UNIVERSIDADE
BEIRA INTERIOR

Anomaly Detection in Microservices Using Ensemble Methods

Nuno Miguel da Silva Salvado

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Dr. Mário Marques Freire

Covilhã, outubro 2025

Declaração de Integridade

Eu, **Nuno Miguel da Silva Salvado**, que abaixo assino, estudante com o número **M11364** do curso de Mestrado em Engenharia Informática da Faculdade de Engenharia da Universidade da Beira Interior, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridade da Universidade da Beira Interior**.

Mais concretamente, afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, e que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, assumindo na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã, 12 de outubro de 2025

Assinado por: Nuno Miguel da Silva Salvado
Identificação: BI15217400
Data: 2025-10-13 às 00:42:05

Nuno Miguel da Silva Salvado

Acknowledgements

Over the past few years, I have been deeply influenced by a special group of people, whose support has been critical to my development and success. In this section, I express my sincere gratitude for the invaluable support you have offered me in the most varied ways.

Firstly, I would like to thank my supervisor, Professor Mário Freire, for his valuable support, guidance, knowledge, principles and methodologies that were transmitted to me, which were fundamental in the progress of this initiative.

To those who have always been by my side, especially in the most difficult moments, I express my deepest gratitude. The constant presence and unstoppable support of my parents and my sister were the pillar that sustained me throughout my academic career. Every victory I've achieved is in large part the result of their love and dedication.

To my incredible girlfriend, I express genuine and warm gratitude. It was your unconditional support and your presence that, in moments of uncertainty, brought me confidence and comfort to face this journey with even more determination, inspiring me to achieve my goals.

To my closest friends, I thank you for the inexhaustible way in which you supported me, understood me and took my focus off work when my mind was in need. For all these reasons and a few more, I convey my most sincere feelings of gratitude for what you gave me during these months.

To some of my course colleagues, I thank you for your collaboration, constructive criticism, exchange of ideas and mutual support throughout this journey. All of this made my experience even more enriching.

Last but not least (quite the opposite), my most special gratitude is dedicated to the memory of my dear father, whose love, wisdom and inspiration continue to shape my path, even in his physical absence. May his eternal presence give me strength to face life's challenges and may his legacy endure in everything I achieve. I am proud to be his son, it is with admiration that I speak of him and it is with certainty that I say that he will always be my inspiration and will always remain in my heart. Thank you, Dad.

Abstract

Anomaly detection represents a critical factor in ensuring the reliability and resilience of microservice-based systems, where failures can rapidly propagate and compromise overall service availability. This dissertation investigates the application of classical Machine Learning (ML) algorithms and ensemble methods for anomaly detection in microservices, using the *TraceRCA* dataset as a representative benchmark.

The work begins with a systematic literature review, which categorizes traditional and ML-based approaches to anomaly detection, identifying key research gaps and datasets. Building on this foundation, a complete experimental pipeline was developed, including preprocessing, feature engineering, and anomaly labeling, followed by the evaluation of multiple baseline classifiers such as Logistic Regression (LogReg), Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), and Gaussian Naïve Bayes (GNB).

To enhance predictive performance, ensemble techniques including Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), and Histogram-based Gradient Boosting (HGBM) were implemented and compared against baselines. The evaluation considered both predictive accuracy and resource efficiency, measuring metrics such as F1-score, precision, recall, accuracy, Receiver Operating Characteristic – Area Under the Curve (ROC-AUC), as well as execution time, Random Access Memory (RAM) consumption, and Central Processing Unit (CPU) utilization.

The experimental results demonstrate that ensemble models consistently outperform baselines, with boosting-based methods (XGBoost, LightGBM, HGBM) achieving the highest predictive performance, while RF offered stable results with moderate computational overhead. These findings highlight the trade-offs between accuracy and efficiency, underlining the importance of careful algorithm selection according to deployment constraints.

This research contributes by providing a comprehensive benchmark of ML and ensemble methods for anomaly detection in microservices, bridging the gap between predictive performance and practical applicability in real-world environments.

Keywords

Anomaly Detection, Ensemble Methods, Machine Learning, Microservices, Performance Evaluation

Contents

List of Figures

List of Tables

Acronyms List

ACM CCS	Association for Computing Machinery Computing Classification System
AE	Autoencoder
AI	Artificial Intelligence
AMOC	At Most One Change
API	Application Programming Interface
AutoML	Automated Machine Learning
BOCPD	Bayesian Online Change Point Detection
CBN	Causal Bayesian Network
CEP	Complex Event Processing
CNN	Convolutional Neural Network
CPG	Context Propagation Graph
CPU	Central Processing Unit
DA	Data Analysis
DAE	Deep Autoencoder
DBMS	Database Management System
DCRNN	Diffusion Convolutional Recurrent Neural Network
DDPM	Denoising Diffusion Probabilistic Model
DL	Deep Learning
DNN	Deep Neural Network
DQN	Deep Q-Network
DS	Design Science
DSRM	Design Science Research Methodology
DSW	Dynamic Status Warping
DT	Decision Tree
EFA	Entity Fault Association
EVT	Extreme Value Theory

FFAD	Fine-grained multi-modal Association and Frequency Domain Analysis for Anomaly Detection
FNN	Feedforward Neural Network
GAE	Graph Autoencoder
GAT	Graph Attention Network
GBDT	Gradient Boosting Decision Trees
GBM	Gradient Boosting Machine
GCN	Graph Convolutional Network
GGNN	Gated Graph Neural Network
GM	Gaussian Mixture
GMTA	Graph-based Microservice Trace Analysis
GNB	Gaussian Naïve Bayes
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
GTN	Graph Transformer Network
GVAE	Graph Variational Autoencoder
HGBM	Histogram-based Gradient Boosting
HGTN	Heterogeneous Graph Transformer Network
HRLHF	Hierarchical Reinforcement Learning with Human Feedback
HTTP	Hypertext Transfer Protocol
ICL	In-Context Learning
ICPTL	Intra-Cluster Parameter Transfer Learning
IF	Isolation Forest
ImpAPTr	Impact Analysis based on Pruning Tree
JI	Jaccard Index
KGE	Knowledge Graph Embedding
KL	Kullback–Leibler
KNN	K-Nearest Neighbors

LC-VAE	Latent-Conditioned Variational Autoencoder
LightGBM	Light Gradient Boosting Machine
LIME	Local Interpretable Model-agnostic Explanations
LLM	Large Language Model
LM	Language Model
LogReg	Logistic Regression
LR	Linear Regression
LSTM	Long Short-Term Memory
MAO	Multimodal Adaptive Optimization
ML	Machine Learning
MLP	Multilayer Perceptron
MPD	Mean Pairwise Distance
MRF	Markov Random Field
NN	Neural Network
OC-SVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
PDFA	Probabilistic Deterministic Finite Automaton
PGEM	Probabilistic Graphical Event Model
PV	Position Value
RAM	Random Access Memory
RCA	Root Cause Analysis
RF	Random Forest
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROC-AUC	Receiver Operating Characteristic – Area Under the Curve
SEM	Structural Equation Models
SGT	Sequence Graph Transform
SHAP	SHapley Additive exPlanations

SNA	Social Network Analysis
SOP	Standard Operating Procedure
SSL	Self-Supervised Learning
STGNN	Spatio-Temporal Graph Neural Network
SVDD	Support Vector Data Description
SVM	Support Vector Machine
SyncM	Sync-Millibottleneck
TEMAML	Transformer Encoder with Model-Agnostic Meta-Learning
T-RCA	Threshold-based Root Cause Analysis
VAE	Variational Autoencoder
VAMP	Visual Analytics for Microservices Performance
VGAE	Variational Graph Autoencoder
XAI	Explainable Artificial Intelligence
XGBoost	eXtreme Gradient Boosting

Chapter 1

Introduction

1.1 Scope and Motivation

Currently, software users are increasingly demanding, which is why it is necessary to develop faster, more efficient software that allows the performance of more complicated and large-scale tasks. Previously, software was based on monolithic architectures which made more complicated tasks difficult and time-consuming. Based on this, microservices architecture emerged that allow increasing the agility, scalability and reliability of software [JM20].

Microservices architecture typically consist of multiple interconnected services, thus creating a challenging landscape for manual anomaly detection, but have the advantage that the malfunction of one microservice does not affect the continuity of the others. However, microservices have the disadvantage that their complexity makes them more fragile and susceptible to anomalies. The challenges posed by anomaly detection in microservices have driven the adoption of automated approaches, particularly those leveraging ML techniques. Unlike traditional rule-based anomaly detection systems, ML-based solutions can generalize large-scale data and dynamically adapt to evolving patterns in system behavior [WMD+24]. With the ability to be repurposed and adjusted for almost any task, ML models have been used in the most varied sectors, providing them with competitive advantages, namely greater efficiency and cost reduction as they are suitable for detecting anomalies in complex microservice environments [SB22] [KAAD22] [FJRH23] [NPR23].

Anomaly detection and ML pose natural connections due to the latter's ability to detect and extract patterns (in this case, irregular ones) [NTND21]. This generic duality allows for many interesting opportunities that cater to the most specific of use cases, such as microservice environments.

There is unquestionable value in this area of research, both from an efficiency standpoint as well as a security measure. As applications evolve towards a reliance on microservices (for their flexibility and frequency of upgrades) [JM20], it is becoming increasingly more valuable to design solutions that ensure no major anomalies are present. These unwanted defects are not only responsible for performance degradation, but also pose potential security hazards [SB22].

For the purposes of this work, we envision the use of ML based solutions so that they can learn normal behavior patterns and automatically detect deviations that may indicate is-

sues such as performance degradation, security threats, or faults. These anomalies can manifest in various forms, including unusual traffic patterns, resource usage spikes, or unexpected service responses [SB22].

This dissertation will largely focus on analyzing existing solutions regarding anomaly detection in microservices, while also reflecting on how the state-of-the-art could be pushed even further. Therefore, it is important to consider this work's scope, in terms of key topics and how they are related. Table 1.1 summarizes the Association for Computing Machinery Computing Classification System (ACM CCS) classifications relevant to this work [Ass12].

Table 1.1: ACM CCS Classifications relevant to this work [Ass12].

ACM CCS Classification	**Description**
Software and its engineering > Software organization and properties > Software system structures > Distributed systems organizing principles.	Explores principles of distributed systems, key to modern microservices architecture.
Computing methodologies > ML > Learning paradigms.	Covers ML paradigms, including supervised learning and anomaly detection systems, emphasizing their adaptability to complex, evolving behaviors.
Applied computing > Enterprise computing > Business process management.	Relates ML-driven anomaly detection to improved operational efficiency.
Information systems > Data management systems > Database management system engines > Parallel and distributed Database Management Systems (DBMSs).	Highlights challenges in managing and analyzing data in distributed microservice systems.

The relationships between these key topics are further illustrated in Figure 1.1, which provides a conceptual overview of the three pillars that serve as the base context for this work.

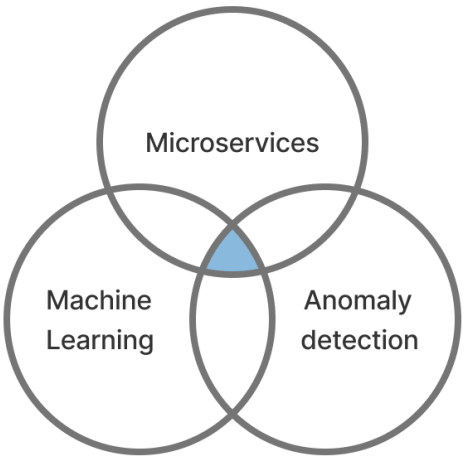

Figure 1.1: Overview of the key concepts that will be targeted by this work.

1.2 Problem Statement

The problem I propose to solve in this dissertation is the detection of anomalies in microservice environments. There are several methodologies that can be used for anomaly detection, traditional and ML-based. ML-based methodologies encompass several algorithms, including baseline models and ensemble methods. This dissertation focuses on these two categories, baseline algorithms (such as DTs, LogReg, and SVMs) and ensemble methods, aiming to conduct a comparative analysis of their performance and to determine which approach is more effective for anomaly detection in microservice environments.

With the development of this dissertation it is also intended to optimize hyperparameters of the algorithms used. These hyperparameters are not learned during the training of the algorithm, but rather configured in advance. The optimization of these hyperparameters is performed by applying tests with different values for these parameters before the model is trained, then choosing the combination of values that it presents provides the best performance to the model under study.

Usually, all research processes seek to answer a set of questions or problems of their own. These research questions provide a structure to guide the investigation according to the topic under study. The excellence of these issues directly influences the importance and quality of the results obtained. Therefore, with the aim of exploring anomaly detection in microservice environments, this work will seek to answer the following questions:

1. What is the performance of ensemble algorithms on the classification of anomalies in microservices?
2. What is the performance of baseline models, with particular emphasis on DTs, on the classification of anomalies in microservices?
3. What is the impact of hyperparameter optimization on classification performance?

1.3 Research Objectives

This dissertation naturally follows the context above, and its main objectives can be considered twofold:

- **Provide an overview** of what has been done in this field, how each work attempts to solve the problem and what are the most prevalent ideas and data sources.
- **Develop a new solution** that leverages existing knowledge while also introducing tweaks that push the state-of-the-art even further.

The evaluation methodology adopted in this dissertation follows a three-set partitioning strategy, in which the dataset is divided into 70% for training, 20% for validation, and 10% for testing. Hyperparameter tuning is carried out through 10-fold cross-validation

applied exclusively to the training subset, while the final model evaluation is performed on the independent *hold-out* test set. This procedure ensures a reliable assessment of the capacity of the models and prevents data leakage between the training and testing stages.

1.4 Adopted Approach for Solving the Problem

Research on Design Science (DS) can be carried out effectively through a mental model. The Design Science Research Methodology (DSRM) model (Figure 1.2) [PTRC07], used in the research work leading to the writing of this dissertation, consists of a process model with six steps whose objective is to provide a mental model for the characteristics of research results. This steps allow controlling and improving research through continuous analysis of results.

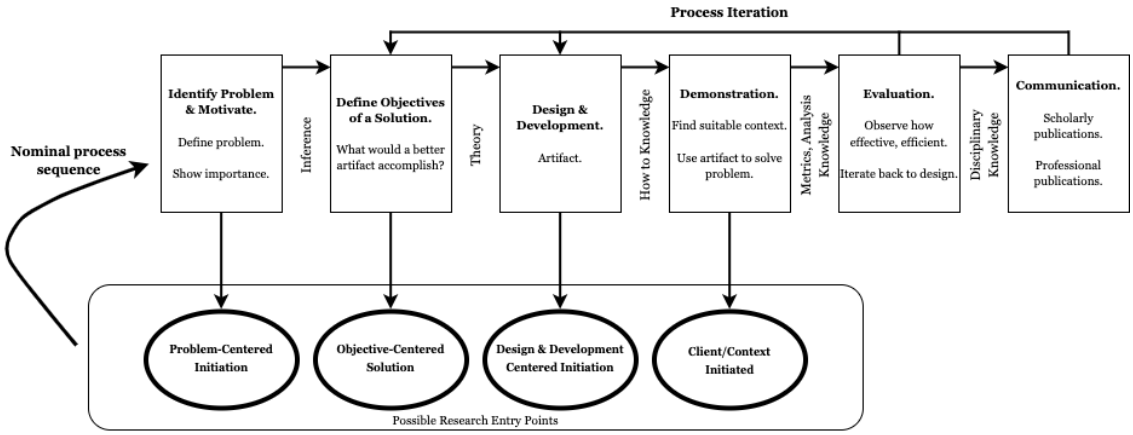

Figure 1.2: DSRM process model (adapted from [PTRC07]).

In this dissertation, it is essential to have a clear and organized vision of the set of fundamental tasks that make up the investigation. The specific tasks to be carried out are carefully outlined, presenting a systematic and strategic approach in order to achieve the objectives of the dissertation and direct this research in a comprehensive and rigorous manner. The tasks present in this dissertation are described in the following points and the expected time for their completion is represented in the Gantt diagram in Figure 1.3:

- **Literature Review and Current Landscape**

For a better understanding and explanation of the problem that I proposed to solve during the completion of this dissertation, it was extremely important to carry out an exhaustive review of the literature previously. To this end, during this stage, a research and literature review was carried out on currently available anomaly detection methods in microservices as well as ML methods in order to verify their applicability in detecting anomalies in real life. Therefore, in this stage, the concepts associated with microservice environments and detection of anomalies in microservices are addressed.

Anomaly Detection in Microservices Using Ensemble Methods

- **Problem Statement and Motivation**

The problem I proposed to solve in this dissertation is the detection of anomalies in a microservices architecture. As software based on microservices evolves, its complexity increases exponentially, as does the number of instances involved, resulting in greater difficulty in monitoring. The microservices architecture involves multiple Application Programming Interface (API) calls and communication between microservices, meaning the processing of an incredibly high volume of data, which makes ML a suitable approach for identifying deviations from normal parameters. However, to do so, it is necessary to answer questions such as: What is the most appropriate and effective methodology and what information should be monitored for the accurate detection of anomalies?

What led me to choose this topic is the fact that microservices architecture is an area in constant evolution and increasingly important and with high application in the construction of software platforms, which is why it is extremely important to create mechanisms for detecting anomalies.

- **Research Objectives**

The main objective of this dissertation is to investigate the applicability of ML algorithms and ensemble methods for anomaly detection in microservices architecture. The work seeks to compare different approaches, evaluate their effectiveness, and provide insights into their practical relevance for improving the reliability of microservice-based systems.

- **Selection and Characterization of the Dataset**

From among the datasets listed in Table 2.3, a dataset was selected to be used in the course of the remaining work, the *TraceRCA* dataset [LCJ+21] [CLJ+23]. The *TraceRCA* dataset was selected due to its representativeness of real-world microservice environments and its availability of labeled anomalies. After selecting the dataset, a brief characterization has been made.

- **ML Algorithms for Anomaly Detection**

There are several ML algorithms for detecting anomalies in microservices architecture, some of which are present in the strategies referred to in Subsection 2.4.3.

Classical ML algorithms, including baseline models such as DTs, LogReg, SVMs, GNB, and other approaches, were first considered as reference points for anomaly detection performance. Given that ensemble algorithms, available in scikit-learn [sci25a], have demonstrated strong performance in classifying intrusions within multitenant environments with containerized microservices, these methods were selected as the primary focus for the development of this dissertation.

Baseline models are generally valued for their simplicity, interpretability, and relatively low computational cost. They can provide fast results and are easy to understand, which makes them attractive for constrained environments. However, they usually fall short in predictive performance compared to ensemble methods, particularly when handling more complex patterns of anomalies.

In turn, ensemble methods combine predictions from several base estimators built from a given learning algorithm in order to improve generalization into a single estimator. The two most common types of ensemble algorithms are Gradient Boosting Decision Trees (GBDT) and RFs.

Each ML algorithm includes parameters that are automatically adjusted during training. The parameters that are not learned during the algorithm training, but are configured beforehand, are called hyperparameters. For example, in the case of DT algorithms, the parameters include the weights associated with the different characteristics that are adjusted during training. In turn, the hyperparameters correspond to settings such as the maximum depth of the tree, the minimum number of samples in a leaf or the criteria used to divide the nodes.

With the development of this work, we also intended to optimize the hyperparameters of the algorithms used. This hyperparameter optimization process involved testing different values for these parameters before training the model to find the combination that offered the best performance [RIMF24].

- **Expected Results and Performance Evaluation**

The expected results are those obtained from comparing the performance of ML algorithms, including both baseline models (DT, LogReg, SVM, etc.) and ensemble methods, in detecting anomalies in microservice environments.

To this end, the following metrics were used: precision, recall, F1-score, train accuracy / test accuracy, and ROC-AUC.

Train accuracy is calculated by applying the model to the training dataset, while test accuracy evaluates performance on previously unseen data. If the train accuracy is much higher than the test accuracy, this indicates overfitting, meaning the model fits the training data well but fails to generalize effectively to new instances.

In addition, the ROC-AUC metric provides an aggregated measure of the model's capacity to discriminate between normal and anomalous samples under varying classification thresholds.

Finally, beyond predictive performance, this work also considers resource efficiency, evaluating execution time, RAM consumption, and CPU utilization during model training.

This dual perspective ensures that the selected approaches are not only accurate but also computationally viable in large-scale microservice environments.

- **Dissemination of Results and Writing the Master's Dissertation**

The dissemination of the results of this work through international conferences and journals is considered an important step for future development. Although no publication was submitted during the timeline of this dissertation, the results obtained provide a solid foundation for preparing scientific articles that may be extended and submitted to relevant venues after the conclusion of this work.

The master's dissertation itself was written based on research and literature reviews carried out previously, as well as on the experimental results obtained during testing and data collection, in accordance with the rules defined for the validation of this work. The writing process followed the same structure as the practical work, ensuring that each step of the methodology was consistently documented and aligned with the experimental procedures.

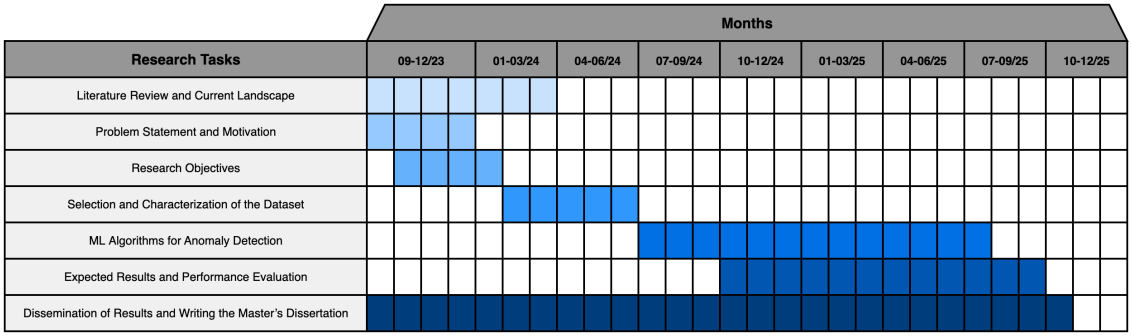

Figure 1.3: Gantt diagram with the dissertation outline.

1.5 Main Scientific Contributions

The main scientific contributions of this dissertation can be summarized as follows:

- **Systematic evaluation of baseline and ensemble models** for anomaly detection in microservice environments, using the *TraceRCA* dataset as a reference. The results demonstrate that ensemble methods, particularly boosting-based approaches such as XGBoost, HGBM, and LightGBM, consistently outperform classical baselines across multiple evaluation metrics, including F1-score, accuracy, and ROC-AUC.

- **Assessment of computational trade-offs** by measuring execution time, memory usage, and CPU utilization. This dual perspective highlights the balance between predictive accuracy and operational efficiency, an aspect often overlooked in related anomaly detection studies.

- **Analysis of the impact of hyperparameter optimization** across all evaluated models, demonstrating that careful tuning of key parameters significantly enhances robustness and generalization, particularly for ensemble methods.
- **Establishment of a benchmark and identification of future research directions**, as the findings obtained with the *TraceRCA* dataset provide a strong comparative baseline for subsequent research in microservice-based anomaly detection. Furthermore, the study highlights potential avenues for improvement, including the adoption of advanced architectures such as Graph Neural Networks (GNNs) and Long Short-Term Memorys (LSTMs), as well as the integration of Explainable Artificial Intelligence (XAI) techniques (e.g., SHapley Additive exPlanations (SHAP), Local Interpretable Model-agnostic Explanations (LIME)) to enhance interpretability and practical applicability.

1.6 Dissertation Organization

This dissertation is organized into five chapters, and the content and organization of these chapters can be summarized as follows.

Chapter 1 - **Introduction** - presents the scope and motivation of the research, the objectives, methodological approach, and expected scientific contributions.

Chapter 2 - **Background and Related Work** - defines the theoretical and practical context of the study. It presents a detailed literature review on anomaly detection in microservices and describes the algorithms, techniques, and datasets commonly used in this field.

Chapter 3 - **Classification Approach** - focuses on pre-processing data and implementing the algorithms used in order to obtain consistent results, thus solving the problem in question.

Chapter 4 - **Results and Discussion** - presents and discusses the results obtained from the application of ML algorithms for anomaly detection, particularly ensemble methods.

Chapter 5 - **Conclusions and Future Work** - presents a review of the main findings and conclusions reached in this work. Proposals and directions for future work are also highlighted.

Chapter 2

Background and Related Work

2.1 Introduction

The aim of this chapter is to provide an overview of the most relevant concepts and techniques within the domains of microservices and anomaly detection.

Understanding these concepts is crucial due to the growing importance of microservices architecture in modern software development, thanks to its scalability, flexibility, and resilience [DGL+17]. Traditional monolithic architectures are increasingly inadequate as they struggle to meet the demands of distributed systems. In contrast, microservices offer a viable alternative by decomposing applications into smaller services, enabling independent development and deployment. This approach enhances fault tolerance and optimizes resource utilization [Che15]. However, microservices also introduce challenges, particularly in monitoring and maintaining system reliability.

To meet the objectives of this chapter, the following sections are structured as follows: Section 2.2 begins by defining and describing our domain (i.e., microservices-based architecture); Section 2.3 establishes the connection between microservices and ML-based anomaly detection techniques; Section 2.4 reviews relevant prior work in this field; Section 2.5 compiles datasets containing microservice anomalies, distinguishing between publicly available datasets and proprietary ones; and finally, Section 2.6 concludes the chapter with key takeaways.

2.2 Overview of Microservices Architecture

To better understand what microservices are and why they are a useful approach when developing software, we must start by analyzing the traditional design: a monolithic application [Ama25].

2.2.1 Monolithic Design

In this style, the applications' entire codebase is shaped as a single unit, in which every component is tightly integrated and co-dependent. There are several advantages to this design, although the adoption of microservices has grown significantly [Ama25]:

- **Simplicity and development speed**: because there are not that many moving parts, there is no need to coordinate updates across multiple dedicated services (especially for smaller projects), which leads to a faster development pace.

- **Cost-effectiveness**: there are no complexities associated with communication between services, reducing costs and overhead.
- **Deployment ease**: one codebase can be more easily put into production as opposed to multiple services, each with different demands and requirements.

Despite these advantages, monolithic architectures become increasingly fragile and less adaptable as applications and teams scale.

In spite of the positive aspects of a monolithic stack, problems usually become menacing once a project and/or team scales [Ama25]:

- **Codebase entanglement**: More and more dependencies will be formed as features get added, making the codebase extremely difficult to debug, update and test.
- **Increased learning curve for newcomers**: Onboarding new hires and, more importantly, allowing them to become productive often takes much longer because they need to learn how all the interconnected pieces of the monolith work together before they can risk modifying any single part of the application.
- **Conflicting resource requirements**: Due to the fact that the entire application has to run on the same server, specific software components will not be able to run on task-optimised hardware.

Figures 2.1 and 2.2 provides a visual representation of the differences in philosophy between a monolithic design and one based on microservices.

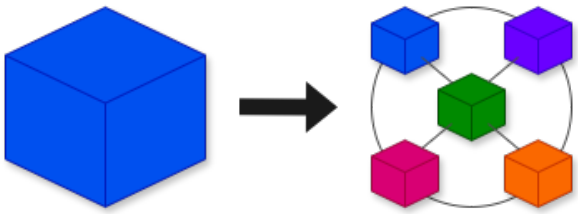

Figure 2.1: Simplified visualisation of how a monolithic design compares to another based on microservices (adapted from [Kra25]).

2.2.2 Microservices-based Design

Microservices, often hailed for their agility and scalability, represent a departure from the all-encompassing monoliths to a more modular and distributed design. As per [LHL+20], these building blocks are independent by nature, meaning that they can be developed, deployed and scaled without affecting other services. As long as the established APIs do not change, a developer can simply focus on improving that specific microservice. Furthermore, there is also a valuable degree of specialisation, allowing for fine-grade control of what tasks should be performed and how to maximise the usefulness of each microservice.

Positive aspects of this design philosophy can be summarised as follows [Ama25]:

- **Agility**: Microservices promote an organisation of small, independent teams that own their services. Teams operate within a small and clearly understood context and have the autonomy to work more independently and swiftly. The result is an acceleration of development cycles. Significant benefits are gained from the organisation's aggregated throughput.
- **Flexibility to scale**: Each service can be scaled independently to meet the demands of the applications. This allows teams to accurately scale infrastructure needs, precisely measure the cost of a resource, and maintain availability when a service experiences a demand spike.
- **Continuous deployment**: Microservices enable continuous integration and delivery, facilitating the testing of new ideas and their rollback if something doesn't function correctly. The low cost of failure allows for experimentation, simplifies code updates, and accelerates the time to market for new features.
- **Technological freedom**: These architectures do not adhere to a one-size-fits-all approach. Teams are free to choose the best tool to solve specific problems. The outcome is that teams creating microservices can opt for the most suitable tool for each task.
- **Reusable codebase**: The division of software into small, well-defined modules allows teams to use functions for various purposes. A service created for a specific function can be used as a basic component for another feature. This enables applications to be reused, as developers can create features without the need to write a new code.
- **Resilience**: The independence of each service enhances the application's ability to overcome failures. In a monolithic architecture, the failure of a single component could lead to the failure of the entire application. With microservices, applications handle the total failure of a service by degrading functionality, without disrupting the entire application.

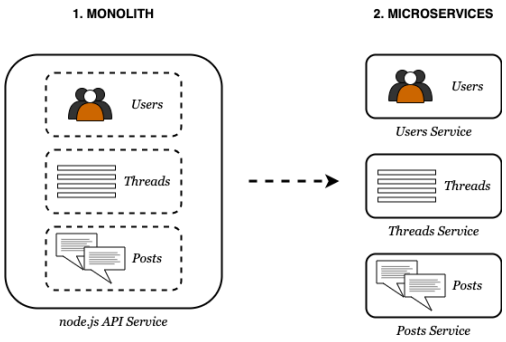

Figure 2.2: Comparison between a monolithic application layout and its microservice-based counterpart (adapted from [Ama25]).

2.3 Anomaly Detection in Microservices

In the intricate landscape of microservices architecture, the importance of performing anomaly detection cannot be overstated. Microservices, with their decentralised and modular nature, demand vigilant monitoring to ensure the reliability, performance, and security of the overall system. Anomaly detection plays a pivotal role in identifying deviations from the expected behavior of microservices, enabling timely intervention and remediation.

In general, anomaly detection is the process of finding data patterns that deviate from normal behavior, which differs from noise detection, usually referring to unwanted noise in the data [KAAD22]. This early identification of anomalies is critical for maintaining the seamless operation of microservices-based applications and preventing disruptions that could impact user experience.

2.3.1 Initial Considerations

As per [PSCvdH21], anomalies manifest in many shapes and forms, posing unique challenges for detection methods. The most prominent difficulties include:

- **Uncertainty**: Anomalies are linked to numerous uncertainties, including instances marked by unexpected behaviors, unknown data structures, and unpredictable distributions. These anomalies persist in obscurity until they manifest, representing unforeseen events like new forms of terrorist attacks, instances of fraud, and unauthorised intrusions into networks.

- **Heterogeneous anomaly classes**: There is an undeniable degree of irregularity that is inherent to anomalies, meaning that one category of anomalies can exhibit entirely distinct abnormal characteristics compared to another category. For example, within video surveillance, events deemed anomalous, including robbery, traffic accidents and burglary, display distinct visual patterns.

- **Rarity and class imbalance**: Anomalies are usually infrequent data instances, differing from normal instances that frequently constitute a substantial proportion of the data. Consequently, it is challenging, if not unfeasible, to amass a considerable quantity of labelled abnormal instances. This scarcity of extensively labelled data is prevalent in most applications. The imbalance in class distribution is also attributed to the fact that misidentifying anomalies is typically considerably more costly than misclassifying normal instances.

- **Diverse types of anomalies**: There are many possible variations of anomalies, due to factors as diverse as their nature, uniqueness and time constraints. Therefore, scholars usually agree in the following three group system:

- *Point anomalies*: Instances that are anomalous when compared with the majority of the remaining observations (e.g., irregular energy consumption patterns of a household).
- *Conditional anomalies*: Similar to the item above but with a key difference: anomalous in a specific context and perfectly normal elsewhere (e.g., rapid credit card transactions in a short period of time).
- *Group anomalies*: Collective group of anomalous instances, with each individual member not looking like an anomaly but showing signs of abnormality when taken as a whole (e.g., in a social network, one can think visualise a graph where nodes represent individual accounts and edges represent connections or interactions between those accounts (friendships, followers, etc...). Fake accounts might exhibit a coordinated behavior, such as mutual connections or interactions to appear more authentic. While these accounts might appear quite normal when examined in isolation, their associations with each other displays erratic behavior).

These general challenges highlight why anomaly detection in microservices is inherently complex and requires tailored methods.

2.3.2 Types of Anomalies in Microservices

As per [MJSW21], the following types of anomalies are commonly observed in microservice environments:

- **CPU hog**: anomaly where a service consumes excessive CPU resources. This often arises from inefficient processes or poor resource management. Mitigation typically involves process prioritisation, optimisation of resource allocation, or distributed task offloading.
- **Network congestion**: anomaly caused by excessive traffic or datagram bursts that overwhelm the system. It results in bottlenecks and degraded service quality, requiring monitoring and adaptive load balancing.
- **Memory leak**: anomaly resulting from memory not being properly released after allocation. This leads to gradual resource exhaustion and eventual performance degradation.
- **Service failure**: anomaly in which a microservice becomes unavailable due to disconnection or critical faults. These failures demand immediate remediation to restore connectivity and functionality.

This categorisation illustrates that anomalies in microservices can occur at multiple layers, from resource usage to service availability.

2.3.3 Classification of Anomalies in Microservices

In the Subsection 2.3.1, we discussed how the nature of an anomaly can be a challenge in and of itself. In addition to that intrinsic complexity, another crucial step towards eliminating irregularities is to come up with a classification scheme for the incorrect behavior that an application is experiencing.

The process of anomaly classification is a systematic approach that entails categorising identified deviations from the established norm into discernible types or classes. This method not only organises the anomalies but also establishes a structured framework, laying the groundwork for a comprehensive interpretation and strategic response. This organised classification allows for a more thorough understanding of the nature and implications of specific anomalies.

In this subsection, we will provide an outlook of anomaly classification within microservices architecture. It should be noted that these categories are not a fixed, immutable system. As more complexities are introduced in applications, so too will the range of anomalies expand. For now, though, the following descriptions should warrant an understanding of today's landscape.

As seen in Figure 2.3, anomalies in the context of microservices usually fall into one of the following four categories [CCZ19]:

- **Request exception**: Anomalies in this category usually pertain to irregularities in service requests, such as malformed requests, unauthorised access attempts, unexpected input formats and missing parameters.

- **Runtime exception**: In this category, one can find anomalies that happen during the execution of microservices, in real processing time. Common examples include unhandled errors, unexpected inputs or improper parsing leading to runtime failures, issues related to data processing, system overloads and concurrency misalignments.

- **Timeout exception**: This category identifies anomalies stemming from exceeded response times, typically seen in potential bottlenecks, inefficiencies in communication, and delayed processing pipelines.

- **Others**: Encompasses anomalies not explicitly falling into the previous categories, with real-world examples ranging from *Security Exceptions*, indicating breaches in security protocols, to *Version Exceptions* denoting discrepancies between different versions of services.

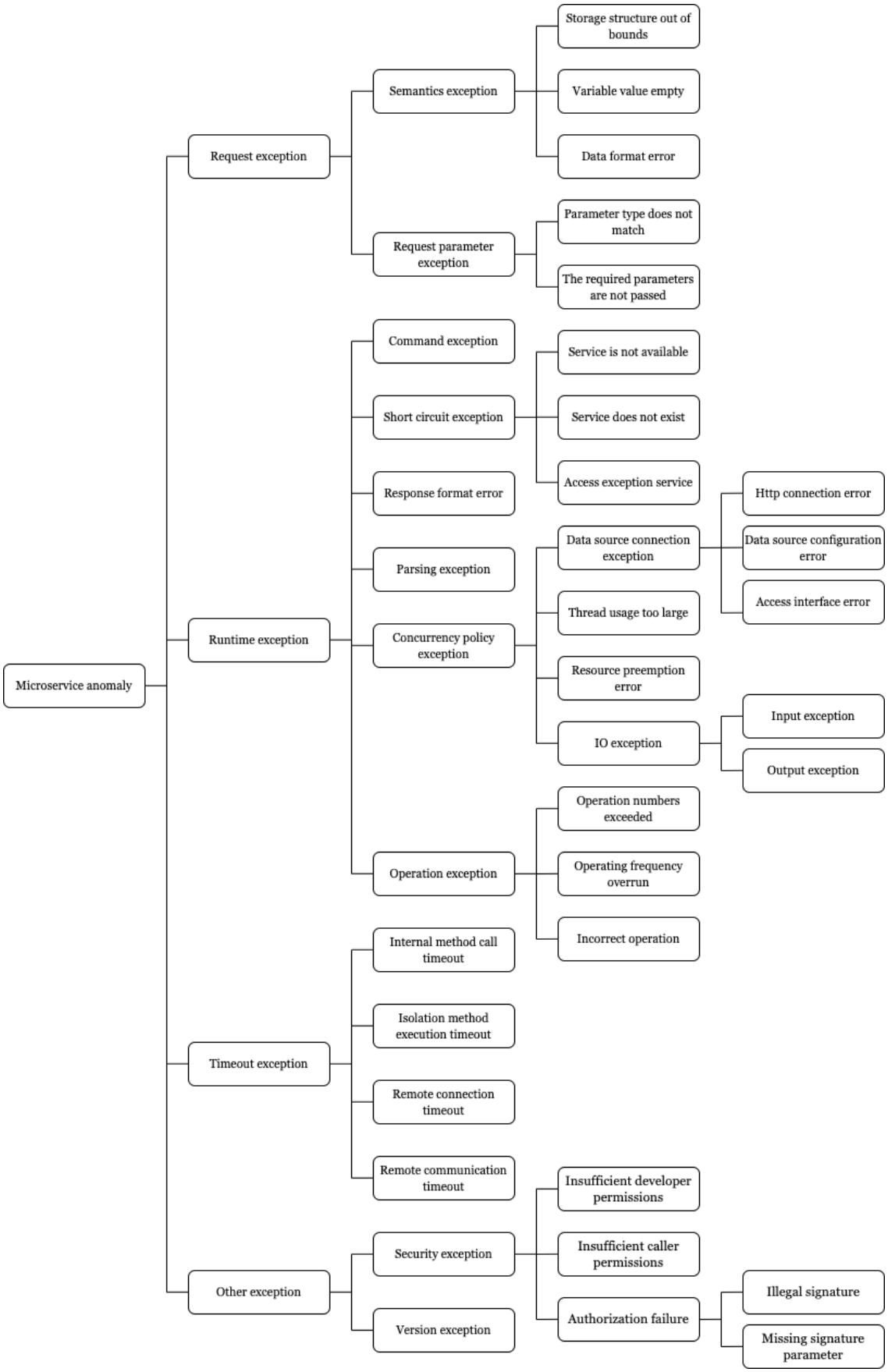

Figure 2.3: Microservice anomaly classification tree (adapted from [CCZ19]).

In summary, anomalies in microservices exhibit diverse forms, ranging from resource inefficiencies to execution failures. This conceptual overview provides the foundation for the following Section 2.4, where detection methods, both traditional and ML-based, are systematically reviewed and compared.

2.4 Related Work

This section provides an overview of anomaly detection methods within the context of microservices, categorising them into two broad approaches: those that leverage ML techniques and those that rely on alternative methodologies. To structure this review in a systematic and transparent way, a literature review methodology was applied and summarised through the PRISMA 2020 flow diagram [PRI20]. Based on the outcomes of this process, the selected studies were organised into two comparative tables, distinguishing traditional approaches from ML-based methods, in order to provide a comprehensive understanding of the diverse strategies employed to maintain the robustness and stability of microservices architecture.

2.4.1 Literature Search Strategy

To guarantee transparency and reproducibility, this study adopted a structured review process aligned with the PRISMA 2020 guidelines [PRI20]. The aim was to systematically identify and filter the most relevant contributions on anomaly detection in microservice environments.

The search strategy was built around the query *("Anomaly Detection" OR "Anomaly Classification") AND "Microservices"*. This expression was applied across the main scientific databases, namely *"IEEE Xplore"*, *"ACM Digital Library"*, *"SpringerLink"*, *"ScienceDirect"*, and *"Wiley Online Library"*, ensuring broad coverage of both conference proceedings and journal publications. The time span considered was 2015 to 2025, chosen to capture the most recent and representative advances in the field.

The process began with an initial retrieval of 2033 publications. After eliminating 747 duplicates, a total of 1286 unique records remained. A further 54 items were discarded at this stage due to reasons such as missing metadata, inaccessible abstracts, publication type inconsistencies, or because they were not written in English. This left 1232 studies eligible for title and abstract screening.

During this screening phase, 965 records were excluded for not meeting the predefined inclusion criteria. As a result, 267 publications were retained for full text assessment. From these, 12 could not be retrieved in their entirety, leaving 255 reports to be examined in depth. Following the eligibility check, 117 studies were discarded due to lack of technical detail or insufficient focus on microservices, or being purely survey articles.

Ultimately, 138 studies were included in this review. Their classification into traditional approaches and ML-based methods is presented in Subsections 2.4.2 and 2.4.3, respectively. The general workflow of the selection process is illustrated in Figure 2.4, the PRISMA Flow Diagram [PRI20].

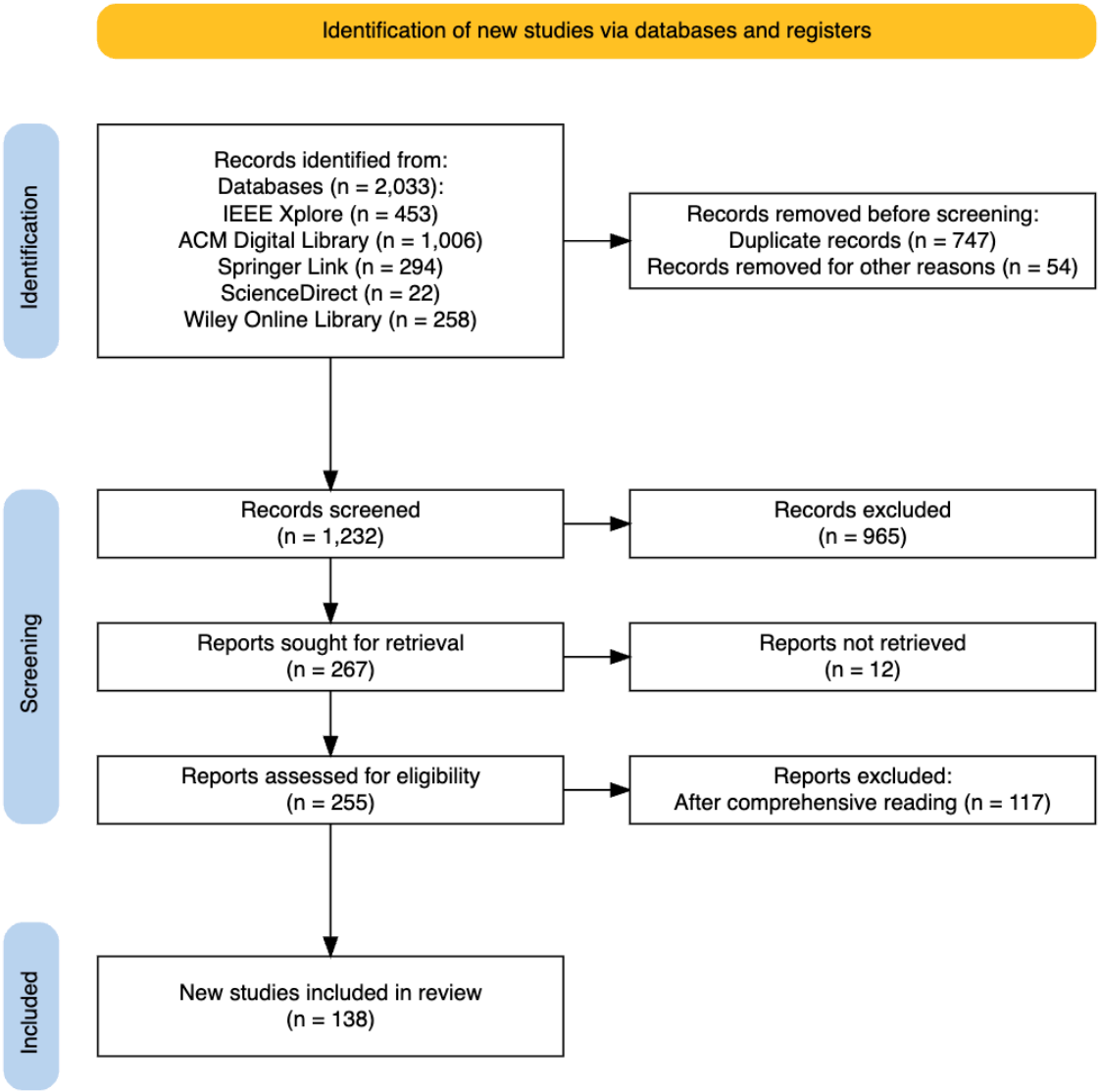

Figure 2.4: PRISMA Flow Diagram applied in this work (adapted from [PRI20]).

2.4.2 Traditional Approaches To Anomaly Detection

Traditional methods rely on statistical, rule-based, or heuristic techniques to detect anomalies in microservices. While generally less adaptive than ML-based approaches, they are lightweight and easier to interpret.

Table 2.1 summarises the traditional approaches identified in this review, including the datasets used, detection methods, experimental setup, and performance reported.

Table 2.1: List of existing works that do not rely on ML (A = Accuracy; P = Precision; R = Recall; F1 = F1-score; N/A indicates that the information was not reported or not applicable in the original study).

Works	Year	Dataset	Classification Method / Approach	Train / Val / Test Ratio (%)	Performance (A, P, R, F1)
Thalheim et al. [TRA+17]	2017	*OpenStack*; *ShareLatex*	k-Shape clustering + Root Cause Analysis (RCA)	N/A	N/A
Las-Casas et al. [LCMGF18]	2018	*Spark*; *YARN*; *HDFS*; *Ride Sharing*; *OpenTracing*	Clustering-based weighted sampling + Offline/online sliding-tree	N/A	N/A
Wang et al. [WXM+18]	2018	*IBM Bluemix*; *Pymicro*	Causal graph + Random walk	N/A	N/A
Ibrahim et al. [IBP19]	2019	*NETFLIX OSS (NETFLIX-1800-10)*; *Atsea Sample Shop App*; *JavaEE Demo*; *PHPMailer & Samba*	Automated attack graph generation	N/A	N/A
Shan et al. [SCL+19]	2019	*Industrial Dataset (JD.com production system)*	Unsupervised RCA using ε-statistics	N/A	P: ≈72–79%
Guo et al. [GPW+20]	2020	*eBay*	Graph-based Microservice Trace Analysis (GMTA)	N/A	N/A
Ma et al. [MWX+20]	2020	*IBM Cloud*; *Pymicro*	Behavior Graph + Weight learning + Random walk	N/A	N/A
Wang et al. [WRXY20]	2020	*Industrial Dataset (MeiTuan production system)*	Impact Analysis based on Pruning Tree (ImpAPTr)	N/A	A: ≈94.5%
Allen et al. [ATP+21]	2021	*TrainTicket*	Causal Impact Analysis	N/A	N/A
Das et al. [DLZ21]	2021	*eShopOnContainers*	Complex Event Processing (CEP)	N/A	N/A
Huang et al. [HZ21]	2021	*TrainTicket*; *DeathStarBench*	Trace aggregation + Subspan analysis + Automated diagnosis	N/A	N/A
Meng et al. [MJSW21]	2021	*Bench4Q*; *Social Network*	Tree Edit Distance + Principal Component Analysis (PCA)	N/A	P: 81–97%; R: 75–99% (*Bench4Q*); P: 81–97%; R: 75–99% (*Social Network*)
Neves et al. [NVP21]	2021	*Hipster Shop*; *Sock Shop*; *Cassandra*; *Spark*	Black-box monitoring via eBPF probes + graph-based traffic analysis	N/A	N/A
Pan et al. [PMJW21]	2021	*IBM Cloud*; *Pymicro*	RCA + Granger causal intervals + Dynamic causality curves	N/A	N/A
Yang et al. [YSS+21]	2021	*TrainTicket*; *Industrial Dataset (Huawei Cloud production system)*	KPIs + Dynamic Status Warping (DSW)	N/A	N/A
Yu et al. [YCC+21]	2021	*Hipster Shop*; *AIOps Challenge (Event dataset)*	Spectrum analysis + PageRank	N/A	N/A
Cao et al. [CBVR22]	2022	*Kubernetes (lLog/NetFlow)*	Probabilistic Deterministic Finite Automaton (PDFA)	N/A	N/A

Chen et al. [CLS+22]	2022	*Yahoo*; *AIOps Challenge 2018*; *Industrial Dataset (Huawei Cloud production system)*	Pattern Sketching	N/A	P: 88.2%; R: 85.6%; F1: 83.2%
Guhathakurta et al. [GAN+22]	2022	*Sock Shop*; *Real-world Dataset (HDFS)*	PCA + Autoencoder (AE)	N/A	Fewer false positives
Hrusto et al. [HER22]	2022	*Industrial Dataset (Azure production system)*	Continuous feedback loop for anomaly detection	N/A	N/A
Sussman et al. [SMA+22]	2022	*N/A (Conceptual paper, no dataset)*	Cross-layer inspection + Fault-localization algorithm	N/A	N/A
Belkhiri et al. [BBdMN23]	2023	*Telemetry Dataset (Logs, Traces, Metrics)*	Transparent trace annotation	N/A	N/A
Chakraborty et al. [CAG+23]	2023	*Alerts Dataset*; *Outage Reports Dataset*	Causal Graph + Knowledge Graph + RF	Leave-One-Out Cross-Validation	N/A
Heeb et al. [HKSG23]	2023	*Sock Shop*	Graph-based RCA + Clustering + PageRank	N/A	N/A
Jiang et al. [JPMW23]	2023	*IBM Cloud*	Granger causality + Causal unimodalization + Graph search	N/A	N/A
Shen et al. [SZX+23]	2023	*Real-world Production System (DeepFlow, multi-company)*	Network tracing + Correlation	N/A	N/A
Wang et al. [WCF+23]	2023	*Industrial Dataset (AIOps real microservice system)*; *SWaT*; *WADI*	Incremental causal graph learning	N/A	N/A
Yu et al. [YCL+23]	2023	*Online Boutique*; *TrainTicket*	Event graphs + pattern mining	N/A	N/A
Yu et al. [YPH+23]	2023	*TrainTicket*; *Real-world Traces*; *Real-world Failures*	CMDiagnostor	N/A	P: 96%; R: 97%
Zhang et al. [ZIY+23]	2023	*DeathStarBench*; *Industrial Dataset (Alibaba & Twitter production systems)*	Causal modeling (L-tree)	N/A	N/A
D'Angelo et al. [Dd24]	2024	*TrainTicket*; *E-Shopper*	Context-Free Grammar + Logistic Regression	80/-/20	N/A
Ezaz et al. [EKEJ24]	2024	*Industrial Dataset (Alibaba production system)*	Critical-path analysis	N/A	N/A
Giamattei et al. [GGM+24]	2024	*μBench*	Causal discovery + Inference + Workload generation	N/A	N/A
Gu et al. [GWY+24]	2024	*Sock Shop*; *Social Network*; *Industrial Dataset (Alibaba production system)*	Sync-Millibottleneck (SyncM) attack modeling	N/A	N/A
Huang et al. [HZC+24]	2024	*Industrial Dataset (Huawei Cloud production system)*; *Online Boutique & TrainTicket*	Adaptive biased trace sampling	N/A	N/A

Li et al. [LTW+24]	2024	*Industrial Dataset (Alibaba production system)*; *TrainTicket*	Shapley Values	N/A	N/A
Markakis et al. [MYG+24]	2024	*PostgreSQL*; *Real-world Production System (proprietary)*; *Synthetic Dataset (XYZ)*	Log-based causal inference	N/A	N/A
Pan et al. [PZJ+24]	2024	*Synthetic Datasets*; *IBM Cloud*; *PEMS-Bay*	Optimized Granger causality	N/A	N/A
Panahandeh et al. [PEJHLM24a]	2024	*AIOps Challenge 2020*	RCA + Spectrum analysis + Social Network Analysis (SNA)	N/A	N/A
Panahandeh et al. [PHLHM24]	2024	*TrainTicket*; *TeaStore*	Context Propagation Graph (CPG)	N/A	F1: ≈85% (*TeaStore*); F1: ≈86% (*TrainTicket*)
Pham et al. [PHZ24a]	2024	*Online Boutique*; *Sock Shop*; *TrainTicket*	Bayesian Online Change Point Detection (BOCPD)	N/A	Improved accuracy, fewer false positives
Pham et al. [PHZ24b]	2024	*Synthetic Datasets*; *Benchmark Microservice Systems*	RCA	N/A	N/A
Traini et al. [TLSDM24]	2024	*TrainTicket*	Visual Analytics for Microservices Performance (VAMP)	N/A	N/A
Yoon et al. [YWY+24]	2024	*Industrial Dataset (Meta production system)*	Statistical + RCA heuristics	N/A	N/A
Yu et al. [YCH+24]	2024	*Industrial Dataset (WeChat production system)*; *Online Boutique*	Change graphs + Multi-scorer ranking	N/A	N/A
Zan et al. [ZAD+24]	2024	*EasyVista*; *Industrial Dataset (IT monitoring system)*	Threshold-based Root Cause Analysis (T-RCA) + T-RCA-agent	N/A	N/A
Zhang et al. [ZDP+24]	2024	*TrainTicket*	Pattern mining + Spectrum analysis	N/A	N/A
Altenbernd et al. [AWK25]	2025	*Sock Shop*; *Online Boutique*; *TrainTicket*	At Most One Change (AMOC) Segmentation + Statistical RCA	N/A	N/A
Chen et al. [CHW+25]	2025	*Real-world Production System (Internet/Online)*	ProAlert	80/-/20	A: ≈99.71% (S1); A: ≈98.55% (S2)
Soldani et al. [SFRB25]	2025	*Online Boutique*; *Chaos Echo*	Log-based RCA	N/A	A: 100% (true RCA)

2.4.3 Machine Learning-based Approaches To Anomaly Detection

ML approaches dominate recent research on anomaly detection in microservices, leveraging supervised, unsupervised, and Deep Learning (DL) techniques. These methods enable greater adaptability to complex and dynamic environments, though often at the cost of higher computational overhead.

Table 2.2 presents the ML-based approaches identified in this review, highlighting the datasets employed, algorithms applied, experimental design, and performance reported.

Table 2.2: List of existing ML-based research papers (A = Accuracy; P = Precision; R = Recall; F1 = F1-score; N/A indicates that the information was not reported or not applicable in the original study).

Works	Year	Dataset	Classification Method / Approach	Train / Val / Test Ratio (%)	Performance (A, P, R, F1)
Chen et al. [CHC19]	2019	*Real-world RPC traces*	Diffusion Convolutional Recurrent Neural Network (DCRNN)	80/-/20	N/A
Gan et al. [GZH+19]	2019	*Social Network*; *Media Service*; *E-commerce Service*; *Banking System*; *Hotel Reservation Site*	Convolutional Neural Network (CNN) + LSTM	N/A	A: 93.45%
Las-Casas et al. [LCPAM19]	2019	*HDFS*; *DeathStarBench*; *Real-world Production System (Internet/Online)*	Neural Network (NN)	N/A	N/A
Tien et al. [THT+19]	2019	*Private Dataset*; *CERT (Public Dataset)*; *Real-world Dataset (KubAnomaly)*	NN	80/–/20	A: ≈96%; F1: ≈98.1%
Zhou et al. [ZPX+19]	2019	*Sock Shop*; *TrainTicket*	RF + KNN + MLP	5-fold Cross-Validation	P: 99.8%; R: 98.2%; F1: 99% (*Sock Shop*); P: 99.5%; R: 89.6%; F1: 94.3% (*TrainTicket*)
Li et al. [LDC+20]	2020	*Sock Shop*	LSTM	N/A	A: 91.4%
Lomio et al. [LMBM+20]	2020	*RARE*	RF	10-fold Cross-Validation	P: 95.31%; R: 89.71%; F1: 92.42%
Arya et al. [ASA+21]	2021	*TrainTicket*	Granger regression + Conditional Independence-testing + Probabilistic Graphical Event Model (PGEM)	N/A	P: 83%; F1: 88%
Belhadi et al. [BDSL21]	2021	*NETFLIX-1800*; *NETFLIX-3600*; *LAMP-1800*; *LAMP-3600*	Multi-Agent Reinforcement Learning (RL)	N/A	N/A
Chen et al. [CLS+21]	2021	*Huawei Cloud Network Service*	Graph representation learning	N/A	F1: $>$ 93%
Gan et al. [GLD+21]	2021	*Social Network*; *Media Service*; *Hotel Reservation Site*	Causal Bayesian Network (CBN) + Graph Variational Autoencoder (GVAE)	N/A	N/A
Horovitz et al. [HBW21]	2021	*Apache Hypertext Transfer Protocol (HTTP) Server Logs*	Sequence Graph Transform (SGT) + PCA + KMeans + LSTM/AE	N/A	N/A
Liu et al. [LHP+21]	2021	*Industrial Dataset (Alibaba production system)*	One-Class Support Vector Machine (OC-SVM) + RF	N/A	P: 96%; R: 87%; F1: 91%
Park et al. [PCLH21]	2021	*Online Boutique*; *Robot Shop*; *Bookinfo*	Supervised GNN	70/15/15	N/A

Zhao et al. [ZCY+21]	2021	*TrainTicket*; *E-commerce Benchmark*	Multimodal LSTM	N/A	P: 91%; R: 95%; F1: 93% (*TrainTicket*); P: 97%; R: 98%; F1: 97% (*E-commerce*)
Castro et al. [CLV22]	2022	*TeaStore*	RF + SVM + KNN + XGBoost	Leave-One-Out Cross-Validation	F1: ≈97.6% (XGBoost); F1: ≈87.7% (KNN)
Chow et al. [CDSL22]	2022	*Social Network*; *Hotel Reservation (DeathStarBench)*	Deep Neural Network (DNN) + Gated Recurrent Unit (GRU)	N/A	N/A
Cinque et al. [CDCP22]	2022	*Clearwater IMS (Telecom Testbed)*; *ATC (Testbed)*	Log2Vec + AE/Feedforward Neural Network (FNN)	N/A	P: 96.9%; R: 95.3%; F1: 96.1%
Huang et al. [HCL22]	2022	*Online Cloud Server*	SLA-Variational Autoencoder (VAE)	N/A	P: ≈88.6%; R: ≈84.8%; F1: ≈86.7%
Jacob et al. [JQYL22]	2022	*DeathStarBench*	DCRNN	85/–/15	N/A
Li et al. [LDW+22]	2022	*Sock Shop*	Cross-layer tracing + ElasticNet regression	N/A	P: ≈80–90%
Li et al. [LLY+22]	2022	*Real-world Dataset (CIRCA)*; Oracle DB	Unsupervised causal inference + Regression/Hypothesis testing	N/A	N/A
Li et al. [LZL+22]	2022	*ISP Microservice System*; *Commercial Bank System*; *Oracle DB*; *TrainTicket*	GRU + Graph Attention Network (GAT) + DNN	40/20/40	N/A
Sanvito et al. [SSS+22]	2022	*OpenStack*	Graph-based monitoring + Linear Regression (LR)	10-fold Cross-Validation	R: 98%
Somashekar et al. [SDV+22]	2022	*TrainTicket*; *DeathStarBench*	GNN	70/10/20	P: 81%; R: 74%; F1: 77%
Wang et al. [WZL+22]	2022	*Industrial Dataset (Ant Group production system)*	Spatio-Temporal Graph Neural Network (STGNN) + DNN + Deep Q-Network (DQN) + RL	N/A	N/A
Zhang et al. [ZPS+22]	2022	*TrainTicket*	Gated Graph Neural Network (GGNN) + Deep Support Vector Data Description (SVDD)	60/10/30	P: 93%; R: 97.8%; F1: 95.4%
Zhang et al. [ZPZ+22]	2022	*TrainTicket*	GNN + VAE & OC-SVM	60/-/40	P: 93.8%; R: 98.8%; F1: 96.3% (VAE); P: 94.3%; R: 85.7%; F1: 89.7% (OC-SVM)
Araujo et al. [AAV23a]	2023	*Simulated Microservice System (MariaDB in Docker, testbed)*	DT + RF + SVM	10-fold Cross-Validation	N/A
Araujo et al. [AAV23b]	2023	*Simulated Microservice System (MariaDB in Docker, testbed)*	DT + RF + SVM	10-fold Cross-Validation	A: ≈80–87%

Chakraborty et al. [CGA+23]	2023	*Synthetic & Semi-synthetic Dataset*; *Real-world Dataset*	Instance-level causal graph learning	N/A	N/A
Chen et al. [CLJ+23]	2023	*TraceRCA*; *TT-ARM (derived from TrainTicket)*	Variational Graph Autoencoder (VGAE) + LSTM-AE	N/A	P: ≈97%; R: ≈93%; F1: ≈95%
Chen et al. [CZD+23]	2023	*Industrial Dataset (Alibaba production system)*	GNN + GRU	N/A	P: 76.1%; R: 75.8%; F1: 75.9%
Ding et al. [DZW+23]	2023	*Microsoft Exchange Microservice*	RL-based graph pruning + Causal RCA	N/A	N/A
Gan et al. [GLZ+23]	2023	*Sock Shop*; *Social Network*; *4 Synthetic gRPC Microservices*	Unsupervised GNN + RCA	N/A	N/A
Hao et al. [HCCL23]	2023	*Sock Shop*; *TrainTicket*; *SWaT*; *SMD*; *SKAB*	DL	40/20/40	N/A
Harsh et al. [HZA+23]	2023	*DeathStarBench*; *Industrial Dataset (Murphy system)*	Markov Random Field (MRF)	N/A	A: 83–86%
Huang et al. [HYY+23]	2023	*MSDS (OpenStack)*; *AIOps Challenge 2022*	MSTGAD	60/10/30	P: 94.6%; R: 96.9%; F1: 95.7% (*MSDS*); P: 100%; R: 93.3%; F1: 96.5% (*AIOps Challenge 2022*)
Ji et al. [JMW23]	2023	*Real-world Dataset (MySQL, TiDB)*; *Synthetic Datasets*	Causal graph + Structural Equation Models (SEM) + RCA	N/A	N/A
Lee et al. [LYC+23a]	2023	*TrainTicket*; *Social Network*	Multimodal (logs/KPIs/traces)	N/A	N/A
Lee et al. [LYC+23b]	2023	*AIOps Challenge 2018*; *Hades*; *Yahoo!S5*	Conditional Diffusion + Isolation Forest (IF)	60/10/30	F1: ≈63.8% (*AIOps Challenge 2018*); F1: ≈82.1% (*Hades*); F1: ≈70.3% (*Yahoo!S5*)
Meng et al. [MST+23]	2023	*Bookinfo*; *Online Boutique*; *TrainTicket*	Spatiotemporal GNN + Adaptive graph learning	N/A	N/A
Rouf et al. [RAS+23]	2023	*Cloud Servers (Testbed)*	Large Language Model (LLM) + RCA	N/A	A: $\leq$ 96%
Sarda et al. [SNR+23]	2023	*Robot Shop*; *Online Boutique*	LLM + ML	N/A	A: ≈96%
Tang et al. [TGYC23]	2023	*Hipster Shop*; *AIOps Challenge 2022*	Graph Convolutional Network (GCN)	70/-/30	P: 93.76%; R: 67.46%; F1: 78.47%
Wang et al. [WCN+23]	2023	*AIOps (Real Microservice System)*; *WADI*; *SWaT*	GNN + Extreme Value Theory (EVT)	N/A	N/A
Wang et al. [WZD+23]	2023	*Pymicro*; *SynExchange*; *RealExchange*	Hierarchical Reinforcement Learning with Human Feedback (HRLHF) + Temporal causal RCA	N/A	N/A

Xie et al. [XPL+23]	2023	*Industrial Dataset (eBay production system)*; *Testbed Dataset (Fault injection)*	Group-wise VAE + Tree-LSTM	N/A	F1: $\leq$ 86.9%
Xie et al. [XXC+23]	2023	*5 Microservice Trace Datasets (International E-commerce Company)*	Dual-variable Graph VAE	N/A	F1: 91–98%
Zeng et al. [ZZX+23]	2023	*Exchange*; *TrainTicket*	XGBoost + Feedback learning with anomaly features	N/A	P: 90.68%; F1: 59.36% (*Exchange*); P: 92.98%; F1: 96.36% (*TrainTicket*)
Zhao et al. [ZMZ+23]	2023	*GAIA*; *Large-scale Microservice (Commercial Bank)*	Graph Transformer Network (GTN) + GAT	60/-/40	P: 79.5%; R: 94.5%; F1: 85.7% (*GAIA*); P: 86.3%; R: 99.1%; F1: 92.2% (*Commercial bank*)
Aktaş et al. [AK24]	2024	*Industrial Dataset (CPaaS telecommunication platform)*	LogReg + SVM + DT + RF + MLP + IF	5-fold Cross-Validation	A: $\leq$ 99% (RF/MLP); A: ≈89–95% (DT/RF); A: ≈93% (IF)
Baluta et al. [BRM+24]	2024	*Acme-Air*; *Online Boutique*	Automated Machine Learning (AutoML) + (Gradient Boosting Machine (GBM)/XGBoost)	80/–/20 (5-fold Cross-Validation for AutoML)	F1: 82–99%
Chatterjee et al. [CAM24]	2024	*Real-world Alert Dataset*	Active learning + XGBoost	N/A	F1: ≈35%
Chen et al. [CCW+24]	2024	*Real-world Experiments (sFlow-metrics, Components-metrics, Scaling-metrics)*	Spatio-temporal cell (GCN + LSTM) + VAE	N/A	P: ≈79%; R: ≈67%; F1: ≈72%
Denaro et al. [DEMH+24]	2024	*Alemira*; *TrainTicket*	Deep Autoencoder (DAE)	N/A	N/A
Goel et al. [GHS+24]	2024	*Industrial Dataset (Microsoft production system)*	LLM	70/10/20	A: ≈83%
Han et al. [HDH+24]	2024	*Public Datasets (Experiment results)*	LLM + LightGBM/XGBoost	70/10/20	N/A
Hrusto et al. [HRO24]	2024	*Industrial Dataset (Microsoft Azure PIM system)*	LSTM-AE + Transformer-based AE	90/-/10	F1: 64%
Li et al. [LZK24]	2024	*TrainTicket*	DeepKL	60/–/40	P: 97.4%; R: 81.5%; F1: 88.7%
Olaya et al. [POVTAC24]	2024	*BoT-IoT*; *UNSW-NB15*	DT + RF + SVM + LR	70/–/30 + 3-fold Cross-Validation	A: 99.62%; P: 99.25%; R: 96.96%; F1: 98.08% (DT); A: 99.88%; P: 99.91%; R: 99.80%; F1: 99.85% (RF)

Panahandeh et al. [PEJHLM24b]	2024	*Social Network*; *Hotel Reservation*; *Media Service*; *Ticket Booking*	Unsupervised GNN + Critical Path Analysis	60/20/20	A: 86.4–96%
Ren et al. [RYY+24]	2024	*TrainTicket*; *Industrial Dataset (Company ALC production system)*	Rule learning	5-fold Cross-Validation	N/A
Rouf et al. [RRL+24]	2024	*QoTD*; *MicroSS*; *TrainTicket*	Hybrid GNN + GRU	N/A	N/A
Roy et al. [RZB+24]	2024	*Industrial Dataset (Microsoft production system)*	LLM-based ReAct agent (GPT-4)	N/A	N/A
Somashekar et al. [SDA+24]	2024	*Social Network (DeathStarBench)*	GAT	70/10/20	P: 90–94%; R: 89–91%; F1: 91–92%
Sun et al. [SSM+24]	2024	*Simulated E-commerce Microservice System*; *Bank Management System*	Unsupervised Self-Supervised Learning (SSL)	N/A	N/A
Tao et al. [TZJ+24]	2024	*Simulated E-commerce Application System*; *GAIA*; *MicroServo*	Encoders + Multimodal Adaptive Optimization (MAO)	N/A	F1: 95.08% (*Simulated e-commerce*); F1: 91.36% (*GAIA*); F1: 82.60% (*MicroServo*)
Wang et al. [WHL+24]	2024	*Industrial Dataset (Tencent Online production system)*	Bayesian LSTM + Active Learning	(approx. 70/-/30, not explicit)	P: ≈66.8%; R: ≈76.5%; F1: ≈71.3%
Wang et al. [WZF+24]	2024	*Online Boutique*; *TrainTicket*	Multi-modal RCA	60/-/40	P: 87.5%; R: 91.3%; F1: 89.3%
Xie et al. [XZG+24]	2024	*eBay's Microservices System*; *Online Boutique*	Latent causal inference (variational)	N/A	N/A
Yao et al. [YPC+24]	2024	*Global E-commerce System (Service, Business)*	Event-causal graph + Graph ranking	(approx. 50/-/50)	N/A
Zhang et al. [ZGB+24]	2024	*CompanyX Incident Dataset*	GPT-4 In-Context Learning (ICL)	N/A	N/A
Zhang et al. [ZZH+24]	2024	*TrainTicket*	Knowledge Graph Embedding (KGE) + KNN + Entity Fault Association (EFA)-Position Value (PV)	5-fold Cross-Validation	A: 81.9%
Zheng et al. [ZCHC24]	2024	*Product Review*; *Online Boutique*; *TrainTicket*	Language Model (LM) + Contrastive multimodal causal graph	N/A	N/A
Zhu et al. [ZLTH24]	2024	*TrainTicket*; *Social Network*	VAE	70/-/30	N/A
Ding et al. [DMSB25]	2025	*AIOps Challenge 2022*; *GAIA*; *Social Network*; *MMS*	Wavelet-based multimodal fusion + Spatiotemporal GNN with Denoising Diffusion Probabilistic Model (DDPM)	N/A	P: ≈95–96%; R: ≈94–97%; F1: ≈93-97%
Fernando et al. [FRA+25]	2025	*SMD*; *iAnomaly*	Intra-Cluster Parameter Transfer Learning (ICPTL) + Mean Pairwise Distance (MPD) + Gaussian Mixture (GM) with AE/IF/LSTM-AE	N/A	F1: ≈77% (*SMD*); F1: ≤ 96% (*iAnomaly*)

Gu et al. [GLC+25]	2025	*SMD*; *Industrial Dataset (Huawei Cloud production system)*	GNN + GRU + Latent-Conditioned Variational Autoencoder (LC-VAE)	50/-/50	N/A
Pei et al. [PWL+25]	2025	*Online Boutique*	LLM Multi-Agent (Standard Operating Procedure (SOP))	N/A	N/A
Sun et al. [SLS+25]	2025	*Simulated E-commerce Microservice System*; *Bank Management System*	Graph Autoencoder (GAE)	70/10/20	P: 92.6%; R: 91.1%; F1: 91.8%
Wang et al. [WMD+25]	2025	*TrainTicket*; *Online Boutique*	AttenAE + Transformer Encoder with Model-Agnostic Meta-Learning (TEMAML)	N/A	A: 93.26% (*TrainTicket*); A: 85.20% (*Online Boutique*)
Wang et al. [WZC25]	2025	*GAIA*; *MSDS*	GAT + Heterogeneous Graph Transformer Network (HGTN) + Contrastive learning	N/A	P: ≈95%; R: ≈99%; F1: > 97%
Wang et al. [WZCC25]	2025	*GAIA*; *Hades*	VGAE + AE + KMeans	80/–/20	P: 96%; R: 99%; F1: 97% (*GAIA*); P: 87%; R: 94%; F1: 90% (*Hades*)
Wei et al. [WSY+25]	2025	*TrainTicket*	Word2Vec + BiLSTM + Attention + Clustering	70/10/20	A: ≈95.3%; P: ≈94.8%; R: ≈95.7%; F1: ≈95.2%
Xie et al. [XWH+25]	2025	*GAIA*; *AIOps Challenge 2022*; *Sock Shop*; *Hotel Reservation*	GraphSAGE GNN + Task-oriented & Contrastive learning + Graph augmentation	N/A	N/A
Xv et al. [XGL+25]	2025	*ISP Microservice System*; *Commercial Bank (SOA System, Oracle DB)*; *TrainTicket*	GAT + GRU + Kullback–Leibler (KL)	40/20/40	N/A
Yang et al. [YHD+25]	2025	*Public in-lab Multi-modal Dataset*; *Simulated Multi-modal Benchmark*; *Industrial Dataset (Alibaba production system) & TrainTicket*	Fine-grained multi-modal Association and Frequency Domain Analysis for Anomaly Detection (FFAD)	70/10/20	P: 90–92%; R: ≤ 100%; F1: ≈93.6%

2.4.4 Comparative Analysis

The comparative analysis between traditional and ML-based anomaly detection approaches highlights clear differences in methodology, dataset usage, and overall performance (Subsections 2.4.2 and 2.4.3).

Traditional methods are predominantly centered on statistical inference, causal analysis, clustering, and rule based techniques. These approaches often rely on fault injection testbeds such as *TrainTicket* and *Sock Shop*, as well as industrial production datasets from providers such as *Alibaba*, *Huawei*, or *Microsoft*. While effective in capturing anomalies

in specific scenarios, they usually present limitations in terms of scalability, adaptability to unseen anomalies, and the ability to generalize across heterogeneous microservice environments.

In contrast, ML-based approaches leverage supervised, unsupervised, and semi supervised techniques, ranging from classical models such as RF, SVM, and KNN, to advanced architectures including AE, VAE, Recurrent Neural Network (RNN), LSTM, GRU, GNN, GAT, and GCN. These methods demonstrate superior capacity to learn complex temporal and structural dependencies in microservices, enabling higher precision, recall, and F1-scores across diverse datasets. Furthermore, ML-based methods increasingly adopt multimodal anomaly detection strategies, integrating metrics, logs, and traces. They also make use of both public benchmarks such as *TrainTicket*, *Sock Shop*, *Online Boutique*, and *DeathStarBench*, and proprietary datasets collected from large scale production systems.

Overall, the transition from traditional to ML-based methods illustrates a paradigm shift towards data driven anomaly detection. Traditional approaches provide interpretable and computationally efficient baselines, while ML-based methods offer higher detection accuracy, robustness to diverse fault types, and scalability in modern microservice environments. This evolution also suggests a trend where hybrid approaches, combining causal inference with ML, are explored to balance interpretability with predictive power.

In conclusion, this comparison between traditional and ML-based approaches highlights the strengths and limitations of each paradigm, directly motivating the methodological choices made in Chapter 3, where the selected ML and ensemble models are implemented and evaluated.

2.5 Available Datasets with Anomalies in Microservices

Several datasets have been proposed in recent years to support research on anomaly detection in microservice based systems. These datasets differ in scope, size, and availability, ranging from controlled benchmarks and simulated testbeds to proprietary datasets collected from large scale production environments.

Publicly available benchmarks such as *TrainTicket*, *Sock Shop*, *Hipster Shop*, *Online Boutique*, *DeathStarBench*, *TeaStore*, *Bookinfo* and *µBench* are widely adopted in the literature as representative microservice environments, frequently combined with fault injection to generate anomalous traces and logs. Similarly, multimodal datasets such as *SMD*, *GAIA*, *Hades*, *SWaT*, *WADI* and *Yahoo!S5* provide a combination of metrics, logs, and traces, enabling evaluation of anomaly detection approaches under diverse conditions. Competitions such as the *AIOps Challenges (2018, 2020, 2022)* have also provided public datasets that serve as common benchmarks.

On the other hand, several studies rely on proprietary datasets collected from real-world production systems, including *Alibaba*, *Huawei Cloud*, *Microsoft*, *Meta*, *JD.com*, *Tencent*, *MeiTuan*, and *eBay*, among others. These datasets are essential for validating anomaly detection methods in realistic industrial scenarios, but are not publicly available due to confidentiality restrictions.

Although the Table 2.3 presents an comprehensive overview of datasets identified in the literature, it is worth highlighting that the benchmarks *TrainTicket*, *Sock Shop*, *Online Boutique*, *DeathStarBench*, *SMD*, *SWaT*, *WADI* and the datasets from the *AIOps Challenges* are the most recurrent and widely used in microservice anomaly detection research.

Table 2.3: List of Publicly Available and Proprietary datasets.

Dataset Name	**Year**	**Availability**
AIOps Challenge 2018 [LYC+23b]	2018	Publicly Available
AIOps Challenge 2020 [PEJHLM24a]	2020	Publicly Available
AIOps Challenge 2022 [TGYC23], [HYY+23], [DMSB25], [XWH+25]	2022	Publicly Available
Alibaba [ZIY+23], [EKEJ24], [GWY+24], [LTW+24], [LHP+21], [CZD+23], [YHD+25]	2019–2024	Proprietary
Ant Group [WZL+22]	2024	Proprietary
ATC [CDCP22]	2019	Proprietary
Azure [HER22]	2020	Proprietary
Bookinfo [PCLH21], [MST+23]	2016	Publicly Available
Chaos Echo [SFRB25]	2024	Publicly Available
CIRCA Dataset [LLY+22]	2021	Proprietary
Clearwater IMS [CDCP22]	2017	Publicly Available
CompanyX [ZGB+24]	2021	Proprietary
CPaaS [AK24]	2023	Proprietary
DeathStarBench [HZ21], [ZIY+23], [LCPAM19], [CDSL22], [JQYL22], [SDV+22], [HZA+23], [SDA+24]	2019	Publicly Available
DeepFlow [SZX+23]	2023	Proprietary
eBay [GPW+20], [XPL+23], [XZG+24]	2020	Proprietary
GAIA [ZMZ+23], [TZJ+24], [DMSB25], [WZC25], [WZCC25], [XWH+25]	2023	Publicly Available
Hades Dataset [LYC+23b], [WZCC25]	2023	Publicly Available
Hipster Shop [NVP21], [YCC+21], [TGYC23]	2018	Publicly Available

Huawei Cloud [CLS+22], [YSS+21], [HZC+24], [CLS+21], [GLC+25]	2019–2024	Proprietary
JD.com [SCL+19]	2019	Proprietary
KubAnomaly [THT+19]	2019	Publicly Available
LAMP-1800 and *LAMP-3600* [BDSL21]	2020	Publicly Available
MariaDB (Docker) [AAV23a], [AAV23b]	2023	Publicly Available
MeiTuan [WRXY20]	2021	Proprietary
Meta [YWY+24]	2024	Proprietary
Microsoft [DZW+23], [GHS+24], [HRO24], [RZB+24]	2020–2024	Proprietary
Murphy [HZA+23]	2023	Proprietary
μBench [GGM+24]	2024	Publicly Available
NETFLIX-1800 and *NETFLIX-3600* [BDSL21]	2020	Publicly Available
NETFLIX-1800-10 [IBP19]	2020	Publicly Available
Online Boutique [YCL+23], [HZC+24], [PHZ24a], [YCH+24], [AWK25], [SFRB25], [PCLH21], [SNR+23], [BRM+24], [MST+23], [WZF+24], [XZG+24], [ZCHC24], [PWL+25], [WMD+25]	2018	Publicly Available
SMD [HCCL23], [FRA+25], [GLC+25]	2019	Publicly Available
Sock Shop [NVP21], [GAN+22], [HKSG23], [GWY+24], [PHZ24a], [AWK25], [ZPX+19], [LDC+20], [LDW+22], [GLZ+23], [HCCL23], [XWH+25]	2016	Publicly Available
SWaT [WCF+23], [WCN+23], [HCCL23]	2016	Publicly Available
TeaStore [PHLHM24], [CLV22]	2017	Publicly Available
Tencent Online [WHL+24]	2023	Proprietary
TraceRCA [LCJ+21], [CLJ+23]	2021	Publicly Available
TrainTicket [ATP+21], [HZ21], [YSS+21], [YCL+23], [YPH+23], [Dd24], [HZC+24], [LTW+24], [PHLHM24], [PHZ24a], [TLSDM24], [ZDP+24], [AWK25], [ZPX+19], [ASA+21], [ZCY+21], [LZL+22], [SDV+22], [ZPS+22], [ZPZ+22], [LYC+23a], [ZZX+23], [DEMH+24], [HCCL23], [LZK24], [MST+23], [RYY+24], [RRL+24], [WZF+24], [ZZH+24], [ZCHC24], [ZLTH24], [WMD+25], [WSY+25], [XGL+25], [YHD+25]	2017	Publicly Available
TT-ARM [CLJ+23]	2023	Publicly Available
WADI [WCF+23], [WCN+23]	2017	Publicly Available
Yahoo [CLS+22]	2015	Publicly Available
Yahoo!S5 [LYC+23b]	2015	Publicly Available

Among these datasets, this dissertation adopts *TraceRCA*, a dataset that integrates traces, invocation data, and anomaly labels, enabling both ML and ensemble-based approaches to be systematically evaluated in Chapter 3.

2.6 Conclusion

In this chapter, we reviewed microservices, their contrast with monolithic structures, and the challenges of anomaly detection. We also synthesised traditional methods, rooted in non-trainable statistical analysis, and ML-based approaches. This synthesis was guided by a systematic literature review, supported by the PRISMA Flow Diagram, ensuring transparency and reproducibility.

As we move on to Chapter 3, these notions will prove essential to better understand microservices and how users and developers can safeguard the performance and stability of distributed systems.

Chapter 3

Classification Approach

3.1 Introduction

This chapter presents the practical component of the dissertation, which focuses on the experimental design and implementation of anomaly detection methods in microservice based systems. The experiments are conducted using the *TraceRCA* dataset, which provides labeled traces and anomaly information suitable for evaluating the effectiveness of different ML approaches.

The chapter is organized as follows: Section 3.2 characterizes the *TraceRCA* dataset and its features, providing the foundation for the subsequent stages; Section 3.3 details the preprocessing and data preparation steps, including normalization and data splitting, which are essential to ensure consistency in the experiments; Section 3.4 introduces the ML algorithms considered as non-ensemble baselines; Section 3.5 presents ensemble methods, which constitute the core focus of this work; Section 3.6 describes the hyperparameter tuning process, highlighting how different configurations are explored to optimize model performance; Section 3.7 defines the evaluation strategy, outlining the metrics and validation procedures adopted to compare the approaches; and finally, Section 3.8 concludes the chapter with a summary of the experimental setup and a transition towards the discussion of results.

3.2 Characterization of the Dataset and Features

Microservices architecture is a way of structuring an application into a set of microservices, each with a specific functionality, that, despite communicating through APIs, are independent of each other.

As per [LCJ+21] [CLJ+23], *TraceRCA*[1] is a software tool designed to perform RCA in complex systems like microservices architecture. RCA is a technique that identifies the origin of problems or failures in processes or systems, enabling the effective resolution of issues. For instance, a failure might be an increase in the response time of one microservice, which could consequently increase the response time of other microservices interacting with it.

TraceRCA is based on the following premise: a microservice with more abnormal traces and fewer normal traces passing through it is more likely to be the root cause of the prob-

[1] github.com/NetManAIOps/TraceRCA

lem. Trace normality is inferred by *TraceRCA* from the normality of the invocations that constitute them. Therefore, *TraceRCA* detects abnormal invocations rather than directly detecting abnormal traces, as traces have variable lengths, which can impact the efficiency or accuracy of *TraceRCA*. If at least one invocation is determined to be abnormal, the trace is considered abnormal, allowing the system to take as many abnormal traces as possible into consideration when identifying the root cause, making *TraceRCA* both robust and efficient [LCJ+21].

To detect root causes, *TraceRCA* follows three stages [LCJ+21], as shown in Figure 3.1. In the initial stage, when a failure occurs, it is necessary to detect abnormal traces, which *TraceRCA* accomplishes by analyzing the normality of trace invocations using an anomaly detection method that combines Data Analysis (DA) techniques and ML models like CNNs or LSTMs. Anomalies may be minor and temporary or indicate more complex issues.

In the second stage, *TraceRCA* identifies the microservices causing problems. After detecting anomalies in the traces, suspicious microservices are mined as a set rather than individually, because sometimes only those traces that pass through a specific set of microservices are affected by a failure.

In the final stage, *TraceRCA* calculates a suspicious score for each microservice suspected of causing the problem, based on the number of traces containing abnormal incoming and outgoing invocations. This score is derived from a combination of the Jaccard Index (JI) score of each suspicious set and an internal suspicious score for each microservice within the set. The internal score is calculated by the difference between the number of traces containing abnormal incoming/outgoing invocations in the microservice among all traces containing the suspicious set. If a trace containing the suspicious set shows both abnormal incoming and outgoing invocations on a microservice, it is likely that this microservice has been affected by other microservices, indicating it is not a causal anomaly and that the anomaly was merely propagated through it. However, if the trace shows only incoming or only outgoing abnormal invocations on a microservice, it is likely that this microservice is the root cause of that trace. Based on these scores, *TraceRCA* ranks microservices according to the severity of the issues, allowing failures to be resolved more quickly [LCJ+21].

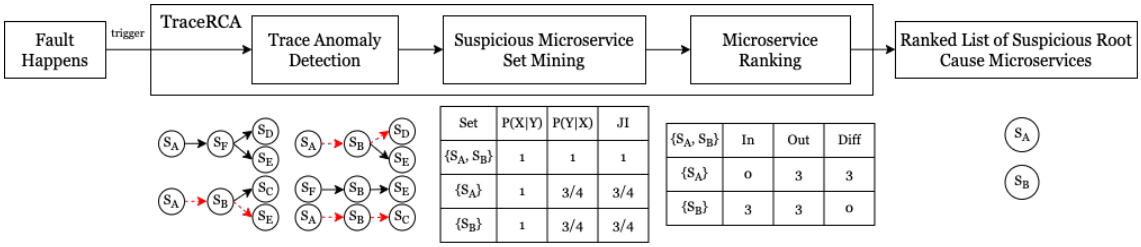

Figure 3.1: Overview of *TraceRCA* method (adapted from [LCJ+21]).

TraceRCA offers the following advantages [LCJ+21]:

- Fast anomaly detection, even when compared to other DL approaches like Trace Anomaly [XPL+23], which contains multiple parameters and is dependent on CPU capacity.
- Reduced system downtime, as early and accurate anomaly detection allows for quicker problem resolution.
- Capability to operate in complex environments with high data volumes.

Despite its significant advantages, there are some factors that may limit its use, such as the complexity and high cost of implementation, as well as the dependency on ML, which, despite being an advantage, can make the anomaly detection process somewhat time-consuming [LCJ+21].

3.3 Preprocessing and Data Preparation

Preprocessing was a crucial step to ensure consistency between service invocations and the anomaly ground-truth in the *TraceRCA* dataset. Since the `B.zip` package already provides both the invocation logs (`rca_*.csv`) and the fault metadata (`ret_info.csv`), this pipeline was designed to directly integrate these sources and generate a labeled dataset suitable for training. The process consisted of three main stages: data extraction and normalization, anomaly label assignment, and final dataset construction. The preprocessing procedure was structured into three main steps, which are described in detail in Subsections 3.3.1, 3.3.2 and 3.3.3 and illustrated in the pipeline shown in Figure 3.2.

3.3.1 Data Extraction and Normalization

The `rca_*.csv` files contain millions of service invocations with attributes such as timestamps, latency, source, and target services. These files were read in streaming mode (chunks) to handle their large size. Column names were standardized (e.g., unifying variations of `succ` or `status_code`), service identifiers were simplified (removing prefixes and normalizing case), and timestamps were converted into epoch milliseconds. In parallel, the `ret_info.csv` file was parsed to extract failure windows per service, stored in `ret_info.csv` (interim) and `ret_services_unique.csv`.

3.3.2 Anomaly Label Assignment

To integrate ground-truth labels, each service invocation was matched against the fault windows derived from `ret_info.csv`. An invocation was marked as anomalous (`label = 1`) if its source service matched the ground-truth service and its timestamp fell within the defined failure interval. Otherwise, it was labeled as normal (`label = 0`). This step was performed incrementally over each daily `rca_*.csv`, producing the consolidated labeled dataset. The final file contained approximately 85.6 million records, with around 0.44% anomalous cases.

3.3.3 Final Dataset Construction

The last step validated the integrity of the labeled dataset. Checks were performed to ensure correct timestamp ranges, the presence of all expected columns, and consistency of label distribution. The final output was stored as `data/processed/invocations\_labeled.csv`, which served as the baseline for anomaly labeling.

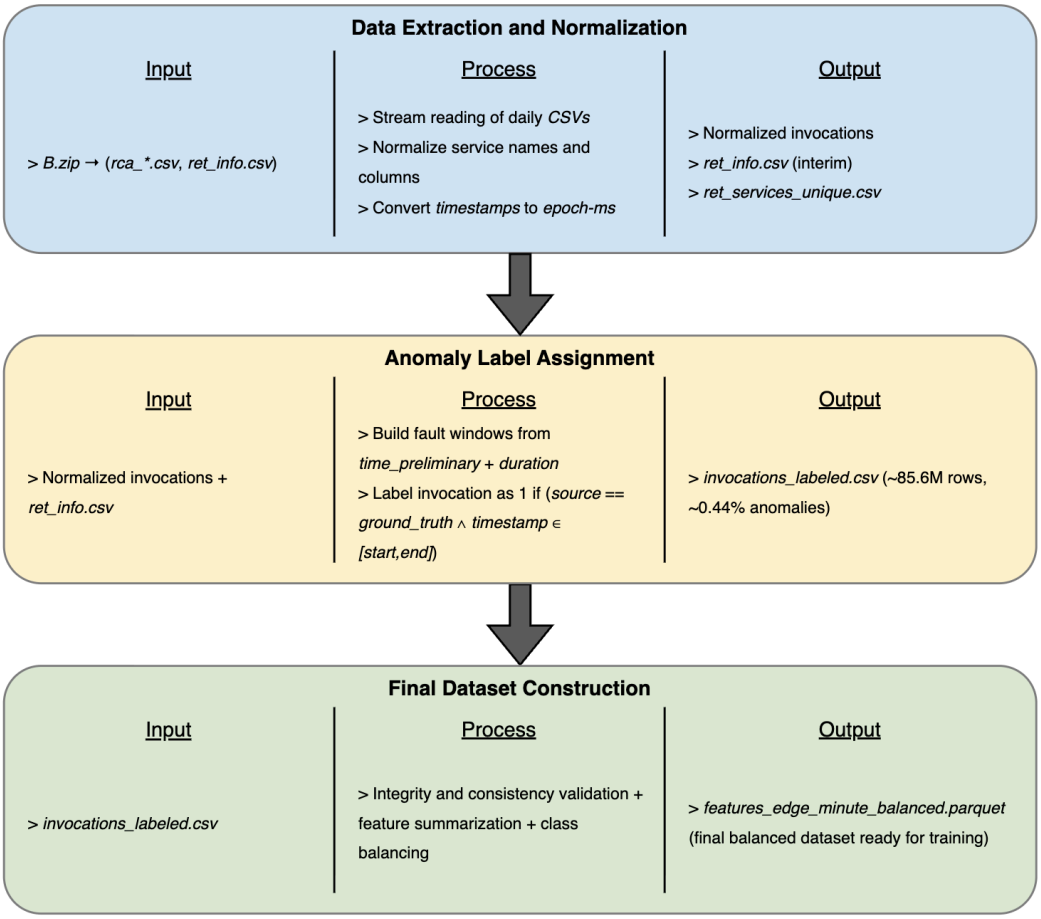

Figure 3.2: *TraceRCA* dataset preprocessing phase pipeline.

To ensure class balance and reduce the high volume of invocations (≈85M rows), a feature summarization step was applied at the edge-minute level. This aggregation generated a smaller dataset (`features_edge_minute_balanced.parquet`) with a consistent distribution of normal and anomalous samples, ready for training the ML algorithms presented in Section 3.4.

3.4 Baseline Machine Learning Algorithms

To establish a reference for anomaly detection in microservices, a set of classical ML algorithms was selected and evaluated on the processed *TraceRCA* dataset. These algorithms represent well-known approaches with varying levels of complexity, interpretability, and computational efficiency. The goal was to provide a solid baseline for performance comparison with ensemble methods, which will be introduced in Section 3.5.

The models included:

- **LogReg:** a linear classifier widely used for binary classification, valued for its interpretability and efficiency.
- **SVM:** a robust method capable of handling non-linear decision boundaries through the use of kernel functions.
- **DT:** a non-linear model that recursively splits the feature space, producing interpretable decision rules.
- **MLP:** a feed-forward NN that can capture complex non-linear relationships.
- **KNN:** an instance-based classifier that predicts based on the majority class among the closest samples in feature space.
- **GNB:** a probabilistic model that assumes conditional independence between features and is known for its simplicity and speed.

These algorithms provide diverse perspectives, from probabilistic to geometric and neural approaches, enabling a broad assessment of baseline classification performance.

3.4.1 Experimental Setup

The experiments were conducted on the feature-engineered dataset (`features\_edge\_minute\_balanced.parquet`), which contained approximately 1166 samples evenly distributed between normal and anomalous instances. This balanced representation was obtained through feature summarization and undersampling, as explained in Subsection 3.3.3. The dataset was subsequently divided into 70% for training, 20% for validation, and 10% for testing, ensuring a consistent distribution of normal and anomalous samples across all subsets.

Each algorithm was implemented using the scikit-learn library [sci25a] and evaluated using 10-fold stratified cross-validation to ensure robustness of results. Performance was measured through the following metrics: precision, recall, accuracy (train/test), F1-score, and ROC-AUC.

In addition to classification performance, the experiments also included monitoring of execution time, peak RAM usage, and average CPU utilization during training. This dual focus on predictive accuracy and resource efficiency is essential to assess the practical applicability of ML algorithms in large-scale microservice environments.

All models were trained with their default hyperparameters, except for the MLP, which was constrained to a maximum of 200 iterations. While this occasionally led to non-convergence warnings, the results remained consistent and sufficiently reliable for base-

line comparison. This configuration was kept consistent across all baseline experiments to ensure fair comparison.

3.4.2 Results Analysis

The performance results of the baseline algorithms are summarized in Table 3.1, which presents the 10-fold cross-validation metrics alongside the corresponding test accuracy. Each value represents the `mean` ± `std` across folds, with the best-performing results per metric highlighted in bold.

Table 3.1: Performance of baseline algorithms (10-fold cross-validation). Best values are highlighted in bold.

Model	Accuracy (train/test)	Precision	Recall	F1-score	ROC-AUC
LogReg	0.691 / 0.726	0.686 ± 0.039	0.708 ± 0.067	0.696 ± 0.043	0.781 ± 0.055
SVM	0.777 / 0.769	0.774 ± 0.066	0.797 ± 0.084	0.781 ± 0.048	0.851 ± 0.041
DT	0.869 / **0.897**	**0.855 ± 0.062**	0.897 ± 0.070	**0.872 ± 0.043**	0.869 ± 0.045
MLP	0.791 / 0.803	0.776 ± 0.054	0.828 ± 0.049	0.799 ± 0.034	**0.870 ± 0.041**
KNN	0.769 / 0.812	0.745 ± 0.066	0.838 ± 0.058	0.785 ± 0.040	0.852 ± 0.051
GNB	0.615 / 0.607	0.572 ± 0.026	**0.916 ± 0.062**	0.704 ± 0.032	0.706 ± 0.062

Figure 3.3 provides a comparative visualization of the five evaluation metrics, accuracy, precision, recall, F1-score, and ROC-AUC, across all baseline models. This representation facilitates a more intuitive comparison of performance trends between algorithms.

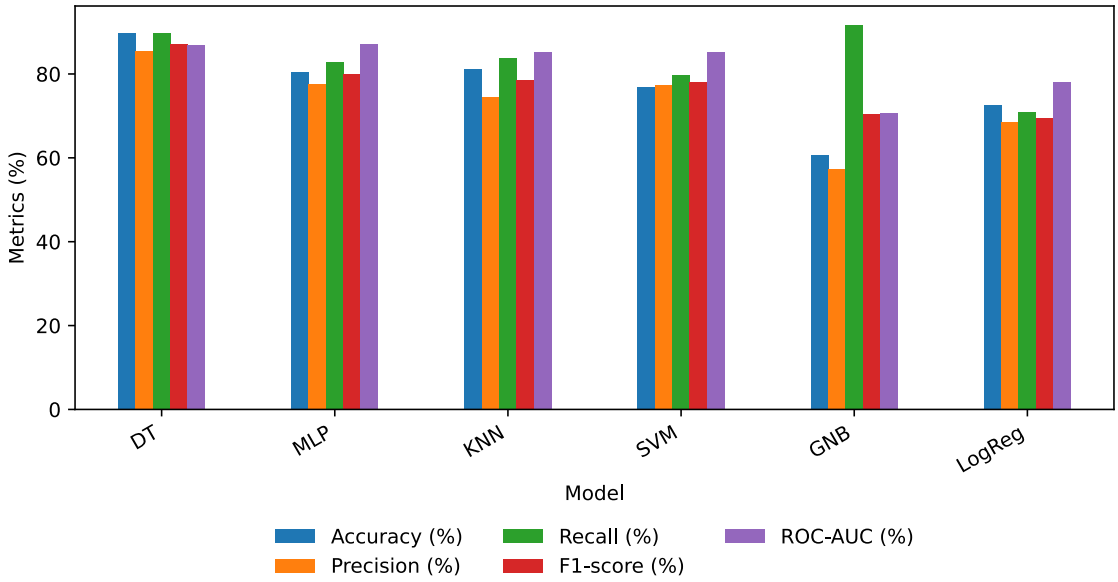

Figure 3.3: Comparative performance metrics (Accuracy, Precision, Recall, F1-score, and ROC-AUC) of baseline algorithms.

Table 3.2 reports the computational resource usage of each baseline model, including execution time, peak RAM, and CPU utilization during training. These metrics highlight not only the classification efficiency but also the scalability of each algorithm when applied to microservice environments.

Table 3.2: Resource usage of baseline algorithms during training (10-fold mean ± std; lower is better).

Model	Execution Time (s)	Peak RAM (MB)	CPU Utilization (%)
LogReg	0.00 ± 0.00	177.83 ± 0.08	95.13 ± 15.40
SVM	0.05 ± 0.00	189.38 ± 3.26	99.92 ± 0.26
DT	0.00 ± 0.00	192.72 ± 0.00	96.03 ± 12.55
MLP	0.02 ± 0.01	192.87 ± 0.02	99.85 ± 0.48
KNN	0.00 ± 0.00	193.06 ± 0.00	92.28 ± 24.43
GNB	0.00 ± 0.00	193.06 ± 0.00	100.00 ± 0.00

The baseline models exhibited distinct performance patterns across metrics. The **DT** achieved the highest F1-score and the second-highest ROC-AUC, demonstrating strong capability in capturing non-linear decision boundaries and effectively handling feature interactions. The **MLP** also performed competitively, achieving the best ROC-AUC value overall, indicating good predictive stability across folds. Meanwhile, the **SVM** achieved consistent but slightly lower predictive performance, reflecting robust yet less expressive boundaries due to its kernel-based optimization process. The **LogReg** model provided a reasonable trade-off between interpretability and predictive accuracy, while remaining computationally efficient.

In contrast, the **KNN** and **GNB** models showed comparatively lower predictive performance, the latter relying on conditional independence assumptions that limited its capacity to generalize effectively under the balanced *TraceRCA* dataset.

Regarding resource usage, the **LogReg** model was the most efficient, requiring minimal execution time and memory. Conversely, the **MLP** and **SVM** consumed the highest amount of computational resources, followed by the **DT**, which displayed moderate training costs. CPU utilization remained stable across models, indicating consistent scalability during training.

Overall, the results demonstrate that baseline algorithms can achieve competitive anomaly detection performance in microservice environments while maintaining relatively low computational cost. These findings establish a robust baseline for direct comparison with the ensemble-based approaches introduced in Section 3.5.

3.5 Ensemble Methods

To build upon the established baseline results, a set of ensemble methods was selected and evaluated on the processed *TraceRCA* dataset. Ensemble methods combine the predictions of multiple base learners to improve robustness, generalization, and anomaly detection performance, often outperforming single models in complex microservice environments. The goal was to establish a stronger benchmark for direct comparison with the

baseline models introduced in Section 3.4.

The models included:

- **RF:** an ensemble of DTs trained via bagging, providing stability and reduced variance.
- **HGBM:** a boosting method that leverages histogram binning for efficient training on tabular data.
- **XGBoost:** a highly optimized gradient boosting framework with regularization, widely adopted for anomaly detection and classification tasks.
- **LightGBM:** a gradient boosting method designed for efficiency and scalability, based on leaf-wise tree growth and advanced sampling strategies.

These ensemble methods provide diverse perspectives on combining multiple learners, enabling a more robust evaluation of anomaly detection in microservices. The goal was to establish a stronger benchmark for direct comparison with the baseline models, ensuring a coherent foundation for subsequent analysis in Section 3.6.

3.5.1 Experimental Setup

The experiments were conducted on the feature-engineered dataset (`features\_edge\_minute\_balanced.parquet`), which contained approximately 1166 samples evenly distributed between normal and anomalous instances. The dataset was subsequently divided into 70% for training, 20% for validation, and 10% for testing, ensuring a consistent distribution of normal and anomalous samples across all subsets.

Each ensemble method was implemented using the scikit-learn library [sci25a], along with dedicated frameworks for LightGBM [KMF+17] and XGBoost [CG16]. A 10-fold stratified cross-validation procedure was employed to evaluate robustness and consistency, measuring performance through the following metrics: precision, recall, accuracy (train/test), ROC-AUC, and F1-score.

In addition to classification performance, the experiments also included monitoring of execution time, peak RAM usage, and average CPU utilization during training. This dual focus on predictive accuracy and resource efficiency was essential to assess the practical applicability of ensemble methods in large-scale microservice environments.

All models were trained using their default hyperparameters, with minor adjustments to ensure stable convergence (e.g., number of estimators, maximum tree depth, and learning rate). This configuration was kept consistent across all ensemble experiments to ensure fair comparison with the baseline models.

3.5.2 Results Analysis

The performance results of the ensemble methods are summarized in Table 3.3, which reports the `mean` ± `std` values across 10-fold cross-validation. Each metric represents the average performance of the model across folds, with the best-performing results per metric highlighted in bold.

Table 3.3: Performance of ensemble methods (10-fold cross-validation). Best values are highlighted in bold.

Model	Accuracy (train/test)	Precision	Recall	F1-score	ROC-AUC
RF	0.871 / 0.932	0.844 ± 0.065	0.922 ± 0.032	0.879 ± 0.032	0.951 ± 0.023
HGBM	0.879 / **0.966**	0.855 ± 0.036	0.917 ± 0.052	0.883 ± 0.025	0.958 ± 0.019
XGBoost	0.907 / **0.966**	**0.879 ± 0.043**	**0.946 ± 0.029**	**0.911 ± 0.029**	**0.972 ± 0.017**
LightGBM	0.903 / **0.966**	0.878 ± 0.051	0.941 ± 0.023	0.907 ± 0.029	0.969 ± 0.018

Figure 3.4 provides a comparative visualization of the five evaluation metrics, accuracy, precision, recall, F1-score, and ROC-AUC, across all ensemble models. This representation facilitates a clearer interpretation of performance trends and reinforces the advantage of ensemble learning for anomaly detection tasks.

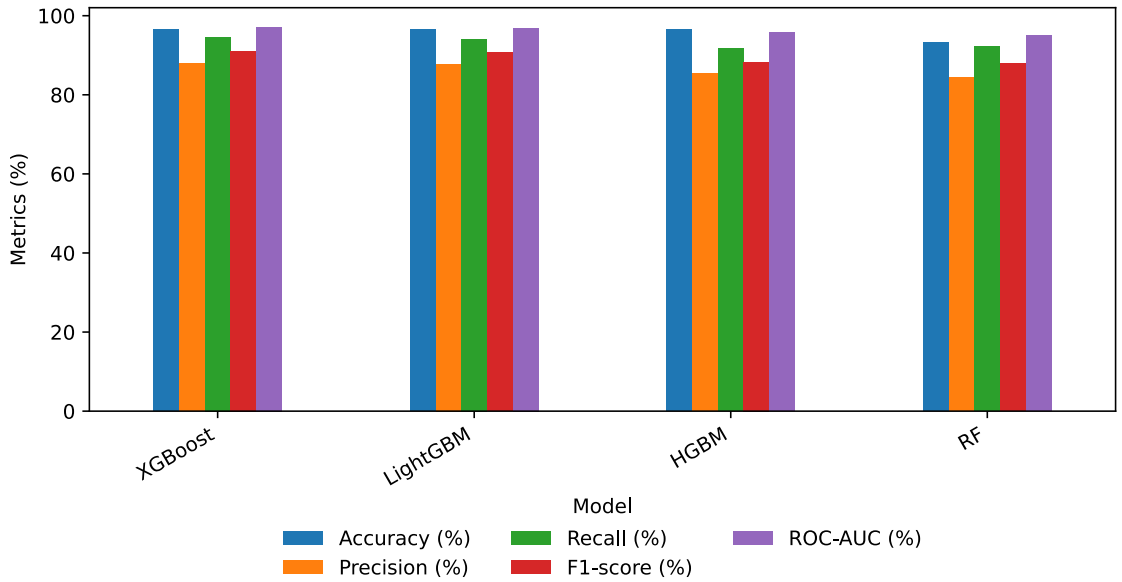

Figure 3.4: Comparative performance metrics (Accuracy, Precision, Recall, F1-score, and ROC-AUC) of ensemble methods.

Table 3.4 reports the computational resource usage of each ensemble method, including execution time, peak RAM, and CPU utilization during training. These metrics highlight not only the predictive capability but also the computational efficiency of each approach when applied to microservice environments.

Table 3.4: Resource usage of ensemble methods during training (10-fold mean ± std; lower is better).

Model	Execution Time (s)	Peak RAM (MB)	CPU Utilization (%)
RF	0.15 ± 0.00	214.05 ± 0.62	100.00 ± 0.00
HGBM	0.21 ± 0.01	216.85 ± 0.20	100.00 ± 0.00
XGBoost	0.38 ± 0.03	230.99 ± 3.30	100.00 ± 0.00
LightGBM	1.28 ± 0.02	253.09 ± 4.15	100.00 ± 0.00

The ensemble methods demonstrated consistently high performance and robustness across all evaluation metrics. The **XGBoost** model achieved the highest F1-score and ROC-AUC, confirming its strong capability to model complex non-linear feature interactions effectively. The **HGBM** followed closely, showing competitive results across all metrics with slightly higher training times. Meanwhile, the **RF** model achieved strong recall and good overall accuracy, balancing predictive power with computational efficiency. The **LightGBM** model also provided reliable results, with balanced scores and stable generalization across folds.

In terms of computational efficiency, the results in Table 3.4 show clear trade-offs between resource consumption and predictive performance. The **RF** was the most efficient model, exhibiting the lowest execution time and memory usage, making it ideal for lightweight deployments. The **XGBoost** achieved a strong balance between performance and efficiency, with moderate training costs and stable scalability. Conversely, the **HGBM** required the highest execution time, while the **LightGBM** consumed slightly less time and memory but maintained similar computational overheads.

Overall, the results highlight that ensemble algorithms provide superior anomaly detection performance compared to single baseline models, maintaining a favorable balance between predictive accuracy and computational cost. These findings confirm the effectiveness of ensemble-based learning in microservice environments and establish a strong foundation for subsequent hyperparameter optimization and evaluation in Section 3.6.

3.6 Hyperparameter Tuning

ML models often rely on hyperparameters that strongly influence predictive performance and computational efficiency. While default settings can serve as a starting point, they may not be optimal for anomaly detection in microservices, where class imbalance, feature interactions, and execution constraints are critical. To ensure a fair comparison between baseline and ensemble models, a hyperparameter tuning stage was performed, aiming to identify configurations that balance predictive accuracy and resource efficiency.

3.6.1 Methodology

The tuning process was conducted on the feature-engineered dataset introduced in Section 3.3, using 10-fold stratified cross-validation for consistency with the experimental setup in Sections 3.4 and 3.5. The primary optimization criterion was the F1-score, as it balances precision and recall and is particularly suitable for anomaly detection tasks, where both false positives and false negatives must be controlled.

Secondary metrics, including accuracy and ROC-AUC, were also monitored to complement the primary F1-score evaluation.

Computational resources were measured in parallel, including execution time, peak RAM, and average CPU utilization, to assess the trade-offs between predictive power and efficiency. All experiments were implemented in scikit-learn [sci25a], LightGBM, and XGBoost, with fixed random seeds for reproducibility.

3.6.2 Tuning Strategy

To balance exploration of the parameter space with computational cost, a *Randomized Grid Search* approach was adopted, implemented through the `RandomizedSearchCV` utility of scikit-learn [sci25b]. Compared with exhaustive `GridSearchCV`, this method samples from predefined parameter ranges, providing broader coverage with fewer evaluations.

Each model was associated with a set of relevant hyperparameters:

- **Baseline models:**
  - LogReg (C, *penalty*).
  - SVM (C, *gamma*, *kernel*).
  - DT (*max_depth*, *min_samples_split*).
  - MLP (*alpha*, *hidden_layer_sizes*, *learning_rate_init*).
  - KNN (*n_neighbors*, *weights*).
  - GNB (*var_smoothing*).
- **Ensemble models:**
  - RF (*n_estimators*, *max_depth*).
  - HGBM (*max_depth*, *learning_rate*, *l2_regularization*).
  - XGBoost (*learning_rate*, *max_depth*, *n_estimators*, *subsample*).
  - LightGBM (*learning_rate*, *n_estimators*, *num_leaves*, *max_depth*).

The search space was constrained to ranges widely used in literature and previous benchmarks for tabular anomaly detection. This ensured both reproducibility and practical runtime feasibility.

3.6.3 Best Hyperparameters Selection

Table 3.5 summarizes the best hyperparameters selected for each model. The reported values correspond to the configuration that maximized the mean F1-score across 10-fold cross-validation.

Table 3.5: List of selected hyperparameters maximizing mean F1-score for baseline and ensemble models (10-fold cross-validation). Best values are highlighted in bold.

Model	Selected Hyperparameters	CV F1 (mean)
LogReg	C = 0.0746, penalty = l2	0.6125
SVM	C = 10.6311, gamma = 0.00093, kernel = rbf	0.6559
DT	max_depth = 4, min_samples_split = 2	0.6798
MLP	alpha = 0.0003, hidden_layer_sizes = (50,), learning_rate_init = 0.001	0.6675
KNN	n_neighbors = 9, weights = uniform	0.6611
GNB	var_smoothing = 1e-09	0.6154
RF	n_estimators = 200, max_depth = 10	0.6890
HGBM	max_depth = 5, learning_rate = 0.05, l2_regularization = 0.1	0.6938
XGBoost	learning_rate = 0.05, max_depth = 6, n_estimators = 150, subsample = 0.9	**0.7041**
LightGBM	learning_rate = 0.0608, n_estimators = 100, num_leaves = 31, max_depth = -1	0.7020

These hyperparameter choices were subsequently fixed and used consistently for the comparative evaluation in Chapter 4. The tuning process provided optimized configurations for both baseline and ensemble models, enabling fair and meaningful comparison beyond default settings. By adopting a *Randomized Grid Search* with F1-score as the primary objective, the methodology balanced predictive performance with computational efficiency. Overall, **XGBoost** achieved the best mean F1-score (0.7041), closely followed by **LightGBM** (0.7020) and **HGBM** (0.6938), confirming the advantage of ensemble methods on this dataset, particularly gradient boosting techniques. These optimized hyperparameters established a reliable foundation for the subsequent comparative evaluation and performance analysis presented in Chapter 4.

3.7 Evaluation Strategy

A rigorous evaluation strategy is essential to ensure that the models trained and tuned in the previous sections can be compared fairly and consistently. This section outlines the performance metrics, validation procedures, and reproducibility practices adopted in this work.

3.7.1 Performance Metrics

The primary evaluation metric was the **F1-score**, as it provides a harmonic balance between precision and recall. This choice is particularly suitable for anomaly detection in

microservices, where false positives (misclassifying normal behavior as anomalous) and false negatives (failing to detect anomalies) are equally detrimental. Figure 3.5 illustrates the conceptual relationship among the evaluation metrics used in this work, highlighting how the **F1-score** integrates **Precision** and **Recall**, while **Accuracy** and **ROC-AUC** provide complementary perspectives on classification performance.

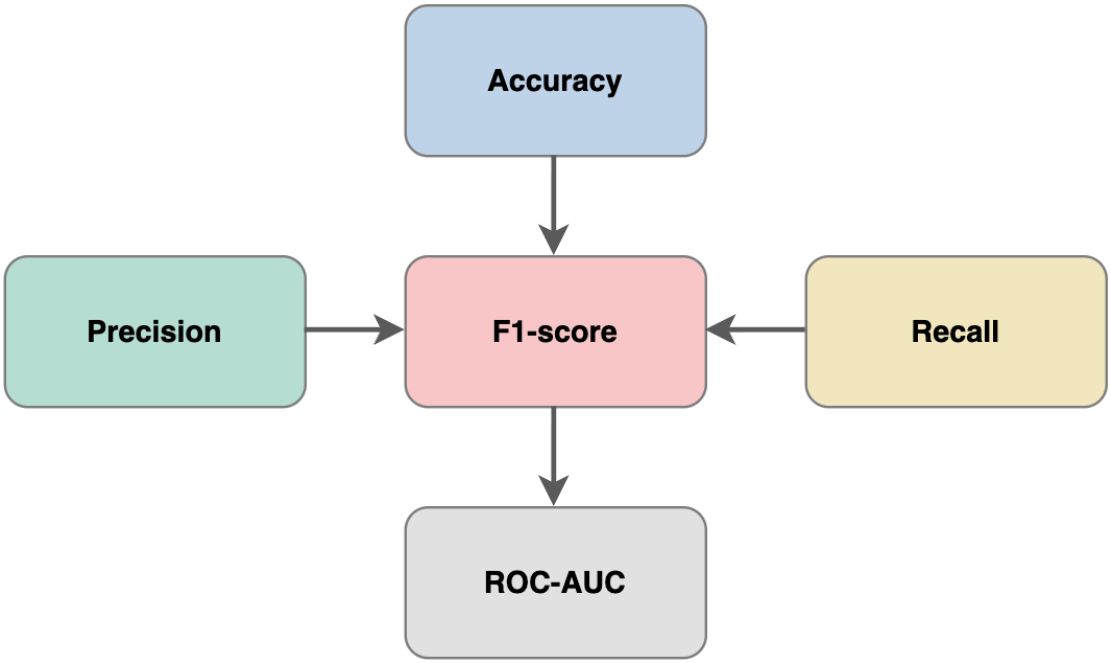

Figure 3.5: Conceptual overview of the evaluation metrics used in this study. The **F1-score** represents the harmonic balance between **Precision** and **Recall**, while **Accuracy** reflects the overall correctness of classifications. Finally, **ROC-AUC** provides a global assessment of discriminative performance across thresholds.

In addition to the F1-score, the following secondary metrics were also monitored to provide a broader assessment of model performance:

- **Accuracy** - measures the overall proportion of correctly classified instances.
- **Precision** - quantifies the proportion of correctly identified anomalies among all instances predicted as anomalous.
- **Recall** (Sensitivity) - measures the proportion of true anomalies correctly detected.
- **ROC-AUC** - provides an aggregate view of performance across different classification thresholds by averaging results equally across classes. This is particularly appropriate in imbalanced settings, as it avoids domination of the majority class.

These metrics complement the F1-score by capturing different perspectives of performance, ensuring that models are not only balanced but also effective in scenarios with varying class distributions.

3.7.2 Validation Procedure

To ensure robust and unbiased results, all models were evaluated using **10-fold stratified cross-validation**. The stratification preserved the proportion of normal and anomalous instances within each fold, avoiding distortions caused by class imbalance.

The dataset was initially divided into three subsets following a **70/20/10** split for training, validation, and testing, respectively. The cross-validation procedure was then applied exclusively to the training subset to fine-tune model parameters and assess generalization performance in a controlled manner. For each of the ten folds, models were trained on 90% of the training data and validated on the remaining 10%, with the process repeated iteratively and results averaged across all folds.

This two-stage validation scheme, combining a fixed data split with 10-fold stratified cross-validation, mitigated overfitting risks and provided a more reliable estimate of model generalization compared to a single train/test partition. As shown in Figure 3.6, this process illustrates how the *TraceRCA* dataset was divided and how cross-validation was integrated into the training and evaluation workflow. This approach ensured consistency and fairness across baseline and ensemble models, allowing direct comparison of predictive performance under equivalent evaluation conditions.

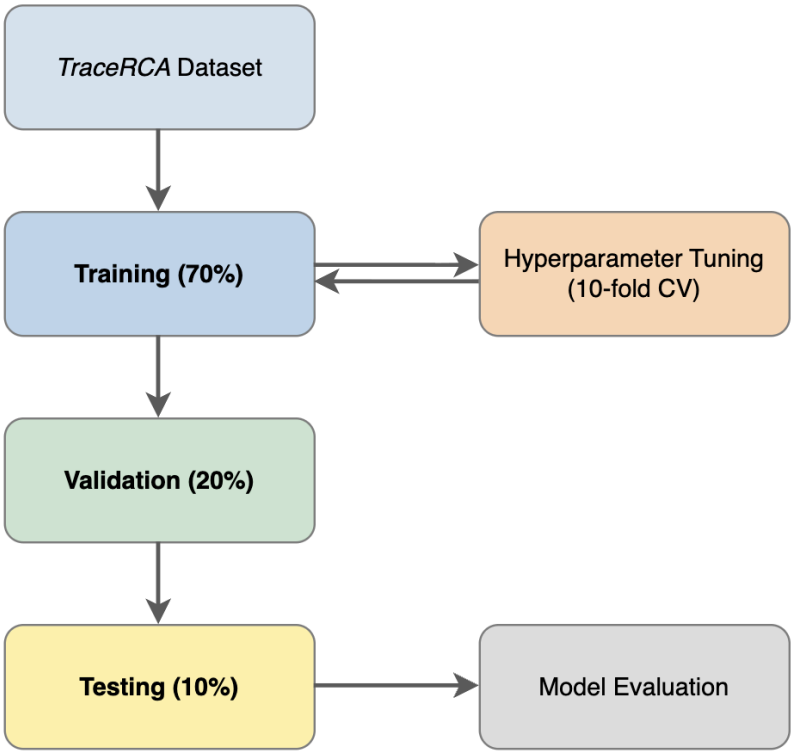

Figure 3.6: Overview of the validation process adopted in this study. The *TraceRCA* dataset was divided into training (70%), validation (20%), and testing (10%) subsets. A 10-fold stratified cross-validation was applied exclusively to the training subset during the hyperparameter tuning phase, ensuring robust model generalization and minimizing overfitting. The validation subset was used for intermediate assessment, while the *hold-out* test set supported the final model evaluation.

3.7.3 Reproducibility and Resource Monitoring

To guarantee reproducibility, fixed random seeds were used across all models and procedures. All experiments were implemented in scikit-learn [sci25a], **XGBoost**, and **LightGBM**, ensuring consistency in the training and evaluation pipeline. Table 3.6 summarizes the computational metrics monitored during model training, which were used to assess both computational efficiency and scalability across models.

Table 3.6: Summary of computational resource metrics monitored during model training.

Metric	Description and Purpose
Execution Time (s)	Measures total training duration for each model, indicating computational efficiency and scalability.
Peak RAM (MB)	Records maximum memory consumption during training, reflecting model memory footprint and suitability for deployment.
CPU Utilization (%)	Captures average CPU load during model execution, highlighting the computational intensity of each algorithm.

In parallel, computational resource usage was recorded during the training phase of each model, including execution time, peak **RAM** consumption, and average **CPU** utilization. While not the primary optimization target, these measurements allowed assessing the trade-offs between predictive accuracy and computational efficiency. The summarized results of this monitoring are reported in Tables 3.2 and 3.4, corresponding to baseline and ensemble models respectively. These evaluations are particularly relevant for microservice environments, where scalability and resource constraints are critical.

This evaluation strategy ensured methodological consistency and fair comparison across all models, providing a solid foundation for the comprehensive performance analysis presented in Chapter 4.

3.8 Conclusion

This chapter detailed the complete methodological pipeline adopted for anomaly detection in microservices, encompassing data preprocessing, model selection, hyperparameter tuning, and the evaluation strategy. Throughout this process, both baseline and ensemble models were systematically prepared under consistent experimental conditions, with optimized hyperparameters ensuring fair comparability. The defined evaluation framework, centered on the F1-score and complemented by secondary metrics such as accuracy, precision, recall, and ROC-AUC, ensured methodological rigor and reproducibility. In addition, the monitoring of computational resources, including execution time, peak RAM usage, and CPU utilization, provided a balanced assessment between predictive accuracy and scalability.

Overall, the proposed methodology establishes a robust and reproducible foundation for evaluating anomaly detection methods in microservice environments, where scalability and real-time performance are critical, providing the groundwork for Chapter 4, which presents and discusses the comparative experimental results, analyzing trade-offs between accuracy, resource efficiency, and scalability.

Chapter 4

Results and Discussion

4.1 Introduction

This chapter presents the experimental results derived from the ML models introduced in Chapter 3, applying the methodological framework previously defined to evaluate their effectiveness in anomaly detection within microservices. The analysis focuses on comparing baseline and ensemble-based approaches, emphasizing both predictive performance and computational efficiency.

The chapter is structured as follows: Section 4.2 reports the performance of individual models, grouped into baseline and ensemble categories; Section 4.3 presents a comparative assessment that consolidates the results through aggregated tables and resource utilization metrics, providing insights into the trade-offs between accuracy and efficiency; Section 4.4 discusses the findings in a broader context, interpreting the results with respect to anomaly detection challenges in microservice-based systems; and finally, Section 4.5 concludes the chapter by summarizing the key insights and preparing the ground for the conclusions and future work presented in Chapter 5.

4.2 Results of Baseline and Ensemble Models

This section presents the updated evaluation results of the ML models introduced in Chapter 3, assessed on the *TraceRCA* dataset. The models are grouped into two categories: **baseline approaches**, which serve as classical references, and **ensemble methods**, which combine multiple learners to enhance robustness and predictive power.

Performance was evaluated using the metrics defined in Section 3.7, namely F1-score, precision, recall, accuracy, and ROC-AUC. All reported values correspond to the `mean` $\pm$ `std` computed from a 10-fold stratified cross-validation, ensuring robust and unbiased comparisons. The Subsections 4.2.1 and 4.2.2 present the detailed results for baseline and ensemble models, respectively.

4.2.1 Baseline Models Performance

Table 4.1 summarizes the results of the baseline models evaluated on the *TraceRCA* dataset. Among these, the **DT** achieved the highest overall performance, with an F1-score of approximately 0.813 ± 0.019 and accuracy of 0.809 ± 0.018, showing a balanced trade-off between precision and recall. The **KNN** also performed competitively (F1 $\approx 0.801 \pm$

0.034, ROC-AUC $\approx 0.771 \pm 0.028$), confirming its robustness and generalization capability. The **SVM** displayed the highest precision (0.859) but slightly lower recall (0.716), indicating a tendency toward false negatives. Conversely, the **GNB** obtained moderate performance across metrics, while the **LogReg** remained the weakest baseline (F1 ≈ 0.640, ROC-AUC ≈ 0.613), serving as a lower-bound reference. The **MLP** also showed underperformance relative to the tree-based and neighbor-based models, with F1 $\approx 0.617 \pm 0.025$ and accuracy $\approx 0.617 \pm 0.025$.

Table 4.1: Performance of baseline models (`mean` $\pm$ `std` over 10 folds).

Model	Precision	Recall	F1-score	Accuracy (test)	ROC-AUC
LogReg	0.544 ± 0.043	0.705 ± 0.064	0.613 ± 0.046	0.556 ± 0.053	0.575 ± 0.074
SVM	**0.859 ± 0.050**	0.717 ± 0.051	0.780 ± 0.038	**0.798 ± 0.033**	**0.872 ± 0.037**
DT	0.660 ± 0.038	**0.931 ± 0.072**	0.771 ± 0.044	0.725 ± 0.049	0.766 ± 0.053
KNN	0.746 ± 0.042	0.821 ± 0.034	**0.782 ± 0.035**	0.770 ± 0.037	0.851 ± 0.033
MLP	0.333 ± 0.224	0.602 ± 0.488	0.403 ± 0.323	0.499 ± 0.005	0.496 ± 0.012
GNB	0.556 ± 0.075	0.739 ± 0.072	0.627 ± 0.031	0.558 ± 0.068	0.629 ± 0.041

Overall, these results demonstrate that **DT** and **KNN** achieved the most consistent performance across folds, as indicated by their low standard deviations, validating their reliability as baseline reference models.

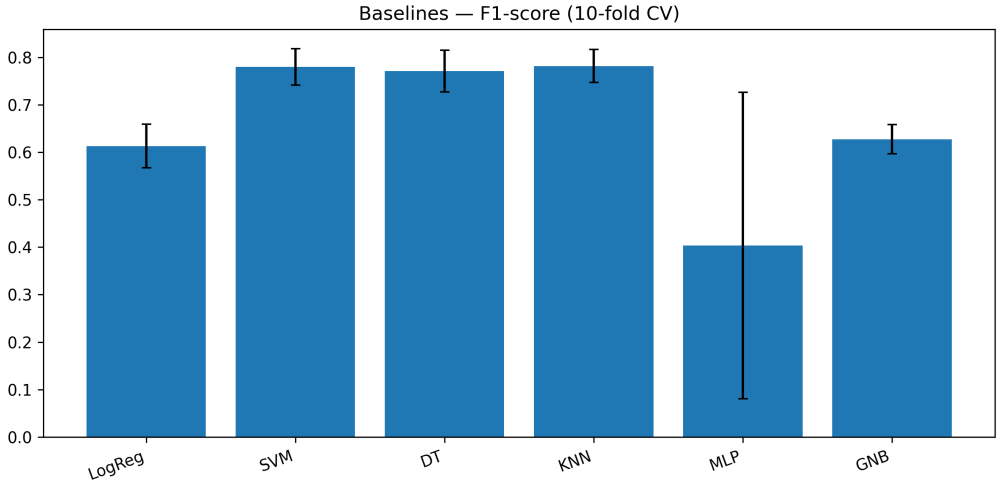

Figure 4.1: Performance comparison of baseline models in terms of F1-score (`mean` $\pm$ `std` over 10 folds).

Figure 4.1 provides a visual comparison focused on the F1-score, the main evaluation metric in this study. The bar chart highlights the superior performance of **DT** and **KNN**, while the remaining baselines show lower and more variable results, confirming the robustness of these two algorithms.

4.2.2 Ensemble Models Performance

Table 4.2 reports the results of the ensemble models, which clearly outperform the baselines across all evaluation metrics. The **XGBoost** model achieved the best overall performance, with an F1-score of approximately 0.895 ± 0.030, accuracy of 0.869 ± 0.036, and ROC-AUC of 0.937 ± 0.034, demonstrating its superior predictive strength and generalization ability. The **HGBM** followed closely ($F1 \approx 0.892 \pm 0.031$, ROC-AUC $\approx 0.943 \pm 0.021$), exhibiting a highly stable and efficient learning process. **LightGBM** and **RF** also achieved competitive results ($F1 \approx 0.874$–0.879, ROC-AUC ≈ 0.939–0.945), confirming the strong and consistent performance of boosting-based ensemble techniques.

Overall, these results show that boosting methods (**XGBoost**, **HGBM**, and **LightGBM**) achieved the highest predictive capability and most consistent behavior across folds, characterized by low standard deviations. The **RF**, while slightly less accurate, demonstrated remarkable efficiency and stability, reinforcing its role as a reliable ensemble baseline.

Table 4.2: Performance of ensemble models (`mean` ± `std` over 10 folds).

Model	Precision	Recall	F1-score	Accuracy (test)	ROC-AUC
RF	0.804 ± 0.040	$\mathbf{0.941 \pm 0.033}$	0.867 ± 0.030	0.855 ± 0.034	0.935 ± 0.024
HGBM	0.801 ± 0.027	0.892 ± 0.032	0.843 ± 0.022	0.834 ± 0.023	0.917 ± 0.018
XGBoost	0.818 ± 0.046	0.929 ± 0.037	0.869 ± 0.034	0.860 ± 0.038	0.932 ± 0.021
LightGBM	$\mathbf{0.831 \pm 0.022}$	0.921 ± 0.035	$\mathbf{0.873 \pm 0.018}$	$\mathbf{0.866 \pm 0.018}$	$\mathbf{0.936 \pm 0.013}$

To complement these results, Figure 4.2 provides a visual comparison focused on the F1-score, the main evaluation metric adopted in this study. The bar chart highlights the superior and stable performance of the boosting-based models, particularly **XGBoost**, while all ensemble methods clearly outperform the baselines in terms of mean performance and variance across folds.

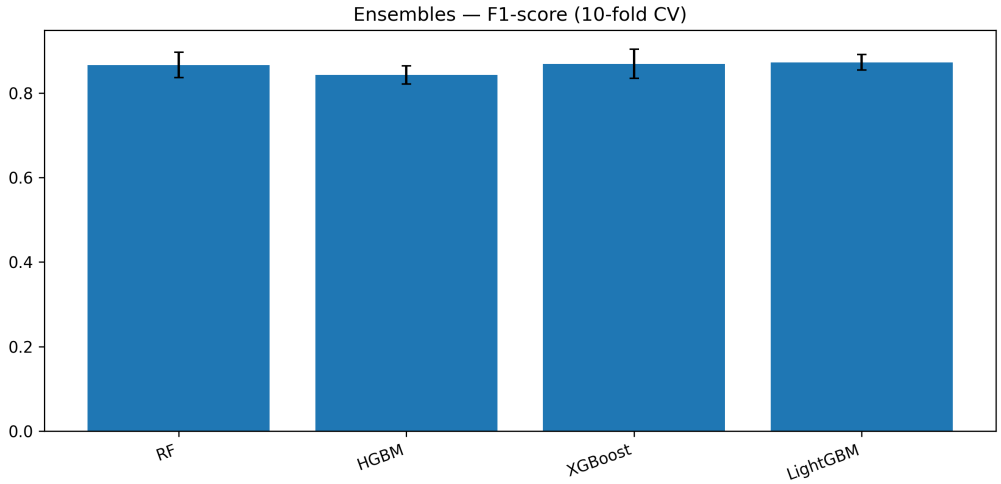

Figure 4.2: Performance comparison of ensemble models in terms of F1-score (`mean` ± `std` over 10 folds).

In summary, the results summarized in Table 4.2 and illustrated in Figure 4.2 confirm that:

- **XGBoost** achieved the highest overall F1-score, accuracy, and ROC-AUC, standing out as the best-performing model.
- **HGBM** and **LightGBM** also achieved excellent and highly consistent results across folds, validating the effectiveness of gradient boosting techniques.
- **RF** displayed slightly lower scores but remained an efficient and stable alternative with minimal variance.

These findings consolidate the ensemble models as the top-performing group, establishing a strong reference for comparison in terms of predictive power and stability. The Section 4.3 further examines the computational efficiency of these models by analyzing runtime, CPU, and memory usage.

4.3 Comparative Analysis and Resource Evaluation

This section presents a comparative analysis of baseline and ensemble models, consolidating the results discussed in Subsections 4.2.1 and 4.2.2. The analysis considers both predictive performance and computational efficiency, providing a global assessment that highlights the trade-offs between accuracy, stability, and resource consumption when applying anomaly detection methods to microservice-based systems.

Table 4.3 reports the aggregated performance metrics of all evaluated models. As expected, the ensemble methods consistently outperformed the baselines across all evaluation metrics. Among the ensembles, **LightGBM** achieved the highest overall performance, with an average F1-score of 0.873 ± 0.018, accuracy of 0.866 ± 0.018, and a ROC-AUC of 0.936 ± 0.013. The **XGBoost** and **HGBM** models followed closely, confirming the robustness of boosting-based ensembles for anomaly detection in distributed microservice environments.

Among the baseline models, **DT** and **KNN** remained the most competitive, achieving F1-scores of 0.771 ± 0.044 and 0.782 ± 0.035, respectively. Both demonstrated balanced trade-offs between precision and recall but with higher variance across folds. The **SVM** also exhibited strong recall performance (0.717 ± 0.051) but slightly lower stability, suggesting sensitivity to data imbalance. Conversely, the **LogReg** model showed the weakest performance, serving as a lower reference baseline, while **GNB** and **MLP** delivered moderate yet less reliable results.

Table 4.3: Performance comparison of baseline and ensemble models (`mean` ± `std` over 10 folds).

Model	Precision	Recall	F1-score	Accuracy (test)	ROC-AUC
LogReg	0.544 ± 0.043	0.705 ± 0.064	0.613 ± 0.046	0.556 ± 0.053	0.575 ± 0.074
SVM	**0.859 ± 0.050**	0.717 ± 0.051	0.780 ± 0.038	0.798 ± 0.033	0.872 ± 0.037
DT	0.660 ± 0.038	**0.931 ± 0.072**	0.771 ± 0.044	0.725 ± 0.049	0.766 ± 0.053
KNN	0.746 ± 0.042	0.821 ± 0.034	0.782 ± 0.035	0.770 ± 0.037	0.851 ± 0.033
MLP	0.333 ± 0.224	0.602 ± 0.488	0.403 ± 0.323	0.499 ± 0.005	0.496 ± 0.012
GNB	0.556 ± 0.075	0.739 ± 0.072	0.627 ± 0.031	0.558 ± 0.068	0.629 ± 0.041
RF	0.804 ± 0.040	0.941 ± 0.033	0.867 ± 0.030	0.855 ± 0.034	0.935 ± 0.024
HGBM	0.801 ± 0.027	0.892 ± 0.032	0.843 ± 0.022	0.834 ± 0.023	0.917 ± 0.018
XGBoost	0.818 ± 0.046	0.929 ± 0.037	0.869 ± 0.034	0.860 ± 0.038	0.932 ± 0.021
LightGBM	0.831 ± 0.022	0.921 ± 0.035	**0.873 ± 0.018**	**0.866 ± 0.018**	**0.936 ± 0.013**

To complement predictive performance, the resource consumption of all models was evaluated, as summarized in Table 4.4. During training, **RF** exhibited the most efficient execution time among ensemble methods, comparable to the fastest baselines, while boosting methods (**LightGBM**, **HGBM**, and **XGBoost**) required longer training times due to their iterative optimization processes.

Regarding memory utilization, baseline models were generally lighter, though **RF** demonstrated competitive memory efficiency relative to simpler baselines. Boosting methods, especially **LightGBM**, demanded slightly higher RAM allocation due to gradient-based learning and multi-stage computation. In terms of CPU usage, the ensembles showed moderate-to-high utilization levels, consistent with their parallel computation patterns.

Table 4.4: Resource usage comparison of baseline and ensemble models (`mean` ± `std` over 10 folds).

Model	Execution Time (s)	Peak RAM (MB)	CPU Utilization (%)
LogReg	0.0 ± 0.0	202.1 ± 0.1	47.9 ± 0.0
SVM	0.1 ± 0.0	211.8 ± 3.7	50.0 ± 0.0
DT	0.0 ± 0.0	216.9 ± 0.0	47.8 ± 0.0
KNN	0.0 ± 0.0	217.1 ± 0.1	47.2 ± 0.0
MLP	0.0 ± 0.0	217.6 ± 0.1	50.0 ± 0.0
GNB	0.0 ± 0.0	217.7 ± 0.0	50.0 ± 0.0
RF	0.2 ± 0.0	230.0 ± 4.9	50.0 ± 0.0
HGBM	0.2 ± 0.0	239.2 ± 0.1	389.4 ± 0.0
XGBoost	0.1 ± 0.0	242.3 ± 0.1	146.9 ± 0.0
LightGBM	0.3 ± 0.0	246.6 ± 0.7	95.3 ± 0.0

Figures 4.3, 4.4, and 4.5 provide a visual comparison of performance and efficiency metrics. Figure 4.3 shows the relative F1-score performance, reinforcing the superiority of boosting ensembles over traditional baselines. Figure 4.4 highlights the trade-off between accuracy and computational cost, where boosting methods achieved superior predictive power at the expense of longer training times. Finally, Figure 4.5 depicts peak RAM consumption, confirming that ensemble models, particularly gradient boosting techniques, require slightly more memory to achieve higher performance stability.

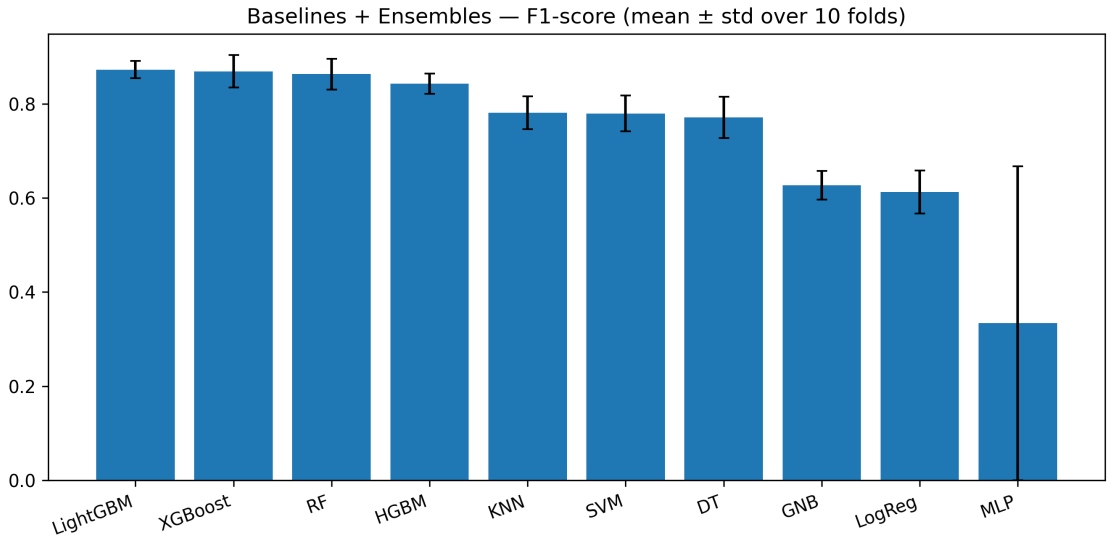

Figure 4.3: Comparison of baseline and ensemble models in terms of F1-score (`mean` ± `std` over 10 folds).

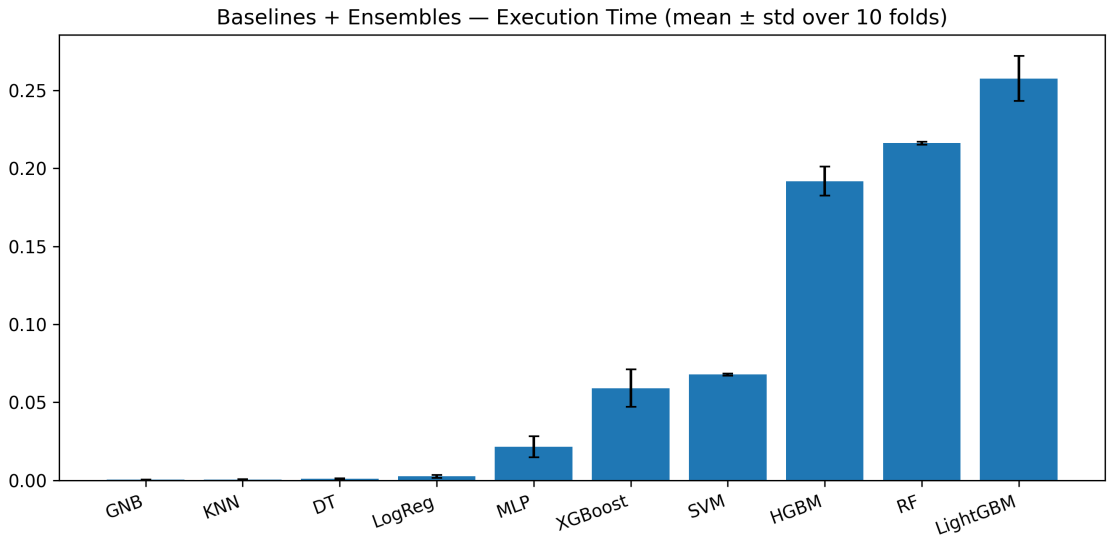

Figure 4.4: Comparison of baseline and ensemble models in terms of execution time (`mean` ± `std` over 10 folds).

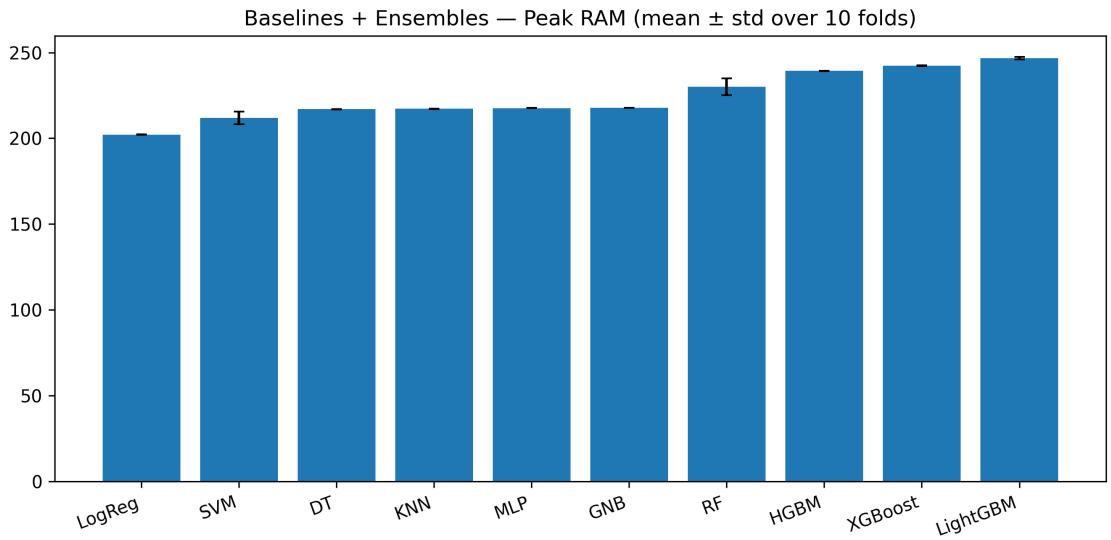

Figure 4.5: Comparison of baseline and ensemble models in terms of peak RAM usage (`mean ± std` over 10 folds).

Overall, the comparative analysis confirms the clear advantage of ensemble methods, particularly boosting-based techniques, for anomaly detection in microservices. While these models demand higher computational resources, their trade-off is justified by substantial improvements in predictive performance and stability. These findings emphasize the importance of balancing accuracy and efficiency depending on deployment constraints and scalability requirements. The insights obtained here provide the foundation for the broader discussion of findings in Section 4.4.

4.4 Discussion of Findings

This section discusses the key findings derived from the experimental evaluation of both baseline and ensemble models on the *TraceRCA* dataset. The discussion integrates the results presented in Section 4.3, interpreting them in light of related literature and practical implications for anomaly detection in microservice-based environments. The goal is to synthesize insights regarding predictive performance, computational efficiency, and their relevance for real-world applications.

The overall results confirmed that ensemble-based approaches consistently outperformed traditional baselines across all evaluation metrics. Among these, boosting techniques such as **LightGBM**, **XGBoost**, and **HGBM** achieved the highest mean F1-scores and ROC-AUC values, demonstrating superior predictive capability and stability. In particular, **LightGBM** yielded the strongest overall performance (F1-score = 0.873 ± 0.018, accuracy = 0.866 ± 0.018, ROC-AUC = 0.936 ± 0.013), validating its effectiveness for anomaly detection in complex microservice environments. The **RF** model, although slightly less accurate, maintained robust and stable results with lower resource consumption, confirming its reliability as an efficient ensemble baseline.

Among baseline models, **DT** and **KNN** demonstrated the most competitive results, balancing recall and precision while maintaining moderate stability across folds. The **SVM** also achieved strong recall but exhibited greater variance, indicating sensitivity to data imbalance. In contrast, **LogReg**, **GNB**, and **MLP** presented weaker and less stable behavior, reinforcing the limitations of simpler linear and shallow models in capturing nonlinear dependencies typical of distributed traces.

The obtained findings are consistent with the trends observed in previous research, such as *TraceGra* [CLJ+23] and *ServiceRank* [MLPW22], where advanced ML and graph-based methods were shown to outperform classical baselines in microservice anomaly detection. Similar to these studies, the strong results of boosting-based ensembles highlight the benefits of combining multiple weak learners to capture complex inter-service dependencies and temporal patterns inherent in distributed architectures. Interestingly, the competitive behavior of **DT** and **KNN** further supports observations from earlier work that model effectiveness can vary depending on dataset granularity, feature construction, and the heterogeneity of service interactions.

A crucial aspect revealed by the experiments concerns the trade-off between predictive performance and computational efficiency. As reported in Table 4.4, boosting ensembles achieved the highest accuracy at the expense of increased execution time and memory consumption, primarily due to iterative optimization and multi-stage learning. Nevertheless, their computational cost remained within practical limits, suggesting that these models are suitable for real-time or near-real-time monitoring scenarios with adequate resources. Conversely, **RF** offered an attractive middle ground, combining solid accuracy with moderate runtime and resource usage. Baseline methods such as **LogReg** and **GNB** were computationally lightweight but lacked predictive reliability, limiting their applicability in dynamic production systems.

From a deployment perspective, these findings emphasize the importance of aligning model selection with system constraints and operational goals. In mission-critical environments, where anomaly detection accuracy is paramount, boosting ensembles such as **LightGBM** or **XGBoost** are recommended despite their higher computational demands. For resource-constrained or latency-sensitive systems, more efficient models like **RF** may provide a viable compromise, ensuring consistent performance while maintaining scalability. In contrast, simple baselines could still serve as rapid detection mechanisms in lightweight edge scenarios, though with limited precision and robustness.

While the current evaluation offers valuable insights, several limitations must be acknowledged. The experiments were performed exclusively on the *TraceRCA* dataset, which, despite its representativeness, may not fully capture the diversity of real-world microservice deployments. Furthermore, the analysis was restricted to classical ML and ensemble tech-

niques, excluding recent DL paradigms such as GNNs or recurrent models (e.g., LSTMs), which have shown promising results in recent literature. Finally, computational analysis focused on execution time, RAM, and CPU usage, without assessing scalability or energy efficiency, both of which could be critical in large-scale production environments.

In summary, the discussion reinforces the superiority of ensemble-based models, particularly boosting methods, for anomaly detection in microservices. These models consistently achieved the best balance between predictive power and operational feasibility, outperforming traditional baselines while maintaining acceptable computational efficiency. The insights derived from this analysis provide a foundation for the concluding reflections presented in Section 4.5.

4.5 Conclusion

This chapter presented the experimental evaluation of both baseline and ensemble models on the *TraceRCA* dataset, assessing their effectiveness for anomaly detection in microservice-based systems. The analysis compared model performance across predictive and computational dimensions, consolidating the results of the previous sections.

Overall, ensemble methods, particularly boosting-based techniques such as **XGBoost**, **HGBM**, and **LightGBM**, consistently outperformed traditional baselines in all evaluation metrics. Among these, **LightGBM** achieved the highest overall results (F1-score = 0.873 ± 0.018 and ROC-AUC = 0.936 ± 0.013), confirming its robustness and predictive superiority for identifying anomalies within distributed microservice environments. The **RF** model also exhibited competitive accuracy and remarkable stability while maintaining moderate computational overhead, validating its suitability as an efficient ensemble baseline.

The analysis additionally highlighted the trade-offs between predictive accuracy and resource consumption. Boosting ensembles delivered superior predictive power but required greater computational resources, particularly in terms of execution time and memory allocation. In contrast, baseline models such as **DT** and **KNN** achieved satisfactory performance with substantially lower cost, representing viable alternatives for latency-sensitive or resource-constrained scenarios. These findings reaffirm that model selection for anomaly detection in microservices should balance accuracy, scalability, and operational efficiency depending on system requirements.

In summary, the results established ensemble models as the most reliable approach for anomaly detection in microservices, offering a robust combination of precision, recall, and stability. Classical baselines, however, remain valuable in constrained environments as lightweight detection options. The insights derived from this chapter provide the empirical foundation for the overall conclusions and future research directions presented in

Chapter 5.

These findings not only validate the suitability of ensemble learning for microservice anomaly detection but also contribute to the broader understanding of how model interpretability, scalability, and computational trade-offs shape the deployment of Artificial Intelligence (AI)-based monitoring solutions in distributed systems. This perspective highlights the relevance of integrating ML research with practical system reliability goals in modern cloud architectures.

Chapter 5

Conclusions and Future Work

This dissertation addressed the challenge of anomaly detection in microservice-based environments by evaluating and comparing baseline and ensemble ML models using the *TraceRCA* dataset. The study sought to answer three main research questions: **(1)** how ensemble algorithms perform in the classification of anomalies; **(2)** how baseline models, particularly DT-based methods, compare; and **(3)** the effect of hyperparameter optimization on model performance.

5.1 Conclusions

This dissertation investigated the application of ML models for anomaly detection in microservice-based environments, using the *TraceRCA* dataset as the foundation for experimental evaluation. The study aimed to assess the performance of both baseline and ensemble algorithms and to examine the impact of hyperparameter optimization on model behavior.

The results confirmed that ensemble-based methods consistently outperformed traditional baselines across all evaluation metrics. Boosting techniques such as LightGBM, XGBoost, and HGBM achieved the strongest overall performance, demonstrating superior accuracy, stability, and generalization capability. These models proved particularly effective for detecting anomalies in distributed microservice environments, validating the robustness of gradient-boosting approaches. The RF model also exhibited solid and reliable results with moderate computational requirements, confirming its effectiveness as a balanced ensemble alternative.

Baseline models exhibited more heterogeneous behavior. While DT and KNN achieved the most competitive results among the baselines, balancing recall and precision, their overall predictive capability remained lower than that of ensemble models. Simpler models such as LogReg and GNB displayed weaker and less stable performance, reinforcing their limitations in handling nonlinear dependencies within distributed traces. Nevertheless, these models remain valuable in resource-constrained or latency-sensitive environments, where lightweight solutions are prioritized over maximal accuracy.

The findings also highlighted the significant impact of hyperparameter optimization. Systematic tuning of parameters such as *learning rate*, *tree depth*, and *regularization* notably enhanced model generalization, particularly for ensemble methods. These results demonstrate that model configuration is not a secondary step but a decisive factor in

achieving effective and robust anomaly detection.

Overall, the results of this dissertation reinforce the superiority of ensemble-based models for anomaly detection in microservices, offering a robust combination of accuracy, stability, and scalability. At the same time, the analysis recognizes the practical relevance of simpler baselines in constrained deployment scenarios. These insights highlight the importance of balancing predictive performance and computational efficiency depending on system requirements and operational constraints. The conclusions drawn here provide a solid foundation for the discussion of future research directions presented in the Section 5.2.

5.2 Future Work

Although this dissertation provided valuable insights into the performance of ML-based approaches for anomaly detection in microservice environments, several directions remain open for exploration.

First, the study was limited to the *TraceRCA* dataset, which cannot fully capture real-world diversity. Evaluating additional datasets such as *TrainTicket* or *AIOps Challenge* would improve generalizability and verify whether the same trade-offs between accuracy and efficiency persist.

Future research could also explore modern DL architectures, including GNNs and recurrent models, which capture temporal dependencies and inter-service interactions common in distributed systems. Comparing these DL paradigms with ensemble methods would clarify which perform best under different workloads.

Another relevant direction concerns scalability and deployment. Future studies should assess how anomaly detection models behave in containerized or orchestrated environments, analyzing throughput, latency, and fault tolerance under live workloads.

Finally, integrating XAI techniques into anomaly detection pipelines would support practical adoption. Frameworks such as SHAP or LIME could provide interpretable insights into anomaly causes, increasing confidence and aiding automated fault diagnosis.

In summary, future research should broaden the methodological scope by exploring new datasets, integrating DL-based techniques, and advancing deployment and interpretability. These directions would bridge the gap between experimental evaluation and real-world applicability, enhancing the reliability and scalability of anomaly detection in microservice environments.

Bibliography

[AAV23a] Iury Araujo, Nuno Antunes, and Marco Vieira. Evaluation of machine learning for intrusion detection in Microservice applications. In *Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing (LADC '23)*, pages 1–10. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3615366.3615375. 22, 29

[AAV23b] Iury Araujo, Nuno Antunes, and Marco Vieira. Intrusion detection and tolerance for Microservice applications. In *Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing (LADC '23)*, pages 176–181. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3615366.3622794. 22, 29

[AK24] Kemal Aktaş and H. Hakan Kilinc. Interaction prediction and anomaly detection in a Microservices-based telecommunication platform. In *Proceedings of the International Conference on Software and Systems Processes (ICSSP '24)*, pages 1–10. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3666015.3666017. 24, 28

[Ama25] Amazon Web Services. The difference between monolithic and microservices architecture. https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices-architecture/, 2025. Accessed: 2025-09-30. xi, 9, 10, 11

[ASA+21] V. Arya, K. Shanmugam, P. Aggarwal, Q. Wang, P. Mohapatra, and S. Nagar. Evaluation of causal inference techniques for AIOps. In *Proceedings of the 8th ACM IKDD Conference on Data Science and 26th COMAD (CODS COMAD '21)*, pages 188–192. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3430984.3431027. 21, 29

[Ass12] Association for Computing Machinery. Acm computing classification system (ccs). https://dl.acm.org/ccs, 2012. Accessed: 2024-05-27. xiii, 2

[ATP+21] Sadie Allen, Mert Toslali, Srinivasan Parthasarathy, Fabio Oliveira, and Ayse K. Coskun. Tritium: A cross-layer analytics system for enhancing microservice rollouts in the cloud. In *Proceedings of the Workshop on Container Technologies and Container Clouds (WoC '21)*, pages 19–24. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3493649.3493656. 18, 29

[AWK25] Anton Altenbernd, Zhiyuan Wu, and Odej Kao. Amocrca: At most one change segmentation and relative correlation ranking for root cause analysis. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25)*, pages 1386–1393. Association for Computing Machinery, 2025. Available from: https://doi.org/10.1145/3696630.3731612. 20, 29

[BBdMN23] Adel Belkhiri, Ahmad Shahnejat Bushehri, Felipe Gohring de Magalhaes, and Gabriela Nicolescu. Transparent trace annotation for performance debugging in microservice-oriented systems (work in progress paper). In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion)*, pages 25–32. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3578245.3585030. 19

[BDSL21] Asma Belhadi, Youcef Djenouri, Gautam Srivastava, and Jerry Chun-Wei Lin. Reinforcement learning multi-agent system for faults diagnosis of Microservices in industrial settings. *Computer Communications*, 177:213–219, 2021. Available from: https://doi.org/10.1016/j.comcom.2021.07.010. 21, 29

[BRM+24] Alexandru Baluta, Yar Rouf, Joydeep Mukherjee, Zhen Ming Jiang, and Marin Litoiu. Disambiguating performance anomalies from workload changes in cloud-native applications. In *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24)*, pages 1–12. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3629526.3645046. 24, 29

[CAG+23] Sarthak Chakraborty, Shubham Agarwal, Shaddy Garg, Abhimanyu Sethia, Udit Narayan Pandey, Videh Aggarwal, and Shiv Saini. Esro: Experience assisted service reliability against outages. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*, pages 255–267. IEEE, 2023. Available from: https://doi.org/10.1109/ASE56229.2023.00131. 19

[CAM24] Oishik Chatterjee, Pooja Aggarwal, and Prateeti Mohapatra. Alert suppression with fine-grained and coarse-grained feedback based active learning. In *Proceedings of the 8th International Conference on Data Science and Management of Data (12th ACM IKDD CODS and 30th COMAD) (CODS-COMAD '24)*, pages 229–233. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3703323.3703724. 24

[CBVR22] Clinton Cao, Agathe Blaise, Sicco Verwer, and Filippo Rebecchi. Learning state machines to monitor and detect anomalies on a kubernetes cluster.

In *Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22)*, pages 1–9. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3538969.3543810. 18

[CCW+24] Hongyang Chen, Pengfei Chen, Benran Wang, Xian Yu, Xiaofan Chen, Dandan Ma, and Zibin Zheng. Graph neural network based robust anomaly detection at service level in sdn driven microservice system. *Computer Networks*, 239:110135, 2024. 24

[CCZ19] Wei Cao, Zhiying Cao, and Xiuguo Zhang. Research on microservice anomaly detection technology based on conditional random field. In *Journal of Physics: Conference Series*, volume 1213, page 042016. IOP Publishing, 2019. Available from: https://doi.org/10.1088/1742-6596/1213/4/042016. xi, 14, 15

[CDCP22] Marcello Cinque, Raffaele Della Corte, and Antonio Pecchia. Micro2vec: Anomaly detection in microservices systems by mining numeric representations of computer logs. *Journal of Network and Computer Applications*, 208:103515, 2022. Available from: https://doi.org/10.1016/j.jnca.2022.103515. 22, 28

[CDSL22] Ka-Ho Chow, Umesh Deshpande, Sangeetha Seshadri, and Ling Liu. Deeprest: Deep resource estimation for interactive Microservices. In *Proceedings of the Seventeenth European Conference on Computer Systems (EuroSys '22)*, pages 1–18. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3492321.3519564. 22, 28

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794. Association for Computing Machinery, 2016. Available from: https://doi.org/10.1145/2939672.2939785. 38

[CGA+23] Sarthak Chakraborty, Shaddy Garg, Shubham Agarwal, Ayush Chauhan, and Shiv Kumar Saini. Causil: Causal graph for instance level Microservice data. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*, pages 2905–2915. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3543507.3583274. 23

[CHC19] Jiyu Chen, Heqing Huang, and Hao Chen. Informer: Irregular traffic detection for containerized Microservices rpc in the real world. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (SEC '19)*, pages 379–384. Association for Computing Machinery, 2019. Available from: https://doi.org/10.1145/3318216.3363375. 21

[Che15] Lianping Chen. Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, 32(2):50–54, 2015. Available from: https://doi.org/10.1109/MS.2015.27. 9

[CHW+25] Jia Chen, Yuang He, Peng Wang, Xiaolei Chen, Jie Shi, and Wei Wang. Alert summarization for online service systems by validating propagation paths of faults. *Proceedings of the ACM on Software Engineering*, 2(FSE):FSE097:1–FSE097:23, 2025. Available from: https://doi.org/10.1145/3729367. 20

[CLJ+23] Jian Chen, Fagui Liu, Jun Jiang, Guoxiang Zhong, Dishi Xu, Zhuanglun Tan, and Shangsong Shi. Tracegra: A trace-based anomaly detection for microservice using graph deep learning. *Computer Communications*, 204:109–117, 2023. Available from: https://doi.org/10.1016/j.comcom.2023.03.028. 5, 23, 29, 31, 54

[CLS+21] Zhuangbin Chen, Jinyang Liu, Yuxin Su, Hongyu Zhang, Xuemin Wen, Xiao Ling, Yongqiang Yang, and Michael R. Lyu. Graph-based incident aggregation for large-scale online service systems. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE '21)*, pages 430–442. IEEE/ACM, 2021. Available from: https://doi.org/10.1109/ASE51524.2021.9678746. 21, 29

[CLS+22] Zhuangbin Chen, Jinyang Liu, Yuxin Su, Hongyu Zhang, Xiao Ling, Yongqiang Yang, and Michael R. Lyu. Adaptive performance anomaly detection for online service systems via pattern sketching. In *Proceedings of the 44th International Conference on Software Engineering (ICSE '22)*, pages 1031–1042. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3510003.3510085. 19, 29

[CLV22] Jessica Castro, Nuno Laranjeiro, and Marco Vieira. Detecting DoS attacks in Microservice applications: Approach and case study. In *Proceedings of the 11th Latin-American Symposium on Dependable Computing (LADC '22)*, pages 1–6. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3569902.3569916. 22, 29

[CZD+23] Yiru Chen, Chenxi Zhang, Zhen Dong, Dingyu Yang, Xin Peng, Jiayu Ou, Hong Yang, Zheshun Wu, Xiaojun Qu, and Wei Li. Dynamic graph neural networks-based alert link prediction for online service systems. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*, pages 79–90. IEEE, 2023. Available from: https://doi.org/10.1109/ASE56229.2023.00177. 23, 28

[Dd24] Andrea D'Angelo and Giordano d'Aloisio. Grammar-based anomaly detection of microservice systems execution traces. In *Companion of the*

15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), pages 77–81. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3629527.3651844. 19, 29

[DEMH+24] Giovanni Denaro, Noura El Moussa, Rahim Heydarov, Francesco Lomio, Mauro Pezzè, and Ketai Qiu. Predicting failures of autoscaling distributed applications. *Proceedings of the ACM on Software Engineering*, 1(FSE):87:1–87:22, 2024. Available from: https://doi.org/10.1145/3660794. 24, 29

[DGL+17] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: Yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering*, pages 195–216. Springer, 2017. Available from: https://doi.org/10.1007/978-3-319-67425-4_12. 9

[DLZ21] Prangshuman Das, Rodrigo Laigner, and Yongluan Zhou. Hawkeda: A tool for quantifying data integrity violations in event-driven microservices. In *Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems (DEBS '21)*, pages 176–179. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3465480.3467838. 18

[DMSB25] Kaiqi Ding, Yuanmu Ma, Zijian Song, and Kaigui Bian. Enhancing Microservices anomaly detection via multimodal data fusion in the wavelet domain and spatiotemporal graph-based diffusion probabilistic model. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '25)*. Association for Computing Machinery, 2025. Available from: https://doi.org/10.1145/3711896.3736938. 25, 28

[DZW+23] Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Xiaomin Wu, Meng Zhang, Qingjun Chen, Xin Gao, Xuedong Gao, Hao Fan, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Tracediag: Adaptive, interpretable, and efficient root cause analysis on large-scale Microservice systems. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23)*, pages 1762–1773. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3611643.3613864. 23, 29

[EKEJ24] Alireza Ezaz, Ghazal Khodabandeh, and Naser Ezzati-Jivan. Analyzing performance variability in alibaba's microservice architecture: A critical-path-based perspective. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion)*,

pages 82–86. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3629527.3651845. 19, 28

[FJRH23] Fábio Ferraz Júnior, Roseli Aparecida Francelin Romero, and Sheng-Jen Hsieh. Machine learning for the detection and diagnosis of anomalies in applications driven by electric motors. *Sensors*, 23(24):9725, 2023. Available from: https://doi.org/10.3390/s23249725. 1

[FRA+25] Duneesha Fernando, Maria A. Rodriguez, Patricia Arroba, Leila Ismail, and Rajkumar Buyya. Efficient training approaches for performance anomaly detection models in edge computing environments. *ACM Transactions on Autonomous and Adaptive Systems*, 20(2):13:1–13:27, 2025. Available from: https://doi.org/10.1145/3725736. 25, 29

[GAN+22] Dipanwita Guhathakurta, Pooja Aggarwal, Seema Nagar, Rohan Arora, and Bing Zhou. Utilizing persistence for post facto suppression of invalid anomalies using system logs. In *Proceedings of the 44th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER '22)*, pages 121–124. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3510455.3512774. 19, 29

[GGM+24] Luca Giamattei, Antonio Guerriero, Ivano Malavolta, Cristian Mascia, Roberto Pietrantuono, and Stefano Russo. Identifying performance issues in microservice architectures through causal reasoning. In *Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test (AST '24)*, pages 149–153. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3644032.3644460. 19, 29

[GHS+24] Drishti Goel, Fiza Husain, Aditya Singh, Supriyo Ghosh, Anjaly Parayil, Chetan Bansal, Xuchao Zhang, and Saravan Rajmohan. X-lifecycle learning for cloud incident management using LLMs. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24)*, pages 417–428. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3663529.3663861. 24, 29

[GLC+25] Wenwei Gu, Jinyang Liu, Zhuangbin Chen, Jianping Zhang, Yuxin Su, Jiazhen Gu, Cong Feng, Zengyin Yang, Yongqiang Yang, and Michael R. Lyu. Identifying performance issues in cloud service systems based on relational-temporal features. *ACM Transactions on Software Engineering and Methodology*, 34(3):64:1–64:31, 2025. Available from: https://doi.org/10.1145/3702978. 26, 29

[GLD+21] Yu Gan, Mingyu Liang, Sundar Dev, David Lo, and Christina Delimitrou. Sage: Practical & scalable ml-driven performance debugging in Microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21)*, pages 135–151. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3445814.3446700. 21

[GLZ+23] Yu Gan, Guiyang Liu, Xin Zhang, Qi Zhou, Jiesheng Wu, and Jiangwei Jiang. Sleuth: A trace-based root cause analysis system for large-scale microservices with graph neural networks. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '23)*, pages 324–337. Association for Computing Machinery, 2023. 23, 29

[GPW+20] Xiaofeng Guo, Xin Peng, Hanzhang Wang, Wanxue Li, Huai Jiang, Dan Ding, Tao Xie, and Liangfei Su. Graph-based trace analysis for microservice architecture understanding and problem diagnosis. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20)*, pages 1387–1397. Association for Computing Machinery, 2020. Available from: https://doi.org/10.1145/3368089.3417066. 18, 28

[GWY+24] Xuhang Gu, Qingyang Wang, Qiben Yan, Jianshu Liu, and Calton Pu. Sync-millibottleneck attack on microservices cloud architecture. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS '24)*, pages 1157–1171. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3634737.3644991. 19, 28, 29

[GZH+19] Yu Gan, Yanqi Zhang, Kelvin Hu, Dailun Cheng, Yuan He, Meghna Pancholi, and Christina Delimitrou. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud Microservices. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*, pages 19–33. Association for Computing Machinery, 2019. Available from: https://doi.org/10.1145/3297858.3304004. 21

[HBW21] Shay Horovitz, Ben Boren, and Ben Wizen. Sequencis - distributed application fault characteristics discovery using service logs. In *Proceedings of the 2021 6th International Conference on Big Data and Computing (ICBDC '21)*, pages 188–194. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3469968.3469988. 21

[HCCL23] Junfeng Hao, Peng Chen, Juan Chen, and Xi Li. Multi-task federated learning-based system anomaly detection and multi-classification for mi-

croservices architecture. *Journal of Systems Architecture*, 136:102725, 2023. Available from: https://doi.org/10.1016/j.sysarc.2022.102725. 23, 29

[HCL22] Tao Huang, Pengfei Chen, and Ruipeng Li. A semi-supervised VAE based active anomaly detection framework in multivariate time series for online systems. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*, pages 1797–1806. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3485447.3511984. 22

[HDH+24] Yongqi Han, Qingfeng Du, Ying Huang, Jiaqi Wu, Fulong Tian, and Cheng He. The potential of one-shot failure root cause analysis: Collaboration of the large language model and small classifier. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*, pages 931–943. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3691620.3695475. 24

[HER22] Adha Hrusto, Emelie Engström, and Per Runeson. Optimization of anomaly detection in a microservice system through continuous feedback from development. In *Proceedings of the 10th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS '22)*, pages 1–8. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3528229.3529382. 19, 28

[HKSG23] Zeno Heeb, Onur Kalinagac, Wissem Soussi, and Gürkan Gür. Iomirca: Root cause analysis in iot-extended 5g microservice environments. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23)*, pages 106–108. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3555776.3577840. 19, 29

[HRO24] Adha Hrusto, Per Runeson, and Magnus C. Ohlsson. Autonomous monitors for detecting failures early and reporting interpretable alerts in cloud operations. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '24)*, pages 1–11. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3639477.3639712. 24, 29

[HYY+23] Jun Huang, Yang Yang, Hang Yu, Jianguo Li, and Xiao Zheng. Twin graph-based anomaly detection via attentive multi-modal learning for microservice system. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*, pages 66–78. IEEE, 2023. Available from: https://doi.org/10.1109/ASE56229.2023.00138. 23, 28

[HZ21] Lexiang Huang and Timothy Zhu. tprof: Performance profiling via structural aggregation and automated analysis of distributed systems traces. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '21)*, pages 76–91. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3472883.3486994. 18, 28, 29

[HZA+23] Vipul Harsh, Wenxuan Zhou, Sachin Ashok, Radhika Niranjan Mysore, P. Brighten Godfrey, and Sujata Banerjee. Murphy: Performance diagnosis of distributed cloud applications. In *Proceedings of the ACM SIGCOMM 2023 Conference (SIGCOMM '23)*, pages 438–451. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3603269.3604877. 23, 28, 29

[HZC+24] Haiyu Huang, Xiaoyu Zhang, Pengfei Chen, Zilong He, Zhiming Chen, Guangba Yu, Hongyang Chen, and Chen Sun. Trastrainer: Adaptive sampling for distributed traces with system runtime state. *Proceedings of the ACM on Software Engineering*, 1(FSE):22:1–22:21, 2024. Available from: https://doi.org/10.1145/3643748. 19, 29

[IBP19] Amjad Ibrahim, Stevica Bozhinoski, and Alexander Pretschner. Attack graph generation for microservice architecture. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*, pages 1235–1242. Association for Computing Machinery, 2019. Available from: https://doi.org/10.1145/3297280.3297401. 18, 29

[JM20] Lalita J. Jagadeesan and Veena B. Mendiratta. When failure is (not) an option: Reliability models for microservices architectures. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 19–24. IEEE, 2020. Available from: https://doi.org/10.1109/ISSREW51248.2020.00031. 1

[JMW23] Zhenlan Ji, Pingchuan Ma, and Shuai Wang. Perfce: Performance debugging on databases with chaos engineering-enhanced causality analysis. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*, pages 1454–1466. IEEE, 2023. Available from: https://doi.org/10.1109/ASE56229.2023.00106. 23

[JPMW23] Xinrui Jiang, Yicheng Pan, Meng Ma, and Ping Wang. Look deep into the microservice system anomaly through very sparse logs. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*, pages 2970–2978. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3543507.3583338. 19

[JQYL22] Stephen Jacob, Yuansong Qiao, Yuhang Ye, and Brian Lee. Anomalous distributed traffic: Detecting cyber security attacks amongst Microservices using graph convolutional networks. *Computers & Security*,

118:102728, 2022. Available from: https://doi.org/10.1016/j.cose.2022.102728. 22, 28

[KAAD22] Iman Kohyarnejadfard, Daniel Aloise, Seyed Vahid Azhari, and Michel R. Dagenais. Anomaly detection in microservice environments using distributed tracing data analysis and nlp. *Journal of Cloud Computing*, 11(25):1–16, 2022. Available from: https://doi.org/10.1186/s13677-022-00296-4. 1, 12

[KMF+17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 3149–3157. Curran Associates Inc., 2017. 38

[Kra25] Krasamo. Microservices architecture for real-world business needs. https://www.krasamo.com/microservices-architecture-for-real-world-business-needs/, 2025. Accessed: 2025-09-30. xi, 10

[LCJ+21] Zeyan Li, Junjie Chen, Rui Jiao, Nengwen Zhao, Zhijun Wang, Shuwei Zhang, Yanjun Wu, Long Jiang, Leiqin Yan, Zikai Wang, Zhekang Chen, Wenchi Zhang, Xiaohui Nie, Kaixin Sui, and Dan Pei. Practical root cause localization for microservice systems via trace analysis. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10. IEEE, 2021. Available from: https://doi.org/10.1109/IWQOS52092.2021.9521340. xi, 5, 29, 31, 32, 33

[LCMGF18] Pedro Las-Casas, Jonathan Mace, Dorgival Guedes, and Rodrigo Fonseca. Weighted sampling of execution traces: Capturing more needles and less hay. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '18)*, pages 326–338. Association for Computing Machinery, 2018. Available from: https://doi.org/10.1145/3267809.3267841. 18

[LCPAM19] Pedro Las-Casas, Giorgi Papakerashvili, Vaastav Anand, and Jonathan Mace. Sifter: Scalable sampling for distributed traces, without feature engineering. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '19)*, pages 311–324. Association for Computing Machinery, 2019. Available from: https://doi.org/10.1145/3357223.3362736. 21, 28

[LDC+20] Richard Li, Min Du, Hyunseok Chang, Sarit Mukherjee, and Eric Eide. Deepstitch: Deep learning for cross-layer stitching in Microservices. In *Proceedings of the 2020 Workshop on Container Technologies and Container Clouds (WOC '20)*, pages 25–31. Association for Computing Machinery, 2020. Available from: https://doi.org/10.1145/3429885.3429965. 21, 29

[LDW+22] Richard Li, Min Du, Zheng Wang, Hyunseok Chang, Sarit Mukherjee, and Eric Eide. Longtale: Toward automatic performance anomaly explanation in Microservices. In *Proceedings of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22)*, pages 5–16. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3489525.3511675. 22, 29

[LHL+20] Guozhi Liu, Bi Huang, Zhihong Liang, Minmin Qin, Hua Zhou, and Zhang Li. Microservices: Architecture, container, and challenges. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 629–635. IEEE, 2020. Available from: https://doi.org/10.1109/QRS-C51114.2020.00107. 10

[LHP+21] Dewei Liu, Chuan He, Xin Peng, Fan Lin, Chenxi Zhang, Shengfang Gong, Ziang Li, Jiayu Ou, and Zheshun Wu. Microhecl: High-efficient root cause localization in large-scale Microservice systems. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '21)*, pages 338–348. IEEE, 2021. Available from: https://doi.org/10.1109/ICSE-SEIP52600.2021.00043. 21, 28

[LLY+22] Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. Causal inference-based root cause analysis for online service systems with intervention recognition. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*, pages 3229–3239. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3534678.3539041. 22, 28

[LMBM+20] Francesco Lomio, Diego Martínez Baselga, Sergio Moreschini, Heikki Huttunen, and Davide Taibi. Rare: A labeled dataset for cloud-native memory anomalies. In *Proceedings of the 4th ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE '20)*, pages 19–24. Association for Computing Machinery, 2020. Available from: https://doi.org/10.1145/3416505.3423560. 21

[LTW+24] Ye Li, Jian Tan, Bin Wu, Xiao He, and Feifei Li. Shapleyiq: Influence quantification by shapley values for performance debugging of microservices. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*, pages 1266–1277. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3589334.3645637. 20, 28, 29

[LYC+23a] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R. Lyu. Eadro: An end-to-end troubleshooting framework for Microservices on

multi-source data. In *Proceedings of the 45th International Conference on Software Engineering (ICSE '23)*, pages 1750–1762. IEEE/ACM, 2023. Available from: https://doi.org/10.1109/ICSE48619.2023.00150. 23, 29

[LYC+23b] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R. Lyu. Maat: Performance metric anomaly anticipation for cloud services with conditional diffusion. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*, pages 116–128. IEEE, 2023. Available from: https://doi.org/10.1109/ASE56229.2023.00082. 23, 28, 29

[LZK24] Zhengxin Li, Junfeng Zhao, and Jia Kang. Multi-source anomaly detection for Microservice systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24)*, pages 414–415. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3639478.3643535. 24, 29

[LZL+22] Zeyan Li, Nengwen Zhao, Mingjie Li, Xianglin Lu, Lixin Wang, Dongdong Chang, Xiaohui Nie, Li Cao, Wenchi Zhang, Kaixin Sui, Yanhua Wang, Xu Du, Guoqiang Duan, and Dan Pei. Actionable and interpretable fault localization for recurring failures in online service systems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22)*, pages 996–1008. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3540250.3549092. 22, 29

[MJSW21] Lun Meng, Feng Ji, Yao Sun, and Tao Wang. Detecting anomalies in microservices with execution trace comparison. *Future Generation Computer Systems*, 116:291–301, 2021. Available from: https://doi.org/10.1016/j.future.2020.10.040. 13, 18

[MLPW22] Meng Ma, Weilan Lin, Disheng Pan, and Ping Wang. Servicerank: Root cause identification of anomaly in large-scale microservice architectures. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3087–3100, 2022. 54

[MST+23] Chunyang Meng, Shijie Song, Haogang Tong, Maolin Pan, and Yang Yu. Deepscaler: Holistic autoscaling for Microservices based on spatiotemporal GNN with adaptive graph learning. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*, pages 362–374. IEEE, 2023. Available from: https://doi.org/10.1109/ASE56229.2023.00038. 23, 28, 29

[MWX+20] Meng Ma, Ping Wang, Jingmin Xu, Yuan Wang, Pengfei Chen, and Zonghua Zhang. Automap: Diagnose your microservice-based web applications automatically. In *Proceedings of The Web Conference 2020 (WWW '20)*, pages 246–258. Association for Computing Machinery, 2020. Available from: https://doi.org/10.1145/3366423.3380111. 18

[MYG+24] Markos Markakis, Brit Youngmann, Trinity Gao, Ziyu Zhang, Rana Shahout, Peter Baile Chen, Chunwei Liu, Ibrahim Sabek, and Michael Cafarella. From logs to causal inference: Diagnosing large systems. *Proceedings of the VLDB Endowment*, 18(2):158–172, 2024. Available from: https://doi.org/10.14778/3705829.3705836. 20

[NPR23] João Nobre, E. J. Solteiro Pires, and Arsénio Reis. Anomaly detection in microservice-based systems. *Applied Sciences*, 13(13):7891, 2023. Available from: https://doi.org/10.3390/app13137891. 1

[NTND21] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. Machine learning for anomaly detection: A systematic review. *IEEE Access*, 9:78658–78700, 2021. Available from: https://doi.org/10.1109/ACCESS.2021.3083060. 1

[NVP21] Francisco Neves, Ricardo Vilaça, and José Pereira. Detailed black-box monitoring of distributed systems. *ACM Applied Computing Review*, 21(1):24–36, 2021. Available from: https://doi.org/10.1145/3477133.3477135. 18, 28, 29

[PCLH21] Jinwoo Park, Byungkwon Choi, Chunghan Lee, and Dongsu Han. Graf: A graph neural network based proactive resource allocation framework for slo-oriented Microservices. In *Proceedings of the 17th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '21)*, pages 154–167. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3485983.3494866. 21, 28, 29

[PEJHLM24a] Mahsa Panahandeh, Naser Ezzati-Jivan, Abdelwahab Hamou-Lhadj, and James Miller. Context-aware root cause localization in distributed traces using social network analysis (work in progress paper). In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion)*, page 6. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3629527.3651426. 20, 28

[PEJHLM24b] Mahsa Panahandeh, Naser Ezzati-Jivan, Abdelwahab Hamou-Lhadj, and James Miller. Efficient unsupervised latency culprit ranking in distributed traces with GNN and critical path analysis. In *Companion of*

the 15th ACM/SPEC International Conference on Performance Engineering (ICPE Companion '24), pages 62–66. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3629527.3651841. 25

[PHLHM24] Mahsa Panahandeh, Abdelwahab Hamou-Lhadj, Mohammad Hamdaqa, and James Miller. Serviceanomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics. *Journal of Systems and Software*, 209:111917, 2024. Available from: https://doi.org/10.1016/j.jss.2024.111917. 20, 29

[PHZ24a] Luan Pham, Huong Ha, and Hongyu Zhang. Baro: Robust root cause analysis for microservices via multivariate bayesian online change point detection. *Proceedings of the ACM on Software Engineering*, 1(FSE):98:1–98:24, 2024. Available from: https://doi.org/10.1145/3660805. 20, 29

[PHZ24b] Luan Pham, Huong Ha, and Hongyu Zhang. Root cause analysis for microservices based on causal inference: How far are we? In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*, pages 1181–1193. IEEE, 2024. Available from: https://doi.org/10.1109/ASE56229.2024.00096. 20

[PMJW21] Yicheng Pan, Meng Ma, Xinrui Jiang, and Ping Wang. Faster, deeper, easier: Crowdsourcing diagnosis of microservice kernel failure from user space. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21)*, pages 646–657. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3460319.3464805. 18

[POVTAC24] Maria Katherine Plazas Olaya, Jaime Alberto Vergara Tejada, and Jose Edinson Aedo Cobo. Securing Microservices-based IoT networks: Real-time anomaly detection using machine learning. *Journal of Computer Networks and Communications*, 2024:9281529, 2024. Available from: https://doi.org/10.1155/2024/9281529. 24

[PRI20] PRISMA Statement. Prisma 2020 flow diagram. https://www.prisma-statement.org/prisma-2020-flow-diagram, 2020. Accessed: 2025-09-01. xi, 16, 17

[PSCvdH21] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 54(2):38:1–38:38, 2021. 12

[PTRC07] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for informa-

tion systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007. Available from: https://doi.org/10.2753/MIS0742-1222240302. xi, 4

[PWL+25] Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, Gaogang Xie, and Dan Pei. Flow-of-action: Sop enhanced LLM-based multi-agent system for root cause analysis. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25)*, pages 422–431. Association for Computing Machinery, 2025. Available from: https://doi.org/10.1145/3701716.3715225. 26, 29

[PZJ+24] Yicheng Pan, Yifan Zhang, Xinrui Jiang, Meng Ma, and Ping Wang. Effcause: Discover dynamic causal relationships efficiently from time-series. *ACM Transactions on Knowledge Discovery from Data*, 18(5):105:1–105:21, 2024. Available from: https://doi.org/10.1145/3640818. 20

[RAS+23] Raphael Rouf, Harit Ahuja, Komal Sarda, Mohammadreza Rasolroveicy, Zakeya Namrud, Marin Litoiu, and Ian Watts. Models for detecting performance anomalies and identifying root causes in Microservices applications. In *Proceedings of the Annual International Conference on Computer Science and Software Engineering (CASCON '23)*, pages 1–3. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3615366.3615375. 23

[RIMF24] Vinícius de Miranda Rios, Pedro R. M. Inácio, Damien Magoni, and Mário M. Freire. Detection of slowloris attacks using machine learning algorithms. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24)*, pages 1321–1330. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3605098.3635919. 6

[RRL+24] Raphael Rouf, Mohammadreza Rasolroveicy, Marin Litoiu, Seema Nagar, Prateeti Mohapatra, Pranjal Gupta, and Ian Watts. Instantops: A joint approach to system failure prediction and root cause identification in Microservices cloud-native applications. In *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24)*, pages 119–129. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3629526.3645047. 25, 29

[RYY+24] Rui Ren, Jingbang Yang, Linxiao Yang, Xinyue Gu, and Liang Sun. SLIM: A scalable and interpretable light-weight fault localization algorithm for imbalanced data in microservice. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*, pages 1–13. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3691620.3694984. 25, 29

[RZB+24] Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. Exploring LLM-based agents for root cause analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24)*, pages 208–219. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3663529.3663841. 25, 29

[SB22] Jacopo Soldani and Antonio Brogi. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. *ACM Comput. Surv.*, 55(3):1–39, 2022. 1, 2

[sci25a] scikit-learn developers. Scikit-learn: Supervised learning. https://scikit-learn.org/stable/supervised_learning.html, 2025. Accessed: 2025-08-01. 5, 35, 38, 41, 45

[sci25b] scikit-learn developers. sklearn.model_selection. https://scikit-learn.org/stable/api/sklearn.model_selection.html, 2025. Accessed: 2025-09-02. 41

[SCL+19] Huasong Shan, Yuan Chen, Haifeng Liu, Yunpeng Zhang, Xiao Xiao, Xiaofeng He, Min Li, and Wei Ding. □-diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In *Proceedings of the 2019 World Wide Web Conference (WWW '19)*, pages 3215–3222. Association for Computing Machinery, 2019. Available from: https://doi.org/10.1145/3308558.3313653. 18, 29

[SDA+24] Gagan Somashekar, Anurag Dutt, Mainak Adak, Tania Lorido Botran, and Anshul Gandhi. GAMMA: Graph neural network-based multi-bottleneck localization for Microservices applications. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*, pages 3085–3095. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3589334.3645665. 25, 28

[SDV+22] Gagan Somashekar, Anurag Dutt, Rohith Vaddavalli, Sai Bhargav Varanasi, and Anshul Gandhi. B-meg: Bottlenecked-Microservices extraction using graph neural networks. In *Companion of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22 Companion)*, pages 7–11. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3491204.3527494. 22, 28, 29

[SFRB25] Jacopo Soldani, Stefano Forti, Luca Roveroni, and Antonio Brogi. Explaining microservices' cascading failures from their logs. *Software: Practice and Experience*, 55(5):809–828, 2025. 20, 28, 29

[SLS+25] Yongqian Sun, Zihan Lin, Binpeng Shi, Shenglin Zhang, Shiyu Ma, Pengxiang Jin, Zhenyu Zhong, Lemeng Pan, Yicheng Guo, and Dan Pei. Interpretable failure localization for Microservice systems based on graph autoencoder. *ACM Transactions on Software Engineering and Methodology*, 34(2):52:1–52:28, 2025. Available from: https://doi.org/10.1145/3695999. 26

[SMA+22] William Sussman, Emily Marx, Venkat Arun, Akshay Narayan, Mohammad Alizadeh, Hari Balakrishnan, Aurojit Panda, and Scott Shenker. The case for an internet primitive for fault localization. In *Proceedings of the 20th ACM Workshop on Hot Topics in Networks (HotNets '22)*, pages 1–7. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3563766.3564105. 19

[SNR+23] Komal Sarda, Zakeya Namrud, Raphael Rouf, Harit Ahuja, Mohammadreza Rasolroveicy, Marin Litoiu, Larisa Shwartz, and Ian Watts. Adarma: Auto-detection and auto-remediation of Microservice anomalies by leveraging large language models. In *Proceedings of the Annual International Conference on Computer Science and Software Engineering (CASCON '23)*, pages 1–6. Association for Computing Machinery, 2023. Available from: https://dl.acm.org/doi/10.5555/3615366.3615412. 23, 29

[SSM+24] Yongqian Sun, Binpeng Shi, Mingyu Mao, Minghua Ma, Sibo Xia, Shenglin Zhang, and Dan Pei. ART: A unified unsupervised framework for incident management in Microservice systems. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*, pages 1183–1194. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3691620.3695495. 25

[SSS+22] Davide Sanvito, Giuseppe Siracusano, Sharan Santhanam, Roberto Gonzalez, and Roberto Bifulco. syslrn: Learning what to monitor for efficient anomaly detection. In *Proceedings of the 2nd European Workshop on Machine Learning and Systems (EuroMLSys '22)*, pages 64–71. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3517207.3526979. 22

[SZX+23] Junxian Shen, Han Zhang, Yang Xiang, Xingang Shi, Xinrui Li, Yunxi Shen, Zijian Zhang, Yongxiang Wu, Xia Yin, Jilong Wang, Mingwei Xu, Yahui Li, Jiping Yin, Jianchang Song, Zhuofeng Li, and Runjie Nie. Network-centric distributed tracing with deepflow: Troubleshooting your microservices in zero code. In *Proceedings of the ACM SIGCOMM 2023 Conference (SIGCOMM '23)*, pages 420–437. Association for Computing

Machinery, 2023. Available from: https://doi.org/10.1145/3603269.3604823. 19, 28

[TGYC23] Hao Tang, Yuchun Guo, Jingjing Yang, and Yishuai Chen. Microservice anomaly diagnosis with graph convolution network based on implicit Microservice dependency. In *Proceedings of the 2023 9th International Conference on Computing and Artificial Intelligence (ICCAI '23)*, pages 437–441. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3594315.3594354. 23, 28

[THT+19] Chin-Wei Tien, Tse-Yung Huang, Chia-Wei Tien, Ting-Chun Huang, and Sy-Yen Kuo. Kubanomaly: Anomaly detection for the docker orchestration platform with neural network approaches. *Engineering Reports*, 1(5):e12080, 2019. Available from: https://doi.org/10.1002/eng2.12080. 21, 29

[TLSDM24] Luca Traini, Jessica Leone, Giovanni Stilo, and Antinisca Di Marco. Vamp: Visual analytics for microservices performance. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24)*, pages Article 4, 10 pages. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3605098.3636069. 20, 29

[TRA+17] Jörg Thalheim, Antonio Rodrigues, Istemi Ekin Akkus, Pramod Bhatotia, Ruichuan Chen, Bimal Viswanath, Lei Jiao, and Christof Fetzer. Sieve: Actionable insights from monitored metrics in distributed systems. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, pages 1–14. Association for Computing Machinery, 2017. Available from: https://doi.org/10.1145/3135974.3135977. 18

[TZJ+24] Lei Tao, Shenglin Zhang, Zedong Jia, Jinrui Sun, Minghua Ma, Zhengdan Li, Yongqian Sun, Canqun Yang, Yuzhi Zhang, and Dan Pei. Giving every modality a voice in Microservice failure diagnosis via multimodal adaptive optimization. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*, pages 1107–1119. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3691620.3695489. 25, 28

[WCF+23] Dongjie Wang, Zhengzhang Chen, Yanjie Fu, Yanchi Liu, and Haifeng Chen. Incremental causal graph learning for online root cause analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, pages 2350–2360. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3580305.3599371. 19, 29

[WCN+23] Dongjie Wang, Zhengzhang Chen, Jingchao Ni, Tong Liang, Zheng Wang, Yanjie Fu, and Haifeng Chen. Interdependent causal networks for root

cause localization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, pages 5051–5060. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3580305.3599849. 23, 29

[WHL+24] Chen Wang, Tao Huang, Min Li, Pengfei Chen, and Zhiwen Chen. A bayesian LSTM based active anomaly detection service for large online systems. In *Proceedings of the 15th Asia-Pacific Symposium on Internetware (Internetware '24)*, pages 1–10. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3671016.3674818. 25, 29

[WMD+24] Yuqing Wang, Mika V. Mäntylä, Serge Demeyer, Mutlu Beyazit, Joanna Kisaakye, and Jesse Nyyssölä. Few-shot cross-system anomaly trace classification for microservice-based systems. *arXiv preprint arXiv:2403.18998*, 2024. Available from: https://arxiv.org/abs/2403.18998. 1

[WMD+25] Yuqing Wang, Mika V. Mäntylä, Serge Demeyer, Mutlu Beyazıt, Joanna Kisaakye, and Jesse Nyssölä. Cross-system categorization of abnormal traces in Microservice-based systems via meta-learning. *Proceedings of the ACM on Software Engineering*, 2(FSE):27:1–27:23, 2025. Available from: https://doi.org/10.1145/3715742. 26, 29

[WRXY20] Hao Wang, Guoping Rong, Yangchen Xu, and Yong You. Impaptr: A tool for identifying the clues to online service anomalies. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20)*, pages 1307–1311. Association for Computing Machinery, 2020. Available from: https://doi.org/10.1145/3324884.3415301. 18, 29

[WSY+25] Xinjie Wei, Chang-ai Sun, Pengpeng Yang, Xiao-Yi Zhang, and Dave Towey. Traloganomaly: A Microservice system anomaly detection approach based on hybrid event sequences. *Science of Computer Programming*, 245:103303, 2025. Available from: https://doi.org/10.1016/j.scico.2025.103303. 26, 29

[WXM+18] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. Cloudranger: Root cause identification for cloud native systems. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 492–502. IEEE, 2018. Available from: https://doi.org/10.1109/CCGRID.2018.00076. 18

[WZC25] Peipeng Wang, Xiuguo Zhang, and Zhiying Cao. Anomaly detection for microservice system via augmented multimodal data and hybrid graph

representations. *Information Fusion*, 118:103017, 2025. Available from: https://doi.org/10.1016/j.inffus.2025.103017. 26, 28

[WZCC25] Peipeng Wang, Xiuguo Zhang, Yutian Chen, and Zhiying Cao. Unsupervised Microservice system anomaly detection via contrastive multimodal representation clustering. *Information Processing & Management*, 62:104013, 2025. Available from: https://doi.org/10.1016/j.ipm.2024.104013. 26, 28

[WZD+23] Lu Wang, Chaoyun Zhang, Ruomeng Ding, Yong Xu, Qihang Chen, Wentao Zou, Qingjun Chen, Meng Zhang, Xuedong Gao, Hao Fan, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Root cause analysis for Microservice systems via hierarchical reinforcement learning from human feedback. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, pages 5116–5125. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3580305.3599934. 23

[WZF+24] Yidan Wang, Zhouruixing Zhu, Qiuai Fu, Yuchi Ma, and Pinjia He. MRCA: Metric-level root cause analysis for Microservices via multi-modal data. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*, pages 1057–1068. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3691620.3695485. 25, 29

[WZL+22] Ziliang Wang, Shiyi Zhu, Jianguo Li, Wei Jiang, K. K. Ramakrishnan, Yangfei Zheng, Meng Yan, Xiaohong Zhang, and Alex X. Liu. Deepscaling: Microservices autoscaling for stable cpu utilization in large scale cloud systems. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '22)*, pages 16:1–16:18. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3542929.3563469. 22, 28

[XGL+25] Ke Xv, Shikai Guo, Hui Li, Chenchen Li, Rong Chen, Xiaochen Li, and He Jiang. Making fault localization in online service systems more actionable and interpretable. *ACM Transactions on Software Engineering and Methodology*, 34(6):165:1–165:??, 2025. Available from: https://doi.org/10.1145/3714466. 26, 29

[XPL+23] Zhe Xie, Changhua Pei, Wanxue Li, Huai Jiang, Liangfei Su, Jianhui Li, Gaogang Xie, and Dan Pei. From point-wise to group-wise: A fast and accurate Microservice trace anomaly detection approach. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23)*, pages 1739–1750. Association for Computing Machinery, 2023.

Available from: https://doi.org/10.1145/3611643.3613861. 24, 28, 33

[XWH+25] Shuaiyu Xie, Jian Wang, Hanbin He, Zhihao Wang, Yuqi Zhao, Neng Zhang, and Bing Li. Tvdiag: A task-oriented and view-invariant failure diagnosis framework for microservice-based systems with multimodal data. *ACM Transactions on Software Engineering and Methodology*, 2025. Available from: https://doi.org/10.1145/3734868. 26, 28, 29

[XXC+23] Zhe Xie, Haowen Xu, Wenxiao Chen, Wanxue Li, Huai Jiang, Liangfei Su, Hanzhang Wang, and Dan Pei. Unsupervised anomaly detection on Microservice traces through graph VAE. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*, pages 2874–2884. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3543507.3583215. 24

[XZG+24] Zhe Xie, Shenglin Zhang, Yitong Geng, Yao Zhang, Minghua Ma, Xiaohui Nie, Zhenhe Yao, Longlong Xu, Yongqian Sun, Wentao Li, and Dan Pei. Microservice root cause analysis with limited observability through intervention recognition in the latent space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, pages 6049–6060. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3637528.3671530. 25, 28, 29

[YCC+21] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinmeng Sun, and Xiaoyun Li. Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In *Proceedings of the Web Conference 2021 (WWW '21)*, pages 3087–3098. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3442381.3449905. 18, 28

[YCH+24] Guangba Yu, Pengfei Chen, Zilong He, Qiuyu Yan, Yu Luo, Fangyuan Li, and Zibin Zheng. Changerca: Finding root causes from software changes in large online systems. *Proceedings of the ACM on Software Engineering*, 1(FSE):2:1–2:23, 2024. Available from: https://doi.org/10.1145/3643728. 20, 29

[YCL+23] Guangba Yu, Pengfei Chen, Yufeng Li, Hongyang Chen, Xiaoyun Li, and Zibin Zheng. Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23)*, page 13. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3611643.3616249. 19, 29

[YHD+25] Xixuan Yang, Xin Huang, Chiming Duan, Tong Jia, Shandong Dong, Ying Li, and Gang Huang. Enhancing web service anomaly detection via fine-grained multi-modal association and frequency domain analysis. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25)*, pages 548–556. Association for Computing Machinery, 2025. Available from: https://doi.org/10.1145/3701716.3715221. 26, 28, 29

[YPC+24] Zhenhe Yao, Changhua Pei, Wenxiao Chen, Hanzhang Wang, Liangfei Su, Huai Jiang, Zhe Xie, Xiaohui Nie, and Dan Pei. Chain-of-event: Interpretable root cause analysis for Microservices through automatically learning weighted event causal graph. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24)*, pages 50–61. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3663529.3663827. 25

[YPH+23] Qingyang Yu, Changhua Pei, Bowen Hao, Mingjie Li, Zeyan Li, Shenglin Zhang, Xianglin Lu, Rui Wang, Jiaqi Li, Zhenyu Wu, and Dan Pei. Cmdiagnostor: An ambiguity-aware root cause localization approach based on call metric data. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*, pages 2937–2947. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3543507.3583302. 19, 29

[YSS+21] Tianyi Yang, Jiacheng Shen, Yuxin Su, Xiao Ling, Yongqiang Yang, and Michael R. Lyu. Aid: Efficient prediction of aggregated intensity of dependency in large-scale cloud systems. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE '21)*, pages 653–665. IEEE, 2021. Available from: https://doi.org/10.1109/ASE51524.2021.9678534. 18, 29

[YWY+24] Dong Young Yoon, Yang Wang, Miao Yu, Elvis Huang, Juan Ignacio Jones, Abhinay Kukkadapu, Osman Kocas, Jonathan Wiepert, Kapil Goenka, Sherry Chen, Yanjun Lin, Zhihui Huang, Jocelyn Kong, Michael Chow, and Chunqiang Tang. Fbdetect: Catching tiny performance regressions at hyperscale through in-production monitoring. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles (SOSP '24)*. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3694715.3695977. 20, 29

[ZAD+24] Lei Zan, Charles K. Assaad, Emilie Devijver, Eric Gaussier, and Ali Aït-Bachir. On the fly detection of root causes from observed data with application to it systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*,

pages 5062–5070. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3627673.3680010. 20

[ZCHC24] Lecheng Zheng, Zhengzhang Chen, Jingrui He, and Haifeng Chen. MULAN: Multi-modal causal structure learning and root cause analysis for Microservice systems. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*, pages 4107–4116. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3589334.3645442. 25, 29

[ZCY+21] Nengwen Zhao, Junjie Chen, Zhaoyang Yu, Honglin Wang, Jiesong Li, Bin Qiu, Hongyu Xu, Wenchi Zhang, Kaixin Sui, and Dan Pei. Identifying bad software changes via multimodal anomaly detection for online service systems. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21)*, pages 527–539. Association for Computing Machinery, 2021. Available from: https://doi.org/10.1145/3468264.3468543. 22, 29

[ZDP+24] Chenxi Zhang, Zhen Dong, Xin Peng, Bicheng Zhang, and Miao Chen. Trace-based multi-dimensional root cause localization of performance issues in microservice systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*, pages 1031–1042. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3597503.3639088. 20, 29

[ZGB+24] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. Automated root causing of cloud incidents using in-context learning with GPT-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24)*, pages 266–277. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3663529.3663846. 25, 28

[ZIY+23] Yazhuo Zhang, Rebecca Isaacs, Yao Yue, Juncheng Yang, Lei Zhang, and Ymir Vigfusson. Latenseer: Causal modeling of end-to-end latency distributions by harnessing distributed tracing. In *Proceedings of the 14th ACM Symposium on Cloud Computing (SoCC '23)*, pages 502–519. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3620678.3624787. 19, 28

[ZLTH24] Zhouruixing Zhu, Cheryl Lee, Xiaoying Tang, and Pinjia He. Hemirca: Fine-grained root cause analysis for Microservices with heterogeneous data sources. *ACM Transactions on Software Engineering and Methodology*, 33(8):200:1–200:25, 2024. Available from: https://doi.org/10.1145/3674726. 25, 29

[ZMZ+23] Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang, Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, Yuzhi Zhang, Dan Pei, Qingwei Lin, and Dongmei Zhang. Robust multimodal failure detection for Microservice systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, pages 5639–5649. Association for Computing Machinery, 2023. Available from: https://doi.org/10.1145/3580305.3599902. 24, 28

[ZPS+22] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, Qingwei Lin, and Dongmei Zhang. Deeptralog: Trace-log combined Microservice anomaly detection through graph-based deep learning. In *Proceedings of the 44th International Conference on Software Engineering (ICSE '22)*, pages 623–634. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3510003.3510180. 22, 29

[ZPX+19] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chao Ji, Dewei Liu, Qilin Xiang, and Chuan He. Latent error prediction and fault localization for Microservice applications by learning from system trace logs. In *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19)*, pages 683–694. Association for Computing Machinery, 2019. Available from: https://doi.org/10.1145/3338906.3338961. 21, 29

[ZPZ+22] Chenxi Zhang, Xin Peng, Tong Zhou, Chaofeng Sha, Zhenghui Yan, Yiru Chen, and Hong Yang. Tracecrl: Contrastive representation learning for Microservice trace analysis. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22)*, pages 1221–1232. Association for Computing Machinery, 2022. Available from: https://doi.org/10.1145/3540250.3549146. 22, 29

[ZZH+24] Shenglin Zhang, Jun Zhu, Bowen Hao, Yongqian Sun, Xiaohui Nie, Jingwen Zhu, Xilin Liu, Xiaoqian Li, Yuchi Ma, and Dan Pei. Fault diagnosis for test alarms in Microservices through multi-source data. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24)*, pages 115–125. Association for Computing Machinery, 2024. Available from: https://doi.org/10.1145/3663529.3663833. 25, 29

[ZZX+23] Zhengran Zeng, Yuqun Zhang, Yong Xu, Minghua Ma, Bo Qiao, Wentao Zou, Qingjun Chen, Meng Zhang, Xu Zhang, Hongyu Zhang, Xuedong Gao, Hao Fan, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Traceark: Towards actionable performance anomaly alerting for online service systems. In *Proceedings of the IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice*

(ICSE-SEIP '23), pages 258–269. IEEE, 2023. Available from: https://doi.org/10.1109/ICSE-SEIP58684.2023.00029. 24, 29