	2	3	4
0.7	72	27	0.727
0.8	70	29	0.707
0.9	68	31	0.687
0.9545	67	32	0.677
0.9973	63	36	0.636

Таблиця 4 – Результат виявлення аномального вікна на наборі даних NAB за допомогою RPCA для різних порогових значень

ε	Знайдено (total_found)	Пропущено (total_missed)	TPR загальний (total_tpr)
0.0	99	0	1.000
0.1	98	1	0.990
0.2	97	2	0.980
0.3	97	2	0.980
0.4	95	4	0.960
0.5	93	6	0.939
0.6	93	6	0.939
0.6827	93	6	0.939
0.7	92	7	0.929
0.8	92	7	0.929
0.9	86	13	0.869
0.9545	83	16	0.838
0.9973	75	24	0.758

Таблиця 4.9 – Результат виявлення аномального вікна на наборі даних компанії за допомогою HTM Anomaly Detector для різних порогових значень

ε	Знайдено (total_found)	Пропущено (total_missed)	TPR загальний
0.0	117	0	1.000
0.1	116	1	0.991
0.2	107	10	0.915
0.3	107	10	0.915
0.4	107	10	0.915
0.5	107	10	0.915
0.6	103	14	0.880
0.6827	103	14	0.880
0.7	103	14	0.880
0.8	101	16	0.863
0.9	98	19	0.838
0.9545	98	19	0.838
0.9973	97	20	0.829

Таблиця 4.10 – Результат виявлення аномального вікна на наборі даних компанії за допомогою RPCA Anomaly Detector для різних порогових значень

ε	Знайдено (total_found)	Пропущено (total_missed)	TPR загальний
1	2	3	4
0.0	117	0	1.000
0.1	116	1	0.991
0.2	116	1	0.991
0.3	116	1	0.991
0.4	116	1	0.991
0.5	116	1	0.991
0.6	116	1	0.991

Продовження таблиці 4.10

1	2	3	4
0.6827	114	3	0.974
0.7	114	3	0.974
0.8	114	3	0.974
0.9	114	3	0.974
0.9545	113	4	0.966
0.9973	110	7	0.940

Оскільки мітки відомих інцидентів були доступні для набору даних компанії, вони були співвіднесені з аномаліями, виявленими двома детекторами. Результати наведено в таблиці 4.11. Оскільки мітки відомих інцидентів не містять точних дат початку та закінчення інциденту, для співвіднесення позначеної аномалії з інцидентом використовується близькість виявленої аномалії та інциденту. Близькість інциденту використовується для створення вікна так само, як і для вікон аномалій.

Таблиця 4.11 – Виявлено інциденти на кожному пороговому рівні

Детектор	Знайдено	Пропущено	Співвідношення
HTM	101	0	1.0
RPCA	101	0	1.0

4.3 Обговорення результатів експериментів

З часових показників у таблиці 4.1 видно, що реалізація детектора RPCA набагато швидша, ніж детектора HTM. Одним з параметрів, що впливає на продуктивність HTM-детектора, є необхідність обробляти кожне значення послідовно, а отже, кожне значення спричиняє оновлення базової HTM-мережі, що вимагає досить важких обчислень. Алгоритм RPCA

обробляє всю серію за один раз і не покладається на модель, яка потребує оновлення.

Вимірювання за допомогою бінарного класифікатора слід сприймати з певним скептицизмом як для набору даних NAB, так і для набору даних компанії, оскільки не кожна точка у вікні аномалій є аномалією. Як обговорювалося вище, деякі точки за межами вікна все ще можуть бути аномаліями, навіть у наборі даних NAB, які нібито мають точні мітки. Набори даних також дуже незбалансовані, оскільки негативний клас (не аномалії) набагато більший за позитивний клас (аномалії). Цей дисбаланс призводить до того, що в метриці значною мірою домінують негативні класи FP і TN. Зважаючи на ці обмеження, результати, однак, дають змогу зробити деякі цікаві висновки

Цікавим результатом для обох наборів даних є те, що, порівнюючи TPR і FPR детекторів, видно, що вікна аномалій були більш "позитивно щільними" порівняно з рештою даних для обох детекторів. Пропорція аномальних точок була вищою у позначених вікнах.

ВИСНОВКИ

Як видно з результатів, і RPCA-детектор, і HTM-детектор добре справлялися з виявленням аномалій, і обидва алгоритми змогли працювати принаймні так само добре, як і поточні налаштування виявлення інцидентів для тестованих часових рядів. Обидва детектори змогли знайти всі відомі інциденти в наборі даних, і навіть при високих порогах ймовірності обидва алгоритми змогли знайти більшість позначених вікон аномалій. Алгоритм RPCA здається загалом більш чутливим, ніж HTM-детектор, однак, оскільки виявлення аномалій і чутливість детекторів також залежать від порогових значень, обидва алгоритми можна налаштувати на більшу чутливість за бажанням. Чутливість також залежить від набору даних, оскільки детектор HTM мав більше позначених значень, ніж детектор RPCA на наборі даних компанії, тоді як для набору даних NAB спостерігалася протилежна картина. Для набору даних компанії детектор RPCA зміг виявити 94% всіх вікон аномалій, при цьому рівень помилкових спрацьовувань склав 0,2%. Детектор HTM ніколи не досягав такого рівня продуктивності для набору даних компанії, маючи мінімальний рівень помилкових спрацьовувань 0,6% і знаходячи 88% вікон аномалій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Francis Ysidro Edgeworth. “Xli. on discordant observations”. In: The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 23.143 (1887), pp. 364–375.

2. Nicola Dragoni et al. “Microservices: yesterday, today, and tomorrow”. In: Present and Ulterior Software Engineering. 2017.

3. Javier Martínez Fernandez-Yanez. private communication. 2018.

4. Kyle Brown. Microservices and cloud-native development versus traditional development. 2017. URL: https : / / www . ibm . com / blogs/bluemix/2017/09/microservices-and-cloudnative-development-vs-traditional-development/

5. Numanta. https://numenta.org/

6. Netflix. Surus. https : / / github . com / Netflix / Surus. 2015.

7. Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly Detection: A Survey”. In: ACM Comput. Surv. 41.3 (2009), 15:1– 15:58

8. Subutai Ahmad et al. “Unsupervised real-time anomaly detection for streaming data”. In: Neurocomputing 262 (2017). Online Real-Time Learning Strategies for Data Streams, pp. 134–147

9. Anthony Bagnall et al. “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances”. In: Data Mining and Knowledge Discovery 31.3 (2017), pp. 606–660

10. Charu C. Aggarwal. Data Mining: The Textbook. Springer Publishing Company, Incorporated, 2015

11. Maximilian Christ, Andreas W. Kempa-Liehr, and Michael Feindt. “Distributed and parallel time series feature extraction for industrial big data applications”. In: ArXiv e-prints (2016). arXiv: 1610.07717 [cs.LG].

12. Yanping Chen et al. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/

13. Nicola Dragoni et al. “Microservices: yesterday, today, and tomorrow”. In: Present and Ulterior Software Engineering. 2017

14. Bing Hu, Yanping Chen, and Eamonn Keogh. “Time Series Classification under More Realistic Assumptions”. In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp. 578– 586

15. Emmanuel. J. Candes et al. “Robust Principal Component Analysis?” In: CoRR abs/0912.3599 (2009)

16. Matthias Scholz. “Approaches to analyse and interpret biological profile data”. doctoral thesis. Universität Potsdam, 2006.

17. Jeffrey Wong et al. RAD Outlier Detection on Big Data". 2015. URL: https://medium.com/netflix-techblog/rad-outlierdetection-on-big-data-d6b0494371cc

18. Françoise Beaufays. The neural networks behind Google Voice transcription. 2015. URL: https://research.googleblog.com/ 2015/08/the-neural-networks-behind-google-voice. Html

19. Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. “Continuous online sequence learning with an unsupervised neural network model”. In: Neural Computation 28 (2016)

20. Jeff Hawkins and Subutai Ahmad. “Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex”. In: Frontiers in Neural Circuits 10 (2016), p. 23

21. Numenta. NAB. https://github.com/numenta/NAB. 2017

22. Erik Grafarend and Joseph L Awange. Linear and nonlinear models. Springer, Berlin, Heidelberg, 2012

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