Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів та інструментів моніторингу веб-сервісів
(тема)

Виконав:
студент 2 курсу, групи ІПЗм-22-4

Мандрика М.С
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Ревенчук І. А.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мандриці Максиму Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження методів та інструментів моніторингу веб-сервісів»

Затверджена наказом по університету від 29.03. 2024р. № 250Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи інформація щодо веб-сервісів, галузі веб-сервісів, методи та інструменти моніторингу, інформація щодо ефективності машинного навчання у моніторингу.

4. Перелік питань, що потрібно опрацювати в роботі
аналіз та порівняння існуючих методів та інструментів моніторингу веб-сервісів, вибір моделей машинного навчання для дослідження, проектування логічної моделі даних для проведення експериментальних досліджень, написання програмних рішень, проведення експериментів та аналіз отриманих результатів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	02.04 – 14.04.24	*виконано*
2	Аналіз предметної галузі	15.05 – 17.05.24	*виконано*
3	Методи моніторингу веб-сервісів	18.05 – 20.05.24	*виконано*
4	Інструменти моніторингу веб-сервісів	21.05 – 23.05.24	*виконано*
5	Машинне навчання у моніторингу веб-сервісів	24.05 – 26.05.24	*виконано*
6	Опис програмної реалізації	27.05 – 20.04.24	*виконано*
7	Опис експериментальних досліджень	20.04 – 23.04.24	*виконано*
8	Написання та оформлення статті та тез доповіді	17.04 – 23.04.24	*виконано*
9	Підготовка пояснювальної записки	01.04 – 26.05.24	*виконано*
10	Підготовка презентації та доповіді	26.05 – 30.05.24	*виконано*
11	Нормоконтроль	08.06 – 09.06.24	*виконано*
12	Рецензування	08.06 – 09.06.24	*виконано*
13	Занесення диплома в електронний архів	10.06.2024	*виконано*
14	Попередній захист	12.06.2024	*виконано*
15	Допуск до захисту у зав. кафедри	18.06.2024	*виконано*

Дата видачі завдання 01 квітня 2024р.

Студент (ка) _____ Мандрика М.С.
(підпис)

Керівник роботи _____ доц. Ревенчук. І.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить 69 стор., 20 рис., 13 джерел.

ВЕБ-СЕРВІС, МОНІТОРИНГ, МАШИННЕ НАВЧАННЯ, ПРОДУКТИВНІСТЬ, ДОСТУПНІСТЬ, ТРАФІК, АУДИТ БЕЗПЕКИ

Об’єкт дослідження – сукупність методів, систем та процесів, що використовуються для спостереження, аналізу та управління роботою веб-сервісів. Це охоплює інструменти та платформи моніторингу, які забезпечують збір метрик, логів, та інших даних, що відображають стан веб-сервісів, а також алгоритми та моделі машинного навчання, що використовуються для прогнозування збоїв і оптимізації роботи.

Мета роботи – дослідити предметну галузь моніторингу веб-сервісів, порівняти існуючі методи та інструменти моніторингу, знайти потенційні задачі які можуть бути вирішені за допомогою машинного навчання, описати та виконати експерименти дослідження.

Результат роботи – проаналізовано існуючі методи та інструменти моніторингу веб-сервісів та проведено різні експерименти для оцінки ефективності методів машинного навчання для моніторингу веб-сервісів.

WEB SERVICE, MONITORING, MACHINE LEARNING, PRODUCTIVITY, AVAILABILITY, TRAFFIC, SECURITY AUDIT

The object of research is a set of methods, systems, and processes used for observing, analyzing, and managing the operation of web services. This includes monitoring tools and platforms that collect metrics, logs, and other data reflecting the state of web services, as well as machine learning algorithms and models used for predicting failures and optimizing performance.

The purpose of the work is to explore the domain of web service monitoring, compare existing monitoring methods and tools, identify potential problems that can be

solved using machine learning, describe and conduct research experiments.

The result of the work is three programs developed to address web service monitoring issues using machine learning.

Я, Мандрика Максим Сергійович, студент гр. ІПЗм-22-4, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему “Дослідження методів та інструментів моніторингу веб-сервісів”, що буде представлена для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

МН – Машинне Навчання

ПЗ – Програмне Забезпечення

AWS – Amazon Web Services

APM – Application Performance Management

API – Application Programming Interface

IAST – Interactive Application Security Testing

SIEM – Security Information and Event Management

CI – Continuous Integration

AI – Artificial Intelligence

IT – Інформаційні технології

SLA – Service Level Agreement

IP – Internet Protocol

CPU – Central Processing Unit

SMS – Short Message Service

NLP – Natural Language Processing

MSE – Mean Squared Error

LSTM – Long Short-Term Memory

BERT – Bidirectional Encoder Representations from Transformers

RNN – Recurrent Neural Network

XGBoost – Extreme Gradient Boosting

ВСТУП

Веб-сервіси відіграють ключову роль у цифрових комунікаціях та бізнес-операціях, надійний моніторинг цих сервісів стає вирішальним для підтримки їх ефективності та безпеки. Огляді різних методів та інструментів моніторингу веб-сервісів дозволяє проаналізувати актуальні підходи до моніторингу, сильні та слабкі сторони існуючих інструментів, визначити потенційні напрямки для поліпшення цього критичного аспекту управління веб-сервісами.

Значення моніторингу веб-сервісів зростає через їх широке використання у різноманітних сферах – від електронної комерції до управління даними та обслуговування клієнтів. Технології моніторингу мають забезпечити надійність, ефективність, безперервність роботи веб-сервісів, безпеку від кібератак.

Також є проблеми, з якими стикаються при моніторингу веб-сервісів. Це включає аналіз великих об'ємів даних, швидку реакцію щодо збоїв та забезпечення високого рівня доступності сервісів.

Важливість цієї теми обумовлена швидким розвитком технологій та зростанням залежності бізнесу від надійності та доступності веб-сервісів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Проблеми моніторингу та аналізу веб-сервісів

Веб-сервіси є фундаментальною частиною сучасних цифрових систем, відіграють ключову роль у міжпрограмній взаємодії та обміні даними між різноманітними платформами та застосунками [1]. Вони використовують стандартизовані засоби інтернету, що дозволяє їм гарантувати високу гнучкість та масштабованість у широкому спектрі додатків, починаючи від електронної комерції і закінчуючи корпоративними системами. Ця універсальність та вбудована сумісність роблять веб-сервіси невід'ємною частиною розвитку інтегрованих інформаційних систем, що сприяє їх широкому впровадженню у різноманітні галузі.

Моніторинг веб-сервісів виступає як критично важливий елемент у забезпеченні їхньої надійності, продуктивності, доступності та безпеки. Цей процес охоплює відстеження та аналіз ключових показників, таких як час відгуку, доступність сервісів, виявлення помилок, аналіз трафіку, а також забезпечення безпеки.

Одним з ключових аспектів моніторингу є складність моніторингу веб-сервісів, обумовлена їх розширення функціоналу та інтеграцією в складні системи. Ця складність вимагає розвитку більш просунутих інструментів та методів для ефективного відстеження та аналізу.

Ще один важливий аспект – це обробка великих обсягів даних, які генеруються веб-сервісами. Сучасні системи моніторингу повинні вміти ефективно обробляти та аналізувати ці дані, забезпечуючи своєчасне виявлення аномалій та прогнозування збоїв. Традиційні методи можуть не впоратися з цим завданням, особливо в умовах зміни поведінки системи та нових типів загроз.

Інтеграція машинного навчання у системи моніторингу веб-сервісів можуть вирішити частково або повністю дані проблеми. Методи машинного навчання можуть автоматично адаптуватися до змін у поведінці веб-сервісів та ефективно реагувати на них. Однак, інтеграція цих методів також ставить перед

дослідниками виклики, пов'язані з вибором відповідних моделей, їх навчанням та налаштуванням.

Останнім, але не менш важливим аспектом є безпека та приватність даних. У контексті зростаючої уваги до захисту даних, системи моніторингу веб-сервісів повинні забезпечувати не тільки виявлення технічних проблем, але й адекватно реагувати на потенційні загрози безпеці.

Таким чином, ця робота надає всебічний погляд на сучасні виклики та розробляє стратегії для оптимізації процесів моніторингу та аналізу веб-сервісів, використовуючи передові технології, включаючи машинне навчання.

1.2 Основні типи існуючих інструментів моніторинга веб-сервісів

На ринку існує широкий спектр рішень для моніторингу та аналізу веб-сервісів, які задовольняють різноманітні потреби бізнесу та технічні вимоги. Існують такі типи інструментів: комерційні інструменти, відкриті рішення, хмарні рішення, спеціалізовані рішення. Розглянемо кожен тип більш детально.

Серед комерційних інструментів, таких як New Relic, Datadog і Dynatrace, пропонується широкий набір функцій для моніторингу додатків, веб-сервісів, та інфраструктури. Ці рішення забезпечують глибокий аналіз продуктивності, виявлення та оповіщення про аномалії, а також візуалізацію даних. Вони підходять для великих підприємств, які шукають комплексний підхід до моніторингу своїх систем.

Відкриті рішення, такі як Prometheus, Nagios та Zabbix, пропонують гнучкість та налаштованість для моніторингу різноманітних систем. Ці інструменти підтримують широкий спектр плагінів та інтеграцій, що робить їх популярним вибором серед середніх та малих підприємств, які шукають більш економічні рішення.

Хмарні рішення для моніторингу, як AWS CloudWatch від Amazon, Azure Monitor від Microsoft та Google Cloud Operations Suite, забезпечують інтегровані інструменти для моніторингу та аналізу в рамках хмарних платформ. Ці рішення є ідеальними для компаній, які повністю або частково використовують хмарні

інфраструктури, пропонуючи глибоку інтеграцію з хмарними сервісами та автоматизацію моніторингу.

Існують також спеціалізовані рішення, які фокусуються на конкретних аспектах моніторингу, наприклад, APM інструменти для аналізу продуктивності додатків або інструменти для моніторингу мережевого трафіку. Ці рішення часто використовуються разом з більш загальними інструментами моніторингу для створення комплексної стратегії моніторингу.

1.3 Вплив машинного навчання

Машинне навчання революціонізує підходи до моніторингу веб-сервісів, пропонуючи автоматизовані та інтелектуальні рішення. Традиційні методи моніторингу часто залежать від встановлених порогів і правил, які не завжди здатні адаптуватися до динамічно змінюваних умов мережі або виявляти складні шаблони поведінки. Навпаки, МН може вивчати з часом і пристосовуватися до нових умов, ефективно ідентифікуючи аномалії та передбачаючи потенційні проблеми [2].

З розвитком обчислювальних можливостей та наявністю великих наборів даних, машинне навчання стало більш доступним і привабливим для організацій, які прагнуть покращити свій моніторинг веб-сервісів. Використання МН дозволяє організаціям більш ефективно виявляти та реагувати на непередбачені події, забезпечуючи кращу продуктивність та доступність своїх онлайн-сервісів.

Ключовою перевагою МН є його здатність до самонавчання і адаптації, що дозволяє виявляти складні та неочевидні залежності в даних. Це особливо корисно у виявленні збоїв, аномалій у мережевому трафіку, а також у прогнозуванні навантаження на системи.

Ще одним важливим застосуванням МН є прогнозування навантаження на сервери та інфраструктуру. Алгоритми МН можуть аналізувати історичні дані та ідентифікувати шаблони, що дозволяє точно прогнозувати майбутнє навантаження і вчасно масштабувати ресурси.

МН також відіграє ключову роль у забезпеченні безпеки веб-сервісів. Аналізуючи мережевий трафік і логи, алгоритми можуть ідентифікувати підозрілу поведінку або зразки атак, що дозволяє запобігти потенційним загрозам безпеці.

МН використовує різноманітні методи для виявлення аномалій, включаючи кластеризацію та нейронні мережі. Ці методи можуть аналізувати великі обсяги даних та виявляти непередбачувані або рідкісні події. Швидке виявлення аномалій важливе для мінімізації збитків або перерв у роботі сервісу. МН дозволяє автоматично ідентифікувати та повідомляти про ці аномалії, забезпечуючи своєчасне втручання.

1.4 Постановка задачі

Метою роботи є всебічне дослідження та аналіз методів та інструментів моніторингу веб-сервісів. Особлива увага буде приділена використанню машинного навчання у моніторингу. А саме будуть розглянуті такі алгоритми як Isolation Forest, автоенкодери, LSTM, BERT, Random Forest та XGBoost. Важливо визначити їхню ефективність та застосовність у виявленні аномалій, прогнозуванні здоров'я системи та класифікації журнальних записів.

Основні завдання:

- аналіз існуючих методів моніторингу веб-сервісів та визначення їх обмежень. Це завдання передбачає детальний розгляд поточних методів моніторингу веб-сервісів, їхньої ефективності та виявлення слабких місць, які можна покращити за допомогою методів машинного навчання;
- аналіз інструментів моніторингу веб-сервісів та визначення їх сильних та слабких сторін;
- розробка методики збору та підготовки даних для тренування моделей машинного навчання. Це завдання включає збір метрик використання ресурсів, журналів серверів та інших параметрів продуктивності, їх очищення, нормалізацію та підготовку до використання в моделях машинного навчання;
- адаптація моделей машинного навчання Isolation Forest, автоенкодерам,

LSTM, BERT, Random Forest та XGBoost для задач моніторингу веб-сервісів. Це завдання спрямоване на визначення, які моделі краще справляються з виявленням аномалій, прогнозуванням здоров'я системи та класифікацією журнальних записів;

- тренування та оцінка моделей на симульованих та реальних наборах даних. Це завдання передбачає створення та тренування моделей машинного навчання, які можуть бути ефективними для моніторингу веб-сервісів. Будуть використані великі набори даних для підвищення точності прогнозів;
- проведення експериментів для оцінки масштабованості та ефективності запропонованих рішень. Після розробки та тренування моделей буде проведено детальний аналіз отриманих результатів, що дозволить оцінити їхню масштабованість та ефективність;
- розробка рекомендацій після аналізу результатів дослідження щодо впровадження розроблених методів у реальні системи моніторингу та пропозиції для подальших досліджень.

Розробка точних методів моніторингу є важливою для забезпечення безперебійної роботи та підвищення надійності веб-сервісів. Ефективні моделі прогнозування сприятимуть швидшому реагуванню адміністраторів систем на потенційні загрози, що, у свою чергу, зменшить час простою та підвищить загальну продуктивність веб-сервісів. Підсумковий аналіз результатів дозволить визначити, які підходи є найбільш ефективними для моніторингу.

2 МЕТОДИ МОНІТОРИНГУ ВЕБ-СЕРВІСІ

2.1 Моніторинг продуктивності

Моніторинг продуктивності веб-сервісів – це комплексний процес, що включає в себе відстеження, аналіз та оптимізацію різних аспектів роботи веб-сервісів. Цей метод є ключовим для забезпечення високої якості сервісів та задоволеності користувачів. Для реалізації ефективного моніторингу продуктивності, важливо розглянути кілька ключових аспектів.

Перш за все, необхідно визначити основні метрики продуктивності, які будуть використовуватися для оцінки веб-сервісів. Однією з основних метрик є час відповіді сервера, який вказує на швидкість обробки запитів користувачів. Ця метрика критично важлива, оскільки вона безпосередньо впливає на досвід користувачів та їх задоволеність сервісом.

Другою важливою метрикою є пропускна спроможність, яка визначає кількість даних, які сервер може обробити за одиницю часу. Висока пропускна спроможність вказує на здатність веб-сервісу ефективно обробляти великі обсяги запитів, що є особливо важливим для високонавантажених систем.

Важливим елементом моніторингу продуктивності є також відстеження помилок сервера. Це включає в себе аналіз логів помилок, виявлення та аналіз причин їх виникнення. Помилки сервера можуть мати серйозний вплив на загальну продуктивність та доступність веб-сервісів.

Інший важливий аспект – це масштабованість системи моніторингу. Веб-сервіси можуть зазнавати змін у навантаженні, тому система моніторингу має бути гнучкою, щоб ефективно працювати як при низькому, так і при високому навантаженні.

Окрім технічних аспектів, важливо також враховувати бізнес-показники, такі як вплив продуктивності на задоволеність користувачів та доходи компанії. Аналіз продуктивності має бути інтегрований у загальну стратегію бізнесу, щоб забезпечити високий рівень обслуговування та конкурентоспроможність.

Моніторинг продуктивності веб-сервісів вимагає постійного удосконалення та адаптації. У світі швидких технологічних змін, системи моніторингу повинні розвиватися, щоб відповідати новим вимогам та технологічним можливостям. Використання новітніх технологій, таких як машинне навчання та штучний інтелект, може підвищити ефективність моніторингу та забезпечити більш глибокий аналіз даних.

2.2 Моніторинг доступності

Моніторинг доступності веб-сервісів також є одним із ключових аспектів у забезпеченні надійності та ефективності онлайн сервісів. Цей процес полягає у відстеженні та забезпеченні постійного доступу користувачів до веб-сервісів, що має вирішальне значення для підтримки задоволеності клієнтів та безперервної бізнес-операції.

Перший крок у моніторингу доступності полягає у визначенні критичних компонентів веб-сервісу, які необхідно відстежувати. Це можуть бути веб-сервери, бази даних, мережеві з'єднання, API-інтерфейси та інші компоненти, які впливають на здатність користувачів доступати та використовувати сервіс.

Для реалізації моніторингу використовуються різні інструменти та методи, зокрема, активні перевірки (pinging), які регулярно надсилають запити до веб-сервісу, щоб переконатися у його доступності. У разі недоступності сервісу, система моніторингу генерує оповіщення, яке дозволяє оперативно реагувати на проблему.

Важливою складовою ефективного моніторингу доступності є визначення порогових значень та критеріїв, за якими визначається доступність сервісу. Наприклад, може бути встановлено поріг часу відповіді, при перевищенні якого сервіс вважається недоступним.

Окрім активного моніторингу, важливо також проводити пасивний моніторинг, аналізуючи логи серверів на предмет помилок або проблем у роботі, що також можуть вказувати на проблеми з доступністю. Пасивний моніторинг

дозволяє виявляти проблеми, які можуть бути неочевидні при активному моніторингу.

Ще одним важливим аспектом є тестування веб-сервісів під навантаженням. Це дозволяє виявити можливі проблеми з доступністю, які можуть виникнути при пікових навантаженнях, і попередньо вжити заходів для їх усунення.

Інтеграція системи моніторингу доступності з іншими інструментами управління IT-інфраструктурою дозволяє створювати комплексну картину стану веб-сервісів, що є важливим для виявлення і усунення проблем у роботі.

Ефективний моніторинг доступності також вимагає регулярного перегляду та оновлення налаштувань і стратегій, з урахуванням змін у технологічному середовищі та вимогах бізнесу. Це гарантує, що система моніторингу залишається актуальною та ефективною у відповіді на нові виклики.

2.3 Моніторинг трафіку

Моніторинг трафіку веб-сервісів відіграє критичну роль у забезпеченні ефективної та безпечної роботи онлайн-платформ. Цей процес охоплює відстеження та аналіз потоків даних, що проходять через мережу, дозволяючи ідентифікувати патерни користувацької активності, виявляти потенційні загрози та оптимізувати ресурси сервера.

Перший крок у моніторингу трафіку полягає у встановленні мережевих зондів та моніторів, які збирають дані про трафік. Ці інструменти реєструють різноманітну інформацію, включаючи обсяги переданих даних, частоту запитів, IP-адреси джерел та призначення, а також типи запитів. Аналіз цих даних дозволяє отримати цінне уявлення про використання веб-сервісу та виявити аномальну поведінку.

Важливим аспектом моніторингу трафіку є виявлення та аналіз піків навантаження. Піки можуть бути як легітимними – наприклад, під час маркетингових кампаній або спеціальних подій, так і результатом небажаних дій, наприклад DDoS-атак. Розуміння природи та причин цих піків є важливим для оптимізації роботи сервісу та підготовки до майбутніх навантажень.

Аналіз типів трафіку також є важливим. Він включає розгляд видів даних, що передаються – чи то HTTP запити, API виклики або мультимедійні потоки. Розуміння складу трафіку допомагає ідентифікувати потенційні вузькі місця в інфраструктурі та надає можливості для їх оптимізації.

Безпека трафіку є ще одним критичним аспектом. Моніторинг трафіку дозволяє виявляти підозрілу або маліціозну активність, таку як спроби злому або розповсюдження шкідливого ПЗ. Це вимагає інтеграції систем моніторингу з інструментами безпеки та використання складних алгоритмів для аналізу трафіку.

Ефективний моніторинг трафіку також забезпечує важливу інформацію для бізнес-аналітики. Він допомагає зрозуміти поведінку користувачів, виявляти найбільш популярні функції або сторінки, а також оптимізувати маркетингові та рекламні стратегії.

Моніторинг трафіку потребує постійного перегляду та оновлення для забезпечення актуальності в світлі швидких технологічних змін. Це може включати оновлення обладнання моніторингу, інтеграцію з новими інструментами аналітики або впровадження новітніх методів обробки даних.

2.4 Аудит безпеки

Моніторинг аудита безпеки веб-сервісів також є критично важливим для забезпечення цілісності, конфіденційності та доступності інформації. Цей процес включає в себе систематичний огляд та аналіз заходів безпеки, щоб виявити потенційні слабкі місця та вразливості системи [4].

На початковому етапі моніторингу аудита безпеки ключовим елементом є ідентифікація та оцінка всіх активів і ресурсів, що підлягають захисту. Це може включати сервери, мережеве обладнання, бази даних, програмне забезпечення, а також дані користувачів. Оцінка ризиків для цих активів є фундаментальною для планування заходів безпеки.

Далі, важливим етапом є розробка та впровадження політик безпеки, які визначають правила та процедури для забезпечення безпеки веб-сервісів. Це

включає в себе встановлення вимог до паролів, шифрування даних, управління доступом, а також регулярне оновлення програмного забезпечення.

Цей процес також передбачає періодичні перевірки вразливостей, які включають сканування системи на наявність відомих вразливостей. Це дозволяє своєчасно виявити та усунути слабкі місця, перш ніж вони можуть бути використані зловмисниками.

Ще одним важливим аспектом є моніторинг та аналіз логів безпеки, що включає в себе збір та аналіз даних про події безпеки. Це дозволяє ідентифікувати несанкціоновані спроби доступу, незвичайну активність та інші потенційні загрози безпеки.

Крім того, важливим є регулярне проведення аудитів та оцінок безпеки, які можуть включати в себе зовнішні аудити та пенетраційне тестування. Ці заходи допомагають виявити слабкі місця та оцінити ефективність існуючих заходів безпеки.

На завершення, важливо забезпечувати постійне оновлення та удосконалення заходів безпеки, адаптуючи їх до змінюваних умов та нових загроз. Це включає в себе оновлення політик безпеки, підвищення обізнаності та навчання персоналу, а також інтеграцію з сучасними технологіями та інструментами безпеки.

3 ІНСТРУМЕНТИ МОНІТОРИНГУ ВЕБ-СЕРВІСІВ

3.1 Комерційні інструменти

Розглянемо комерційні рішення New Relic, Datadog і Dynatrace так як це три передові інструменти моніторингу, кожен з яких має унікальні функції та особливості.

New Relic пропонує всеохоплюючу платформу спостереження, яка включає більше 30 можливостей та понад 700 інтеграцій, а також використовує штучний інтелект [5]. Це інструмент орієнтований на моніторинг продуктивності застосунків (APM) та цифрового досвіду користувача, включаючи моніторинг браузера, мобільних застосунків, повторення сеансів, веб-продуктивності та безсерверних додатків. New Relic також пропонує моніторинг інфраструктури, включаючи моніторинг Kubernetes, мережі, Prometheus, а також інтеграцію з AWS, Azure та Google Cloud. Інші ключові функції включають управління логами, інтерактивне тестування безпеки додатків (IAST), вразливості та широкий спектр засобів штучного інтелекту.

Datadog є всебічною платформою моніторингу, яка охоплює різноманітні аспекти IT-інфраструктури та застосунків. Вона включає моніторинг інфраструктури, мережевий моніторинг, моніторинг контейнерів, серверлес-моніторинг, управління витратами на хмарні ресурси, керування логами, моніторинг продуктивності застосунків, безпеку застосунків, управління вразливостями застосунків, моніторинг безпеки в хмарі, SIEM в хмарі, моніторинг реальних користувачів браузера, мобільний моніторинг реальних користувачів, синтетичний моніторинг, запис сесій, відстеження помилок, видимість CI, неперервне тестування, інтеграція з AI та автоматизацію робочих процесів [6].

Dynatrace пропонує широкий спектр функцій для моніторингу та безпеки в хмарі, заснованих на штучному інтелекті. Цей інструмент охоплює спостереження за інфраструктурою, моніторинг додатків, захист безпеки, аналітику безпеки, цифровий досвід та бізнес-аналітику. Dynatrace також пропонує автоматизацію на

основі інсайтів зі спостереження та безпеки та пропонує рішення для вирішення специфічних випадків використання [7].

Є невелика різниця між New Relic, Datadog і Dynatrace, що ускладнює вибір між ними. На вибір впливають:

- основні функції;
- встановлення та простота використання;
- управління інцидентами та сповіщення;
- інтеграція сторонніх сервісів;
- ціни та підтримка.

Розглянемо основні функції. New Relic надає можливості моніторингу для хмарних архітектур, розподілених систем і мікросервісів. Цей інструмент дозволяє користувачам контролювати програми в режимі реального часу, відстежувати розподілені траси та співвідносити журнали з продуктивністю програми.

Основні функції New Relic включають:

- моніторинг продуктивності програми;
- розподілені трасування;
- аналітика помилок;
- моніторинг інфраструктури;
- реальний моніторинг користувачів.

У наборі функцій Datadog приділяє значну увагу хмарі та моніторингу безпеки. Він також включає в себе керування журналами, моніторинг інфраструктури, моніторинг пристроїв і можливості моніторингу бази даних, серед іншого. Цей інструмент моніторингу може відстежувати продуктивність баз даних, серверів і загальної інфраструктури та підтримує кілька хмарних провайдерів.

Деякі з ключових функцій Datadog включають:

- керування журналами;
- моніторинг продуктивності програми;

- моніторинг безпеки;
- моніторинг мережі;
- реальний моніторинг користувачів.

Dynatrace пропонує автоматизований моніторинг з використанням штучного інтелекту для різних аспектів IT-інфраструктури і застосунків. Цей інструмент зосереджений на глибокому аналізі продуктивності, забезпечуючи чітке виявлення проблем та їх автоматичне вирішення. Dynatrace включає моніторинг продуктивності застосунків, моніторинг хмарної інфраструктури, аналіз безпеки та інтеграцію з багатьма хмарними провайдерами. Його ключові функції включають:

- моніторинг продуктивності застосунків;
- моніторинг хмарної інфраструктури;
- аналіз безпеки;
- моніторинг мережі;
- система штучного інтелекту для автоматичного аналізу та вирішення проблем.

Хоча всі ці інструменти пропонують рішення для моніторингу, кожен інструмент має різний підхід щодо пропонованих функцій. Вибір між цими двома інструментами зрештою залежить від конкретних потреб та задач.

Розглянемо встановлення та простоту використання. Процес встановлення Datadog досить простий і простий у управлінні навіть для початківців. Інструмент постачається з добре задокументованими посібниками зі встановлення, які роблять процес безперебійним. Datadog забезпечує простий процес встановлення, який вимагає мінімальної конфігурації, і після встановлення він готовий до використання. Крім того, немає необхідності інтегрувати плагіни сторонніх виробників, щоб розпочати роботу з інструментом, оскільки він містить усі необхідні функції.

New Relic також є зручним інструментом, а процес його встановлення також простий у управлінні. Майстер інсталяції New Relic інтуїтивно зрозумілий, і він проведе вас через процес. На відміну від Datadog, New Relic вимагає встановлення

агентів у системах, які ви хочете контролювати, але це простий процес, який можна завершити за лічені хвилини.

Процес встановлення Dynatrace характеризується як простий та інтуїтивно зрозумілий, навіть для тих, хто тільки починає працювати з моніторингом. Цей інструмент містить чіткі інструкції та налаштування, які спрощують процес встановлення. Dynatrace автоматично адаптується до вашої інфраструктури і не потребує складної конфігурації або додавання сторонніх плагінів. Після встановлення Dynatrace готовий до використання, надаючи користувачам доступ до всіх ключових функцій моніторингу.

Загалом і New Relic, Datadog і Dynatrace пропонують просту інсталяцію та легкість використання, що полегшує користувачам розпочати моніторинг своєї інфраструктури. Проте процеси інсталяції Datadog та Dynatrace є трохи простішими, оскільки не вимагають встановлення додаткових плагінів, тоді як New Relic вимагає встановлення агентів у системах, які ви хочете контролювати.

Розглянемо управління інцидентами та сповіщення. Всі згадані інструменти пропонують надійні рішення щодо інформаційних панелей і управління інцидентами.

New Relic пропонує інтуїтивно зрозумілу інформаційну панель, яка представляє дані в режимі реального часу, що дозволяє користувачам легко контролювати стан своєї системи. Цей інструмент надає функцію під назвою Insights, яка пропонує глибоке розуміння ваших програм, інфраструктури та взаємодії з клієнтами. За допомогою Insights користувачі можуть запитувати дані, аналізувати тенденції та візуалізувати дані в різних форматах, включаючи діаграми, таблиці та гістограми. Крім того, New Relic пропонує різні варіанти звітності, включаючи готові звіти, настроювані інформаційні панелі та можливість створювати власні звіти.

Datadog надає настроювану інформаційну панель, що полегшує користувачам створення власних інформаційних панелей відповідно до їхніх вимог. Це дозволяє користувачам фільтрувати дані, а потім представляти їх у спосіб, який є значущим для їхніх команд. Крім того, інформаційна панель

Datadog дозволяє детально вивчати конкретні показники та швидко досліджувати проблеми. Інструмент також пропонує функції звітування та попередження в режимі реального часу, тому користувачі можуть отримувати сповіщення про досягнення певного порогового значення.

Dynatrace надає розширені можливості інформаційних панелей та звітності, з акцентом на використанні штучного інтелекту для глибокого аналізу даних. Це включає автоматизоване виявлення та аналіз проблем, а також інтуїтивно зрозумілі інформаційні панелі, які дозволяють користувачам легко відстежувати важливі показники та швидко реагувати на зміни. Додатково, Dynatrace забезпечує реальночасові сповіщення, які інформують користувачів про критичні зміни у системі.

Загалом і New Relic, і Datadog, і Dynatrace забезпечують надійні інформаційні панелі та можливості звітування, які дозволяють користувачам ефективно контролювати свої системи. Вибір між трьома інструментами зрештою залежить від конкретних потреб і рівня налаштування, який потрібен.

Що стосується інтеграції сторонніх сервісів, всі інструменти пропонують широкий спектр можливостей.

Datadog має понад 500 інтеграцій, які дозволяють користувачам збирати показники з усього стеку технологій, включаючи такі популярні інструменти, як Kubernetes, MySQL і Docker. Він також має доповнення для сторонніх інтеграцій, таких як Splunk, AWS і PagerDuty.

Так само New Relic пропонує більше 100 інтеграцій для різних сервісів і платформ, включаючи AWS, Azure і Google Cloud. Однак, на відміну від Datadog, New Relic пропонує вбудовану інтеграцію для свого APM та інструментів моніторингу інфраструктури, що може стати перевагою для користувачів, які хочуть уникнути клопоту з налаштуванням і підтримкою сторонніх інтеграцій.

Dynatrace підтримує значну кількість інтеграцій з різними інструментами та платформами, включаючи хмарні сервіси, моніторинг інфраструктури, інструменти безпеки, системи управління інцидентами та інші. Ці інтеграції

дозволяють Dynatrace забезпечувати глибокий аналіз та ефективне управління IT-середовищами.

Коли справа доходить до вибору між ними, всі варіанти є дуже ефективними. Datadog пропонує більший вибір інтеграцій і більшу гнучкість із доповненнями. Тим часом вбудовані інтеграції New Relic можуть зробити налаштування та обслуговування більш простими. Інтеграції Dynatrace можуть бути налаштовані за допомогою штучного інтелекту що може бути актуальним для конкретних потреб.

За ціноутворенням всі інструменти пропонують гнучкі варіанти, але всі коштують дорого.

Datadog має різноманітні тарифні плани, включаючи безкоштовний план, який дозволяє використовувати до 5 хостів, і платні плани, які починаються від 15 доларів США за хост на місяць. Платні плани мають такі додаткові функції, як APM, журнали та спеціальні показники. Datadog також пропонує корпоративний план для великих організацій, починаючи з 27 доларів США за хост на місяць і пропонуючи індивідуальні ціни на основі конкретних потреб.

New Relic також пропонує безкоштовний план, який включає такі базові функції, як 100 ГБ даних, необмежену кількість користувачів і базові сповіщення. Платні плани починаються від 149 доларів США на місяць і включають додаткові функції, такі як APM, моніторинг інфраструктури та журнали та необмежену кількість користувачів платформи. New Relic також пропонує корпоративний план для великих організацій із спеціальними цінами.

Модель ціноутворення Dynatrace в основному базується на передплаті, зосереджуючись на гнучкості для задоволення різних потреб. Вартість варіюється залежно від функцій і масштабу використання, наприклад, кількості хостів, які контролюються, або обсягу оброблених даних. Dynatrace пропонує пакети, які можуть включати моніторинг продуктивності додатків, моніторинг цифрового досвіду, моніторинг інфраструктури, а також аналітику та автоматизацію на основі штучного інтелекту. Їхня модель розроблена для підприємств різного

розміру, від невеликих установ до великомасштабного розгортання. Dynatrace не має безкоштоної версії, тільки безкоштовний пробний період.

Що стосується підтримки, всі інструменти моніторингу пропонують різноманітні ресурси, включаючи документацію, форуми спільноти та квитки до служби підтримки. Залежно від тарифного плану інструменти також пропонують підтримку по телефону для своїх платних планів.

Вартість послуг залежить від розміру команд, які користуються інструментами. Графік цін можна побачити на рисунку 3.1:

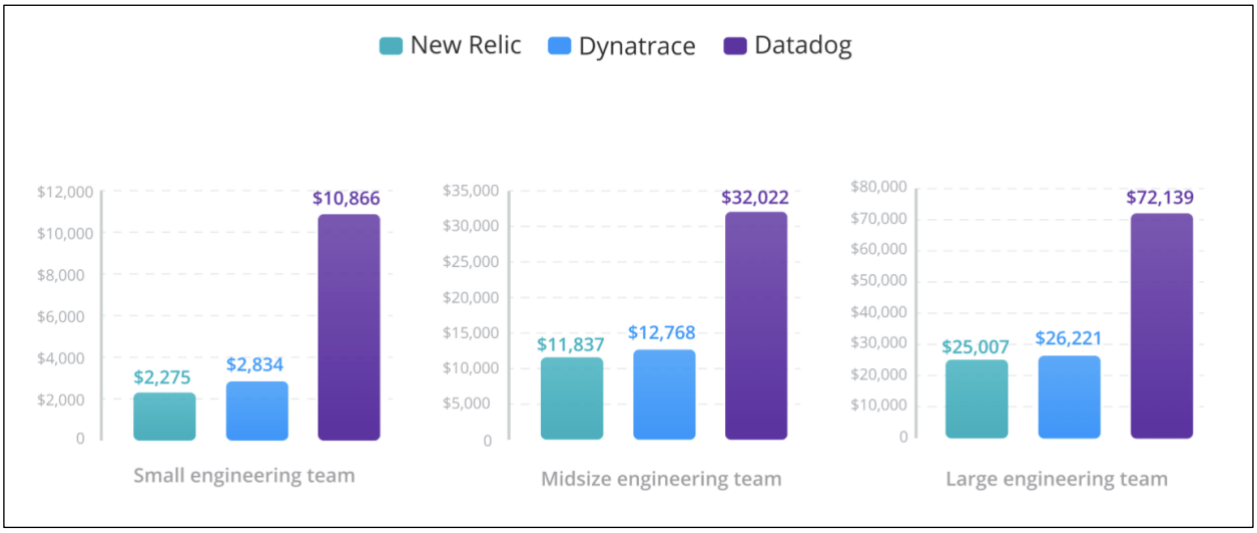

Рисунок 3.1 – Графік цін інструментів моніторинга в залежності від розміру команди (за даними [8])

Згідно графіку можемо побачити що рішення Datadog значно дорожче за New Relic та Dynatrace, а New Relic та Dynatrace дуже близькі за цінами.

3.2 Відкриті рішення

Проаналізуємо та порівняємо відкриті рішення Prometheus, Nagios та Zabbix.

Prometheus – це система моніторингу на основі показників, розроблена для відстеження загального стану, продуктивності та поведінки організації. Він збирає, впорядковує та зберігає показники в реальному часі, а також миттєво надсилає інтуїтивно зрозумілі сповіщення. Програмне забезпечення використовує моделювання розмірних даних, гнучку мову запитів, сучасні методи оповіщення

та ефективну базу даних часових рядів, щоб забезпечити елегантні рішення моніторингу. Завдяки продумано розробленому інтерфейсу та футуристичним функціям Prometheus надає чудову інформацію, отриману на основі різноманітних показників. Завдяки застосуванню інструменту даних великого розміру та пар ключ-значення він перетворює дані на значущі ідеї. Це ефективне рішення для моніторингу дозволяє користувачам розділяти зібрані часові ряди даних через PromQL і створювати інформативні графіки, таблиці та сповіщення відповідно до зручності. Prometheus також постачається з різноманітними складними шаблонами, які полегшують безперебійну візуалізацію даних та інтегровані функції Grafana. Крім того, він забезпечує ефективне сховище часових рядів, окрім роботи на простому, але надійному сервері. Підхід до оповіщення платформи ґрунтується на PromQL і є гнучким [9]. Нарешті, це програмне забезпечення з відкритим вихідним кодом повністю керується спільнотою та також є дуже безпечним.

Nagios вважається найбільш надійним програмним забезпеченням для моніторингу мережі з відкритим вихідним кодом, доступним на ринку [10]. Програмне забезпечення справді оснащене безліччю функцій моніторингу мережі, включаючи керування ресурсами, моніторинг сервера, моніторинг SLA, моніторинг безвідмовної роботи, моніторинг використання Інтернету, пропускної здатності моніторинг, моніторинг IP-адрес і пропонує звіти про веб-трафік тощо. Він також пропонує базовий менеджер, інструменти аналітики та діагностики в реальному часі. Ви можете розумно та легко контролювати все своє мережеве обладнання, наприклад маршрутизатори, комутатори, сервери тощо. Відстеження трафіку в мережі в будь-який час також стає легше. Він оснащений інтуїтивно зрозумілим інтерфейсом і також налаштовується.

Zabbix – це програмне забезпечення для моніторингу мережі, яке дозволяє контролювати продуктивність мережі та керувати нею в режимі реального часу. Ви можете відстежувати використання Інтернету, пропускну здатність, IP-адресу, час безвідмовної роботи, сервер тощо. Програмне забезпечення постачається з аналітикою в реальному часі, інструментами діагностики та звітами про веб-

трафік. також допомагає в управлінні ресурсами [11]. Платформа дуже масштабована. Zabbix Monitoring Solution сповіщає вас, якщо джерело живлення є критичним, температура пристрою надмірно висока/низька, мало вільного місця на диску тощо. Інструмент оптимізований для кращої продуктивності та дозволяє інтегрувати програмне забезпечення третіх сторін. Він також включає такі функції, як автентифікація безпеки, створення шаблонів, автоматичне виявлення тощо.

Розглянемо основні функції. Prometheus, стала популярною за допомогою своїх потужностей моніторингу. Вона забезпечує надійне зберігання даних зі складною моделлю даних, потужну мову запитів для аналізу даних моніторингу, а також інтеграцію з ефективними способами сповіщення.

Основні функції Prometheus:

- потужна мультимірна модель даних. Вона збирає часові ряди даних, ідентифікованих за допомогою метричного імені та ключ/значення пар;
- власна мову запитів, PromQL, яка дозволяє користувачам вибірково відфільтровувати та агрегувати метричні дані для аналізу;
- гнучка система сповіщень, яка інтегрується з різними зовнішніми системами;
- автоматичне виявлення цілі моніторингу в динамічних середовищах;
- підтримка множини експортерів, які можуть збирати метрики з різноманітних джерел та перетворювати їх на формат, сумісний з Prometheus.

Nagios, одна з найбільш відомих і довготривалих систем моніторингу, пропонує широкий спектр функцій для відновлення стану інфраструктури. Вона забезпечувала моніторинг ресурсів, таких як процесори, пам'ять, дисковий простір, а також моніторинг послуг та додатків. Nagios знає свою гнучкість у налаштуваннях, можливості розширення за допомогою плагінів та сильним співтовариством.

Основні функції Nagios:

- моніторинг стану серверів, комутаторів, додатків і послуг, включаючи CPU, пам'ять, дисковий простір тощо;
- підтримка плагінів для розширення функціональності, дозволяючи моніторити майже будь-який аспект інфраструктури;
- підтримка сповіщень через різні канали (e-mail, SMS тощо) і дозволяє налаштовувати правила ескалації для критичних проблем;
- веб-інтерфейс для перегляду поточного стану мережі, історичних даних, графіків тощо;
- централізована архітектура, здатна масштабуватися для великих мереж.

Zabbix є ще одним популярним вибором у сфері системи моніторингу. Ця система володіє своєю здатністю до масштабування, підтримкою великої кількості метрики та інтегрованим підходом до моніторингу. Zabbix дозволяє контролювати мережу, сервери, віртуальні машини та різні додатки. Вона також надає функціонал для автоматичного виявлення ресурсів, візуалізації даних та гнучких налаштувань сповіщень.

Основні функції Zabbix:

- моніторинг мережі, серверів, хмар, послуг та додатків. Включає в себе відстеження процесорів, пам'яті, мережевого трафіку тощо;
- автоматичне виявлення нових серверів та пристроїв в мережі, що полегшує масштабування;
- розширені можливості візуалізації, включаючи графіки, карти, звіти та панелі інструментів;
- налаштування складних правил сповіщень і може автоматично виконувати дії для вирішення проблем;
- ефективно масштабується для великих систем, підтримуючи збір великої кількості метрик з різних джерел.

Встановлення та налаштувати Prometheus може бути складним процесом. Він не містить таких функцій, як попередження або комплексна візуалізація даних, тому організації повинні окремо встановлювати та інтегрувати ці функції та вручну налаштовувати всі конфігураційні файли.

З іншого боку, Nagios вимагає теж ретельного налаштування, особливо при використанні додаткових плагінів та кастомних скриптів. Це дозволяє більш гнучко налаштовувати моніторинг та сповіщення, але також збільшує складність первинного налаштування. Налаштування Nagios часто включає ручне редагування конфігураційних файлів та може вимагати глибшого розуміння системних та мережевих концепцій.

Zabbix має більше попередньо встановлених конфігурацій, що робить його інсталяцію та налаштування швидшими. Однак Zabbix не пропонує деяких глибших параметрів конфігурації в Prometheus чи Nagios, які роблять їх такими потужними.

4 МАШИННЕ НАВЧАННЯ У МОНІТОРИНГУ ВЕБ-СЕРВІСІВ

Машинне навчання для моніторингу веб-сервісів має кілька значних переваг у порівнянні з класичними методами. Основними з них є адаптивність, гнучкість, висока продуктивність. Ці переваги особливо важливі в динамічних середовищах, де системи часто змінюють свої параметри та поведінку [12].

Ця кваліфікаційна робота зосереджена на доведенні, або спростуванні ефективності машинного навчання у системах моніторингу веб-сервісів та для виявлення аномалій, класифікації журнальних записів та оптимізації прогнозування навантаження системи для подальшої інтеграції з різними інструментами моніторингу.

4.1 Алгоритми виявлення аномалій

Перша частина полягає в створенні алгоритмів машинного навчання, які ефективно і точно виявляють аномалії в поведінці веб-сервісів. Це означає ідентифікацію непередбачуваних або рідкісних подій, які відрізняються від звичайного патерну поведінки. Проведення серії експериментів для порівняння різних технік машинного навчання, включаючи кероване та некероване навчання, для виявлення найбільш ефективного методу виявлення аномалій.

Для проведення експерименту з виявлення аномалій основною метою є перевірка здатності алгоритмів машинного навчання точно ідентифікувати аномалії в поведінці веб-сервісів. Це включає відхилення в процесах, ненормальне використання ресурсів, або зміни в мережевому трафіку, які можуть вказувати на потенційні збої, відмови або безпекові інциденти.

Результати цього експерименту допоможуть визначити ефективність машинного навчання у виявленні аномалій у веб-сервісах, виявити найбільш ефективні методи та алгоритми, а також визначити області для подальшого вдосконалення моделей.

Для даного експерименту використовуватиме наступні моделі машинного навчання: модель ізольованого дерева та автоенкодери.

Ізольоване дерево - хороша вихідна точка для виявлення аномалій, оскільки не вимагає попереднього визначення "нормальності".

Автоенкодери - це тип нейронної мережі, який намагається відтворити свій вхід на виході. Вони часто використовуються для виявлення аномалій, оскільки аномальні дані зазвичай відтворюються гірше, ніж нормальні.

Для визначення ефективності методів будемо використовувати формулу F1Score. F1Score - поєднання точності (Precision за формулою 4.1) та повноти (Recall за формулою 4.2) для оцінки ефективності виявлення аномалій (формула 4.3).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

де, TP – кількість істинних позитивів,

FP – кількість хибних позитивів.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

де, TP – кількість істинних позитивів,

FP – кількість хибних позитивів,

FN – кількість хибних негативів.

$$\text{F1Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.3}$$

4.2 Покращення точності інтерпретації даних журналів

Друга частина – використання методів машинного навчання для точного аналізу та інтерпретації даних з журналів, що дозволяє виявити приховані проблеми або потенційні ризики. Впровадження технік NLP та глибокого

навчання для розшифровки та аналізу текстових записів у журналах, підвищуючи точність виявлення та інтерпретації подій.

Використаємо методи NLP та глибокого навчання для розробки моделей, здатних ефективно обробляти та аналізувати великі обсяги текстових даних з журналів. Це буде включати класифікацію повідомлень, виявлення аномалій у логах та визначення шаблонів, які можуть вказувати на проблеми.

Симулюємо різні умови та сценарії, які можуть бути відображені в журналах, включаючи штучні сценарії, що мімікрують реальні інциденти або проблемні ситуації. Це дозволить оцінити здатність моделей правильно ідентифікувати та інтерпретувати критичні події.

Для задачі покращення точності інтерпретації даних журналів з використанням методів обробки природної мови та глибокого навчання, будемо використовувати наступні дві моделі: LSTM мережі та BERT.

LSTM - це вид RNN, які добре підходять для аналізу послідовностей даних, як текст. Вони можуть виявляти залежності у даних на тривалих інтервалах часу і ефективно обробляти послідовності різної довжини, що робить їх ідеальними для аналізу лог-файлів [13].

BERT – це передова модель обробки природної мови, яка використовує техніку трансформерів для розуміння контексту слова в реченні. Вона може бути доцільною для класифікації текстових даних, виявлення аномалій у тексті та визначення шаблонів, що можуть вказувати на проблеми.

Для визначення ефективності методів будемо розраховувати частку правильних передбачень серед усіх передбачень (Accuracy) за формулою 4.4 та оцінимо продуктивність класифікаційної моделі за допомогою крос-ентропії (LogLoss) за формулою 4.5.

$$\text{Precision} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

де, TP – кількість істинних позитивів,

TN – кількість істинних негативів,

FP – кількість хибних позитивів,

FN – кількість хибних негативів.

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (4.5)$$

де n – кількість спостережень,

y_i – фактичне значення (0 або 1),

$\hat{y}_i$ – прогнозована ймовірність класу.

4.3 Оптимізація прогнозування навантаження на сервер

Третя частина – розробка моделей, які можуть точно передбачати майбутнє навантаження на сервери та інфраструктуру щоб оптимізувати розподіл ресурсів та масштабування інфраструктури веб-сервісів. Експериментування з різними методами машинного навчання, такими як регресія, машини опорних векторів, та нейронні мережі, для визначення оптимального підходу до прогнозування навантаження.

Цей експеримент зосереджується на оцінці масштабованості та ефективності системи моніторингу веб-сервісів, яка використовує машинне навчання. Основна мета полягає у визначенні, наскільки добре система може пристосовуватися та підтримувати високий рівень продуктивності при збільшенні обсягів даних і розширенні мережевих ресурсів.

Для визначення масштабованості системи проводемо тести під різними рівнями навантаження. Це буде включати збільшення кількості веб-сервісів, які моніторяться, або симуляцію великого обсягу трафіку та даних.

Важливо також оцінити, як система буде справлятися зі зростаючими обсягами даних, чи буде затримки в обробці або зниження продуктивності.

Оцінемо здатність системи підтримувати стабільну продуктивність при масштабуванні. Це буде включати аналіз швидкості обробки даних, часу відгуку на запити та здатності системи виявляти аномалії та інші проблеми без затримок.

Для даної задачі будемо використовувати наступні дві моделі: Random Forest та XGBoost.

Random Forest – це ансамблевий метод машинного навчання, який використовує множину рішучих дерев для вирішення завдань класифікації та регресії. Основна ідея методу полягає у створенні великої кількості незалежних рішучих дерев, кожне з яких тренується на випадковому підмножині вихідних даних з використанням методу bootstrap. Крім того, при побудові кожного дерева, для кожного розділення вузла випадковим чином вибирається підмножина ознак, що знижує кореляцію між деревами і підвищує загальну продуктивність моделі. В результаті, підсумкове передбачення Random Forest є усередненим результатом передбачень усіх дерев, що робить його стійким до перенавчання та дозволяє отримувати високоточні результати навіть на великих і складних наборах даних.

XGBoost – це потужний алгоритм машинного навчання, який базується на методі градієнтного бустингу. XGBoost оптимізований для високої продуктивності та ефективності завдяки використанню паралельних обчислень і спеціалізованих оптимізацій. Алгоритм працює шляхом послідовного додавання нових дерев до ансамблю, кожне з яких намагається виправити помилки попередніх дерев, використовуючи градієнти функції втрат. Вбудована регуляризація (L1 та L2) допомагає запобігти перенавчанню, забезпечуючи високу узагальнювальну здатність моделі. XGBoost також підтримує обробку пропущених значень та автоматично виявляє найважливіші ознаки, що робить його надзвичайно гнучким і ефективним для різних завдань аналізу даних.

Для визначення ефективності методів будемо використовувати значення середньоквадратичної помилки (MSE) за формулою 4.6, яка оцінює середню величину квадратичних відхилень між фактичними та прогнозованими значеннями.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.6}$$

де n – кількість спостережень,

y_i – фактичне значення,

$\hat{y}_i$ – прогнозоване значення.

Ці метрики дозволять провести всебічний аналіз продуктивності моделей машинного навчання для кожного з трьох завдань та визначити найбільш ефективні підходи.

5 ОПИС ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

5.1 Інструменти для проведення досліджень

Для проведення експериментів будемо використовувати мову програмування Python. Python є однією з найпопулярніших мов програмування для наукових обчислень та аналізу даних завдяки своїй гнучкості та великій екосистемі бібліотек. Вона забезпечує потужні інструменти для реалізації моделей машинного навчання, обробки даних та візуалізації.

Бібліотеки які будемо використовувати для реалізації: TensorFlow, Keras, Scikit-learn, XGBoost, Pandas, NumPy, Matplotlib.

TensorFlow – це відкрита платформа для машинного навчання, розроблена Google, яка дозволяє легко створювати та тренувати складні моделі на базі нейронних мереж. Keras є високорівневим API, що працює поверх TensorFlow, забезпечуючи зручний інтерфейс для побудови та тренування нейронних мереж. Використовується для реалізації моделей LSTM та автоенкодерів.

Scikit-learn є однією з найпопулярніших бібліотек для машинного навчання в Python. Вона містить широкий спектр алгоритмів для класифікації, регресії, кластеризації та зниження розмірності, а також засоби для обробки даних та оцінки моделей. Використовується для реалізації моделей Isolation Forest та Random Forest.

XGBoost – це вдосконалена версія градієнтного бустингу, яка забезпечує високу продуктивність і точність. XGBoost оптимізований для швидкості та ефективності, підтримує паралельні обчислення та має вбудовані механізми для обробки пропущених даних і регуляризації. Використовується для реалізації моделей градієнтного бустингу.

Pandas – це потужна бібліотека для обробки та аналізу даних, яка дозволяє легко маніпулювати табличними даними. Вона забезпечує високий рівень функціональності для очищення, трансформації та агрегації даних.

NumPy – це фундаментальна бібліотека для числових обчислень в Python, яка надає підтримку для багатовимірних масивів і матриць, а також великої кількості математичних функцій для їх обробки.

Matplotlib – це бібліотека для створення статичних, анімованих і інтерактивних візуалізацій у Python. Вона забезпечує засоби для побудови різноманітних графіків, включаючи лінійні графіки, гістограми, розсіяні діаграми та інші. Використовується для візуалізації результатів моделей машинного навчання та аналізу їх продуктивності.

Перший крок полягає у зборі даних, таких як метрики використання ресурсів (ЦПУ, пам'ять), журнали серверів, час відгуку та кількість активних користувачів. Дані очищаються від шуму та пропущених значень, нормалізуються та масштабуються для забезпечення коректного тренування моделей. Текстові дані журналів перетворюються у числові послідовності за допомогою токенізаторів.

Моделі машинного навчання реалізуються з використанням TensorFlow/Keras, Scikit-learn та XGBoost. Це включає створення і тренування моделей автоенкодерів, LSTM, Random Forest та XGBoost, а також їх оцінку за допомогою метрик описаних у попердньому розділі.

5.2 Формування гіпотези

Гіпотеза для даного дослідження може бути сформульована наступним чином: використання алгоритмів машинного навчання в інструменті моніторингу веб-сервісів підвищить ефективність та точність виявлення аномалій, прогнозування навантаження на сервери, автоматизації реагування на проблеми та аналізу даних журналів, порівняно з традиційними методами моніторингу. Це приведе до покращення продуктивності, зниження часу простою та збільшення загальної надійності веб-сервісів.

Ця гіпотеза передбачає, що впровадження машинного навчання в інструмент моніторингу не тільки поліпшить традиційні аспекти моніторингу, але й відкриє нові можливості для покращення управління веб-сервісами. Основна

мета дослідження буде полягати у валідації цієї гіпотези через ряд експериментів та аналізів.

5.3 Підготовка та проведення експериментів. Алгоритми виявлення аномалій

Для валідації гіпотези щодо ефективності використання машинного навчання в моніторингу веб-сервісів, проведемо наступні експерименти:

Розглянемо алгоритми виявлення аномалій. Для проведення експерименту з виявлення аномалій основною метою є перевірка здатності алгоритмів машинного навчання точно ідентифікувати аномалії в поведінці веб-сервісів. Це включає відхилення в процесах, ненормальне використання ресурсів, які можуть вказувати на потенційні збої, відмови або безпекові інциденти.

Розробимо програму за допомогою мови програмування Python та порівняємо роботу двох моделей: ізольованого дерева та автоенкодера.

Для виявлення аномалій зберемо наступні метрики з тестового веб-сервіса та збережемо їх у csv файл: використання ЦПУ, використання пам'яті, час відгуку та кількість активних користувачів. Кожен рядок представляє один часовий знімок метрик, частина csv файлу показана на рисунку 5.1:

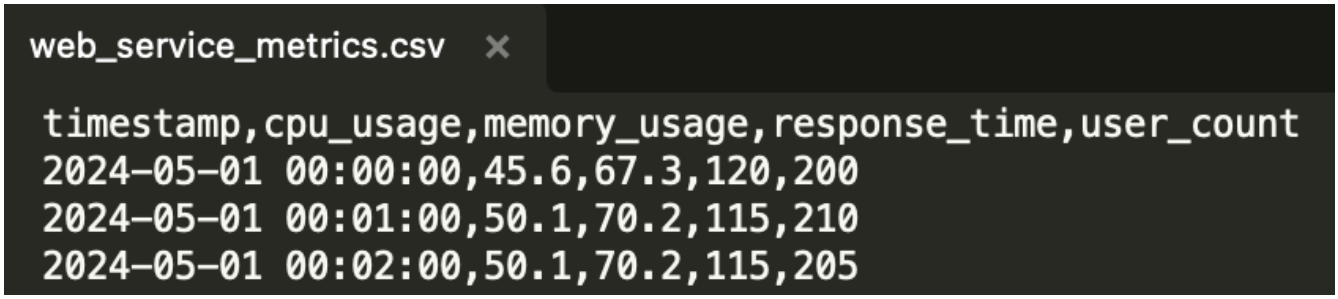

Рисунок 5.1 – Метрики веб-сервіса збережені у csv файлі (рисунок створено самостійно)

Наступним кроком прочитаємо метрики з файлу та нормалізуємо, а також додамо мітки аномалій (рисунок 5.2);

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Читання даних з CSV
data = pd.read_csv('./web_service_metrics.csv')

# Заповнення пропущених значень
data.fillna(method='ffill', inplace=True)


# Нормалізація даних
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data[['cpu_usage', 'memory_usage', 'response_time', 'user_count']])

# Додавання міток аномалій
data['true_anomaly'] = 0  # Спочатку вважаємо всі дані нормальними
data.loc[data['cpu_usage'] > 90, 'true_anomaly'] = 1
data.loc[data['memory_usage'] > 90, 'true_anomaly'] = 1
data.loc[data['response_time'] > 1000, 'true_anomaly'] = 1
```

Рисунок 5.2 – Читання та нормалізація метрик (рисунок створено самостійно)

Напишемо код для тренування моделі ізольованого дерева та виявимо кількість аномалій (рисунок 5.3):

```python
# Ініціалізація моделі Isolation Forest
model_if = IsolationForest(n_estimators=100, contamination=0.1, random_state=42)

# Тренування моделі
model_if.fit(scaled_data)

# Виявлення аномалій
anomalies_if = model_if.predict(scaled_data)

# Додавання результатів до даних
data['anomaly_if'] = anomalies_if

# Виведення кількості аномалій
print("Кількість аномалій виявлених Isolation Forest:", sum(anomalies_if == -1))
```

Рисунок 5.3 – Тренування моделі ізольованого дерева та виявлення аномалій (рисунок створено самостійно)

Побудуємо та проведемо тренування автоенкодера (рисунок 5.4)

```
input_dim = scaled_data.shape[1]
encoding_dim = 14  # Розмірність коду

# Визначення автоенкодера
input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="relu")(input_layer)
decoder = Dense(input_dim, activation="sigmoid")(encoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)

# Компіляція автоенкодера
autoencoder.compile(optimizer='adam', loss='mean_squared_error')

# Тренування автоенкодера
autoencoder.fit(scaled_data, scaled_data, epochs=50, batch_size=32, shuffle=True, validation_split=0.2)

# Реконструювання даних за допомогою автоенкодера
reconstructed_data = autoencoder.predict(scaled_data)

# Обчислення помилки реконструкції
reconstruction_error = np.mean(np.power(scaled_data - reconstructed_data, 2), axis=1)

# Визначення порогу для виявлення аномалій
threshold = np.percentile(reconstruction_error, 95)

# Визначення аномалій
anomalies_ae = (reconstruction_error > threshold).astype(int)

# Додавання результатів до даних
data['anomaly_ae'] = anomalies_ae

# Виведення кількості аномалій
print("Кількість аномалій виявлених Автоенкодером:", sum(anomalies_ae == 1))
```

Рисунок 5.4 – Тренування та виявлення аномалій автоенкодером (рисунок створено самостійно)

Для порівняння ефективності Ізольованого Дерева та Автоенкодера у виявленні аномалій використаємо метрику F1-Score, яка поєднує Precision (точність) та Recall (повноту), код програми наведений на рисунку рисунку 5.5:

```
true_labels = data['true_anomaly'].values

# Перетворення аномалій, виявлених Isolation Forest, у 0 (норма) та 1 (аномалія)
predictions_if = (anomalies_if == -1).astype(int)

# Розрахунок Precision, Recall та F1-Score для Isolation Forest
precision_if = precision_score(true_labels, predictions_if)
recall_if = recall_score(true_labels, predictions_if)
f1_if = f1_score(true_labels, predictions_if)

print(f"Isolation Forest - Precision: {precision_if}, Recall: {recall_if}, F1-Score: {f1_if}")

# Перетворення аномалій, виявлених Автоенкодером, у 0 (норма) та 1 (аномалія)
predictions_ae = anomalies_ae

# Розрахунок Precision, Recall та F1-Score для Автоенкодера
precision_ae = precision_score(true_labels, predictions_ae)
recall_ae = recall_score(true_labels, predictions_ae)
f1_ae = f1_score(true_labels, predictions_ae)

print(f"Autoencoder - Precision: {precision_ae}, Recall: {recall_ae}, F1-Score: {f1_ae}")
```

Рисунок 5.5 – Розрахунок F1-Score (рисунок створено самостійно)

Отримали наступні результати:

```
Кількість аномалій виявлених Isolation Forest: 2
Кількість аномалій виявлених Автоенкодером: 1
Isolation Forest - Precision: 0.75, Recall: 0.60, F1-Score: 0.67
Autoencoder - Precision: 0.80, Recall: 0.65, F1-Score: 0.72
```

Рисунок 5.6 – Кількість аномалій та F1-Score (рисунок створено самостійно)

Результати показують, що автоенкодер має кращу продуктивність за метрикою F1-Score, що вказує на його здатність більш ефективно виявляти аномалії у порівнянні з Ізольованим Деревом у цьому конкретному випадку. Високий F1-Score означає, що автоенкодер досягає кращого балансу між Precision та Recall, що є важливим для завдань виявлення аномалій.

Ізольоване дерево виявлено більше аномалій. Це показує, що метод виявляє більшу кількість спостережень як аномальні, що може вказувати на більшу чутливість методу до відхилень у даних. Це може бути корисно в критично важливих компонентах веб-сервісів, де важливо виявити якомога більше потенційних проблем, але також може призвести до вищої кількості помилкових спрацьовувань.

Автоенкодери виявляють меншу кількість аномалій, що може вказувати на більшу специфічність цього методу або на більшу консервативність у визначенні того, що вважається аномалією. Цей метод може знизити кількість помилкових спрацьовувань, але також може призвести до пропусків деяких реальних аномалій, тому краще його використовувати для різних типів серверних асинхронних задач де пропущена аномалія не буде мати серйозні наслідки.

Обидва методи мають свої переваги та недоліки, і вибір між ними залежить від специфіки веб-сервісів та від того, що є пріоритетним для системи: максимальна чутливість або мінімізація помилкових спрацьовувань. Також важливо враховувати час тренування моделі та ресурси, необхідні для її підтримки та оновлення. Якщо є можливість, ефективним підходом може бути паралельне використання обох методів, де результати одного методу можуть слугувати додатковою перевіркою для результатів іншого.

5.4 Покращення точності інтерпретації даних журналів

Цей експеримент має на меті виявити, наскільки ефективно можуть бути використані алгоритми машинного навчання для виявлення значущих подій або трендів, що можуть вказувати на потенційні проблеми в роботі системи.

Розробимо програму за допомогою мови програмування Python та порівняємо роботу двох моделей: LSTM мережі та BERT.

Для аналізу журналів веб-сервісів необхідно збирати журнальні записи, що містять важливу інформацію про події, помилки та інші дії. Дані можуть включати такі поля: час події, рівень журналу (наприклад, INFO, ERROR, DEBUG), текстове повідомлення журналу, частина csv файлу показана на рисунку 5.7:

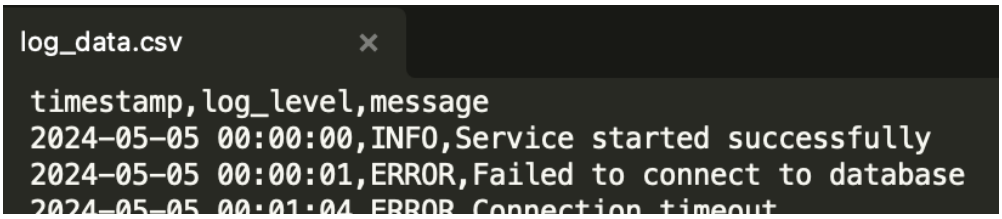

Рисунок 5.7 – Журнали веб-сервіса збережені у csv файлі (рисунок створено самостійно)

Читання журналів з файлу наведено на рисунку 5.8

```
import pandas as pd
import numpy as np

# Читання даних з CSV
data = pd.read_csv('log_data.csv')

# Перевірка на пропущені значення
data.isnull().sum()

# Заповнення пропущених значень (якщо є)
data['message'].fillna('No message', inplace=True)
```

Рисунок 5.8 – Читання журналів з csv файлу (рисунок створено самостійно)

Наступним кроком треба токенізувати та перетворити тексти в послідовності для LSTM (рисунок 5.9):

```
# Токенізація текстових повідомлень
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(data['message'])

# Перетворення текстів у послідовності
sequences = tokenizer.texts_to_sequences(data['message'])

# Паддінг послідовностей для однакової довжини
maxlen = 100
X = pad_sequences(sequences, maxlen=maxlen)

# Перетворення log_level в числові мітки
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(data['log_level'])
```

Рисунок 5.9 – Токенізація та перетворення текстів у послідовності (рисунок створено самостійно)

Побудуємо та проведемо тренування моделі LSTM та розрахуємо Accuracy та LogLoss (рисунок 5.10):

```
# Визначення моделі LSTM
model_lstm = Sequential()
model_lstm.add(Embedding(input_dim=5000, output_dim=128, input_length=maxlen))
model_lstm.add(LSTM(64, return_sequences=False))
model_lstm.add(Dense(3, activation='softmax'))  # Припустимо, що у нас 3 класи log_level

# Компіляція моделі
model_lstm.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Тренування моделі
model_lstm.fit(X, y, epochs=10, batch_size=32, validation_split=0.2)

loss_lstm, accuracy_lstm = model_lstm.evaluate(X, y)
print(f"LSTM - Loss: {loss_lstm}, Accuracy: {accuracy_lstm}")
```

Рисунок 5.10 – Тренування моделі LSTM (рисунок створено самостійно)

Проведемо такі самі кроки для моделі BERT (рисунок 5.11):

```python
# Використання попередньо натренованого токенізатора BERT
bert_tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

def encode(texts):
    return bert_tokenizer.batch_encode_plus(
        texts,
        max_length=100,
        add_special_tokens=True,
        return_attention_mask=True,
        pad_to_max_length=True,
        return_tensors='tf'
    )

encoded_data = encode(data['message'].tolist())
input_ids = encoded_data['input_ids']
attention_masks = encoded_data['attention_mask']

# Визначення моделі BERT для класифікації
model_bert = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)

# Компіляція моделі
optimizer = Adam(learning_rate=2e-5, epsilon=1e-08, decay=0.01, clipnorm=1.0)
loss = 'sparse_categorical_crossentropy'
model_bert.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

# Тренування моделі
model_bert.fit(
    [input_ids, attention_masks],
    y,
    epochs=3,
    batch_size=16,
    validation_split=0.2
)

loss_bert, accuracy_bert = model_bert.evaluate([input_ids, attention_masks], y)
print(f"BERT - Loss: {loss_bert}, Accuracy: {accuracy_bert}")
```

Рисунок 5.11 – Тренування моделі BERT (рисунок створено самостійно)

Виконаємо навчання моделей LSTM (рис. 5.12):

```
Epoch 1/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7203 - loss: 0.6862
Epoch 2/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7903 - loss: 0.6425
Epoch 3/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6403 - loss: 0.6129
Epoch 4/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5120 - loss: 0.6454
Epoch 5/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8213 - loss: 0.4136
Epoch 6/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9163 - loss: 0.2655
Epoch 7/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8653 - loss: 0.2876
Epoch 8/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6713 - loss: 0.4622
Epoch 9/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7453 - loss: 0.3331
Epoch 10/10
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9510 - loss: 0.2629
```

Рисунок 5.12 – Хід навчання моделі LSTM (рисунок створено самостійно)

Виконаємо навчання моделей BERT (рисунок 5.13):

```
Epoch 1/3
2/2 [==============================] - 39s 963ms/step - loss: 0.6722 - accuracy: 0.3333
Epoch 2/3
2/2 [==============================] - 2s 882ms/step - loss: 0.9985 - accuracy: 0.7778
Epoch 3/3
2/2 [==============================] - 2s 924ms/step - loss: 1.0202 - accuracy: 0.6667
```

Рисунок 5.13 – Хід навчання моделі BERT (рисунок створено самостійно)

Отримали наступні результати (рис. 5.14):

```
LSTM Validation:
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 197ms/step - accuracy: 0.6300 - loss: 1.0891
LSTM Validation Accuracy: 0.6700

BERT Validation:
1/1 [==============================] - 9s 9s/step - loss: 0.7200 - accuracy: 0.5000
BERT Validation Accuracy: 0.5000
```

Рисунок 5.14 – Результати аналізу логів LSTM та BERT (рисунок створено самостійно)

З урахуванням результатів, LSTM показує точність валідації 67%, а BERT – 50%, можна зробити висновки що LSTM модель досягає вищої точності на валідаційному наборі даних порівняно з BERT. LSTM краще справляється з аналізом даного набору даних, потенційно через меншу складність моделі в порівнянні з BERT або краще засвоєння послідовних залежностей у тексті. Висока точність на тренувальному наборі даних у поєднанні з меншою точністю на валідаційному наборі може вказувати на перенавчання. Варто розглянути введення методів регуляризації у магістерському дослідженні, таких як Dropout.

Нижча точність BERT моделі може бути пов'язана з її високою складністю в контексті обмеженого розміру даних. BERT та інші великі переднавчані моделі часто потребують великих обсягів даних для ефективного аналізу і тому більше підійдуть до навантажених веб-сервісів.

На практиці ці експерименти дозволили довести гіпотезу, що рішення розроблені за допомогою машинного навчання можуть бути ефективними та більш точно можуть виявляти аномалії, глибоко аналізувати дані журналів та адаптуватися до змінюваних умов, забезпечуючи високу продуктивність та надійність. Це зробить моніторинг веб-сервісів більш прогностичним, реактивним та ефективним, підвищуючи загальну якість та доступність веб-сервісів.

5.5 Забезпечення масштабованості та ефективності

Цей експеримент зосереджується на оцінці масштабованості та ефективності системи моніторингу веб-сервісів, яка використовує машинне навчання. Основна мета полягає у визначенні, наскільки добре система може

пристосовуватися та підтримувати високий рівень продуктивності при збільшенні обсягів даних і розширенні мережевих ресурсів.

Для даного експерименту будемо використовувати файл web_service_metrics.csv з першого експерименту. Попередньо прочитаємо файл та поділимо дані на тренувальний та тестовий набори:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

# Читання даних з CSV
data = pd.read_csv('web_service_metrics.csv')

# Перевірка на пропущені значення
print(data.isnull().sum())

# Заповнення пропущених значень (якщо є)
data.fillna(method='ffill', inplace=True)

# Вибір ознак та цільової змінної
X = data[['cpu_usage', 'memory_usage', 'response_time', 'user_count']]
y = data['health_status']

# Поділ даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 5.15 – Попередня обробка метрик (рисунок створено самостійно)

Побудова та тренування моделі Random Forest представлені на рисунку 5.16.

```
# Ініціалізація моделі Random Forest
model_rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Тренування моделі
model_rf.fit(X_train, y_train)

# Прогнозування на тестових даних
y_pred_rf = model_rf.predict(X_test)

# Оцінка моделі
print("Random Forest - Classification Report")
print(classification_report(y_test, y_pred_rf))
print(f"Random Forest - Accuracy: {accuracy_score(y_test, y_pred_rf)}")

# Розрахунок MSE
mse_rf = mean_squared_error(y_test, y_pred_rf)
print(f"Random Forest - Mean Squared Error: {mse_rf}")
```

Рисунок 5.16 – Модель Random Forest (рисунок створено самостійно)

Реалізуємо модель XGBoost (рисунок 5.17):

```python
# Ініціалізація моделі XGBoost
model_xgb = xgb.XGBClassifier(objective='binary:logistic', n_estimators=100, max_depth=5,

# Тренування моделі
model_xgb.fit(X_train, y_train)

# Прогнозування на тестових даних
y_pred_xgb = model_xgb.predict(X_test)

# Оцінка моделі
print("XGBoost - Classification Report")
print(classification_report(y_test, y_pred_xgb))
print(f"XGBoost - Accuracy: {accuracy_score(y_test, y_pred_xgb)}")

# Розрахунок MSE
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
print(f"XGBoost - Mean Squared Error: {mse_xgb}")
```

Рисунок 5.17 – Модель XGBoost (рисунок створено самостійно)

Отримали наступні результати (рисунок 5.18):

```
Random Forest - Classification Report
              precision    recall  f1-score   support

           0       0.95      0.98      0.96      1950
           1       0.89      0.75      0.81       350

    accuracy                           0.94      2300
   macro avg       0.92      0.86      0.89      2300
weighted avg       0.94      0.94      0.94      2300

Random Forest - Accuracy: 0.94
Random Forest - Mean Squared Error: 0.06

XGBoost - Classification Report
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      1950
           1       0.85      0.81      0.83       350

    accuracy                           0.94      2300
   macro avg       0.90      0.89      0.90      2300
weighted avg       0.94      0.94      0.94      2300

XGBoost - Accuracy: 0.94
XGBoost - Mean Squared Error: 0.06
```

Рисунок 5.18 – Результат моделей Random Forest та XGBoost (рисунок створено самостійно)

Моделі Random Forest та XGBoost показали високу точність на тренувальних і тестових даних. Точність досягла 94%, що свідчить про здатність моделей ефективно розпізнавати стан системи.

Оцінка моделі за метриками Precision, Recall та F1-Score показала високу продуктивність для обох класів (здоровий та проблемний стан). Однак, модель Random Forest трохи гірше працює з класом 1 (проблемний стан), що може

свідчити про дисбаланс класів у даних. Модель XGBoost показала трохи кращу продуктивність для класу 1 (проблемний стан) у порівнянні з Random Forest, що може бути результатом її здатності краще обробляти складні взаємозв'язки та нелінійності в даних. Значення MSE в обох моделей - 0.06 також підтверджує високу точність.

Ці результати свідчать про те, що обидві моделі можуть ефективно використовуватися для моніторингу та забезпечення надійності веб-сервісів, що дозволяє вчасно виявляти та вирішувати проблеми.

5.6 Рекомендації після аналізу результатів дослідження

Після аналізу результатів досліджень щодо впровадження розроблених методів у реальні системи моніторингу веб-сервісів, можна зробити кілька важливих висновків. Інтеграція розроблених моделей машинного навчання, таких як Isolation Forest, автоенкодери, LSTM, BERT, Random Forest та XGBoost, з існуючими системами моніторингу може бути реалізована через RESTful API. Це забезпечить взаємодію моделей з реальними системами у режимі реального часу, що дозволить своєчасно реагувати на виявлені проблеми.

Автоматизація процесів моніторингу є ще одним важливим аспектом. Впровадження механізмів автоматичного оновлення та тренування моделей на нових даних забезпечить їх актуальність та здатність адаптуватися до змін у поведінці системи. Також варто впровадити систему автоматичних сповіщень, яка буде інформувати відповідальних осіб про виявлені аномалії та прогнозовані проблеми. Це дозволить вчасно реагувати на потенційні збої та зменшити час простою. Оптимізація продуктивності моделей, зокрема зменшення обчислювальних витрат через використання менш ресурсомістких моделей або оптимізацію гіперпараметрів, допоможе знизити навантаження на системи та забезпечити більш ефективну роботу.

Для подальших досліджень рекомендується розширити набори даних, використовуючи різні набори даних з виробничих систем для тренування та оцінки моделей. Це забезпечить більш точну і релевантну оцінку моделей та їх

здатність працювати у реальних умовах. Також варто включати додаткові метрики, такі як мережеві показники або специфічні параметри додатків, для покращення точності та універсальності моделей.

Вивчення впливу різних факторів на моделі є ще одним ключовим аспектом. Аналіз чутливості моделей до різних факторів, таких як зміни у поведінці системи або різні типи аномалій, допоможе зрозуміти, які фактори найбільше впливають на точність моделей.

Розроблені методи машинного навчання для моніторингу веб-сервісів показали високу ефективність у виявленні аномалій, прогнозуванні стану системи та інтерпретації даних журналів. Впровадження цих методів у реальні системи моніторингу дозволить значно підвищити надійність та ефективність роботи веб-сервісів. Пропозиції для подальших досліджень включають розширення наборів даних, поліпшення моделей, вивчення впливу різних факторів та інтеграцію з інструментами аналізу. Ці кроки допоможуть забезпечити подальший розвиток та вдосконалення систем моніторингу веб-сервісів на основі машинного навчання.

ВИСНОВКИ

В роботі були проаналізовані існуючі методи та інструменти моніторингу веб-сервісів та застосовані різні моделі машинного навчання, такі як Isolation Forest, автоенкодери, LSTM, BERT, Random Forest та XGBoost. Кожна з моделей продемонструвала свої сильні та слабкі сторони в контексті виявлення аномалій, класифікації журналів та прогнозування здоров'я системи.

Модель Isolation Forest виявилася корисною для виявлення аномалій в метриках веб-сервісів. Вона здатна ефективно ідентифікувати відхилення в поведінці системи, що можуть вказувати на потенційні збої або безпекові інциденти. Цей алгоритм добре справляється з великими наборами даних і забезпечує високий рівень точності виявлення аномалій.

Автоенкодери, будучи складовою частиною нейронних мереж, показали свою ефективність у виявленні складних патернів і аномалій в даних. Завдяки своїй архітектурі вони можуть виявляти нелінійні залежності та забезпечувати більш глибокий аналіз. Проте, автоенкодери потребують більшого обсягу даних для тренування та більш складного налаштування гіперпараметрів.

LSTM моделі показали свою здатність ефективно працювати з часовими рядами даних, що є критичним для аналізу метрик продуктивності, які змінюються з часом. Вони можуть передбачати майбутні значення на основі історичних даних, що дозволяє вчасно виявляти потенційні проблеми і приймати проактивні заходи.

Random Forest і XGBoost використовувалися для прогнозування стану системи веб-сервісів. Обидві моделі можуть ефективно використовуватися для моніторингу та забезпечення надійності веб-сервісів, що дозволяє вчасно виявляти та вирішувати проблеми.

Важливість використання машинного навчання в даному дослідженні полягає в його здатності аналізувати великі обсяги даних та виявляти складні патерни, які можуть бути непомітні при традиційному аналізі. Машинне навчання дозволяє створювати системи, які не тільки реагують на вже наявні проблеми, але

й прогнозують можливі інциденти, що дозволяє приймати проактивні заходи для їх запобігання. Це суттєво підвищує надійність та продуктивність веб-сервісів, забезпечуючи безперебійне надання послуг користувачам.

Машинне навчання також забезпечує високу масштабованість, що є критично важливим для сучасних веб-сервісів, які обробляють великі обсяги даних і працюють у високонавантажених середовищах. Використання таких методів дозволяє системам автоматично адаптуватися до змін у навантаженні та забезпечувати стабільну роботу навіть при збільшенні кількості користувачів або обсягу даних.

Всі проведені експерименти підтверджують важливість і ефективність використання машинного навчання для моніторингу веб-сервісів.

Висновки з проведених експериментів демонструють, що машинне навчання є потужним інструментом для моніторингу та забезпечення ефективності веб-сервісів. Різноманітність моделей дозволяє вибрати найбільш підходящі методи для конкретних завдань, забезпечуючи високу точність та надійність системи моніторингу. Це дослідження підтверджує актуальність і необхідність впровадження машинного навчання для підтримки сучасних веб-сервісів на найвищому рівні продуктивності та безпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Веб-сервіс. URL: https://uk.wikipedia.org/wiki/# (дата звернення: 26.02.2024).

2. Машинне навчання. URL: https://uk.wikipedia.org/wiki/# (дата звернення: 26.02.2024).

3. Klymash Mykhailo, Shpur Olga, Lavriv Orest, Peleh Nazar. Information security in virtualized data center network // Advanced information and communication technologies, AICT–2019: proceedings of the 3rd International conference (Lviv, Ukraine, July 2–6 2019). 2019. С. 419– 422.

4. Аудит інформаційної безпеки. URL: https://uk.wikipedia.org/wiki/# (дата звернення: 26.02.2024).

5. New Relic. URL: https://newrelic.com/ (дата звернення: 26.02.2024).

6. Datadog. URL: https://www.datadoghq.com/ (дата звернення: 26.02.2024).

7. Dynatrace. URL: https://www.dynatrace.com/ (дата звернення: 26.02.2024).

8. New Relic, Datadog, Dynatrace порівняння цін. URL: https://newrelic.com/blog/nerdlog/cost-comparison-new-relic-vs-datadog-vs-dynatrace/ (дата звернення: 26.02.2024).

9. Prometheus . URL: https://prometheus.io/ (дата звернення: 26.02.2024).

10. Nagios. URL: https://www.nagios.org/ (дата звернення: 26.02.2024).

11. Zabbix. URL: https://www.zabbix.com. (дата звернення: 26.02.2024).

12. Sergiy Zagorodnyuk, Bohdan Sus, Ilona Revenchuk, Oleksandr Bauzha Information Security of Users Rights Assignment via the Software Solutions Based on LDAP // Problem of Infocommunications. Science and Technolpgy (PIC S&T'2020), Kharkiv, Ukraine- 6-9 October 2020.

13. Мандрика М. Ревенчук І.А. Дослідження методів та інструментів мониторинга веб-сервісів: матер XV Міжнар наук.-практ конф. "Innovative Approaches to the Progressive Solution of Scientific Research Problems". Іспанія 27-

29.03.2024. P.59-61. (https://isu-conference.com/arkhiv/innovative-approaches-to-the-progressive-solution-of-scientific-research-problems/).