LEVERAGING USER AND ENTITY BEHAVIORAL ANALYSIS AND MACHINE LEARNING FOR LOG-BASED ANOMALY DETECTION

by

SHARIE R NATH, M.S Data Science

DISSERTATION

Presented to the Swiss School of Business and Management Geneva

In Partial Fulfillment

Of the Requirements

For the Degree

DOCTOR OF BUSINESS ADMINISTRATION

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

SEPTEMBER, 2024

LEVERAGING USER AND ENTITY BEHAVIORAL ANALYSIS AND MACHINE LEARNING FOR LOG-BASED ANOMALY DETECTION

by

SHARIE R NATH

Supervised by

Dr.MARIO SILIC

APPROVED BY

Anna Provodnikova, PhD

Dissertation chair

RECEIVED/APPROVED BY:

Admissions Director

Dedication

This thesis is dedicated to the field of cybersecurity and machine learning researchers whose relentless innovation and contributions have advanced this vital area. Your commitment to enhancing digital security through research and technology has been a profound source of inspiration.

Acknowledgments

I express my deepest gratitude to my thesis mentor**, Dr. Mario Silic** for his expertise, guidance, and unwavering support, which have been instrumental in the completion of this research. Your insightful feedback and encouragement have significantly contributed to my academic and personal growth.

I also extend my sincere appreciation to **Dr. Sudan Jha** for your valuable insights and constant support. Your dedication and enthusiasm have greatly enriched this work.

Additionally, I thank **Upgrad** and **SSBM** for making my dream come true.

Finally, I am grateful to my family, friends, and colleagues for their encouragement and support during this challenging process. Your belief in me has been a tremendous source of strength.

Thank you all for making this achievement possible..

ABSTRACT

LEVERAGING USER AND ENTITY BEHAVIORAL ANALYSIS AND MACHINE LEARNING FOR LOG-BASED ANOMALY DETECTION

SHARIE R NATH
2024

Dissertation Chair: <Chair's Name>
Co-Chair: <If applicable. Co-Chair's Name>

As enterprise businesses increasingly migrate to the cloud for enhanced scalability and simplified management, the demand for robust network observability through log analysis has surged. Security Information and Event Management (SIEM) systems are pivotal in this landscape, ensuring smooth operations by analyzing log files. Integrating User and Entity Behavior Analytics (UEBA) with SIEM systems enhances reliability, rapidly identifies abnormal activities, and minimizes potential damage, thus achieving critical business objectives. Despite the promise of UEBA, significant challenges persist in a cloud-based environment, including the analysis of massive log volumes, high false alarm rates, and resource constraints.

The study's objectives include preprocessing structured logs for exploratory data analysis, assigning risk scores using UEBA, and identifying anomalies through machine learning techniques. This approach aims to create a simple yet effective UEBA and ML-based Log Anomaly Detection system that is affordable for small and medium enterprises (SMEs). The research addresses notable gaps in behavioral analysis for log anomaly detection, emphasizing

the need for automated log preprocessing methods to efficiently manage large and complex log data volumes.

This research delves into the existing literature on UEBA-based log anomaly detection within the cloud ecosystem, evaluating computational and performance efficacy using the BGL Dataset. Various machine learning models—such as XGBoost, Random Forest, Neural Networks, and Isolation Forest—are compared to identify a model that optimally balances false alarm rates and computational time. The results indicate that while UEBA techniques generally increase training times, they significantly reduce prediction times, thereby enhancing real-time performance. Among the models studied, XGBoost emerges as the optimal choice due to its high performance and computational efficiency.

In summary, this thesis presents a critical review of UEBA-supported log anomaly detection, aiming to identify performance improvements and optimal machine learning models for superior anomaly detection. This research highlights the necessity for a benchmark UEBA-based log anomaly dataset, further contributing to the advancement of the research community in this domain. This research highlights the potential of UEBA-enhanced machine learning models to provide reliable, efficient, and cost-effective solutions for anomaly detection, particularly benefiting SMEs with limited resources. The findings enable businesses to design robust cloud log-based SIEM systems that achieve significant cost savings, reduce false positives, improve threat response, and proactively mitigate evolving cybersecurity threats, ultimately protecting assets while maintaining stakeholder trust.

KEYWORDS: UEBA, Log Anomaly Detection, Machine Learning, BGL Dataset, Entity Score, SIEM, Cloud Migration

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
ANN	Artificial Neural Networks
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
BGL	BlueGene/L
CNN	Convolutional Neural Network
DL	Deep Learning
DT	Decision Tree
GAN	Generative Adversarial Network)
IDS	Intrusion Detection System
IF	Isolation Forest
LSTM	Long Short-Term Memory
ML	Machine Learning
NB	Naive Bayes
NLP	Natural Language Processing
NN	Neural Networks
RF	Random Forest
RNN	Recurrent Neural Network
SIEM	Security Information and Event Management
SVM	Support Vector Machine
TRA	Theory of Reasoned Action
UEBA	User and Entity Behavioral Analysis

CHAPTER I:

INTRODUCTION

1.1 Introduction

Enterprise business solutions are increasingly migrating to the cloud ecosystem owing to its scalability and simplified management. As the dependency on cloud storage for deploying critical applications and services is increasing, the demand for network observability (Sumologic, 2020) also increases for maintaining reliability and transparency. Log files are the main source of network observability as they contain information about usage, operating system events, and activities. Log analysis plays an important role in implementing Security Information and Event Management systems (SIEM) for any enterprise business to function smoothly.

Logs are auto-generated records that capture computational system events. It contains information regarding system processes, running applications, network accesses, resource allocation, usage behaviors, and error and warning events related to the system. By reviewing the logs system administrators or security analysts can troubleshoot and figure out the root cause of an issue, carry out preventive maintenance, or identify abnormalities. (Yadav, Kumar and Dhavale, 2020). The log analysis helps understand system behavior, malfunctioning detection, security scanning, and failure prediction.

However, the emergence of big data presents challenges in analyzing and detecting anomalies in logs due to the vast volumes of unstructured and varied messages gathered from diverse sources. (Liu et al., 2018; Landauer et al., 2020). Extensive preprocessing and domain knowledge is required for manipulating system logs into a structured entity which involves steps such as overview and filtering, log parsing and extraction of signature, static outlier detection, and sequences and dynamic anomaly detection. Machine learning (ML) and Deep Learning (DL) methods have been proven potent tools for data classification problems and have been applied to various fields of research. Various studies have been conducted successfully using ML and DL

techniques such as NLP, BERT, LSTM, RNN, etc for preprocessing, analysis, and anomaly detection from system logs.

Security Information and Event Management systems (SIEM) play several critical roles within organizations, functioning as a central repository for compliance records, audit trails, and forensic data. They also serve as a monitoring platform for pertinent security alerts and data, establishing a unified and real-time source of prioritized alerts throughout an organization. The present generation of SIEM solutions employs diverse analysis methods such as correlation, statistical deviation, and machine learning to recognize potential threats and other noteworthy events. Their objective is to empower enterprises to convert raw alert data into actionable intelligence, utilizing the most effective analysis method based on specific monitoring goals.

Recently several vendors like Splunk Technologies, Varonis, and Force Points started a new trend of integrating User and Behavior Analytics (UEBA) with SIEM to improve reliability and achieve real business objectives. Traditional antivirus and firewall software scans the file system for any indication of attacks or infection. UEBA systems on the other hand mainly focus on user activities both normal and abnormal, which resources or entities are being used, how frequently being retrieved, who has done and when, and what has been done through behavioral profiling, risk scoring, and timeline analysis.

The Gartner Market Guide for User and Entity Behavior Analytics (UEBA) (Market Guide for User and Entity Behavior Analytics) highlights a trend where few vendors offer standalone UEBA technologies, while the majority integrate UEBA as a service within existing cybersecurity products. Notable standalone UEBA solutions include Aruba IntroSpect, Fortinet FortiInsight, Gurucul Risk Analytics, Splunk-Caspida, and Securonix. These solutions employ advanced machine learning, behavior analytics, and risk scoring for threat detection, offering features such as account takeover detection, regulatory compliance support, and forensic-level reporting. However, most of the SIEM solutions like Splunk(What is User Behavior Analytics (UBA) and User Entity Behavior Analytics (UEBA) | Splunk) deploy anomaly-based or event or instance-based risk scoring. A new trend is set by UEBA solutions like Aruba IntroSpect and

Microsoft Sentil where user or entity-based Risk score calculation is deployed for enhanced SIEM capabilities and cyber attack defence. Quantifying abnormal behavior scores and providing contextualization are crucial for UEBA solutions, requiring the application of data science and machine learning in cybersecurity.

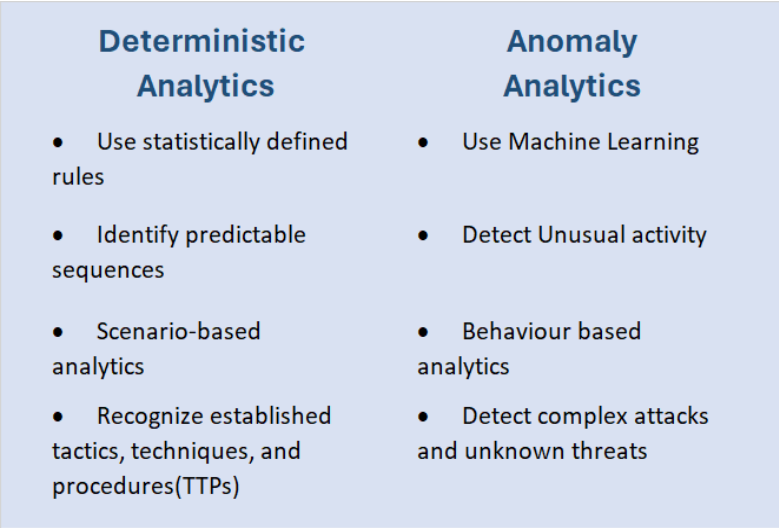


Figure 1-1 Deterministic Vs Anomaly Analytics

The evolving threat landscape necessitates a holistic analytics approach for effective defense against complex attacks. User and Entity Behavior Analytics (UEBA) solutions employ multiple analytical approaches such as deterministic analytics and anomaly analytics (‘A Guide to User and Entity Behavior Analytics (UEBA) ') to effectively defend against attacks. Deterministic or Scenario-based analytics rely on predefined logic rules to identify predictable attack patterns in real-time. Scenario-based analytics recognize Tactics, Techniques, and Procedures (TTPs), while Anomaly analytics, utilize machine learning to detect unusual activity. Anomaly analytics uses behavioral profiling and risk scoring to enhance anomaly detection. Figure 1-1 compares deterministic vs. anomaly analytics.

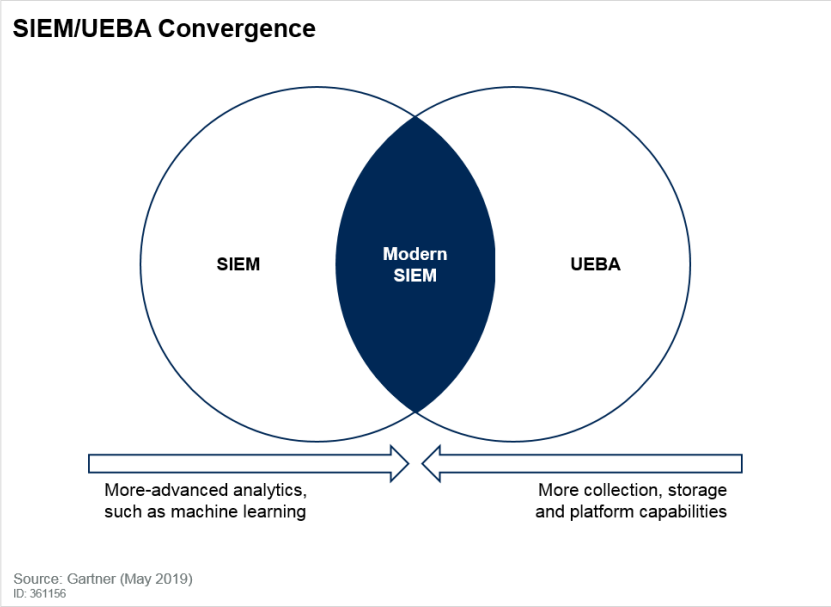


Figure 1-2 SIEM / UEBA Convergence

Source : Gartner(2019)

UEBA techniques by themselves could not prevent threats and attacks in an enterprise system, but when incorporated with state-of-the-art detection techniques, could help detect suspicious activity across the entire spectrum of complex threats instantly minimizing damage and thereby increasing revenue. Also, UEBA-based SIEM helps in reducing false alarms in anomaly detection by profiling normal behaviors. Figure 1-2 depicts that modern SIEM is a convergence of SIEM and UEBA. Corporate giants Amazon and Microsoft are now extensively researching UEBA techniques for enterprise security solutions for extending the principles to areas of APT detection, cyber forensics, identity profiling, and customer behavior modeling. This indicates the scope and future of behavioral analysis-supported anomaly detection for mitigating risks and in-depth analysis in enterprise security solutions.

1.2 Research Problem

A detailed review of the literature indicates that Behavioral Analysis and Anomaly Detection using Log files is a developing topic and of immense significance due to its explanatory and predictive aspect. UEBA is an area of inquiry in data analytics and is limitedly exploited by researchers in a log analysis context. Behavioral analytics and Risk Scoring aids in characterizing anomaly candidates and also reducing false alarms (Liu et al., 2018)

Recently Data Analytics for behavioral research termed Behavioral Analytics is emerging as a research trend in Management Information Systems due to its ability to integrate explanatory and predictive model building. Rapid advancements in social media, sensing technologies, and mobile computing demand novel ways to analyze user perspectives for improving, recommendation systems and adaptive interfacing. The survey by Motiwalla et al., (2019) investigates the numerous studies made on behavioral approach wherein user interactions and behavior are also considered along with user attributes such as emotion detection for text, images, and video, fraud detection in crowdfunding to name a few.

Limited progress has been made in Behavioral analysis for Log Anomaly Detection especially in a cloud-based environment (Liu et al., 2018). The system log behavior approach deals with user and entity-based analysis. The major hurdle in the log-based behavioral analysis would be analyzing massive volumes of logs to extract the WHO, WHICH, and HOW factors of an anomaly rather than the WHAT factor of the anomaly as in conventional approaches. Also, user or entity activity-based risk score calculation for user profiling in UEBA is an underutilized area that can be well explored for performance improvement. In general, from the existing literary works, it can be inferred that although UEBA provides an effective solution in terms of detecting anomalies from the log data certain challenges like increased false alarm rate, risk score calculation, computational time, and resource constraints need to be addressed.

1.3 Purpose of Research

This study aims to leverage user and entity behavioral analysis along with state-of-the-art machine learning techniques to analyze the system logs and identify anomalies with the least manual intervention. The objectives of the study are to:

1. To preprocess the structured logs to conduct exploratory data analysis and feature selection to get valuable insights.
2. To analyze the structured logs using the User and Entity Behavior approach to assign the Entity and Risk scores to the entities.
3. To identify anomalies from the preprocessed log data using machine learning techniques and evaluate the performance

1.4 Significance of the Study

A critical review of the literature shows that the need of the hour is an automatic log analysis and anomaly detection method for online proactive monitoring in SIEM. When such a robust anomaly detection is supported by the UEBA approach the SIEM could bring about the identity of anomalies promptly thereby preventing and reducing the damage caused by cyber attacks. To maximize the effectiveness of such a security solution and to be made affordable by small and medium enterprises (SMEs), simplicity in machine learning models and improved detection capabilities over model complexity are to be prioritized. The study aims to develop such a simple yet affordable UEBA and ML-based Log Anomaly Detection with superior detection capabilities.

1.5 Research Purpose and Questions

On the ground of a comprehensive literature review conducted on UEBA-supported Log-based Anomaly Detection, the research aims to analyze the performance improvement offered by UEBA techniques and to figure out a simple Machine Learning model for superior Log Anomaly Detection.

The following research questions may be formulated:

1. How much computational and performance efficacy can be brought out by leveraging UEBA techniques for log analysis?
2. Which machine learning technique will predict anomalies from the structured log with optimal computational performance and reduced false alarm rate.
3. What is the impact of UEBA-based Log Anomaly Detection on small and medium enterprise businesses?

CHAPTER II:

REVIEW OF LITERATURE

2.1 Literature Review Objectives

The ever-growing dependency on digital systems and the increasing volume of data generated through different entities have posed a significant challenge to maintaining the integrity and security of the data. In the realm of cybersecurity, one of the critical aspects is the detection of anomalous behavior within log data. Logs, which capture various activities and events in a system, serve as valuable sources of information for identifying potential security breaches, system malfunctions, or malicious activities. Traditional rule-based approaches and manual analysis methods are ineffective in dealing with the complexity and scale of log data and this necessitates the implementation of advanced techniques such as machine learning and user and entity behavior analysis (UEBA) for effective log anomaly detection.

Machine learning (ML) techniques have gained significant attention because of their excellent performance in various domains due to their ability to automatically learn patterns and anomalies from data. By leveraging algorithms and statistical models, ML techniques can discover hidden relationships and abnormal behaviors within log data, facilitating the detection of anomalous activities that may indicate security threats or system irregularities. Furthermore, the integration of user and entity behavior analysis adds a layer of context and understanding by considering the behavior of individual users or entities and detecting deviations from their established patterns.

This chapter will provide a brief introduction to the implementation of ML algorithms and UEBA for detecting log anomalies in the system and also discuss the user and entity behavior analysis. The preliminary objective of this chapter is to explore the state-of-the-art

approaches and methodologies in log anomaly detection using ML and UEBA. By reviewing the existing literature, this chapter aims to explore the methodologies involved in the anomaly detection process and provide a comprehensive overview of log anomaly detection. This chapter will serve as a valuable resource for researchers, practitioners, and organizations seeking to enhance their cybersecurity posture by effectively detecting and mitigating anomalous activities within their systems.

The structure of this literature review chapter is as follows: Section 2.2 highlights the important factors for adopting the concept of machine learning. Section 2.3 explores the existing approaches and methodologies proposed in the literature related to the concept of UEBA and reviews existing works related to the concept of UEBA. Section 2.4 discusses the analysis of the existing literature works and different studies related to anomaly detection and user and entity behavior analysis using machine learning. Section 2.5 reviews recent research works done on the implementation of deep learning for log anomaly detection and behavioral analysis, Section 2.6 analyses the Log datasets utilized in the UEBA and IDS landscape. Section 2.7 provides a comprehensive gap analysis focusing on the application of ML, DL, and UEBA methodologies in log anomaly detection. Section 2.8 concludes the literature review by summarizing the key findings and outlining potential research opportunities.

2.2 Important Factors for Technology Adoption

Theory of Reasoned Action (TRA) is one of the extensively used theoretical frameworks that is implemented to understand and analyze human behavior in different domains (LaCaille, 2020). In the context of log anomaly detection using ML and UEBA, the TRA framework provides a valuable lens through which to examine the factors influencing the adoption and effectiveness of such systems. TRA was developed by Martin Fishbein and Icek Ajzen (Ajzen & Fishbein, 1980) which asserts that behavior is influenced by three key factors: behavioral intention, attitude, and subjective norms (Otieno et al., 2016). Behavioral intention reflects a person's readiness and willingness to engage in a specific behavior. Attitude refers to their evaluation of that behavior, while subjective norms involve the perceived social pressure or expectations related to it. According to TRA, these three factors play an important role in shaping an individual's intention to engage in a particular behavior, which, in turn, influences their actual behavior (Glanz et al., 2015).

In this research, TRA is employed to understand and analyze the factors that influence the adoption and usage of ML and UEBA for detecting and mitigating log anomalies. Users' and administrators' attitudes toward these technologies and their perception of their benefits affect their intention to use them. Factors like ease of use, perceived usefulness, and compatibility with existing techniques shape user attitudes. Additionally, subjective norms, including the influence of peers, supervisors, and administrators within an organization, play a crucial role. If these entities consider log anomaly detection valuable, its adoption and utilization are more likely. TRA helps in understanding behavioral intention, attitude, and subjective norms, which is essential for successful implementation.

Culture is a significant factor affecting technology adoption in this context. It encompasses shared beliefs, values, and practices, influencing attitudes and behaviors. Cultural factors impact trust, user interface preferences, and organizational norms, all of which affect the adoption of log anomaly detection techniques. Understanding and addressing cultural aspects can enhance user acceptance and the effectiveness of these technologies.

In conclusion, TRA provides a valuable framework for analyzing the adoption of ML and UEBA for log anomaly detection, while recognizing the critical influence of culture in shaping user behavior and attitudes toward these technologies. This understanding can inform the design and implementation of these techniques to better fit diverse cultural contexts

2.3 User and Behavior Analytics (UEBA)

UEBA is a cybersecurity framework that uses advanced data analytics, ML, and behavioral modeling techniques for detecting and mitigating anomalies in the network (Raguvir & Babu, 2020). UEBA analyzes the behavioral aspects of the users such as employees, administrators, etc., and entities such as applications and servers to identify the abnormalities in the network patterns and detect security threats in the early stage of occurrences. UEBA monitors and analyzes user and entity behavior and identifies anomalies in the network that are not detected by conventional rule-based security techniques. UEBA models collect a huge volume of data from different sources such as log files, access logs, and network traffic database and system events. This data is processed using ML models to distinguish normal behavior from anomalous behavior to identify potential security threats (Sharma et al., 2020).

Several research works have adopted UEBA for log anomaly detection and this section reviews existing works that have implemented UEBA. Currently, most organizations have adopted the policies and compliances of Security Information and Event Management (SIEM) for detecting insider security threats (González-Granadillo et al., 2021). However, the increase in the volume of insider threat information also increases the number of false alarms. It is highly challenging to reduce the number of false alarms by processing such a large volume of data. In addition, this process consumes more time and needs additional manual intervention and computational resources. It is essential to address these problems by developing a robust anomaly detection model using advanced and sophisticated techniques.

The work proposed by (Liu, 2021) addresses insider threat detection in the context of a "digital new era". It highlights the challenges of false alarms and context deficiencies in traditional Security Information and Event Management (SIEM) approaches due to the

increasing volume of insider information data. The proposed solution involves developing an insider threat detection system based on User and Entity Behavior Analysis (UEBA). An improved LSTM-GaN insider threat detection algorithm is introduced. The study discusses the extensibility of the proposed system, considering loose coupling for each module, and suggests further work on exception detection for various attacks. The experimental results indicate an accuracy rate of 91.14% with specific parameter settings.

The study by (Tao *et al.*, 2020) proposes a novel approach for modeling user behavior profiles in mass customization manufacturing by introducing a multi-dimensional semantic activity space. The method integrates data from diverse subsystems, addressing the limitations of individual domain-based models. The authors validate their approach using log data from isolated systems and demonstrate the classification of users based on behavior patterns and statistical indicators. Challenges in multi-source heterogeneous data fusion are acknowledged, and future research directions include applying advanced prediction approaches. The study emphasizes the importance of log preprocessing in effective user behavior mining and presents findings on user activities across a corporate network from various locations.

The paper (Muliukha *et al.*, 2020) introduces an intelligent system prototype focusing on advanced analytics for integrated security in information and cyber-physical systems. Unsupervised machine learning, specifically Isolation Forest and Local Outlier Factor methods, is applied for anomaly detection in corporate networks The study addresses the static nature of security policies in corporate networks, emphasizing the constantly evolving methods of cyber-attacks. The user interface facilitates analytics and control through a web interface. Data analysis involves aggregating information with a specified time interval for machine learning. Results show Isolation Forest's superiority in the proposed task, attributed to its more complex crucial

function. The paper also discusses the use of Security Information and Event Management (SIEM) solutions and explores the limitations of the Mahalanobis distance method, suggesting Siamese networks and autoencoders for improved anomaly detection in certain scenarios.

An UEBA-based approach is proposed (Rengarajan & Babu, 2021) which continuously monitors user profiles along with other details such as data usage, IP address, location of the user their affiliation to the organization, and their period of association. Based on these factors the activities are classified into normal usage and anomalies. The data is represented visually using data visualization tools for creating a visual representation of the data for analyzing and detecting anomalies. The paper discusses the challenges faced by organizations in monitoring and controlling user behavior to prevent data breaches and cyberattacks. The study focuses on the behavior of users accessing corporate networks from various locations, exploring usage frequency, location details, and services utilized. The analysis involves basic statistical methods to identify anomalies in user behavior patterns. The paper highlights the potential of UEBA in enhancing cybersecurity by detecting and preventing anomalous user activities.

A comprehensive analysis of UEBA and its role in detecting insider threats and attacks is presented in (Khaliq et al., 2020). The study discussed different frameworks used in UEBA including both user and role-based detection, calculation of risk score, activity mapping of user and entity, and user profiling approaches. The work also discussed the details of UEBA techniques proposed in existing literary works and provided a structured approach to understanding the possible UEBA solutions. The study also highlighted the fact that there is a deficit of an effective approach that can provide a robust solution for all UEBA problems. UEBA reduces the influence and effect of potential security attacks on the data and minimizes the risk of security breaches by tracking the behavior of normal users and entities. The study emphasizes

that organizations must implement security control using UEBA for securing the data by identifying the threats and attack occurrences in the preliminary stages. However, the cost of deploying such security solutions can be significantly high and it becomes economically difficult for small and medium organizations.

The work presented (Rashid & Miri, 2021) provided a solution for organizations by outsourcing the UEBA to third-party entities. However, there is a risk of disclosing private data while outsourcing the data. Hence, this work proposes a novel approach that integrates differential privacy into UEBA. The analysis of the approach states that instituting noise into the data before outsourcing it to third-party entities for detecting anomalies. The outcome of the analysis shows that the introduction of noise before outsourcing the data can strengthen the integrity and privacy of the data while simultaneously allowing the third parties to accurately analyze UEBA without increasing the cost. The efficacy of the proposed approach is validated through the results.

The authors (Kaur et al., 2022) discussed the analysis of UEBA with the help of a specific case study. Several organizations have adopted Amazon Web Services (AWS) for cloud migration since it is considered one of the leading technologies in the current times. The work mentioned in (Kaur et al., 2022) used AWS as a case study for analyzing the behavior of the user based on their activities. The proposed approach identifies the anomalies and classifies the user according to their activities.

The paper (Sharma, Pokharel and Joshi, 2020a) introduces an unsupervised user behavior modeling approach using an LSTM-based Autoencoder for anomaly detection, focusing on insider threat detection from log data. The method calculates reconstruction errors on a non-anomalous dataset to set a threshold for identifying anomalies. The CERT insider threat dataset

is used, extracting feature vectors from raw events to train the LSTM Autoencoder. The experimental results indicate an Accuracy of 90.17%, True Positives of 91.03%, and False Positives of 9.84%. The research emphasizes the significance of monitoring user behavior to enhance cybersecurity and protect against potential insider threats, showcasing the effectiveness of the proposed approach in automatic anomaly detection.

The paper (Rasheed Yousef and Mahmoud Jazzar, 2021) focuses on evaluating the effectiveness of User and Entity Behavior Analytics (UEBA) for preventing insider threats. The tests and simulated scenarios encompass monitoring user behaviors, establishing baselines, and assessing the impact of false positives. The UEBA system compares the baseline with current user behavior, assigning a risk score to deviations. The results indicate that UEBA is effective in detecting insider threats and reducing false positives, showcasing a 44.3% growth in the global UEBA market from USD 185.6 million in 2017 to an expected USD 2453.4 million in 2024. The experiments were conducted over two weeks, monitoring 530 events and generating 103 events with confidence scores.

This paper (Deng *et al.*, 2021)proposes an approach for analyzing user behavior in complex power grid environments using unsupervised learning. The Adaptive Feature Selection algorithm based on Stacked Autoencoder and Unsupervised Learning (AFS-SAEUL) is introduced to reduce the complexity of user behavior data. Subsequently, a User Behavior Analysis model based on Adaptive Feature Selection and Improved Clustering (UBA-AFSIC) is developed, enhancing unsupervised classification by incorporating an adaptive generation strategy for initial cluster centers. Experiments on real load datasets and a public electric vehicle dataset show that AFS-SAEUL achieves a higher feature selection ratio and better-unsupervised classification performance. UBA-AFSIC outperforms other clustering models, demonstrating a

lower Davies Bouldin Index (DBI) and higher Silhouette Coefficient (SC). The study provides insights into electricity consumption patterns and electric vehicle charging behaviors, offering a basis for future user planning and marketing.

This paper (Alghayadh and Debnath, 2021)introduces a Hybrid Intrusion Detection (HID) system for securing smart home systems against cyber threats. The proposed system utilizes machine learning algorithms, including Random Forest, XGBoost, Decision Tree, K-Nearest Neighbors, and misuse detection techniques. It aims to protect smart homes by analyzing both network and user behavior and addressing vulnerabilities in IoT-based smart home systems. The research emphasizes the need for adaptive security and privacy measures to mitigate the risks associated with smart home devices. Experimental results indicate varying accuracies for different algorithms, with K-Nearest Neighbors achieving the highest accuracy of 95.9% in the CSE-CICIDS2018 dataset. The proposed HID system demonstrates effectiveness in detecting anomalies and securing smart home networks. For the NSL-KDD dataset, Random Forest was the most successful algorithm with an average accuracy rate of 98.6%.

This paper (Anashkin and Zhukova, 2022)discusses the evolutionary development of indicators for monitoring and threat detection, aiming to create a unified descriptive structure for behavioral indicators. The proposed description standard seeks to establish an open database of behavior indicators, forming the basis for a user action profiling system. The integration of behavioral indicators into cyber threat monitoring and detection is highlighted, to develop an accessible indicator database. The resulting behavioral indicator description structure includes fields and typical event sources, facilitating the automated creation of correlation rules for SIEM systems. The paper also emphasizes the advantages of using behavioral indicators for focusing

on malicious intent and detecting time-distributed attacks. Future work will concentrate on developing algorithms for modeling and detecting behavioral indicators from various log events.

A novel framework and design for the UEBA system is presented and deployed by (Lukashin et al., 2020). Experimentation was performed considering devices from Cisco ASA, network firewalls, and different events in the system which exhibited better outcomes in terms of detecting anomalous behavior. The work also described different techniques for handling semi-structured data obtained from different sources with the aid of several anomaly detection techniques.

A technique for constructing the features from hybrid data streams from different SIEM techniques is presented in this work. This work also determined the scalability and efficiency of the proposed approach. Several approaches have been introduced in previous works to reduce the security issues related to network systems with an emphasis on UEBA (Salitin & Zolait, 2018; Dai et al., 2022; Salitin & Zolait, 2023). However, transparency is one of the unsolved challenges which needs more attention.

Traditionally, the data is stored in chronological order for preventing the tampering and exploitation of data and this technique assures the privacy and traceability of the data. A UEBA framework is presented in (Khan et al., 2022) which studies the profile of the users over a period of time and categorizes them either as normal or malicious. The proposed approach incorporates detailed information such as the location of the data stored, organizational information IP address, etc. The study also focuses on applying data science and analytical techniques for creating data visualization for identifying anomalies.

An efficient approach based on UEBA is proposed in (Martin et al., 2021) for improving the security of the Federated Identity Management (FIM) solutions. The UEBA-based

framework deployed in this work enables different entities within the federations to develop a session fingerprint that defines the character of the user's behavior which is accessed based on the available data. In addition, the proposed approach also enabled anomaly detection using the fingerprint details which also incorporated the mechanism of raising alerts upon identifying the anomalies. The approach addresses challenges in FIM, such as pre and post-authentication scenarios, online and offline modes, and integration with existing specifications. The efficacy of the proposed approach is determined and validated using a real-time case study based on a web chat application using an OpenID connect for accurately identifying the users. Experimental results focus on machine learning models for UEBA, with considerations for imbalanced problems and the importance of recall in critical infrastructure settings.

(Martín *et al.*, 2022) presents a novel method for continuous authentication using User and Entity Behavior Analysis (UEBA) techniques to enhance security systems beyond traditional password-based approaches. The method combines information from multiple sources at the feature level, employing Symbolic Aggregate approximation (SAX) and Random Trees Embeddings for temporal representation. Behavioral cores are extracted using density-based clustering, and a risk model is applied for anomaly detection, specifically user impersonation. The proposed method outperforms state-of-the-art models when considering sufficient information. Empirical results demonstrate the accuracy of the approach, with potential applications in various domains beyond continuous authentication, such as detecting temporal anomalies in finance or health monitoring.

In general, UEBA leverages the advantages of anomaly detection methods for recognizing attacks and potential security threats based on the behavior patterns of the user profiles and risk score calculations. The profile data is obtained from the historical log data.

However, it is highly challenging to obtain log data due to the lack of availability of real log datasets which is appropriate for UEBA. Most of the existing datasets are not publicly available which restricts the comparison of different objects and reproducibility of techniques for anomaly detection (Landauer et al., 2022). Most of the existing works make use of synthetic data to alleviate this problem to a certain extent because simulations are not effective enough for handling the dynamic and volatile nature of the user behavior in real-time from the existing log data.

Hence, a real-time log dataset is presented by (Landauer et al., 2022) from a cloud computing environment that consists of more than five thousand users for a period of more than five years. The dataset is evaluated considering a case study related to account hijacking. The proposed framework is designed for detecting the possible injection of attacks and thereby identifying the resulting inferences related to anomaly detection. An adaptive anomaly detection mechanism is employed to analyze the dataset, revealing diverse and erratic user behavior patterns in real cloud computing environments. The dynamic and unstable nature of the log data, influenced by long-term changes in system utilization, is emphasized. The authors provide a method for attack injection, simulating account hijacking, and evaluating the resulting manifestations in the log data using the adaptive anomaly detection mechanism. They also discuss plans for injecting other types of anomalies in the data for future work. The chosen trade-off parameters for the anomaly detection mechanism include a queue size of 30, and a threshold of 0.6, resulting in a true positive rate of 65%, and a false positive rate of 4.6%.

Risk scoring is proven to be a performance-enhancing feature of UEBA solutions and the UEBA-based anomaly detection methods majorly deploy anomaly or event or instance-based risk score calculation. A notable trend of using user profile-based risk scoring is employed in

notable UEBA-based SIEM providers such as Aruba IntroSpect and Microsoft Sentinel to enhance SIEM capabilities. However, user profile or activity-based risk score calculation is under-exploited by existing literature studies. Microsoft Sentinel uses an Investigation Priority Score for its UEBA solution as shown in Figure 2-1. Investigation Priority Score is a weighted score of Alert Score and Activity Score. Alert Scoring comprehends the potential impact of a specific alert on individual users. The scoring considers severity, user impact, and alert prevalence across users, and all entities within the organization. Activity Scoring assesses the likelihood of a specific user engaging in a particular activity through behavioral learning from the user and their peers. Activities identified as highly abnormal receive elevated scores, aiding in prioritizing potential threats effectively.

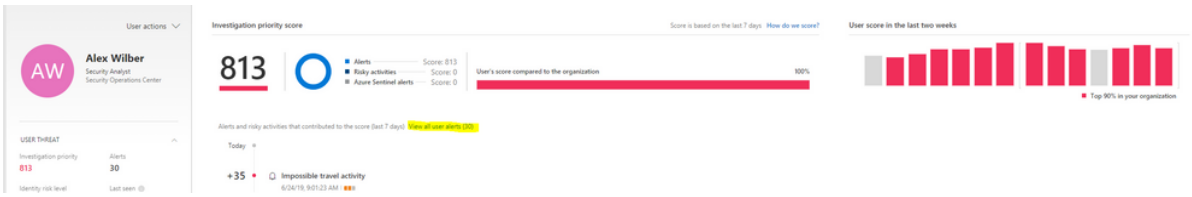


Figure 2.1
User Page with Investigation Priority Score in Microsoft Sentinal
Source: Microsoft. (2023). Introducing investigation priority built on user and entity. Microsoft Tech Community

A detailed review of Risk score-based user profiling done by (Khaliq, Abideen Tariq and Masood, 2020) indicates that the process involves categorizing security events by domain, assigned role, priority, and reliability. The risk score is calculated periodically as the product of event priority, asset value, and the confidence of the detection system. If an event's risk score exceeds a predefined threshold, an alert is generated. The correlation process links different events, resulting in a combined risk score. Aruba IntroSpect, a commercially available UEBA system employs this approach, utilizing over 100 supervised and unsupervised machine learning

algorithms. In the detection process, anomalies or malicious activities trigger events, providing severity and confidence scores ranging from 0 to 100. The severity score indicates the event's importance, while the confidence score reflects the detection system's accuracy probability.

Challenges associated with UEBA

It can be inferred from existing literary works that although UEBA provides an effective solution in terms of detecting anomalies from the log data, certain challenges need to be addressed. This research identifies some of the prominent challenges which are summarized as follows;

(i) Duration of Training Data: The time required for training the model is usually high and deciding an appropriate size for the dataset is a challenging task to develop a baseline profile for analyzing all possible events during log detection.

(ii) Reducing False Positives: Due to large data volumes, it is difficult to detect log anomalies with high accuracy and in most cases, the effectiveness of the anomaly detection techniques is affected due to the increase in the rate of false positives. The increased number of false positives makes it difficult for security professionals to raise alerts.

(iii) Data Quality and Volume: UEBA depends majorly on large volumes of indistinct data sources, such as log files, user activity logs, network traffic data, and system logs. It is significantly complex to understand the quality and reliability of the data in complex and heterogeneous IT systems.

(iv) Resource Constraints: The adoption of UEBA can be resource-intensive since it requires more computational power and storage. Small and medium organizations or those with limited resources may face challenges in adopting UEBA solutions.

(v) Risk Score Constraints: Risk score calculation in UEBA can enhance anomaly detection by giving an empirical dimension to the users, entities, and abnormalities. However, adopting a suitable methodology for risk calculation is required for reducing false alarms.

These challenges can be addressed by combining different technologies with precise domain expertise along with a comprehensive understanding of the security requirements for different scales of organizations.

2.4 Machine Learning Models

Machine learning (ML) is a subcategory of artificial intelligence that emphasizes the design and development of algorithms and statistical models that enable computerized systems to learn and make decisions or predictions without explicit programming. Particularly, ML enables computers to learn from and adapt to data, improving their performance over time without being explicitly programmed for each specific task (Dogo et al., 2019; Al-amri et al., 2021). ML is used in a broad variety of application areas, such as speech processing, language understanding, and Computer vision (Patel & Thakkar, 2020; Chai et al., 2021). Such programs need to get useful information from massive measurements of user behavior information, historical data, etc. which are frequently produced quickly everywhere throughout the world. It was observed that ML algorithms achieve breakthrough performance by solving current and future security problems of network systems. The performance of different ML algorithms is reviewed in this section in terms of log anomaly detection and attack detection.

(i) Support vector machine (SVM): SVM is a supervised ML algorithm and is one of the efficient and potential tools for the classification of faults. In general, SVMs exhibit their tendency not to overfit, but to achieve precise classification in various cases. SVM possesses high generalization, slow convergence speed, and is highly sensitive to local extrema (Weerasinghe et al., 2019). Apart from classification tasks, SVM also performs regression. In most cases, a Radial Basis Function (RBF) or Gaussian kernel is used for training the dataset in

SVM. However, in certain scenarios, anomaly detection models using the kernel function do not perform well with a greater number of training samples or large numbers of features in the input space.

The authors (Fan et al., 2020) implemented SVM along with RBF kernel function for performing adaptive Magnetic anomaly detection (MAD) and the approach aims to enhance the probability of MAD by improving the signal-noise ratio (SNR). Initially, the detection model is designed using both SVM with RBF function and then the anomaly signal in the magnetic field is detected using the orthonormal basis function (OBF) energy and magnetic entropy as the features of the magnetic anomaly. Results of the experimentation process show that the proposed approach significantly minimizes the rate of false alarms and thereby enhances anomaly detection performance with a lower SNR value.

An efficient anomaly detection approach for a cloud computing environment is proposed in Krishnaveni et al.,(2020) employing SVM algorithm for training the model and detecting the intrusions in the early stage. The performance of the SVM was tested on an optimized NSL-KDD dataset and the results show that the attack detection ability of the SVM algorithm was enhanced by the feature set algorithm designed on the information gain ratio. The accuracy of the SVM model was improved by 96.24 % and reduced the false alarm rate (FPR). It can be noted that the ML-based SVM exhibited advantages in terms of detecting intrusions in complicated environments such as cloud computing. Although SVM is highly advantageous as a classifier, there are certain drawbacks such as hypersensitivity to parameter tuning, expensive computation while dealing with large datasets, and lack of capability to handle noisy data and imbalanced datasets.

(ii) Random Forest (RF): The RF algorithm is a supervised classification algorithm. As the name indicates, this algorithm creates the forest with a number of trees. The more trees in the forest the more robust the forest looks. In the same way in the random forest classifier, the higher

the number of trees in the forest the higher the accuracy results (Shaik & Srinivasan, 2019). The RF algorithm can perform both classification and regression tasks and it addresses the drawbacks of the SVM algorithm in terms of handling noisy data and overfitting issues. RF is one of the enhanced ensemble learning algorithms made up of multiple decision trees.

Several research works have used the RF algorithm to classify security issues in the system (Liu et al., 2021; Karthik & Krishnan, 2021; Prathapchandran & Janani, 2021). These works have classified unauthorized intrusions and attacks based on the aggregated data and by performing regression analysis. The RF algorithm has many advantages compared to conventional techniques in terms of superior accuracy among various classification techniques and its efficiency in dealing with larger datasets and data samples with high dimensional functionalities, which reduces the need for performing dimensionality reduction. There is a plot that describes the amount of error for each tree and it shows that 10 trees are suitable for this data. The random forest algorithm suffers from the problem of high variance and hence it is incorporated with bagging techniques to overcome this problem.

The work presented (Primartha & Tama, 2017) analyzed and compared the performance of the exerting RF classifier with other existing techniques in terms of accuracy and false alarm rate. The RF algorithm was designed to identify unknown intrusions and anomalies for securing the network systems. The model was trained using three public intrusion detection datasets such as NSL-KDD, UNSW-NB15, and GPRS datasets. In addition, the model was also accompanied by different combinations of tree sizes, and the best learning parameters were determined using a grid search method. The outcome of the experimental analysis validates the excellency of the RF classifier in terms of detecting unauthorized intrusions compared to other ensemble techniques such as ensemble of Random Tree + Naive Bayes and other single classifiers such as Naive

Bayes and Neural Network in terms of k-cross validation technique. It can be inferred from the results that the RF model achieves an excellent accuracy of 99.57 % for a 10-fold cross-validation performed on the NSL-KDD dataset and achieved an accuracy of 95.5 % for a UNSW -NB15 dataset.

The authors (Lin & Jiang, 2020) proposed an integrated approach that combines an autoencoder algorithm with an RF algorithm for detecting anomalies in credit card fraud scenarios. The hybrid approach is called AERFAD which is designed to detect frauds in credit card transactions. The proposed approach initially uses an autoencoder to reduce the dimensionality of the data and then uses the RF algorithm to classify the data as normal or anomalous. The AEFRAD model was trained to process a large volume of transactional details of the credit card consisting of the information of several European cardholders and thereby identify possible frauds from the transactions. Results show that the proposed model achieves excellent performance with respect to different evaluation metrics such as accuracy, true positive rate, true negative rate, and Matthews correlation coefficient. The model achieved an accuracy of 99.951% compared to other algorithms such as SVM , Adaboost, Neural Network , Naive Bayes, K Nearest Neighbour, and Autoencoder-based clustering.

Elmrabit *et al.*, (2020) evaluated the performance of 12 different ML models with respect to their capacity to identify anomalous behavior in the network systems. The performance was evaluated on three publicly available datasets: UNSW-NB15, CICIDS-2017, and the Industrial Control System (ICS) cyber-attack datasets. The simulation analysis was performed using an ALICE-based high-performance computing facility at the University of Leicester and, the efficacy of the ML algorithm was evaluated in detail it was observed that the RF model achieves excellent results with respect to different evaluation metrics such as Precision, Recall, F1-Score,

accuracy, and Receiver Operating Characteristic (ROC) curves on all these datasets. The RF algorithm outperforms the existing algorithms in terms of binary classification and prediction accuracy and it was inferred that the RF algorithm achieved an accuracy of 88.5%. On the other hand, the accuracy of the RF algorithm was increased to 99 % for the CICIDS-2017 dataset. Results validate the fact that RF algorithms exhibit excellent results while other algorithms achieve better results. This shows that it is highly important to select an optimal ML algorithm based on the dataset.

(iii) Decision Tree (DT): The DT algorithm is one of the popular ML models used for both classification and regression tasks (Lakshminarasimman et al., 2017; Bouke et al., 2022). It builds a tree-like model of decisions and their possible consequences. Each internal node of the tree represents a decision based on a feature, and each leaf node represents the outcome or prediction. The algorithm selects the best feature from the available features as the root node based on the Gini index and information gain. Once the root node is selected, the dataset is split into subsets based on the values of the selected feature wherein each subset represents a different branch or path in the DT. Both SVM and RF algorithms achieve excellent classification accuracy. However, it is difficult to handle the increasing size of the datasets which affects their accuracy.

The authors Vargaftik et al.,(2021) implemented a resource-efficient supervised anomaly detection using decision tree-based ensemble methods known as the RADE method. The DT-based approach is implemented to identify the current anomalies from the dataset. The DT algorithm is trained to be adaptable for complex environments along with resource-consuming environments and identify the anomalies from the input dataset. The DT-based ensemble classifier can address the drawbacks associated with conventional classifiers. The proposed

approach is highly resource-efficient since it increases the size of the dataset to achieve better performance in a resource-constrained environment. The main objective behind implementing the proposed model is to train the algorithm that is capable enough of accurately classifying most of the queries. Further, the DT algorithm uses only subsets of the training data and trains the model using a fewer number of parameters. Usually, the models that are trained using the smaller datasets are at high risk of developing classification errors. The proposed approach implements a scikit learn classifier and the performance evaluation shows that the RADE model can accurately identify the anomalies compared to traditional techniques and thereby improve the memory capacity up to 12 times, training time up to 20 times, and time taken for the classification up to 16 times.

An optimized anomaly and intrusion detection mechanism is proposed in Douiba et al., (2022). The proposed approach was designed using two algorithms namely decision tree (DT) and gradient boosting (GB) for detecting anomalies in the Internet of Things (IoT) environment. Both these algorithms are analyzed through the open-source Catboost. The efficacy of this approach was determined using three datasets namely BoT-IoT, NSL-KDD, and IoT-23 datasets. The Graphics Processing Unit (GPU) was employed to improve the performance of the experimental analysis. The results were compared with the existing techniques and the results show that the GB and DT algorithms achieve outstanding performance with respect to different evaluation metrics such as precision, recall, F1-score, and accuracy metrics which are higher than 99.99% with a higher detection and computational time.

(iv) Naive Bayes (NB): The NB algorithm is a probabilistic ML algorithm based on Bayes' theorem, which is used for classification and sometimes for regression tasks. Despite its simplicity, the NB algorithm is highly effective in several real-time applications, especially in

natural language processing (NLP), text classification, and attack detection (Priya et al., 2021; Tuan et al., 2020). Several research works have emphasized the implementation of the NB algorithm for log anomaly detection.

The authors Kamoona et al., (2019) proposed a model-based technique for detecting anomalies in surveillance video data. The proposed approach is based on the estimation of data sparsity using pre-trained deep features. The study aimed to process the sparsity information of deep features and employ them for distinguishing between anomalous and normal events in the data sequence. The suggested method utilizes a Naive Bayes model within a framework of multiple-instance learning. Findings indicate that this approach improves the precision of detecting anomalies.An improved Naive Bayes (INB) method incorporated with PCA to extract relevant features from the data for achieving accurate classification is presented in Manimurugan et al.(2021). The INB is implemented to design an efficient IDS for attack detection in an integrated IoT-Fog-Cloud computing architecture. The attack-related information is collected from the UNSW-NB 15 dataset and the IDS is designed incorporating feature extraction methods for classifying the attacks. The proposed approach enhances the effectiveness of anomaly detection and the performance is analyzed with respect to its accuracy and detection rate wherein the model achieved an accuracy of 92.48% with a 95.35% detection rate.

2.5 Deep Learning Models

Deep Learning (DL) models address the drawbacks of ML algorithms in terms of reducing computational complexity, improving the accuracy of detection, etc. Considering the advantages of DL algorithms, several research works have implemented DL models for detecting anomalies from log data. The work presented in (Pawar & Attar, 2019) provided a comprehensive review of deep-learning approaches for anomaly detection. The study analyzes the application of various deep-learning models for video-based anomaly detection. It can be observed from the existing literary works that deep learning algorithms were more effective concerning accuracy and precision.

The work presented by Tsogbaatar et al., (2021) employs a novel deep ensemble learning (DeL- IoT) approach for detecting and predicting anomalies in the Software Defined Networks (SDN) environment. The proposed approach employs three components namely anomaly detection, predicting device status, and intelligent flow management utilizing deep and stacked autoencoders (SAE) for extracting handcrafted features that are stacked to form the deep ensemble learning model. The proposed approach provides a robust yet appropriate anomaly detection performance and manages the data flow dynamically.

Further, anomaly-based IDS - Intrusion-detection system for IoT with the evaluation of a real smart scenario utilizing one-class SVM. Eventually, detection accuracy was obtained at 99.71% which shows that the research identifies the present potential solutions (Bagaa et al., 2020). The investigation of utilizing ML-based classification methods for IoT security against DoS threats. The research is mostly executed on the classifier that develops the anomaly-based IDS methods against security threats. Therefore, the performance assessment of every classifier was executed in terms of certain validation and prominent metrics. Various datasets such as

NSL-KDD, UNSW-NB15, and CIDDS-001 utilized in previous research are utilized for benchmark classifiers.

Ribeiro et al.,(2018) Proposed a study of deep convolutional auto-encoders for anomaly detection in videos. The study employs conventional auto encoders (CAE) for anomaly detection focusing on collecting high-level spatial and temporal features from an input image and discusses the effect of these features on the performance of CAE. The performance of the proposed approach was validated by conducting various experiments considering different datasets. Experimental results validate the effectiveness of the proposed approach and support further research in this area. This section reviews some of the prominent DL algorithms for anomaly detection.

(i) Artificial Neural Networks (ANN): ANN is superior compared to its peers because of its reliability, scalability, precise accuracy, quick adopting technique, and its high tolerance towards fault occurrences. An Artificial Neural Network is an information paradigm that is inspired by the way the biological nervous system like the brain processes information. The key component of this model is the novel structure of the information processing system. This model is composed of a large number of interconnected processing elements which are called neurons which work in unison to solve specific problems.

ANN is trained by examples like human beings. It can be configured for particular (specific) applications like data classifications, and pattern recognition only through the learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons and this is true for ANN also when applied for anomaly detection (Kwon et al., 2021; Sahu & Mukherjee, 2020). The authors Abbasi et al.,(2021) implemented ANN for detecting anomalies in IoT applications. In the initial process, the proposed approach

employs a feature extraction technique that can accurately classify the anomalies. The essential features are extracted using a Logistic Regression (LR) and ANN for classification. The model was trained using an IoT dataset which contains the data instances wherein the data is collected from different IoT devices. The attack samples consist of data related to different types of attacks based on the number of criteria for determining the effectiveness of the proposed approach. Results of the simulation analysis show that the ANN model along with LR achieves excellent results with an accuracy of 99.98 %.

(ii) Convolutional Neural Network (CNN): Convolutional neural networks (CNN) is one such technique that exhibits excellent results on many high-level tasks such as image classification, anomaly detection, and more recently attack classification. CNNs constitute local or global pooling layers to integrate the outputs of the neuron clusters. CNNs also possess different composites of convolutional layers and fully connected layers with pointwise nonlinearity applied at the end of or after each layer. This process is inspired by biological phenomena. A comprehensive review of different DL algorithms such as CNN, Autoencoders, and other models for log anomaly detection is presented by Yadav et al., (2020). The study identifies some of the important challenges related to log anomaly detection such as handling unstructured data, instability, log bursts, and lack of availability of public datasets. Results show that DL algorithms are more effective in detecting log anomalies compared to ML algorithms.

(iii) Long Short-Term Memory (LSTM): The LSTM model is a type of Recurrent Neural Network (RNN) that is mainly used in the classification and prediction tasks. The performance of the LSTM model for log anomaly detection has been validated in various research works. The authors Vinayakumar et al.,(2017) applied an LSTM model for detecting anomalies from the log data. For this purpose, the log samples are considered and analyzed for

classifying the data samples as normal and anomalous occurred over a period of time. The LSTM architecture is implemented for detecting events based on the data collected from the Cyber Security Data Mining Competition (CDMC2016) dataset. The experimental evaluation was conducted on the real-time data samples and results show that the LSTM achieves an accuracy of 99.6%.

The work proposed by Du et al.,(2017) implemented an LSTM model for detecting log anomalies from the system data as a natural language sequence. The work proposed a DeepLog approach, which learns the log patterns automatically to identify abnormalities when the log patterns vary from the standard model. An LSTM model was implemented for designing the DeepLog approach which was trained from the log data under normal execution. Moreover, the study also demonstrated that the LSTM-based DeepLog model can be updated online such that it can detect novel log patterns dynamically over a period of time. Moreover, the DeepLog model also constructs the workflow based on the underlying system log. This enables system users to analyze the identified anomaly and conduct a more thorough and effective examination of the anomalies once they are detected. Experimental evaluation shows that the proposed LSTM-based DeepLog approach achieves excellent results compared to other log-based anomaly detection techniques that use conventional data mining approaches.

The authors Baril et al., (2020) implemented an LSTM model for developing a novel model for analyzing the performance anomaly detection that captures temporal deviations from the nominal model, employing a sliding window data representation. This nominal model is trained by a Long Short-Term Memory neural network, which is appropriate to represent complex sequential dependencies. The work presented in this paper assessed the effectiveness of the LSTM model by testing it on both simulated and real datasets. Results show that it is more

robust to temporal variations than current state-of-the-art approaches while remaining as effective.

The researchers Chen et al., (2022) introduced a new model called LogLS, designed for detecting anomalies in system logs. This innovative approach utilizes a dual long short-term memory (LSTM) with a symmetric structure, treating the system log as a natural-language sequence and modeling it based on the preorder and post-order relationships. LogLS is an enhancement of the DeepLog method, addressing the issue of poor predictive performance of LSTM on extended sequences. Through a feedback mechanism, it facilitates the prediction of logs that are not present. The effectiveness of LogLS in log anomaly detection was validated through performance evaluations on two authentic datasets, showcasing promising results for the proposed method.

The work presented by Zhao et al., (2021) presented an unsupervised approach for detecting anomalous behavior in large-scale security logs. The study proposed a novel feature-extracting mechanism that could precisely characterize the features of malicious behaviors. The research conducted an anomaly detection model using Long Short-Term Memory (LSTM) based on the analysis of log content in Knet2016 and R6.2 datasets. The model was trained and tested on extracted log-line text, demonstrating accurate exception detection. Comparative experiments revealed superior performance compared to other models such as PCA, one-class SVM, and GMM in terms of accuracy and efficiency. However, further evaluation against classical models is warranted. Future studies should involve broader datasets for training and testing to ensure generalizability. Additionally, incorporating isolation forest as a baseline in comparative experiments is essential, considering its performance in the DARPA insider threat detection program.

Further investigation is necessary for the comprehensive utilization of logs from diverse sources. It is evident that logs from different sources exhibit distinct structures. Combining these logs directly and employing character-level models for training might not produce optimal results. A logical approach involves separately detecting anomalies in these logs and subsequently integrating the anomaly scores after obtaining individual user anomaly scores.

2.6 Dataset Analysis for IDS and UEBA

The landscape of Intrusion Detection Systems (IDSs) and User and Entity Behavior Analytics (UEBA) heavily relies on the availability of benchmark datasets for evaluation and development. However, the current datasets often lack the real-life characteristics of contemporary network traffic, rendering many anomaly IDSs impractical for production environments Hindy et al., (2020). Moreover, IDSs face challenges in adapting to the dynamic nature of networks, including new nodes, changing traffic loads, and evolving topologies. Consequently, the dependence on outdated datasets hampers the progress of IDS technology. To address this, researchers advocate for the development of new datasets that reflect the dynamic nature of network patterns, emphasizing the importance of datasets that are real, labeled, variant, correct, easily updatable, reproducible, and shareable (Sharafaldin et al., 2017; Viegas, Santin and Oliveira, 2017).

Synthetic data injection is proposed as a strategy to enhance existing datasets or balance attack classes. However, challenges persist in sharing datasets due to confidentiality concerns, limiting research opportunities. Furthermore, simulating real-life scenarios and associated attacks remains complex due to the multitude of parameters required for the model to be viable Hindy et al.,(2020).

In response to these challenges, recent efforts have been made to compile and release prominent datasets for research purposes. For instance, loghub offers a collection of real-world system log datasets, encompassing various software systems such as distributed systems, supercomputers, operating systems, mobile systems, server applications, and standalone software Zhu et al., (2023). These datasets, totaling over 77 GB, provide researchers and practitioners with valuable resources for developing intelligent and automated log analysis techniques. Notably, Loghub datasets include both labeled and unlabeled logs, facilitating a range of log analytics tasks, including anomaly detection, duplicate issues identification, log parsing, log compression, and unsupervised methods Zhu et al., (2023).

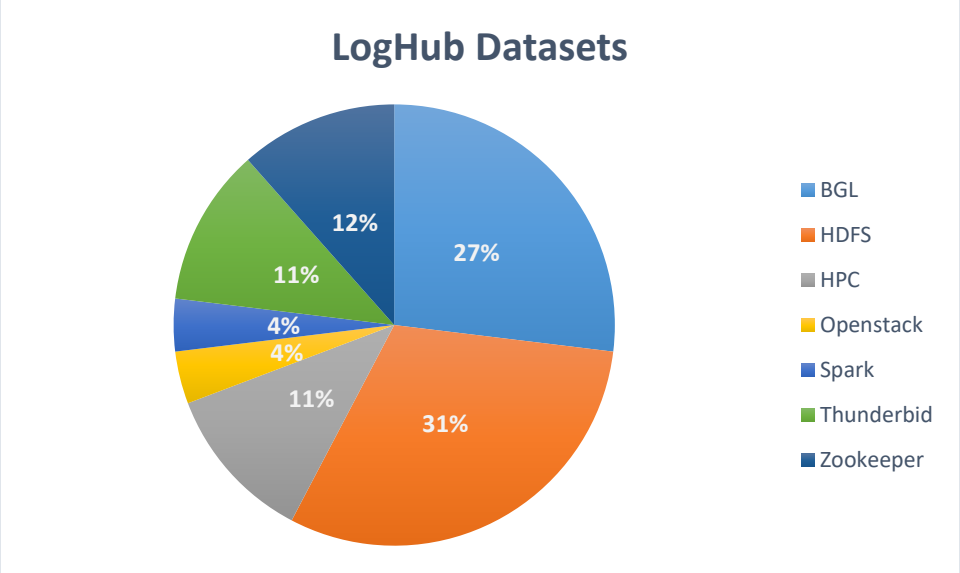


Figure 2-2
Comparative Usage of Loghub Datasets in Literature

The research analyses how datasets from the Loghub cloud environment, such as HDFS, Hadoop, Zookeeper, OpenStack, BGL, HPC, and Thunderbird, are utilized in existing literature, particularly in the context of Log-based Intrusion Detection Systems (IDS). It presents an empirical analysis and plots the findings. According to Figure 2-2, a pie chart is used to illustrate the relative frequency of utilization of these datasets for benchmarking and comparison in Log-based IDS literature. Notably, BGL and HDFS datasets emerge as the most commonly used for research in detecting anomalies in logs.

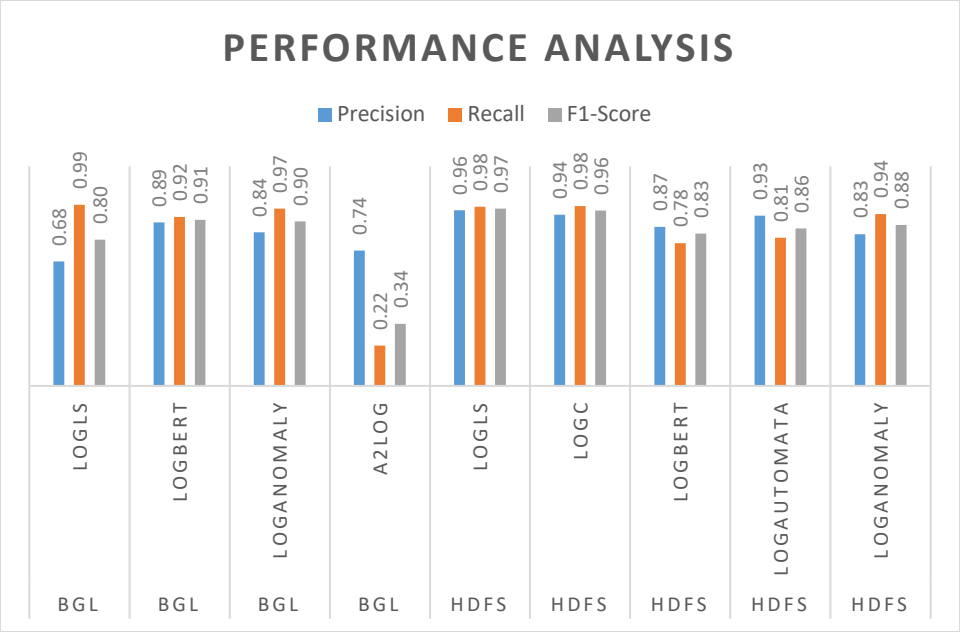


Figure 2-3
Empirical Analysis of Performance on BGL and HDFS dataset

Moreover, Figure 2-3 provides a summary of the empirical evaluation of key performance parameters of prominent Deep Learning (DL) algorithms on the BGL and HDFS datasets. The chart indicates that there is significant research potential for improving the Precision score specifically for the BGL dataset. On the other hand, the results show that the HDFS dataset has achieved remarkable Precision, Recall, and F1-scores, suggesting its effectiveness in log anomaly detection. The data used in the Analysis is shown in Appendix A.

In the realm of UEBA, obtaining suitable log data for anomaly detection poses a significant challenge due to the lack of publicly available real log datasets Landauer *et al.*, (2022). Consequently, most existing works resort to synthetic data, albeit with limited effectiveness in capturing the dynamic and volatile nature of user behavior in real-time scenarios. To address this gap, recent research introduces a real-time log dataset from a cloud computing environment, spanning over five thousand users and five years Landauer *et al.*, (2022). This dataset aims to enable more accurate and robust anomaly detection techniques by capturing

diverse and erratic user behavior patterns. An adaptive anomaly detection mechanism is employed to analyze the dataset, revealing insights into the dynamic and unstable nature of log data influenced by long-term changes in system utilization Landauer *et al.*, (2022).

Furthermore, the study discusses plans for injecting various types of anomalies into the dataset for future research endeavors(Landauer *et al.*, 2022). The evaluation of the proposed framework includes a case study on account hijacking, where attack injection simulations are conducted to assess the framework's effectiveness in identifying anomalous behaviors. Notably, the authors highlight the trade-off parameters employed in the anomaly detection mechanism, resulting in a true positive rate of 65% and a false positive rate of 4.6% Landauer *et al.*, (2022).

In conclusion, while challenges persist in dataset availability and simulation of real-life scenarios, recent initiatives such as Loghub and the development of real-time log datasets contribute significantly to advancing research in IDSs and UEBA. These datasets provide valuable resources for evaluating and developing intelligent security solutions, paving the way for more robust and effective intrusion detection and behavior analytics systems.

2.7 Critical Analysis of Literature Review

As advancements in cybersecurity continue to evolve, the integration of Machine Learning (ML), Deep Learning (DL), and User and Entity Behavior Analytics (UEBA) frameworks has significantly enhanced anomaly detection capabilities. However, amidst these advancements, there remains a critical need for gap analysis to identify areas for further research and improvement. This section aims to conduct a comprehensive gap analysis focusing on the application of ML, DL, and UEBA methodologies in log anomaly detection. By critically examining existing literature, this analysis seeks to uncover areas where current approaches may fall short in addressing emerging challenges such as cultural complexities, data volume management, and resource constraints. Through this gap analysis, the paper endeavors to pave the way for future research endeavors aimed at optimizing anomaly detection techniques and bolstering cybersecurity measures in an increasingly complex digital landscape.

While the Theory of Reasoned Action (TRA) offers a structured approach to understanding technology adoption, its application in the context of ML and UEBA for log anomaly detection has both strengths and limitations. TRA effectively highlights the importance of behavioral intention, attitude, and subjective norms in shaping adoption decisions. However, its cognitive focus may overlook factors like usability and organizational constraints. TRA's limited capacity to address cultural complexities necessitates supplementation with other theories for comprehensive understanding, particularly in diverse settings.

User and Entity Behavior Analytics (UEBA) is a crucial cybersecurity framework utilizing advanced analytics and machine learning to detect anomalies in network behavior. Notable research works, such as (Liu, 2021) and (Sharma, Pokharel and Joshi, 2020) have introduced innovative approaches like LSTM-GaN and LSTM-based Autoencoder for insider threat detection, achieving high accuracy rates. (Rasheed Yousef and Mahmoud Jazzar, 2021) further, validate UEBA's effectiveness in reducing false positives and detecting insider threats.

However, UEBA faces challenges including lengthy training data duration, false positives, data quality, resource constraints, and selecting appropriate methodologies for risk score calculation. Organizations must focus on refining risk score calculation methodologies to minimize false alarms effectively. By leveraging domain knowledge and advanced technologies, UEBA implementation can be optimized, enhancing anomaly detection accuracy and effectively safeguarding networks against evolving cybersecurity threats.

Machine Learning (ML) algorithms are pivotal in network security, particularly in anomaly and attack detection. Support Vector Machine (SVM) excels in classification tasks but can be slow and sensitive to parameter tuning. Studies like (Fan et al., 2020) and (Krishnaveni et al. 2020) showcase their efficacy in enhancing Magnetic Anomaly Detection (MAD) and intrusion detection, respectively. Random Forest (RF) overcomes SVM's limitations, demonstrating high accuracy in detecting intrusions and fraud, as evidenced by (Primartha and Tama, 2017) and (Lin and Jiang, 2020). Decision Tree (DT) methods like the RADE approach and (Douiba et al., 2022) work offer resource-efficient anomaly detection, especially in IoT environments. Naive Bayes (NB), while simple, proves effective in real-time applications such as surveillance video anomaly detection and IoT-based intrusion detection. Each algorithm presents distinct strengths and weaknesses, necessitating careful consideration of dataset and network characteristics when selecting the most suitable one. Understanding the nuanced performance of these ML algorithms is crucial for optimizing anomaly detection systems across diverse security scenarios.

Deep Learning (DL) models, including Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks, have transformed anomaly detection, especially in log data analysis, by adeptly handling complex patterns and dependencies. ANNs, valued for their scalability and fault tolerance, have demonstrated exceptional accuracy, such as (Abbasi et al., 2021) achievement of 99.98% in IoT anomaly detection. CNNs excel in capturing spatial features, as evidenced by (Yadav et al., 2020), who highlighted their effectiveness in log anomaly detection. LSTM networks, designed

for sequential data, have garnered attention for their remarkable accuracy, with studies like (Vinayakumar et al., 2017)reporting 99.6% accuracy in log anomaly detection. Additionally, (Zhao et al., 2021) showcased LSTM's superiority in unsupervised large-scale security log analysis compared to traditional models like PCA, one-class SVM, and GMM.

DL models, particularly ANN, CNN, and LSTM networks, hold substantial promise in log anomaly detection due to their capacity to capture intricate data patterns. However, ongoing research is essential to tackle existing challenges and improve the robustness and applicability of these models in real-world scenarios. Despite advancements, challenges persist in log anomaly detection using DL models, including handling unstructured data and addressing instability.

The landscape of Intrusion Detection Systems (IDSs) and User and Entity Behavior Analytics (UEBA) suffers from outdated datasets, hindering progress in technology. Researchers advocate for new datasets reflecting real-world network dynamics, emphasizing qualities like realism, labeling, variability, correctness, updatability, reproducibility, and shareability. Synthetic data injection is proposed but faces challenges in sharing due to confidentiality concerns. Efforts like Loghub offer vast, labeled, and unlabeled real-world system log datasets, aiding in developing intelligent log analysis techniques for anomaly detection, duplicate issue identification, parsing, compression, and unsupervised methods. These resources provide valuable support for research and practical application. The research underscores the importance of these datasets in advancing the field of Log-based IDS and highlights areas for potential improvement, particularly focusing on enhancing evaluation scores for the BGL dataset while acknowledging the already successful performance of the HDFS dataset in multiple metrics.

In summary, there exists a significant gap in research regarding the utilization of UEBA alongside effective ML or DL algorithms for log anomaly detection, particularly with a focus on risk score analysis. Additionally, there is a need for further investigation into automated log preprocessing methods to efficiently manage large and complex log data volumes in a cost-effective and resource-efficient manner, which would greatly benefit small and medium-sized enterprises. This area requires more thorough examination and exploration. Furthermore, addressing challenges in log anomaly detection using DL models, such as handling unstructured data and addressing instability and applicability in real-world datasets, requires extensive research efforts. Moreover, the development of a benchmark UEBA-based log anomaly dataset would be highly valuable for the research community.

2.8 Summary

This chapter reviews anomaly detection in system logs, focusing on Machine Learning (ML) and User and Entity Behavior Analytics (UEBA). It covers the limitations of traditional rule-based approaches and the benefits of ML and Deep Learning (DL) algorithms like decision trees, random forests, support vector machines, and neural networks for log anomaly detection. The integration of UEBA with ML to enhance detection by analyzing behavioral deviations is discussed, including applications in computer vision. Benchmark datasets are examined for evaluation. The chapter highlights the potential of these advanced techniques, identifies research gaps, and suggests future directions for improving ML and UEBA integration in anomaly detection.

CHAPTER III:

METHODOLOGY

The chapter details the methodology adopted for this study to achieve the research objectives and answer the research questions. It begins with an overview of the research problem, highlighting the challenges faced by enterprise businesses in migrating to the cloud and the increasing demand for robust network observability through log analysis. The section on research problems and questions formulate the guiding research questions. The research design outlines the approach and strategies employed. The dataset details section provides insights into the BGL Dataset used in the study, discussing data sources, structure, and preprocessing steps.

A step-by-step methodology describes the entire process, including data preprocessing, feature selection, the rationale behind specific methods, model development, and performance evaluation. The data analysis section elaborates on techniques for exploratory data analysis, aiming to extract valuable insights. The UEBA score analysis explains how risk scores are assigned based on behavior and examines the effectiveness of UEBA in enhancing anomaly detection. The research design limitations section acknowledges constraints and challenges encountered, including data limitations and computational constraints. The chapter concludes by summarizing the key points, emphasizing the importance of the chosen methods in addressing the research problem and setting the stage for the subsequent chapters.

3.1 Overview of the Research Problem

Behavioral Analysis and Anomaly Detection using log files represent an emerging and significant field of study due to their explanatory and predictive capabilities. User and Entity Behavior Analytics (UEBA) is gaining traction within data analytics, though it remains underexplored in the context of log analysis. Behavioral analytics and risk scoring are pivotal in identifying anomaly candidates and mitigating false alarms, as highlighted by Liu *et al.*, (2018)

The burgeoning field of Behavioral Analytics (Khaliq, Abideen Tariq and Masood, 2020) is recognized as a contemporary research trend within Management Information Systems (MIS), primarily due to its effectiveness in integrating both explanatory and predictive modeling. With rapid advancements in social media, sensing technologies, and mobile computing, there is a pressing need for innovative analytical methodologies to decipher user perspectives, thereby enhancing recommendation systems and adaptive interfacing. Motiwalla *et al.*, (2019) Conducted a comprehensive survey examining various studies adopting a behavioral approach, which considers user interactions and behaviors alongside user attributes, including emotion detection from text, images, and videos, and fraud detection in crowdfunding, among other applications.

Despite these advancements, Behavioral Analysis for Log Anomaly Detection, particularly in cloud-based environments, has seen limited progresss. The system log behavior approach involves analyzing user and entity behavior to understand the WHO, WHICH, and HOW factors of an anomaly, rather than merely identifying the WHAT factor as done in conventional methodologies. This shift necessitates parsing through massive volumes of logs to extract meaningful insights about user and entity behaviors.

The primary challenges in log-based behavioral analysis include the high volume of data, the increased rate of false alarms, and the computational demands associated with real-time risk score calculation. Effective anomaly detection through UEBA requires sophisticated algorithms capable of sifting through extensive log data to identify deviations from normal behavior. Furthermore, computational time and resource constraints pose significant hurdles, necessitating optimization techniques, and scalable architectures to ensure efficient processing.

In summary, while UEBA offers a robust framework for detecting anomalies within log data, addressing the challenges of false alarm rates, computational efficiency, and resource limitations is crucial for its practical implementation and widespread adoption in cloud-based environments. Future research should focus on developing advanced analytical techniques and computational models to enhance the precision and scalability of behavioral anomaly detection systems.

3.2 Research Design

The methodology deployed for the proposed log-based anomaly detection project involves key processes such as pre-processing the chosen data, transforming the data into a structured and comprehensible format, feature learning, building ML-based prediction models, and evaluating the machine learning performance using appropriate evaluation metrics.

This project is using a preprocessed form of the widely used BGL Dataset for conducting experiments sourced from https://github.com/logpai/loghub/tree/master/BGL (*Loghub/BGL at Master · Logpai/Loghub · GitHub*, .) The raw BGL Log is an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) in Livermore, California, with 131,072 processors and 32,768GB memory. The log contains alert and non-alert messages identified by alert category tags. The label information is amenable to alert detection and prediction research. It has been used in several studies on log parsing, anomaly detection, and failure prediction. (Oliner and Stearley, 2007; Zhu *et al.*, 2023)

The primary research method for this study involves a thorough literature review and experimental analysis of various machine learning models on the BGL Log Dataset, focusing on User and Entity Behavior Analysis and Log-Based Anomaly Detection.

3.2.1 Methodology Overview

(i). Data Preprocessing and Feature Extraction:

Data Preprocessing: Perform data cleaning, encoding, oversampling, and normalization ,to prepare the dataset for analysis.

Feature Engineering: Extract and transform key features from the log data, focusing on behavioral, and contextual aspects relevant to anomaly detection.

(ii). User and Entity Behavior Analysis:

Behavioral Feature Extraction: Develop profiles based on user and entity interactions, capturing patterns and anomalies in their behavior.

Risk Scoring: Use ML methods to calculate risk scores, assessing the likelihood of anomalous behavior based on behavioral trends.

(iii). Machine Learning Model Development:

Model Selection: Conduct a comprehensive review of current industry practices and academic research to identify state-of-the-art machine learning and deep learning models suitable for log analysis and anomaly detection.

Model Training: Experiment with a variety of supervised and unsupervised learning algorithms, including Isolation Forest, Random Forest, Gradient Boosting, and Neural Networks.

(iv). Performance Evaluation:

Evaluation Metrics: Assess the models using metrics such as Precision, Recall, F1-Score, and AUC to determine their effectiveness in detecting anomalies.

Optimization: Refine the models to enhance computational efficiency and scalability, addressing resource constraints.

The research study, scheduled from January to June 2024, implemented a detailed experimental setup to validate the proposed methodology for log-based anomaly detection. The flow chart of methodology is depicted in Figure 3-1. This approach aims to overcome the challenges of high false alarm rates, computational inefficiency, and resource constraints, ultimately enhancing the precision and scalability of anomaly detection systems in cloud-based environments. By leveraging a behavioral approach for log analysis, this research intends to provide valuable insights for entity-oriented decision-making in enterprise solutions. Additionally, integrating behavioral analysis into anomaly detection could extend its applications to various fields, including cyber forensics, identity management, and social network profiling. This study aims to showcase the potential of behavioral analytics to transform anomaly detection and improve security and operational efficiency across multiple domains.

3.2.2 Flowchart

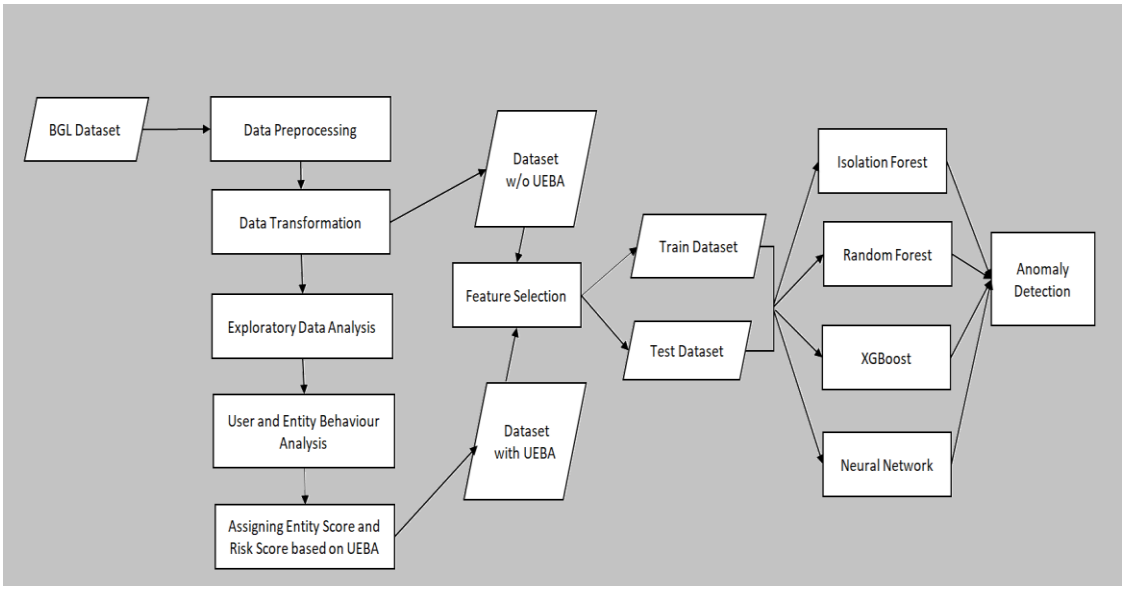


Figure 3-1 Methodology

3.3 Dataset

Overview of the Blue Gene/L (BGL) Dataset

The BGL dataset is a significant resource for system log analysis, originating from the Blue Gene/L supercomputer, developed by IBM and operated at Lawrence Livermore National Laboratory (LLNL). This supercomputer was a groundbreaking system in high-performance computing, holding the title of the world's fastest supercomputer during its operational period. The BGL dataset contains comprehensive system logs that capture various events, including hardware and software errors, warnings, and informational messages typically in plain text, with fields like timestamps (indicating when the event occurred), message types (e.g., ERROR, WARNING, INFO), event IDs (unique identifiers for different types of events), and detailed messages describing the events.

BGL Dataset Applications

Anomaly Detection: The BGL dataset is instrumental in developing and evaluating algorithms to detect unusual patterns in system logs, helping to preemptively address potential system issues.

Fault Diagnosis: It is crucial to diagnose the root causes of system failures by analyzing sequences of logged events, aiding in understanding and mitigating underlying issues.

Predictive Maintenance: By analyzing trends and patterns in log data, researchers can forecast potential system failures, allowing proactive maintenance to enhance system reliability and uptime.

BGL Dataset in LogHub

LogHub is a large collection of diverse system log datasets curated to facilitate research in log analysis, anomaly detection, and related fields. The BGL dataset within LogHub is

particularly valued for its real-world data from a high-performance computing environment. It provides a realistic basis for testing and validating log analysis techniques, as the logs come from an actual operational supercomputer. Moreover the log captures a wide range of system events, offering a rich dataset for analyzing different types of faults and anomalies. It is widely used as a benchmark dataset in academic and industrial research to compare the effectiveness of various log analysis methods and tools.

The BGL dataset has been pivotal in advancing the field of system log analysis. Researchers use it to:

Develop Log Parsing Algorithms: Automatic log parsing is essential for converting raw log data into structured formats that are easier to analyze. The BGL dataset provides the necessary data for training and testing these algorithms.

Event Correlation: By studying the relationships between different log events, researchers can build models that correlate events leading up to failures, crucial for diagnosing complex issues in large-scale systems.

Machine Learning Models: The dataset is used to train machine learning models for tasks such as anomaly detection and fault prediction, automating the monitoring and maintenance of the supercomputing environment.

BGL Dataset Metadata

The raw BGL Log is an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) in Livermore, California, with 131,072 processors and 32,768GB memory. The log contains alert and non-alert messages identified by alert category tags. The label information is amenable to alert detection and prediction research. (Oliner and Stearley, 2007; Zhu *et al.*, 2023).

The preprocessed version of BGL data sourced from Loghub contains 2000 data instances and 13 attributes. The attributes are shown in Figure 3-2 and the BGL dataset snapshot is presented in Table 3-1.

According to the drawings from the literature review of the dataset landscape within the Intrusion Detection Systems (IDS) and User and Entity Behavior Analytics (UEBA) domain, outlined in section 2.6, Figure 2-3 provides an overview of the practical evaluation of essential performance metrics for leading Deep Learning (DL) algorithms applied to both the BGL and Hadoop Distributed File System (HDFS) datasets. The visualization highlights a notable area of research potential focused on enhancing the Precision score, particularly concerning the BGL dataset, which substantiated the selection of this dataset for this research study.

In conclusion, the BGL dataset is a critical resource for understanding and improving the reliability of high-performance computing systems. Its detailed and diverse logs provide invaluable insights into system behavior, helping researchers develop advanced techniques for log analysis, anomaly detection, and fault diagnosis. The inclusion of the BGL dataset in LogHub further underscores its importance and utility in the field of system log research. This dataset supports a wide range of applications, including anomaly detection, fault diagnosis, and predictive maintenance, making it a cornerstone for studies aiming to enhance the reliability and efficiency of large-scale computing systems.

```
BGL_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   LineId         2000 non-null   int64 
 1   Label          2000 non-null   object
 2   Timestamp      2000 non-null   int64 
 3   Date           2000 non-null   object
 4   Node           1965 non-null   object
 5   Time           2000 non-null   object
 6   NodeRepeat     1965 non-null   object
 7   Type           1962 non-null   object
 8   Component      2000 non-null   object
 9   Level          2000 non-null   object
 10  Content        2000 non-null   object
 11  EventId        2000 non-null   object
 12  EventTemplate  2000 non-null   object
dtypes: int64(2), object(11)
memory usage: 203.2+ KB
```

Figure 3-2
BGL Dataset Attributes

LineId	Label	Timestamp	Date	Node	Time	NodeRepeat	Type	Component	Level	Content	EventId	EventTemplate
1	-	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.675872	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	E77	instruction cache parity error corrected
2	-	1117838573	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.53.276129	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	E77	instruction cache parity error corrected
3	-	1117838976	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.49.36.156884	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	E77	instruction cache parity error corrected
4	-	1117838978	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.49.38.026704	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	E77	instruction cache parity error corrected
5	-	1117842440	2005.06.03	R23-M0-NE-C:J05-U01	2005-06-03-16.47.20.730545	R23-M0-NE-C:J05-U01	RAS	KERNEL	INFO	63543 double-hummer alignment exceptions	E3	<*> double-hummer alignment exceptions
6	-	1117842974	2005.06.03	R24-M0-N1-C:J13-U11	2005-06-03-16.56.14.254137	R24-M0-N1-C:J13-U11	RAS	KERNEL	INFO	162 double-hummer alignment exceptions	E3	<*> double-hummer alignment exceptions
7	-	1117843015	2005.06.03	R21-M1-N6-C:J08-U11	2005-06-03-16.56.55.309974	R21-M1-N6-C:J08-U11	RAS	KERNEL	INFO	141 double-hummer alignment exceptions	E3	<*> double-hummer alignment exceptions
8	-	1117848119	2005.06.03	R16-M1-N2-C:J17-U01	2005-06-03-18.21.59.871925	R16-M1-N2-C:J17-U01	RAS	KERNEL	INFO	CE sym 2, at 0x0b85eee0, mask 0x05	E18	CE sym <*>, at <*>, mask <*>
9	APPREAD	1117869872	2005.06.04	R04-M1-N4-I:J18-U11	2005-06-04-00.24.32.432192	R04-M1-N4-I:J18-U11	RAS	APP	FATAL	ciod: failed to read message prefix on control stream (CioStream socket to 172.16.96.116:33569	E33	ciod: failed to read message prefix on control stream (CioStream socket to <*>:<*>
10	APPREAD	1117869876	2005.06.04	R27-M1-N4-I:J18-U01	2005-06-04-00.24.36.222560	R27-M1-N4-I:J18-U01	RAS	APP	FATAL	ciod: failed to read message prefix on control stream (CioStream socket to 172.16.96.116:33370	E33	ciod: failed to read message prefix on control stream (CioStream socket to <*>:<*>
11	-	1117942120	2005.06.04	R30-M0-N7-C:J08-U01	2005-06-04-20.28.40.767551	R30-M0-N7-C:J08-U01	RAS	KERNEL	INFO	CE sym 20, at 0x1438f9e0, mask 0x40	E18	CE sym <*>, at <*>, mask <*>
12	-	1117955341	2005.06.05	R25-M0-N7-C:J02-U01	2005-06-05-00.09.01.903373	R25-M0-N7-C:J02-U01	RAS	KERNEL	INFO	generating core.2275	E67	generating core.<*>
13	-	1117955392	2005.06.05	R24-M1-N8-C:J09-U11	2005-06-05-00.09.52.516674	R24-M1-N8-C:J09-U11	RAS	KERNEL	INFO	generating core.862	E67	generating core.<*>
14	-	1117956980	2005.06.05	R24-M1-NB-C:J15-U11	2005-06-05-00.36.20.945796	R24-M1-NB-C:J15-U11	RAS	KERNEL	INFO	generating core.728	E67	generating core.<*>
15	-	1117957045	2005.06.05	R20-M1-N8-C:J04-U01	2005-06-05-00.37.25.012681	R20-M1-N8-C:J04-U01	RAS	KERNEL	INFO	generating core.775	E67	generating core.<*>
16	-	1117959501	2005.06.05	R24-M0-NE-C:J14-U11	2005-06-05-01.18.21.778604	R24-M0-NE-C:J14-U11	RAS	KERNEL	INFO	generating core.3276	E67	generating core.<*>
17	-	1117959513	2005.06.05	R21-M1-N2-C:J11-U01	2005-06-05-01.18.33.830595	R21-M1-N2-C:J11-U01	RAS	KERNEL	INFO	generating core.1717	E67	generating core.<*>
18	-	1117959563	2005.06.05	R24-M0-N8-C:J04-U11	2005-06-05-01.19.23.822135	R24-M0-N8-C:J04-U11	RAS	KERNEL	INFO	generating core.3919	E67	generating core.<*>
19	-	1117973759	2005.06.05	R31-M0-NE-C:J05-U11	2005-06-05-05.15.59.416717	R31-M0-NE-C:J05-U11	RAS	KERNEL	INFO	generating core.2079	E67	generating core.<*>
20	-	1117973786	2005.06.05	R36-M0-NA-C:J06-U01	2005-06-05-05.16.26.686603	R36-M0-NA-C:J06-U01	RAS	KERNEL	INFO	generating core.1414	E67	generating core.<*>
21	-	1117973919	2005.06.05	R33-M0-N4-C:J02-U11	2005-06-05-05.18.39.396608	R33-M0-N4-C:J02-U11	RAS	KERNEL	INFO	generating core.3055	E67	generating core.<*>
22	-	1117974206	2005.06.05	R22-M0-ND-C:J10-U11	2005-06-05-05.23.26.239153	R22-M0-ND-C:J10-U11	RAS	KERNEL	INFO	generating core.201	E67	generating core.<*>
23	-	1117974463	2005.06.05	R27-M0-N6-C:J10-U01	2005-06-05-05.27.43.336565	R27-M0-N6-C:J10-U01	RAS	KERNEL	INFO	generating core.1125	E67	generating core.<*>
24	-	1117975251	2005.06.05	R26-M1-N8-C:J17-U11	2005-06-05-05.40.51.726735	R26-M1-N8-C:J17-U11	RAS	KERNEL	INFO	generating core.412	E67	generating core.<*>
25	-	1117976658	2005.06.05	R36-M1-N8-C:J17-U01	2005-06-05-06.04.18.406158	R36-M1-N8-C:J17-U01	RAS	KERNEL	INFO	generating core.7828	E67	generating core.<*>
26	-	1117977497	2005.06.05	R33-M1-NB-C:J06-U01	2005-06-05-06.18.17.802159	R33-M1-NB-C:J06-U01	RAS	KERNEL	INFO	generating core.5570	E67	generating core.<*>
27	-	1117979227	2005.06.05	R01-M1-N7-C:J04-U11	2005-06-05-06.47.07.157021	R01-M1-N7-C:J04-U11	RAS	KERNEL	INFO	generating core.8275	E67	generating core.<*>
28	-	1117982609	2005.06.05	R35-M1-NE-C:J05-U01	2005-06-05-07.43.29.979844	R35-M1-NE-C:J05-U01	RAS	KERNEL	INFO	generating core.4183	E67	generating core.<*>
29	-	1117984124	2005.06.05	R36-M1-NF-C:J11-U01	2005-06-05-08.08.44.281729	R36-M1-NF-C:J11-U01	RAS	KERNEL	INFO	generating core.6545	E67	generating core.<*>

Table 3-1 Dataset Snapshot

3.4 Methodology

The primary objective of this methodology is to develop and evaluate machine learning models for log anomaly detection along with UEBA, with a specific emphasis on reducing false alarms and optimizing computational efficiency. This study focuses on exploring the performance of three distinct algorithms—Isolation Forest, Random Forest, and XGBoost—as well as a simple neural network By systematically comparing these models, the study aims to identify the most effective approach for accurate and efficient log anomaly detection. Through a comprehensive process involving data collection, preprocessing, model training, hyperparameter tuning, and validation, this methodology seeks to balance predictive power and computational resource usage, ensuring robust and scalable anomaly detection solutions.

3.4.1 Data Preprocessing and Feature Engineering

Data Preprocessing

In this stage, data cleaning is performed to handle null values and remove duplicate entries. Various attributes are analyzed, and target variables are encoded for binary classification.

Null Value Treatment: The dataset is examined for null values, and null entries are observed in the columns ['Node', 'NodeRepeat', 'Type']. Null values in ['Node', 'NodeRepeat'] are replaced with zero, as these attributes will eventually be converted to numeric. Null values in ['Type'] are replaced with 'UNKNOWN' to facilitate the encoding of this categorical variable.

```
LineId          0.00
Label           0.00
Timestamp       0.00
Date            0.00
Node            1.75
Time            0.00
NodeRepeat      1.75
Type            1.90
Component       0.00
Level           0.00
Content         0.00
EventId         0.00
EventTemplate   0.00
dtype: float64
```

Table 3-2 Null value counts

Attribute	Null replacement value
Node	0
NodeRepeat	0
Type	UNKNOWN

Table 3-3 Null Value Treatment

Duplicate Value Treatment: The dataset is examined for duplicate values, and no duplicate entries are found row-wise. However, the columns 'Node' and 'NodeRepeat' are identical. Therefore, 'NodeRepeat' can be dropped.

Identifying Unique Values: The attributes are analyzed, and the unique values for each attribute are counted.

ATTRIBUTES	UNIQUE VALUES
A	64
B	2
C	16
E	2
G	17
H	2
EventId	120
Type	2
Component	5
Year	2
Month	8
Day	31
Hour	24
weekday	7

Table 3-4 Unique Value Counts

Label Marking: The target labels are designated as NORMAL and ANOMALY for binary classification. The Isolation Forest algorithm predicts the labels as [1, -1], while Random Forest, Gradient Boost, and Neural Network predict the labels as [0, 1]. Therefore, two target variables are defined to facilitate performance evaluation.

Feature Engineering

During this stage, key features are extracted from the log data, with a focus on behavioral and contextual aspects relevant to anomaly detection. The attributes are examined for their data types, and the numerical data is analyzed for statistical spread. Categorical values are

converted to numerical values using dummy variable encoding. The preprocessed, feature-extracted data then undergoes detailed analysis, as explained in the subsequent section. The dataset is normalized, split into training, testing, and validation sets, and then oversampled using SMOTE for model building.

Feature Extraction: Timestamp and Text processing is done for feature extraction.

Timestamp Processing: Temporal parameters such as year, month, date, hour, day, and weekday are extracted from the 'Time' attribute.

Text Processing**:** The 'Node' attribute, which follows a specific format, is parsed to extract multiple fields for further analysis of their impact on the target variable. For example, the 'Node' value " R17-M1-N9-C: J05-U01" can be split into five different alphanumeric attributes and is shown in Table 3-5.

	Timestamp	Component	Level	EventId	Year	Month	Day	Hour	weekday	A	B	C	E	G	H
0	1117838570	KERNEL	INFO	E77	2005	6	3	15	4	R02	M1	N0	C	J12	U11
1	1117838573	KERNEL	INFO	E77	2005	6	3	15	4	R02	M1	N0	C	J12	U11
2	1117838976	KERNEL	INFO	E77	2005	6	3	15	4	R02	M1	N0	C	J12	U11
3	1117838978	KERNEL	INFO	E77	2005	6	3	15	4	R02	M1	N0	C	J12	U11
4	1117842440	KERNEL	INFO	E3	2005	6	3	16	4	R23	M0	NE	C	J05	U01

Table 3-5 Splitting of Node values

Changing Variable Datatypes: After feature extraction, the variables are analyzed for their data types and categorized as numeric, ordinal, or categorical. The data types of certain attributes are then adjusted to be numeric or categorical as appropriate.

```
 0   Timestamp  2000 non-null   int64
 1   Component  2000 non-null   object
 2   Level      2000 non-null   object
 3   EventId    2000 non-null   object
 4   Year       2000 non-null   int64
 5   Month      2000 non-null   int64
 6   Day        2000 non-null   int64
 7   Hour       2000 non-null   int64
 8   weekday    2000 non-null   int64
 9   A          2000 non-null   object
 10  B          2000 non-null   object
 11  C          1896 non-null   object
 12  E          2000 non-null   object
 13  G          2000 non-null   object
 14  H          2000 non-null   object
dtypes: int64(6), object(9)
memory usage: 234.5+ KB
```

Table 3-6 Variable datatypes

Identifying Numerical Variables The categorical variables ['EventId', 'A', 'C', 'G'] contain multiple alphanumeric values. Therefore, the data type can be changed to numeric by converting the strings to integers using ASCII encoding.

Identifying Categorical Variables; The ordinal variables ['Year', 'Month', 'Day', 'Hour', 'Weekday'] are treated as categorical for dummy variable encoding to preserve their temporal effect on the target variables.

Exploratory Data Analysis: Exploratory detailed analysis using Class, Component, Level, and Temporal variables is conducted to gain insights and is detailed in Section 3.6, Data Analysis.

Dummy Variable Encoding: Dummy variable encoding transforms categorical variables into binary vectors (dummy variables), representing each category with binary features. This conversion allows categorical data to be used as input for machine learning algorithms, ensuring they can effectively process and interpret the categorical information.

Normalization : The dummy-encoded categorical variables and numerical variables are then normalized using the Standard Scaler method.

Standard Scaler Normalization

Standard Scaler normalization transforms numerical features to have a mean of zero and a standard deviation of one, ensuring all features are on a comparable scale. This normalization is crucial for many machine learning algorithms that rely on distance metrics or gradient-based optimization. Standard Scaler works by subtracting the mean of each feature and then dividing by the standard deviation of that feature. Mathematically, it can be represented as:

$$Z = \frac{X - \mu}{\sigma}$$

where

- Z is the standardized value,
- X is the original value,
- μ is the mean of the feature,
- σ is the standard deviation of the feature.

Train and Test Data Split

After normalizing the data, the next step involves splitting it into three subsets using a ratio of 0.7:0.3.

1. Training Set: This subset is used to train the machine learning models. It contains a majority portion of the data, typically around 70-80%, depending on the size of the dataset and specific requirements.

2. Validation Set: This subset is used to tune the hyperparameters of the models and to evaluate their performance during training. It helps prevent overfitting by providing an independent dataset that the model has not seen during training. The validation set is used iteratively to adjust the model's parameters until the best performance is achieved.

3. Test Set: This subset is used to evaluate the final performance of the trained model after all parameter tuning and model selection decisions have been made. It serves as an unbiased estimate of the model's performance on unseen data.

Typically, the dataset is split randomly into these subsets while ensuring that each subset represents the overall distribution of the data. This ensures that the evaluation metrics obtained from the test set are reliable indicators of the model's performance in real-world scenarios.

SMOTE Oversampling:

Next, the dataset undergoes oversampling using the SMOTE Algorithm to reduce the data imbalance.SMOTE (Synthetic Minority Over-sampling Technique) oversampling is utilized in scenarios where the dataset exhibits significant class imbalance, such as the case in our dataset where anomalies represent only 7.9% of the total data.

Addressing the imbalance between normal and anomaly classes is crucial for effective anomaly detection using machine learning algorithms. Imbalanced data can bias models, potentially overlooking important anomaly patterns. SMOTE addresses this by creating synthetic instances of the minority class, enhancing its representation in the training dataset. This balanced training data improves the model's ability to generalize and accurately detect anomalies during deployment. It's essential to apply SMOTE exclusively to the training set to maintain unbiased evaluation metrics on the validation and test sets, ensuring robust performance in real-world scenarios.

Overall, SMOTE oversampling stands as a pivotal preprocessing step, enhancing the reliability and effectiveness of anomaly detection models in the face of class imbalance challenges.

3.4.2 Model Development

The primary objective of this stage is to develop and evaluate machine learning models for log anomaly detection, with a specific emphasis on reducing false alarms and optimizing computational efficiency. This study focuses on exploring the performance of three distinct algorithms—Isolation Forest, Random Forest, and XGBoost—as well as a simple neural network.

From the literature review, it is evident that the majority of log anomaly detection problems utilize supervised machine learning algorithms such as Support Vector Machines (SVM), Naive Bayes, Decision Trees, and Random Forests (RF). Additionally, deep learning techniques have been studied for their application in log anomaly detection, particularly Autoencoders, which are used for unsupervised anomaly detection by learning a compressed representation of the logs and identifying anomalies based on reconstruction error, and Long Short-Term Memory (LSTM) networks, which are used for automatic log pattern identification and anomaly detection by capturing temporal dependencies in the log data. Moreover, it has been observed that XGBoost, a robust algorithm widely used for credit card fraud detection and other anomaly detection tasks, is rarely exploited in log anomaly detection scenarios.

Model Selection

. Considering the objective of achieving optimal computational and evaluation performance, the following models are selected for experimentation in log anomaly detection: Isolation Forest (IF), Random Forest (RF), XGBoost, and a simple Neural Network. Each model has been chosen based on its unique characteristics, computational efficiency, and potential to minimize false alarms.

1. Isolation Forest (IF):

Type: Unsupervised

Isolation Forest is an anomaly detection algorithm that isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The logic behind this is that anomalies are few and different, making them easier to isolate. The algorithm constructs an ensemble of trees (isolation trees) to perform the isolation, where the path length to isolate a point is used as the measure of normality.

The anomaly score for a data point is calculated as:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where:

- $E(h(x))$ is the average path length of point x,
- $c(n)$ is the average path length of unsuccessful search in a Binary Search Tree.

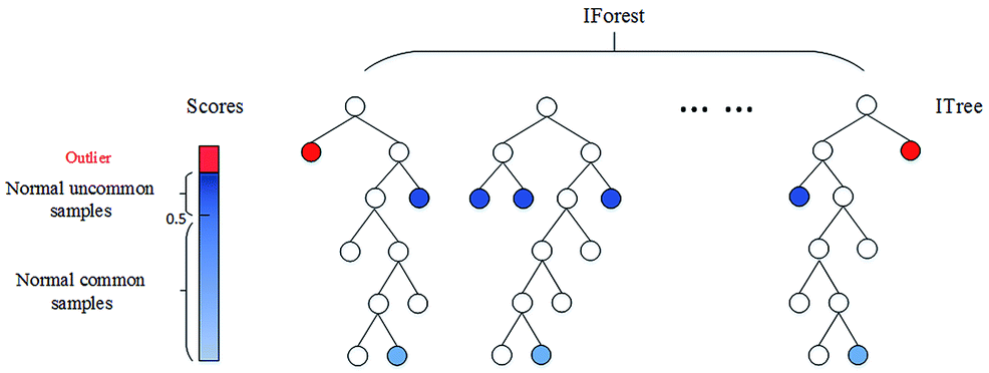


Figure 3-3 Isolation Forest

Source:https://donghwa-kim.github.io/iforest.html

Salient Features:

- Unsupervised: Does not require labeled data.
- Computational Efficiency: Linear time complexity to the number of samples.
- High-Dimensional Data: Effective in identifying anomalies in high-dimensional datasets

Rationale: Isolation Forest is chosen for its efficiency in detecting anomalies in high-dimensional data, especially in scenarios with scarce labeled data. Its unsupervised nature, computational efficiency, and linear time complexity make it ideal for real-time anomaly detection. By isolating observations, it effectively identifies anomalies due to their few and distinct nature, ensuring robust detection.

2. Random Forest (RF):

Type: Supervised

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. Each tree in the forest is built from a bootstrap sample from the training set, and during tree construction, each split is selected from a random subset of features.

The decision function for Random Forest can be represented as:

$$f(x) = \frac{1}{T} \sum_{t=1}^{T} h_t(x)$$

where:

- T is the number of trees,
- h_t is the prediction of the t-th tree.

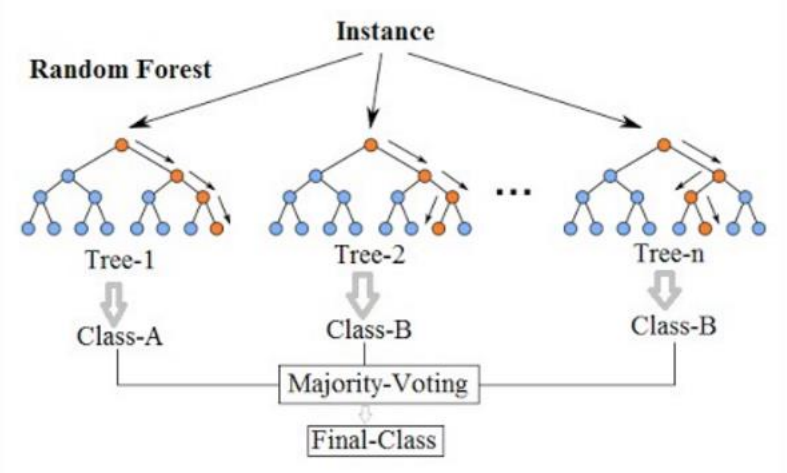


Figure 3-4 Random Forest
Source:https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2

Salient Features:

- Robustness: Reduces overfitting by averaging multiple trees.
- Scalability: Handles large datasets with high-dimensional features.
- Interpretability: Provides feature importance measures.

Rationale: Random Forest is chosen for its robustness and ability to handle large, high-dimensional datasets efficiently. Combining multiple decision trees, RF improves generalization, accuracy, and interpretability, and is less prone to overfitting compared to single decision trees.

3. XGBoost:

Type: Supervised

XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the gradient boosting framework, using decision trees as base learners. XGBoost adds regularization to the boosting procedure, enhancing its performance and preventing overfitting.

The objective function for XGBoost is:

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where:

- l is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i,
- Ω is a regularization term to control the complexity of the model.

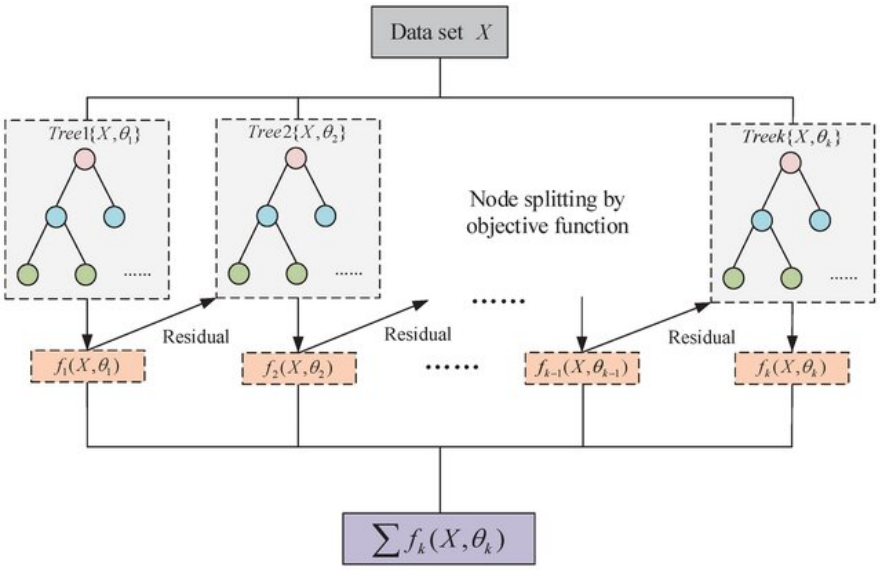


Figure 3-5 XGBoost

Source: Scheme of XGBost operation. Source: (GUO et al., 2020).

Salient Features:

- High Performance: Efficient in terms of speed and accuracy.
- Regularization: Reduces overfitting.
- Scalability: Capable of handling large datasets.

Rationale: XGBoost is chosen for its high performance and scalability in handling large datasets and complex feature spaces. Its use of gradient boosting, combined with regularization techniques, makes it a powerful tool for supervised anomaly detection. Despite its rare use in log anomaly detection, its success in similar tasks suggests it can effectively reduce false alarms and enhance predictive accuracy.

4. Neural Network:

Type: Supervised

A Neural Network is a computational model inspired by the human brain, consisting of interconnected units (neurons) that process information in layers. A simple feedforward neural network, used in this study, consists of an input layer, one or more hidden layers, and an output layer. The network learns to map inputs to outputs through backpropagation, adjusting weights to minimize the loss function.

The output of a neuron is calculated as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where:

- f is the activation function,
- w_i are the weights,
- x_i are the inputs,
- b is the bias term.

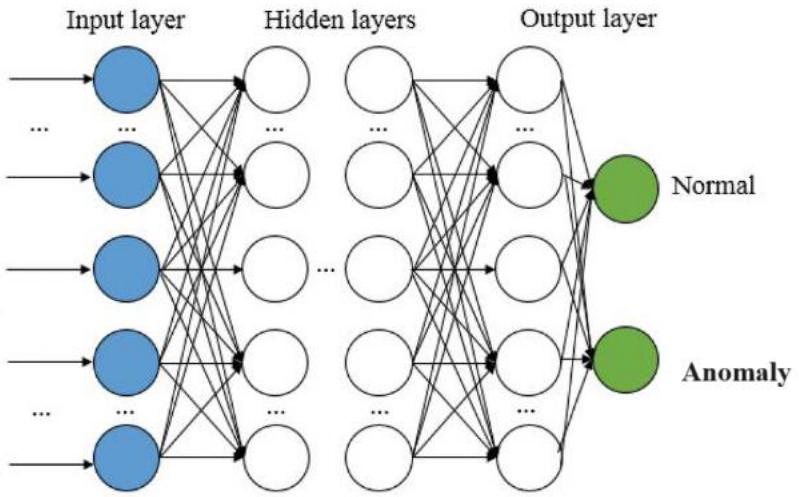


Figure 3-6 Neural Networks

Salient Features:

- Flexibility: Capable of learning complex patterns.
- Scalability: Can be adjusted in complexity by varying the number of layers and neurons.
- Efficiency: A simple architecture provides a balance between learning capability and computational cost.

Rationale: A simple Neural Network is chosen for its balance between complexity and computational efficiency, making it suitable for real-time log anomaly detection tasks. Unlike deep learning models such as LSTMs, which can be computationally intensive, a simple neural network provides sufficient learning capability without high computational costs. This pragmatic approach offers flexibility and scalability in model learning, ideal for scenarios where complexity needs to be balanced with efficiency.

By experimenting with these selected models, the study aims to identify the optimal approach that balances predictive accuracy, computational efficiency, and reduced false alarms in log anomaly detection tasks.

Model Training

The process of training machine learning models involves fitting them to the training data, adjusting their parameters, and tuning hyperparameters to optimize performance. This section outlines the training methodologies for four different models: Isolation Forest (IF), Random Forest (RF), XGBoost, and a simple Neural Network.

1. Isolation Forest (IF): Isolation Forest is an unsupervised learning algorithm used to identify anomalies in the dataset. The model is fitted on the training data to learn the patterns of normal and anomaly logs. Key parameters such as the number of estimators and contamination rate are adjusted to optimize the model's performance. The number of estimators controls the number of trees in the forest, while the contamination rate defines the proportion of outliers in the data.

2. Random Forest (RF): Random Forest is a supervised learning algorithm that builds multiple decision trees and merges them to get a more accurate and stable prediction. The model is trained on labeled training data, and hyperparameters such as the number of trees, maximum depth, and minimum sample split are tuned. Adjusting these parameters helps improve the model's accuracy and prevent overfitting. The number of trees determines how many decision trees are in the forest, the maximum depth limits how deep the trees can grow, and the minimum sample split sets the minimum number of samples required to split an internal node.

3. XGBoost: XGBoost is an efficient and scalable implementation of gradient boosting framework by leveraging decision trees. The model is fitted on labeled training data, and hyperparameters such as learning rate, maximum depth, subsample ratio, and the number of estimators are optimized to enhance predictive performance. The learning rate controls the step size at each iteration while moving toward a minimum of the loss function. The maximum depth sets the maximum depth of a tree, the subsample ratio determines the fraction of samples to be used for fitting the individual base learners, and the number of estimators specifies the number of boosting rounds.

4. Neural Network: The Neural Network model is designed as a simple feedforward architecture with input, hidden, and output layers. The network is trained on labeled data using a suitable optimizer such as Adam and a loss function like binary cross-entropy. Hyperparameters, including the number of layers, neurons per layer, and learning rate, are adjusted to balance complexity and computational efficiency. The number of layers and neurons per layer determine the network's capacity to learn complex patterns, while the learning rate influences the speed and stability of the training process.

Model Validation

Model validation ensures the effectiveness and reliability of machine learning models. This section outlines the process of model validation, which includes a simple train-validation split and hyperparameter tuning, for log anomaly detection models. The steps involved are data splitting, hyperparameter tuning, and model evaluation.

1. Data Split

To ensure a robust evaluation, the balanced data obtained using the Synthetic Minority Over-sampling Technique (SMOTE) is split into training and validation datasets in a 70:30 ratio. The training set is used to train the models, while the validation set is reserved for validating the models' performance. This split helps in maintaining a fair assessment of the models by ensuring that the validation data remains unseen during training.

2. Hyperparameter Tuning

Hyperparameter tuning is conducted using the random search technique on the training set. The Random Search technique involves randomly sampling the hyperparameter space, which is more efficient than evaluating all possible combinations as done in a grid search. By applying random search, the best hyperparameters are identified based on their performance on the validation set. This step is crucial for optimizing the models' performance and ensuring they generalize well to unseen data.

3. Model Evaluation

Once the optimal hyperparameters are identified, the models are evaluated on the validation set using a set of selected evaluation metrics. These metrics include accuracy, precision, recall, F1 score, false positive rate (FPR), and false negative rate (FNR). Evaluating the models with these metrics provides a comprehensive understanding of their performance, highlighting areas that may require further hyperparameter tuning and optimization. The confusion matrix derived for each model is presented in APPENDIX D.

The process of model validation involves splitting the data into training and validation sets, performing hyperparameter tuning using random search, and evaluating the models on the validation set. This approach ensures that the models are not only well-tuned but also thoroughly validated for their performance in detecting anomalies in log data. By using a variety of evaluation metrics, the effectiveness and reliability of the models are assessed, paving the way for further refinement and optimization.

Hyperparameter Tuning and Optimization

Hyperparameter tuning is a crucial step in the model development process, as it aims to identify the best set of hyperparameters that improve the model's performance on the validation dataset. This section outlines the approach for tuning and optimizing the hyperparameters for the selected models: Isolation Forest, Random Forest, XGBoost, and the Neural Network.

The hyperparameter tuning is performed using the Random Search technique. Random Search randomly samples the hyperparameter space instead of evaluating all possible combinations. It is more efficient than Grid Search and often finds good hyperparameters faster.

The hyperparameter optimization process begins by defining the search space, which involves specifying the range or list of values for each hyperparameter to be tuned. Following this, the Random Search technique is employed to identify the best hyperparameters. Once the optimal hyperparameters are determined, the model is evaluated on the validation set to assess its performance. This process may be iterated as necessary, refining the search space based on the results and insights gained from previous iterations, to further enhance the model's accuracy and efficiency.

By following this structured approach to hyperparameter tuning and optimization, the study aims to enhance the performance of the machine learning models for log anomaly detection, ensuring they are well-tuned for accuracy, precision, and computational efficiency.

Hyperparameters to Tune

1. Isolation Forest (IF)

PARAMETER	DESCRIPTION	VALUE
n_estimators	The number of trees in the forest.	155
max_samples	The number of samples to draw from the dataset to train each tree.	5
contamination	The proportion of outliers in the data.	0.0665
max_features	The number of features to consider when looking for the best split.	9

Table 3-7
IF Hyperparameters

2. Random Forest (RF)

PARAMETER	DESCRIPTION	VALUE
max_depth	The maximum depth of the tree.	14
min_samples_split	The minimum number of samples required to split an internal node.	50
min_samples_leaf	The minimum number of samples required to be at a leaf node.	10
max_features	The number of features to consider when looking for the best split.	10

Table 3-8
RF Hyper paarmeters

3. XGBoost

PARAMETER	DESCRIPTION	VALUE
max_depth	The maximum depth of a tree.	3
learning_rate	The step size shrinkage used to prevent overfitting.	0.2
subsample	The proportion of samples to be used for fitting the individual base learners.	0.9
colsample_bytree	The subsample ratio of columns when constructing each tree.	0.5
gamma	The minimum loss reduction required to make a further partition on a leaf node.	0.4

Table 3-9
XGBoost Hyperparameters

4. Neural Network

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                1940

 dense_1 (Dense)             (None, 10)                210

 batch_normalization (Batch  (None, 10)                40
 Normalization)

 dense_2 (Dense)             (None, 4)                 44

 dense_3 (Dense)             (None, 2)                 10

 dense_4 (Dense)             (None, 1)                 3

=================================================================
Total params: 2247 (8.78 KB)
Trainable params: 2227 (8.70 KB)
Non-trainable params: 20 (80.00 Byte)
_________________________________________________________________
```

Table 3-10
Neural Network Model Summary

PARAMETER	DESCRIPTION	VALUE
number of layers	The number of hidden layers	6
activation functions	The activation functions for each layer	RELU
learning rate	The learning rate for the optimizer	0.001
batch size	The number of samples per gradient update	50
epochs	The number of epochs to train the model	20

Table 3-11
Neural Network Hyperparameters

3.4.3 User and Entity Behavior Analysis

User and Entity Behavior Analysis (UEBA) involves detailed analysis of behavioral patterns within datasets to detect anomalies and assign risk scores to users, entities, or components. This section outlines the methodologies used for behavioral analysis, anomaly risk scoring, and entity risk scoring, as well as their integration with machine learning models for performance evaluation.

Entity Behavioral Analysis

Behavioral analysis begins with a detailed examination of dataset attributes at the entity or component level. The attribute **COMPONENT** is treated as the user or entity variable to do the analysis This analysis aims to identify specific behavioral patterns that may indicate anomalous activity. Visual inspection and statistical techniques are employed to capture and understand these patterns comprehensively, which is elaborated in Section 3.6 Data Analysis. This detailed analysis helps in identifying patterns and unusual behavior that could signify potential anomalies. The table lists the entities or components used for UEBA.

SLNO	ENTITY
1	**APP**
2	**DISCOVERY**
3	**HARDWARE**
4	**KERNEL**
5	**MMCS**

Table 3-12 Entity Components for UEBA

Anomaly Risk Scoring

Anomaly Risk scoring leverages the strengths of machine learning algorithms, particularly the high precision of Random Forest and the recall power of XGBoost. These algorithms are used to assign a risk score to each instance in the dataset. Isolation Forest-based scoring is used for a comparative analysis. The process involves training models using labeled data to develop Isolation Forest, Random Forest, and XGBoost models. Once trained, these models are applied to assign anomaly scores to each instance. By evaluating these assigned scores, the likelihood of anomalous behavior is assessed, allowing for the identification of trends and patterns indicative of potential anomalies. Entities or components within the dataset are subjected to detailed analysis based on the risk scores obtained from the machine learning models. This analysis determines which scoring metric better defines anomalies, enabling a deeper understanding of anomalous behavior and its implications. The insights and implications are detailed in Section 3.7.

Entity Score Assignment

From the prediction scores derived earlier, each entity is assigned both a normality score and an anomaly score. Anomaly scores assess the likelihood of anomalous behavior based on behavioral trends, while normality scores indicate regular or expected behavior. These scores quantify the potential risk associated with the entity, helping to identify high-risk candidates and mitigate false alarms. Components with high anomaly scores are flagged as potential risks, guiding proactive risk management strategies.

Integration of UEBA Score with Machine Learning Models

The anomaly risk scores obtained from the UEBA stage are integrated as an additional attribute into the machine learning models. Isolation Forest, Random Forest, XGBoost, and Neural Network algorithms are experimentally evaluated using these scores to assess their performance in anomaly detection. Comparative evaluations focus on accuracy, precision, recall, and other relevant metrics to identify the most effective model leveraging UEBA for detecting and mitigating anomalies in real-world scenarios.

This section explored the application of UEBA techniques for anomaly detection through detailed behavioral analysis and risk scoring. By leveraging machine learning algorithms such as Random Forest and XGBoost, the study empirically scores User and Entity Behavior to identify potential anomalies. The integration of anomaly risk scores into model training and evaluation provides a comprehensive approach to assessing computational and performance efficiency achieved by using UEBA.

Ultimately, this research advances the understanding and application of UEBA in cybersecurity and operational risk management. It aids in identifying high-risk insider threats and determining what constitutes an anomaly and who is behaving anomalously. This approach facilitates faster risk mitigation and enhances the overall security posture.

3.4.4. Performance Evaluation

Evaluating the performance of log anomaly detection models involves understanding how well these models distinguish between normal and anomalous data. Given the imbalanced nature of the dataset, where anomalies are rare compared to normal events, choosing the right evaluation metrics is crucial. Metrics such as confusion matrix, accuracy, precision, recall, F1 score, false positive rate (FPR), false negative rate (FNR), and computational time provide a comprehensive view of model performance from various angles. These metrics ensure that both the detection capability and efficiency are adequately assessed. This section elaborates on the various metrics used to evaluate the performance of both unsupervised and supervised algorithms in log anomaly detection.

1. Confusion Matrix

Definition: A confusion matrix is a table used to describe the performance of a classification model. It summarizes the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

Justification: The confusion matrix provides a detailed breakdown of correct and incorrect classifications, offering insights into specific types of errors the model makes, which is essential for understanding model behavior on imbalanced datasets.

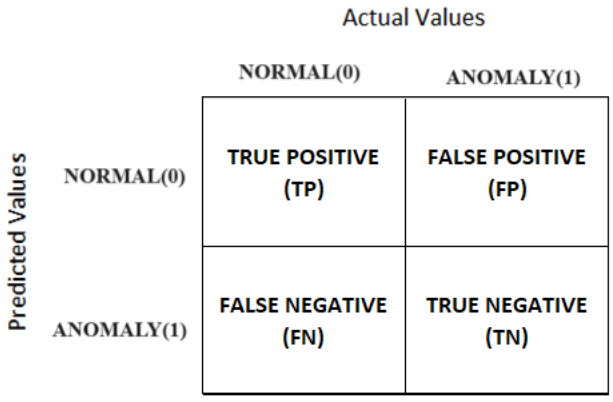


Figure 3-7
Confusion Matrix

2. Accuracy

Definition: Accuracy is the ratio of correctly predicted instances (both TP and TN) to the total number of instances.

Justification: While accuracy is a common metric, it can be misleading with imbalanced data since it may reflect the dominance of the majority class. However, it still provides a baseline measure of overall model performance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

3. Precision

Definition: Precision is the ratio of true positive predictions to the total positive predictions

Justification: Precision measures the accuracy of the positive predictions. High precision is crucial in anomaly detection to minimize false alarms (false positives).

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

4. Recall

Definition: Recall, or sensitivity, is the ratio of true positive predictions to the actual positive instances.

Justification: Recall indicates the model's ability to detect actual anomalies. High recall ensures that most anomalies are identified, which is critical in security contexts.

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

5. F1 Score

Definition: The F1 score is the harmonic mean of precision and recall .

Justification: The F1 score balances precision and recall, providing a single metric that accounts for both false positives and false negatives. It is particularly useful when dealing with imbalanced data.

$$\text{F-1 Score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

6. False Positive Rate (FPR)

Definition: FPR is the ratio of false positives to the total actual negatives

Justification: FPR measures the proportion of normal instances incorrectly classified as anomalies. A low FPR is essential to reduce the occurrence of false alarms.

$$\text{False Positive Rate} = \frac{FP}{FP + TN}$$

7. False Negative Rate (FNR)

Definition: FNR is the ratio of false negatives to the total actual positives

Justification: FNR measures the proportion of anomalies missed by the model. Minimizing FNR is critical to ensure that true anomalies are not overlooked.

$$\text{False Negative Rate} = \frac{FN}{FN + TP}$$

8. Computational Time

Definition: Computational time refers to the total time taken by the model to train and make predictions.

Justification: Computational efficiency is important for real-time anomaly detection, where timely responses are required. Evaluating computational time helps in selecting models that provide a balance between performance and efficiency.

In summary, evaluating log anomaly detection models requires a multi-faceted approach due to the imbalanced nature of the data. Metrics such as confusion matrix, accuracy, precision, recall, F1 score, FPR, FNR, and computational time collectively offer a comprehensive assessment of model performance. These metrics ensure that both the detection capabilities and efficiency of the models are thoroughly evaluated, leading to more reliable and effective anomaly detection in cybersecurity and operational risk management contexts.

3.4.5 Experimental Setup

Hardware Resources :

For this experiment, the hardware resources are specified to ensure that the computational tasks are handled efficiently. The following hardware configuration is used:

CPU: AMD Ryzen 7 4700U with Radeon Graphics, 2.00 GHz, RAM 16.0 GB

The AMD Ryzen 7 4700U is an 8core, 8-thread processor with a base clock speed of 2.00 GHz. It is part of AMD's mobile processor lineup, designed for high performance in portable devices. The integrated Radeon Graphics provide additional computational capabilities, useful for parallel processing tasks often required in machine learning. A total of 16 GB of RAM ensures that the system can handle large datasets and complex computations without significant memory bottlenecks. This amount of RAM is adequate for running multiple applications simultaneously, enabling efficient data processing, model training, and evaluation.

Software Resources:

The experiment utilizes several software resources to implement and evaluate the log anomaly detection models. The chosen programming language and libraries are well-suited for machine learning tasks, providing robust tools for data manipulation, model building, and visualization.

Programming

Python 3.6 : Python 3.6 is selected for its simplicity, readability, and extensive support for machine learning libraries. Python is widely used in the data science community, making it an ideal choice for this experiment

Libraries

Pandas: Used for data manipulation and analysis. Pandas provides data structures like DataFrames, which are essential for handling structured data efficiently.

```
import pandas as pd
```

Numpy: A fundamental library for numerical computing in Python. It provides support for arrays, matrices, and a collection of mathematical functions to operate on these data structures.

```
import numpy as np
```

Scikitlearn : A machine learning library that provides simple and efficient tools for data mining and data analysis. It is used for implementing the Isolation Forest, Random Forest, hyperparameter tuning and evaluation metrics.

```
from sklearn.ensemble import IsolationForest, RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import classification_report, confusion_matrix
```

TensorFlow and Keras: TensorFlow is an open-source platform for machine learning, and Keras is a high-level neural networks API running on top of TensorFlow. These are used to implement and train the Neural Network model.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Seaborn: A data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

```
import seaborn as sns
```

Matplotlib: A plotting library for the Python programming language and its numerical mathematics extension NumPy. It is used for creating static, interactive, and animated visualizations.

```
import matplotlib.pyplot as plt
```

These hardware and software resources collectively provide a powerful platform for developing, training, and evaluating machine learning models for log anomaly detection. The combination of a high-performance CPU, ample memory, and robust programming tools ensures that the experimental setup is well-equipped to handle the demands of this research.

Implementation of Workflow in Python

This section outlines the implementation workflow for log anomaly detection using Python, covering data preprocessing, model training, hyperparameter tuning, scoring, and evaluation. Key techniques include Isolation Forest, Random Forest, XGBoost, and a simple Neural Network, enhanced with UEBA scores for better performance assessment. The workflow ensures data preparation, efficient model training, and comprehensive performance evaluation, with UEBA scores providing a nuanced understanding of anomalous behavior. The detailed Technical workflow is comprehended in APPENDIX D.

3.5 Data Analysis

Exploratory detailed analysis using Class, Component, Level, and Temporal variables is conducted to gain insights and is detailed in the Data Analysis section. This analysis includes univariate, bivariate, and multivariate techniques to obtain meaningful insights. Visual representation of the data is achieved using Seaborn and Matplotlib plots, enhancing the interpretability of the analysis results. Univariate analysis focuses on the individual characteristics of each variable, bivariate analysis examines relationships between two variables, and multivariate analysis explores interactions among multiple variables, providing a comprehensive understanding of the data.

3.5.1 Univariate Analysis

Class Count Analysis

The first chart shows the distribution of the two classes in the dataset. Here is a breakdown:

- Class 0 (Normal): 1857 instances (92.85%)
- Class 1 (Anomaly): 143 instances (7.15%)

This indicates a significant class imbalance, with the anomalous class being a minority class, representing only 7.15% of the data.

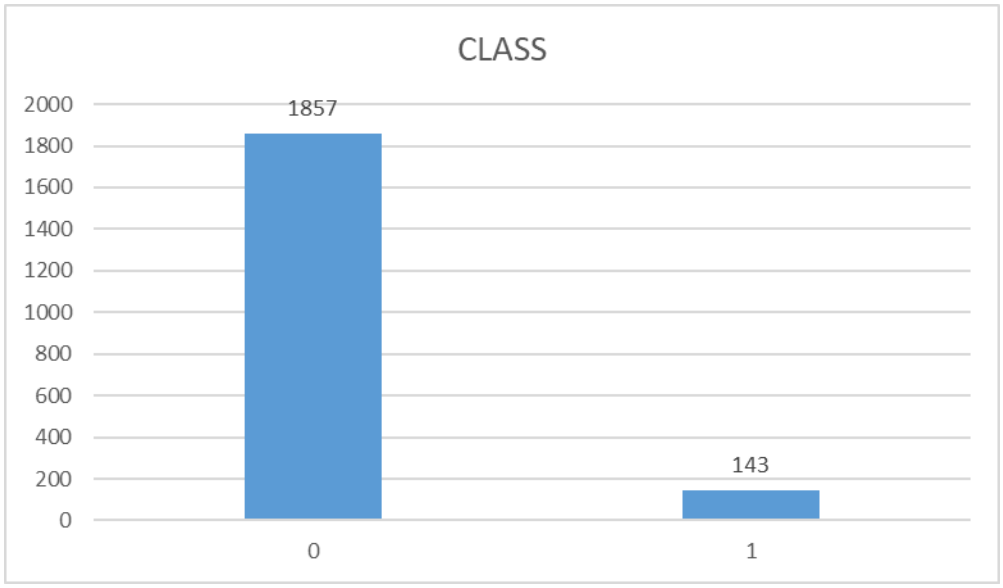


Figure 3-8 Class Count

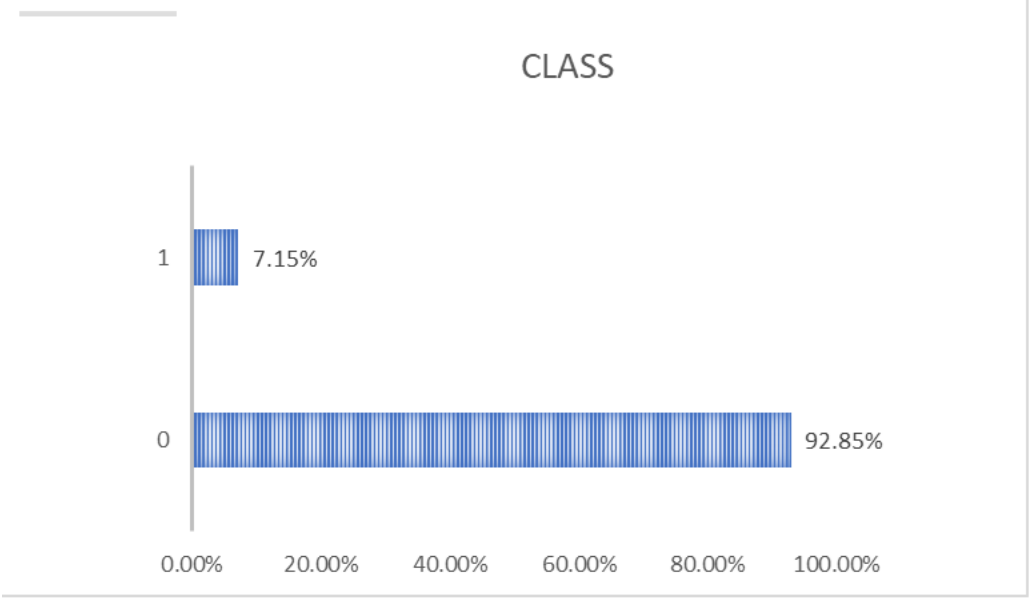


Figure 3-9 Percentage of Anomaly

Label Count Analysis

The second chart shows the distribution of different labels within the anomalous class (Class 1).

Here's a detailed count of each label:

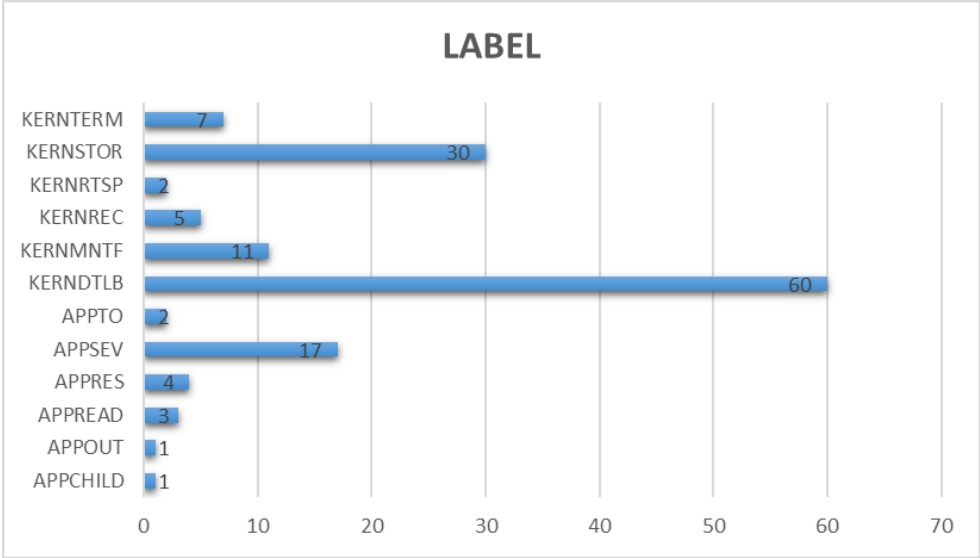


Figure 3-10 Label Distribution

LABELS	COUNT
APPCHILD	1
APPOUT	1
APPREAD	3
APPRES	4
APPSEV	17
APPTO	2
KERNDTLB	60
KERNMNTF	11
KERNREC	5
KERNRTSP	2
KERNSTOR	30
KERNTERM	7
Grand Total	**143**

Table 3-13 Label Counts-

Insights

1. Class Imbalance:

The significant imbalance between Class 0 and Class 1 can pose challenges for training machine learning models. Models trained on this dataset might be biased towards the majority class (Class 0), leading to poor performance in detecting anomalies (Class 1).

2. Label Distribution in Class 1:

Within the anomalous class, the label KERNDTLB has the highest count (60), indicating that this type of anomaly is the most common in the dataset.

KERNSTOR also has a significant count (30), followed by APPSEV (17) and KERNMNTF (11).

Other labels have relatively low counts, with APPCHILD and APPOUT being the least frequent (1 each).

Implications

Handling Class Imbalance: Oversampling Techniques such as SMOTE Oversampling for minority class or using algorithms that are robust to class imbalance (e.g., XGBoost) are necessary to improve model performance.

Label Distribution: Understanding the distribution of labels within the anomalous class provides insights into the nature of anomalies. This helps in feature engineering and selecting appropriate evaluation metrics that consider the imbalance.

By addressing these aspects, the study prepares the data for building robust anomaly detection models.

3.5.2 Bivariate Analysis

Bivariate Analysis- Class

The figure presents four box plots that compare two classes (0 and 1) across four variables: Weekday, Month, Day, and Hour.

For Weekdays, both classes show similar distributions with medians around mid-week days (3-4) and similar ranges, indicating weekdays is not a significant differentiator.

In the Month plot, Class 0 exhibits a wider spread with an interquartile range (IQR) from months 6 to 10, whereas Class 1 has a narrower range and a lower median, suggesting potential seasonal variation.

The Day plot reveals a noticeable difference: Class 0 has a wider spread and higher median day compared to Class 1, which features several outliers, indicating some specific days might be significant.

The Hour plot shows both classes with similar distributions and a median around the middle of the day, but Class 1 has a slightly higher median and wider spread, suggesting some variation in timing. - Class 0 shows a wider spread and a higher median day of the month compared to Class 1.

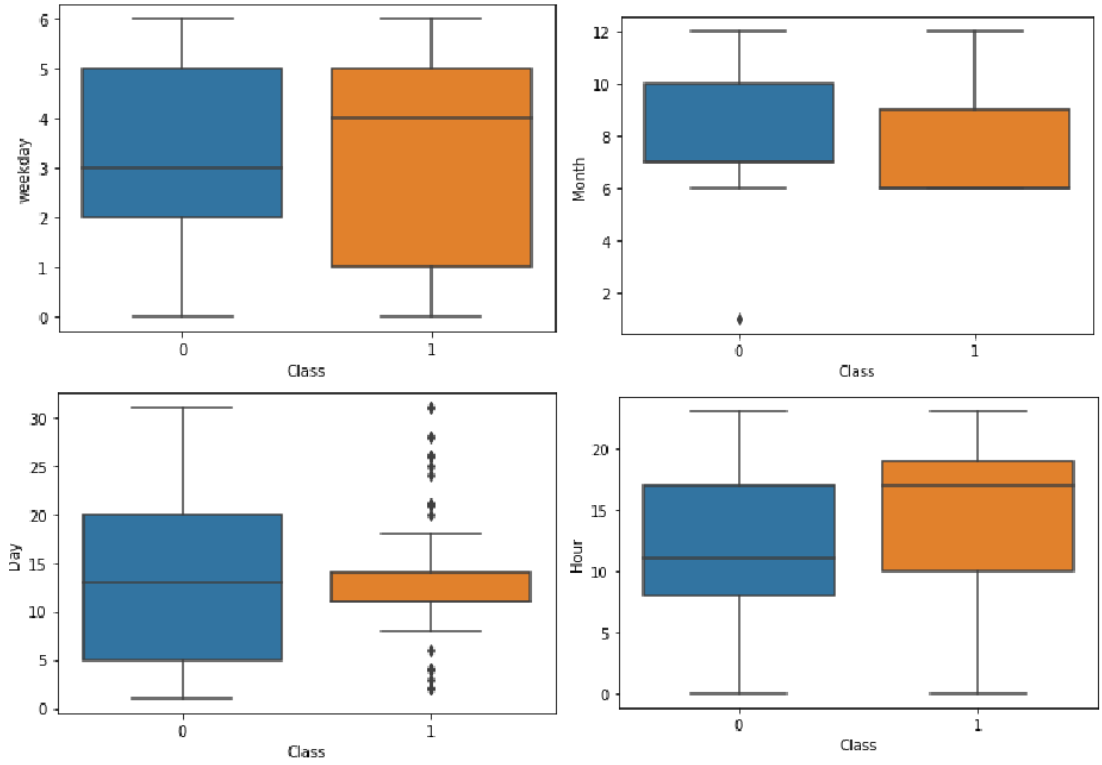


Figure 3-11 Bivariate Analysis -Class

Bivariate Analysis- Component

The figure presents four box plots comparing five components (KERNEL, APP, DISCOVERY, HARDWARE, MMCS) across four variables: Weekday, Month, Day, and Hour.

For Weekday, KERNEL and APP have similar distributions with mid-week medians, while DISCOVERY and HARDWARE show broader distributions, and MMCS has the narrowest spread around mid-week.

In the Month plot, KERNEL shows a wider range, APP has a high median with a narrow range, DISCOVERY has a lower median with some outliers, HARDWARE has a moderate spread, and MMCS shows a consistent distribution around mid-months.

For the Day variable, KERNEL and APP have wide distributions with similar medians, DISCOVERY shows a narrower range, HARDWARE has a low median with minimal spread, and MMCS shows a higher median and widespread.

The Hour plot indicates that KERNEL and APP have similar wide distributions around midday, DISCOVERY shows a wider range with some outliers, HARDWARE has a narrow range, and MMCS has a higher median with a moderate spread. These variations highlight differences in timing and distribution across the components.

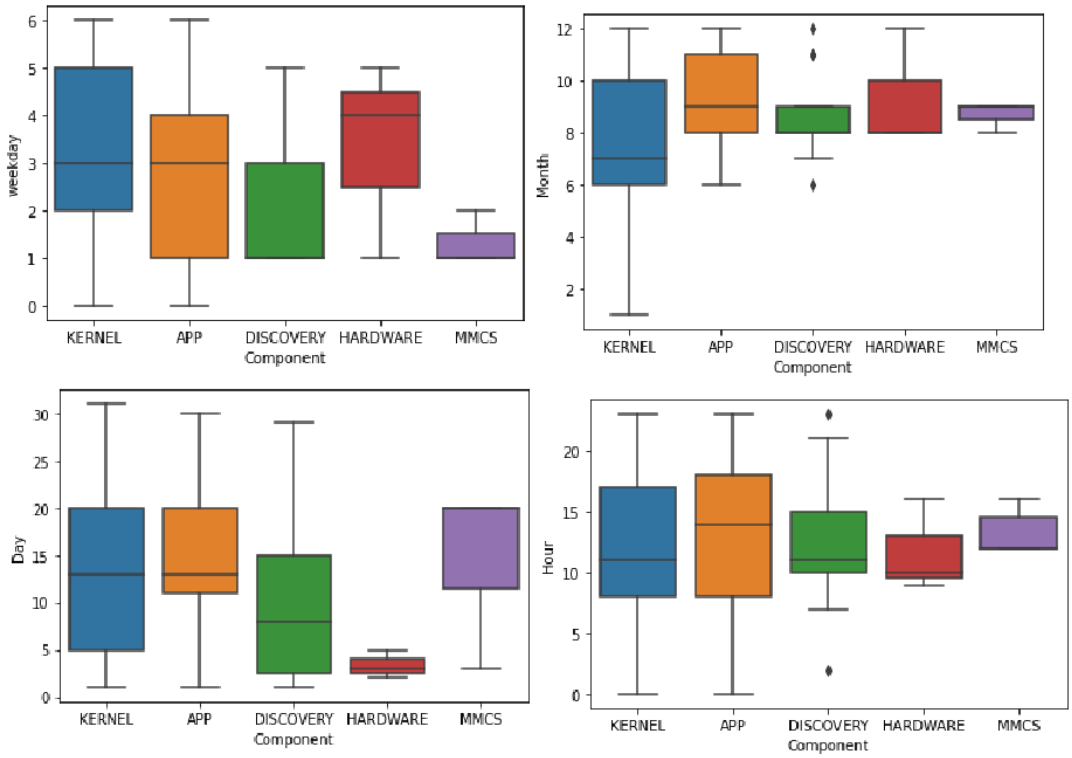


Figure 3-12 Bivariate Analysis -Component

Bivariate Analysis- Level

The figure showcases four box plots comparing five levels (INFO, FATAL, WARNING, SEVERE, ERROR) across four variables: Weekday, Month, Day, and Hour.

Insights

For Weekday, INFO and FATAL show broad distributions with mid-week medians, while WARNING, SEVERE, and ERROR exhibit narrower spreads with different medians, indicating variation across levels.

The Month plot shows INFO with a wide spread and higher median, FATAL with a narrower and lower median, WARNING with a tight range, SEVERE with minimal variation, and ERROR with a moderate spread around mid-months.

The Day variable reveals INFO having a wide distribution and higher median, FATAL with numerous outliers and a lower median, WARNING and SEVERE with narrow ranges and low medians, and ERROR with a higher median and wider spread.

For the Hour variable, INFO and FATAL show similar wide distributions around midday, WARNING has a tight range, SEVERE exhibits minimal spread with a low median, and ERROR shows a moderate spread with a higher median. These patterns highlight differences in timing and distribution across the levels, suggesting varying levels of significance for different time periods and dates.

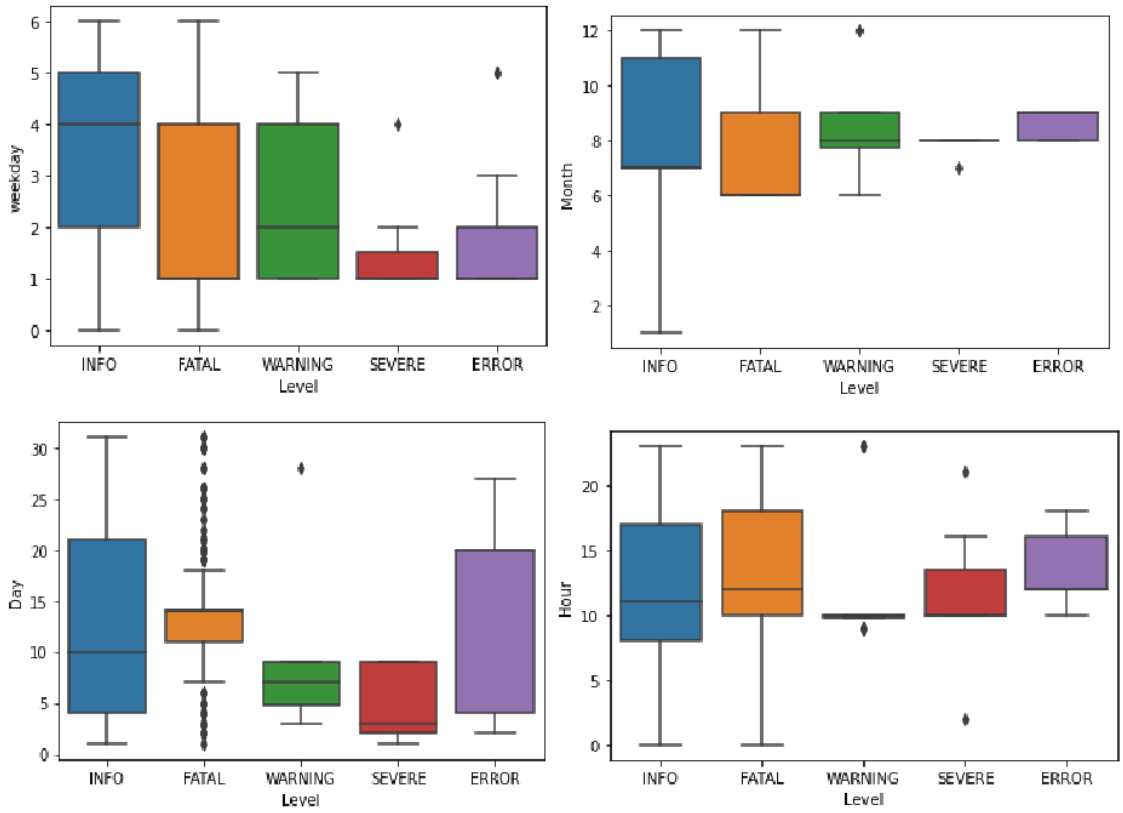


Figure 3-13 Bivariate Analysis -Level

Implications -Bivariate Analysis

The bivariate analysis of class, component, and level, each comparing variables such as Weekday, Month, Day, and Hour through box plots presents the following insights. For class, both classes have similar weekday distributions, but Class 0 shows a wider month spread, higher day median, and slightly varied hour distribution compared to Class 1. In component analysis, KERNEL and APP have mid-week medians, with KERNEL showing a wider month range, while HARDWARE and MMCS exhibit minimal day and hour spread. For level, INFO and FATAL display broad weekday and hour distributions, INFO shows a higher month spread, and ERROR indicates higher day variance.

The bivariant analysis suggests distinct patterns and variations across different categories, highlighting specific time-related characteristics that may influence class, component, and level distributions.

3.5.3 Multivariate Analysis

Multivariate Analysis-Component

The multivariate analysis of Components compares the various components (KERNEL, APP, DISCOVERY, HARDWARE, MMCS) across multiple dimensions: Class, Level, Hour, Month, Day, and Weekday, providing several key insights and implications.

Insights:

1. Component Distribution by Class and Level
    - KERNEL significantly dominates the distribution, with a higher count in both Class 0 and Class 1 compared to other components.
    - APP has a smaller count, while DISCOVERY, HARDWARE, and MMCS have minimal representation.
    - KERNEL has the highest count across all levels, particularly in INFO and ERROR levels.
    - APP, DISCOVERY, HARDWARE, and MMCS have much lower counts, reinforcing KERNEL's prominence across levels.
2. Temporal Distribution:
    - Hour: KERNEL and APP exhibit broader distributions throughout the day, with KERNEL showing high activity across all hours and APP having a peak around midday. DISCOVERY and HARDWARE have more confined hour ranges, while MMCS shows minimal variation.
    - Month: KERNEL has a wide distribution across months, indicating yearround significance, while APP and other components show narrower month ranges with some outliers.
    - Day: KERNEL and APP have wide day distributions, with KERNEL having higher variability. DISCOVERY and HARDWARE have limited day ranges, and MMCS shows minimal day variability.

- Weekday: KERNEL and APP show broad distributions across the week. DISCOVERY and HARDWARE exhibit moderate weekday variability, while MMCS has a very narrow weekday range.

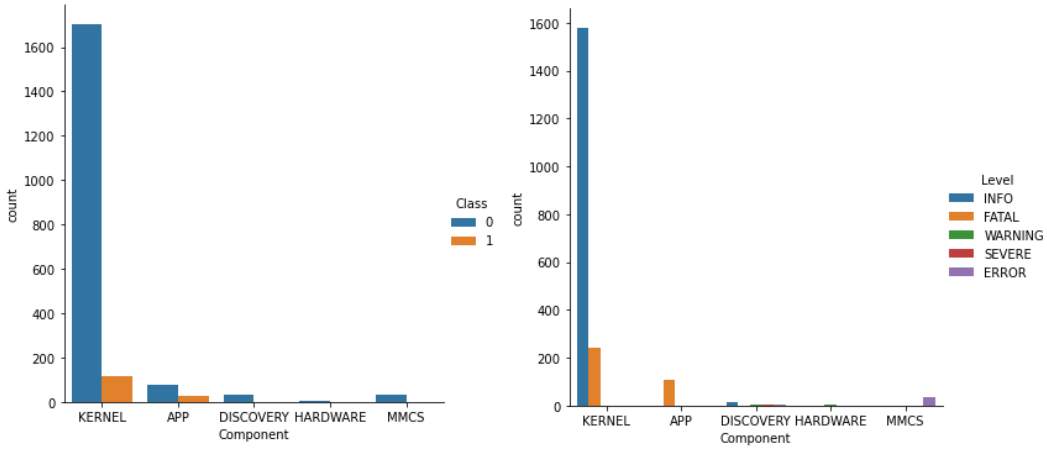


Figure 3-14
Component Distribution by Class and Level

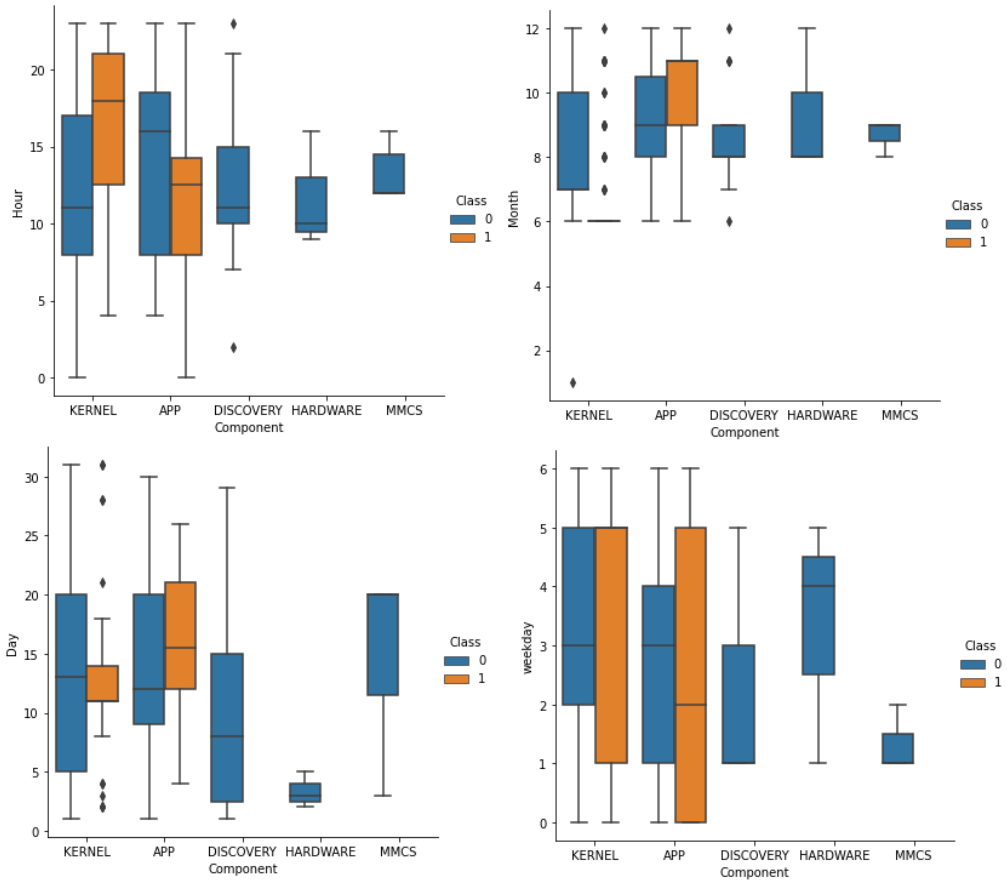


Figure 3-15
Temporal Distribution-Component

Multivariate Analysis-Level

The multivariate analysis of LEVEL compares the various levels (INFO, FATAL, WARNING, SEVERE, ERROR) and components (KERNEL, APP, DISCOVERY, HARDWARE, MMCS) across dimensions like Class, Hour, Month, Day, and Weekday.

Insights:

1. Level Distribution by Class and Component:
    - KERNEL has the highest count across all levels, particularly dominating the INFO level.
    - Class 0 shows significantly higher counts than Class 1 across most levels and components, especially for INFO and FATAL.
    - Components like APP, DISCOVERY, HARDWARE, and MMCS have much lower counts, indicating lesser significance compared to KERNEL.

2. Temporal Distribution:
    - Hour: INFO and FATAL show broad distributions across various hours, with Class 1 peaking in the afternoon for FATAL. Other levels like WARNING and SEVERE have narrower hour ranges.
    - Month: INFO and FATAL have wide month ranges, suggesting consistent issues throughout the year, while other levels show narrower distributions.
    - Day: INFO displays wide day distributions, with FATAL showing numerous outliers around the middle of the month. WARNING and SEVERE levels have more confined day ranges.
    - Weekday: INFO and FATAL have broad weekday distributions, while other levels like WARNING and SEVERE show narrower spreads, indicating specific days might be more prone to these levels of issues.

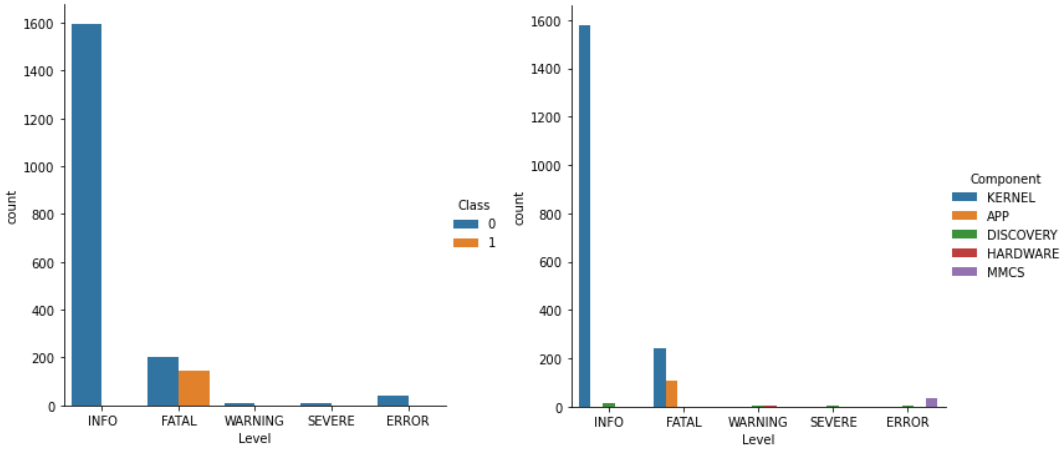


Figure 3-16
Level Distribution by Class and Component

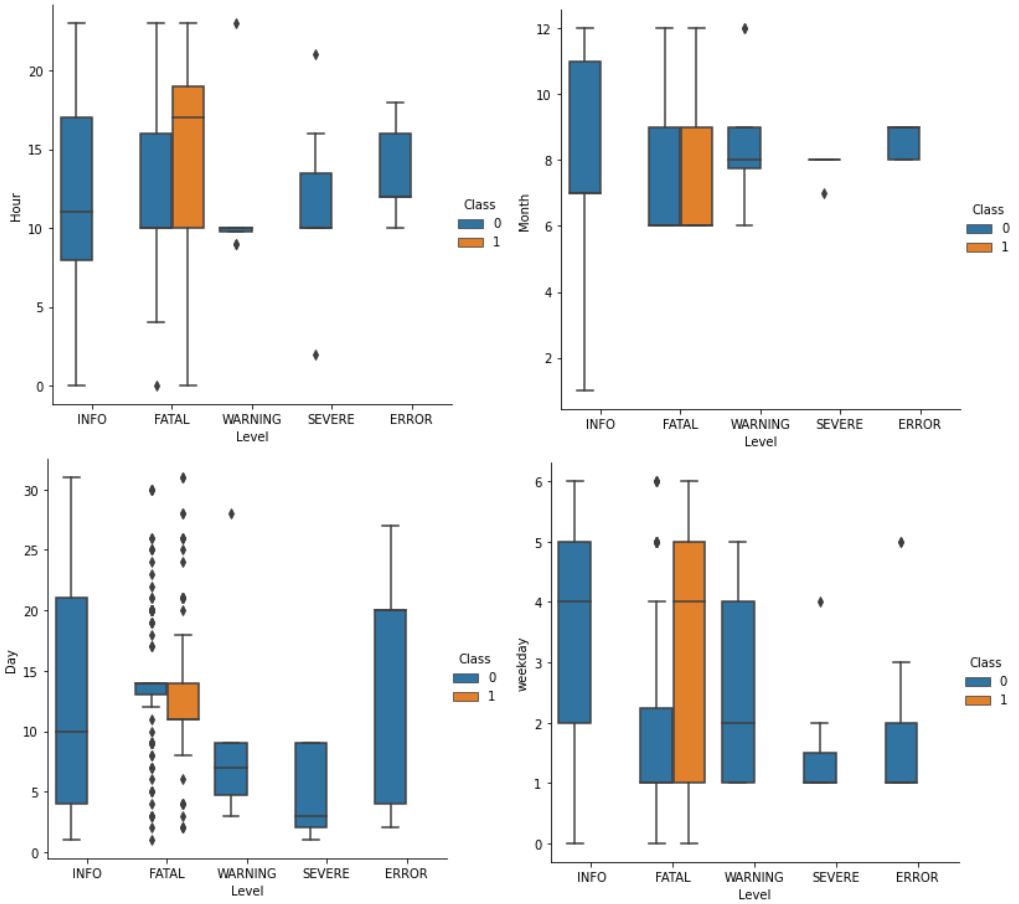


Figure 3-17
Temporal Distribution- Level

Implications-Multivariate Analysis

The multivariate analysis highlights the critical importance of the KERNEL and APP component, especially at the INFO level, suggesting it should be prioritized for monitoring and maintenance to prevent widespread operational impacts. While DISCOVERY is less prominent, they exhibit significant patterns during specific hours and days, necessitating targeted monitoring during peak times to prevent issues. HARDWARE and MMCS have minimal representation, indicating a lower operational impact, but periodic reviews are essential to avoid overlooking potential issues. The variability observed across hours, days, and months provides valuable insights for scheduling maintenance during off-peak times, particularly focusing on high-activity periods for INFO and FATAL levels to minimize disruptions.

Overall, the analysis underscores the need for a differentiated monitoring strategy for Class 0 and Class 1 and emphasizes the importance of focusing resources on key components while maintaining oversight on less significant ones to enhance system reliability and efficiency.

3.5.4 User and Entity Behaviour Analysis of Components

Component-wise UEBA is conducted using Class, Level, and Temporal variables to gain insights and is detailed in this section. The components KERNEL and APP are used as they are predominant in exploratory data analysis (EDA). Hardware, Discovery, and MMCS details are provided in Appendix B. This approach facilitates a comprehensive understanding of user and entity behavior, aiding in effective anomaly detection and risk assessment.

KERNEL

The analysis focuses on the KERNEL component's behavior across, various dimensions such as Class, Level, and Temporal parameters.

Insights

1. Class Distribution:
    - KERNEL occurrences are predominantly in Class 0 at both INFO and FATAL levels.
    - Class 1 shows a notable presence at the FATAL level, indicating critical issues are more frequent in this class.
2. Level Distribution:
    - The majority of KERNEL occurrences are at the INFO level, with a significant number at the FATAL level.
    - This highlights the need to address both routine informational issues and critical problems.
3. Temporal Distribution:
    - Hour: INFO level issues are distributed throughout the day, while FATAL issues for Class 1 peak in the afternoon and early evening.
    - Month: INFO level occurrences are spread throughout the year, while FATAL issues show several outliers, indicating specific months with higher critical issues.
    - Day: INFO level issues have a broad day distribution, while FATAL issues for Class 1 peak around the middle of the month with multiple outliers.

- Weekday: INFO level issues are evenly distributed across the weekdays, while FATAL issues exhibit higher variability, indicating certain days may be more prone to critical issues.

KERNEL-Class Analysis

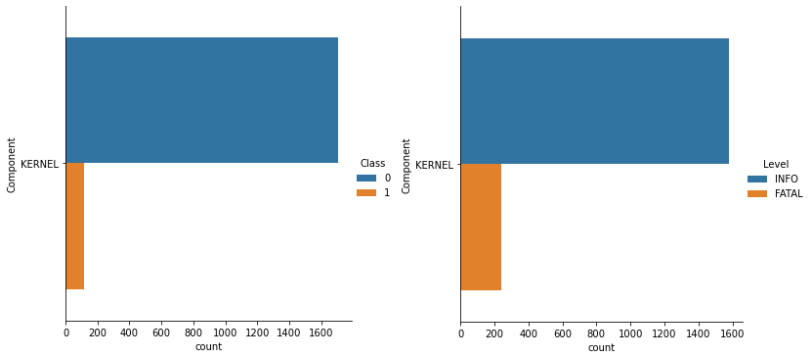


Figure 3-18 KERNEL -Class Analysis

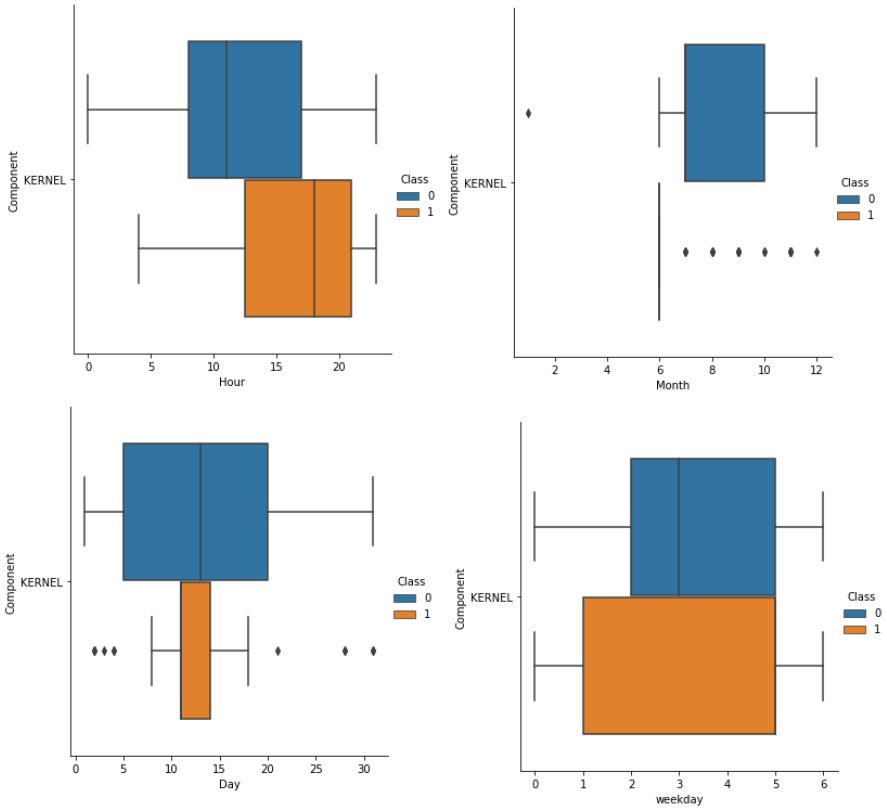


Figure 3-19 KERNEL -Temporal Analysis by Class

KERNEL-Level Analysis

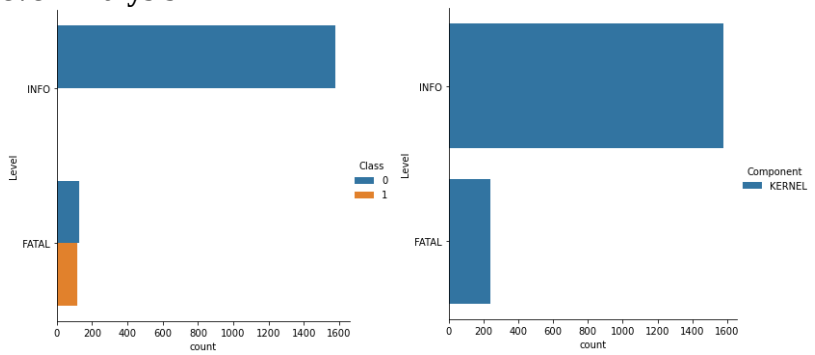


Figure 3-20 KERNEL Level Analysis

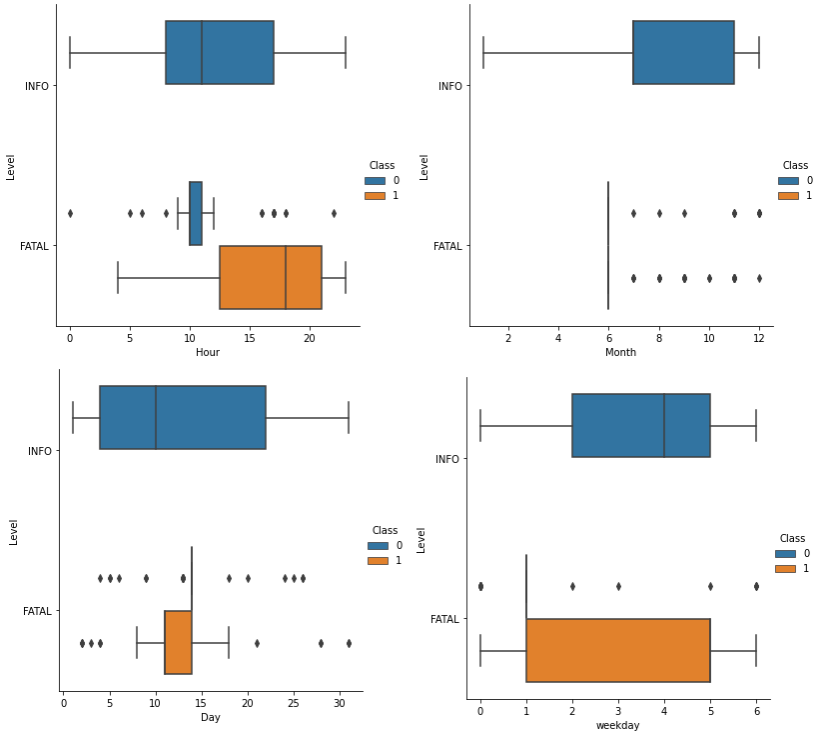


Figure 2-21 KERNEL -Temporal Analysis by Level

APP

The analysis focuses on the APP component's behavior across, various dimensions such as Class, Level, and Temporal parameters.

Insights

1. Class Distribution:

- The APP component shows a higher count in Class 0 compared to Class 1 at the FATAL level.
- Class 1 is significantly present at the FATAL level, indicating critical issues are more frequent in this class.

2. Level Distribution:

- The majority of APP occurrences are at the FATAL level.
- This highlights the need for focused attention on critical issues within the APP component.

3. Temporal Distribution:

- Hour: FATAL issues for APP are distributed throughout the day for Class 0, while Class 1 peaks in the afternoon and early evening.
- Month: FATAL level occurrences are spread across the latter part of the year, with both classes showing variability.
- Day: FATAL issues for APP have a broad day distribution, with Class 1 showing peaks around the middle of the month.
- Weekday: Both classes show a wide weekday distribution, indicating that critical issues are spread across the week without a specific pattern.

APP-Component Analysis

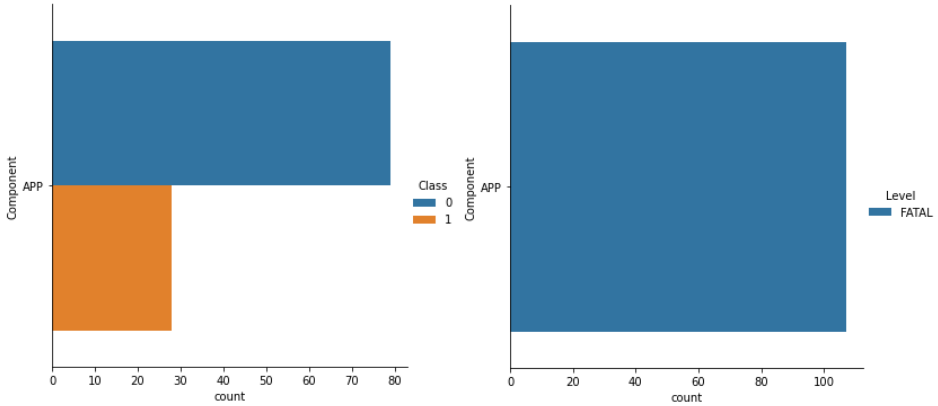


Figure 3-22 APP -Class Analysis

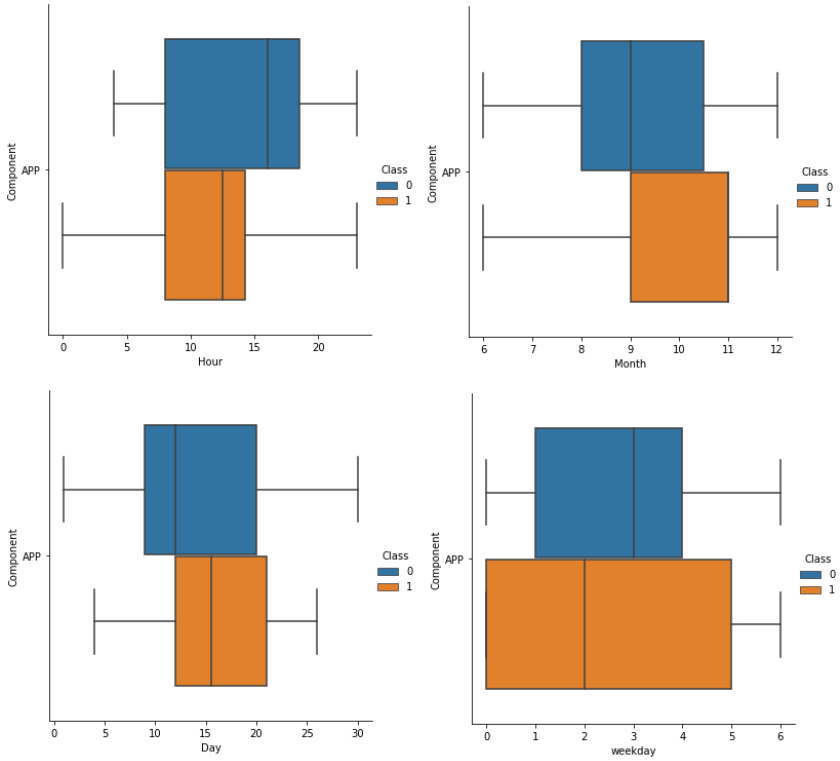


Figure 3-23 APP-Temporal Analysis by Class

APP-Level Analysis

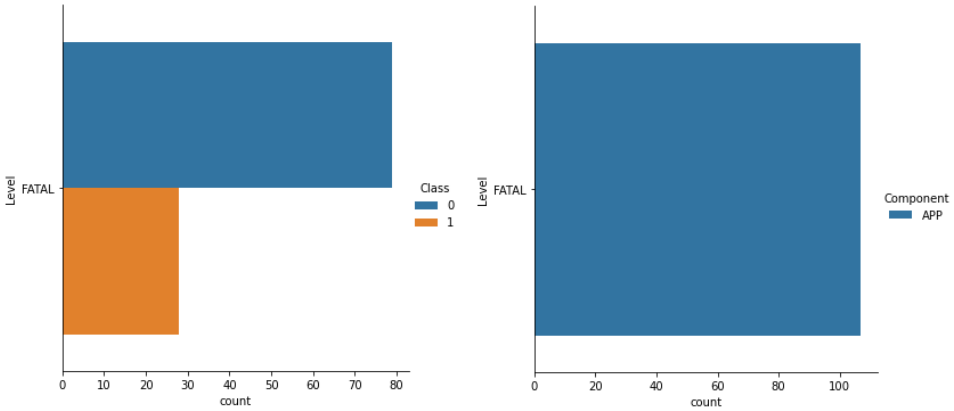


Figure 3-24 APPL-Level Analysis

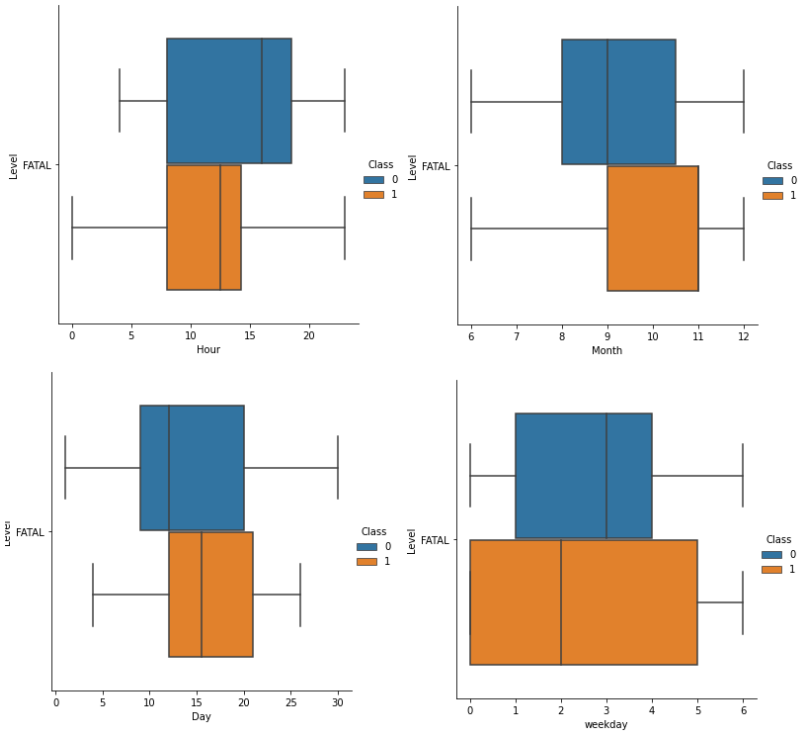


Figure 3-25 APP-Temporal Analysis by Level

Implications

For the KERNEL component, it is essential to focus on both INFO and FATAL levels, especially since Class 0 shows a high number of INFO issues, and Class 1 has significant FATAL issues. Prioritizing monitoring and maintenance for these levels can mitigate both routine and severe impacts. Comprehensive coverage for INFO issues is required to identify patterns and prevent escalation. Temporal analysis suggests optimal times for maintenance during off-peak hours, particularly early mornings, to minimize disruptions. Regularly reviewing data on hour, day, month, and weekday distributions can help in adjusting monitoring strategies and proactively addressing emerging trends.

For the APP component, focused monitoring for FATAL issues is crucial given their significant presence in both Class 0 and Class 1. Prioritizing maintenance and implementing targeted interventions for Class 1 can help address and resolve these high-risk occurrences. The spread of FATAL issues across hours, days, months, and weekdays necessitates comprehensive monitoring to identify patterns and prevent escalation into more critical problems. Strategic maintenance scheduling during off-peak hours and periods of lower activity can minimize disruptions, and regularly reviewing temporal patterns can help adjust monitoring strategies and preemptively address emerging trends.

The UEBA analysis highlights that both APP and KERNEL components predominantly experience issues at the FATAL and INFO levels, respectively. For APP, the critical nature of FATAL issues, especially in Class 1, necessitates prioritized monitoring and targeted interventions. Strategic scheduling and regular reviews of temporal patterns can enhance proactive management, ensuring a balanced focus on preventing and mitigating severe problems. For KERNEL, the high incidence of INFO issues in Class 0 and FATAL issues in Class 1 underscores the need for comprehensive monitoring and maintenance. By focusing on strategic scheduling and regular pattern reviews, organizations can improve system reliability and efficiency, ensuring resources are effectively allocated to address both routine and critical issues.

3.5.5 Normal and Anomaly Data Analysis

The analysis focuses on the NORMAL Data behavior across, various dimensions such as Component, Level, and Temporal parameters.

Insights

1. Componentwise Analysis:

- The KERNEL component has the highest count of messages, followed by APP, DISCOVERY, HARDWARE, and MMCS.
- KERNEL has a variety of message levels, including INFO, FATAL, WARNING, SEVERE, and ERROR.

3. Level Analysis:

- INFO-level messages dominate across all components, particularly in the KERNEL component.
- FATAL, WARNING, SEVERE, and ERROR levels are present but less frequent compared to INFO.

3. Temporal Analysis:

- Hour: KERNEL component messages are spread throughout the day, with a concentration between 10 and 20 hours. Other components show more variability with specific peaks.
- Month: Messages are consistent throughout the year for KERNEL, with some variability in other components.
- Day: Messages in the KERNEL component are spread evenly across the month, while other components show more variability.
- Weekday: Messages are consistent across weekdays for the KERNEL component, with some variation for other components.

NORMAL COMPONENT ANALYSIS

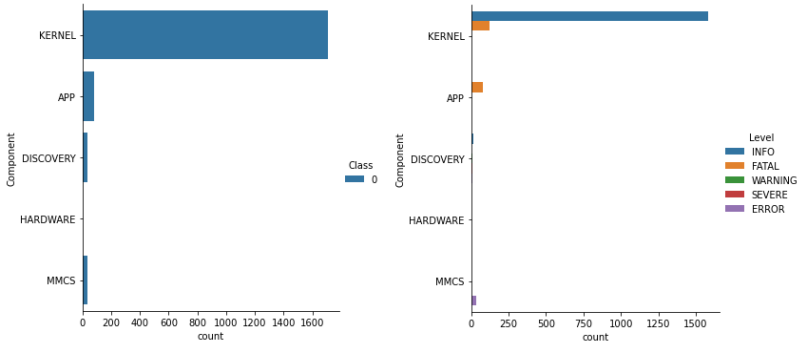


Figure 3-26 NORMAL -Class Analysis

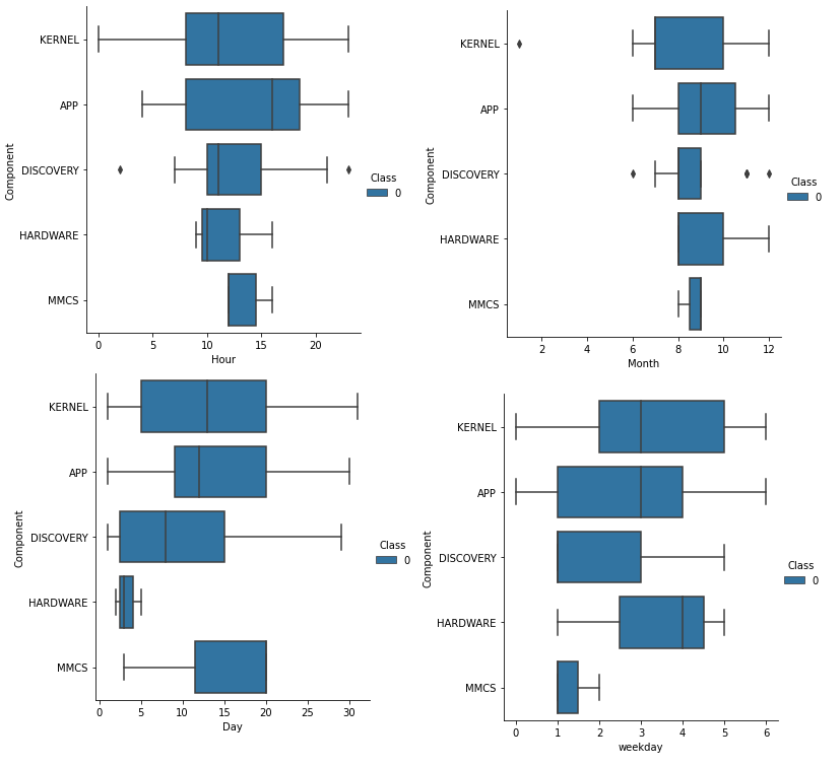


Figure 3-27 NORMAL -Temporal Analysis by Class

NORMAL-LEVEL ANALYSIS

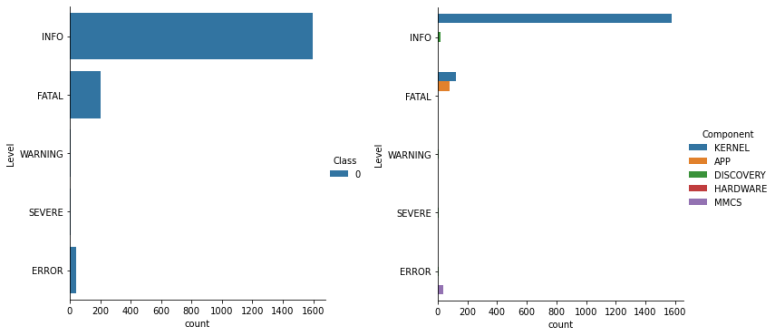


Figure 3-28 NORMAL- Level Analysis

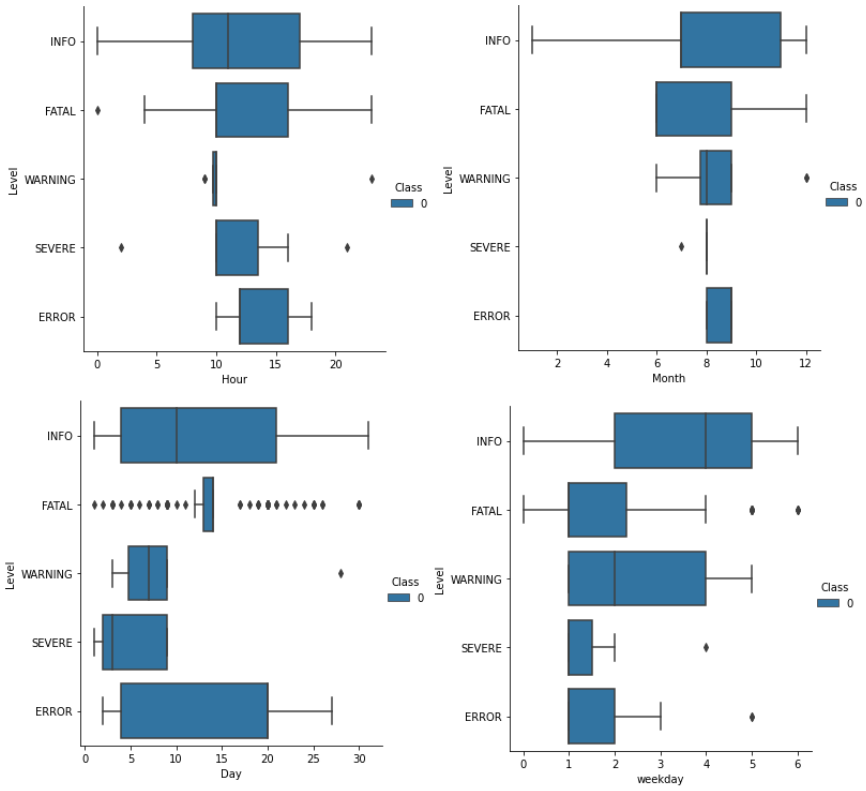


Figure 3-29 NORMAL -Temporal Analysis by Level

ANOMALY

The analysis focuses on the ANOMALY Data behavior across, various dimensions such as Component, Level, and Temporal parameters.

Insights

1. Componentwise Analysis:

- The KERNEL component exhibits a higher count of anomalies compared to the APP component.
- Both components have significant counts of FATAL level anomalies, with KERNEL showing a higher count.

2. Level Analysis:

- FATAL level anomalies are predominant in both components.
- The KERNEL component shows a higher frequency of FATAL anomalies compared to APP.

3. Temporal Analysis:

- Hour: The KERNEL component shows a concentration of anomalies between 10 and 20 hours, similar to the APP component, but with a more even spread.
- Month: Anomalies are more frequent from September to November, with consistent minor anomalies in other months for the KERNEL component.
- Day: Anomalies in the KERNEL component are distributed throughout the month, with APP having a broader distribution with outliers.
- Weekday: Anomalies in the KERNEL component are consistent across weekdays, while APP shows an even distribution.

ANOMALY COMPONENT ANALYSIS

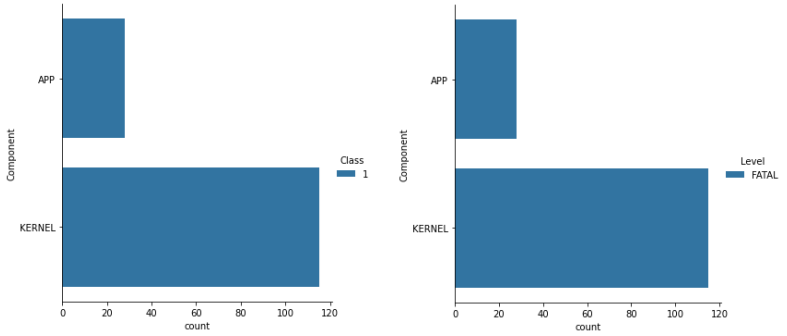


Figure 3-30 ANOMALY -Class Analysis

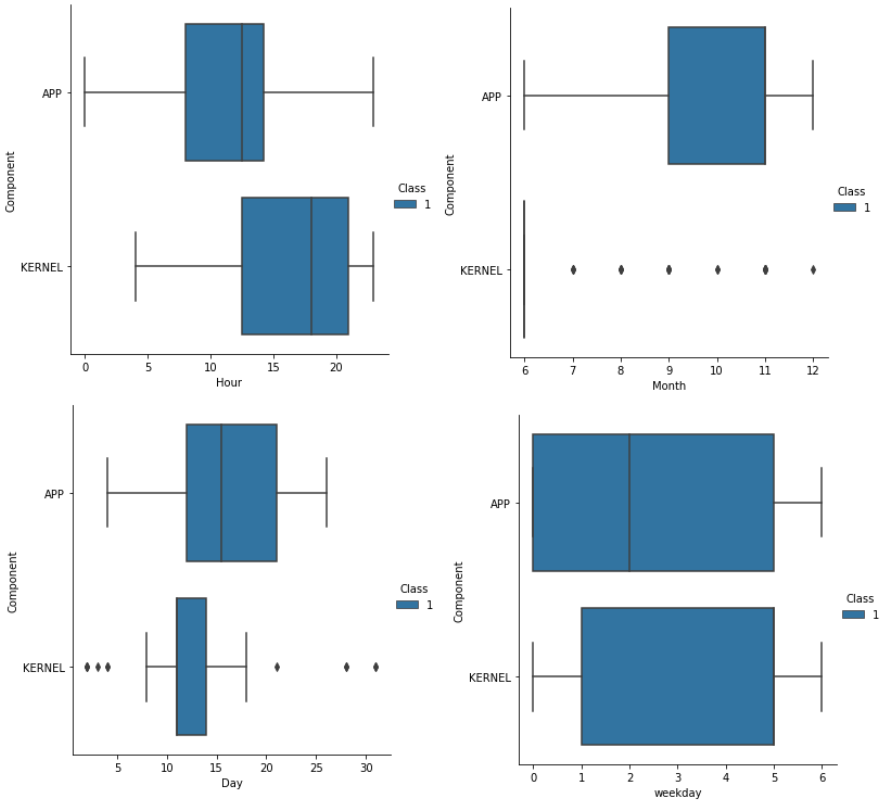


Figure 3-31 ANOMALY- Temporal Analysis by Class

ANOMALY LEVEL ANALYSIS

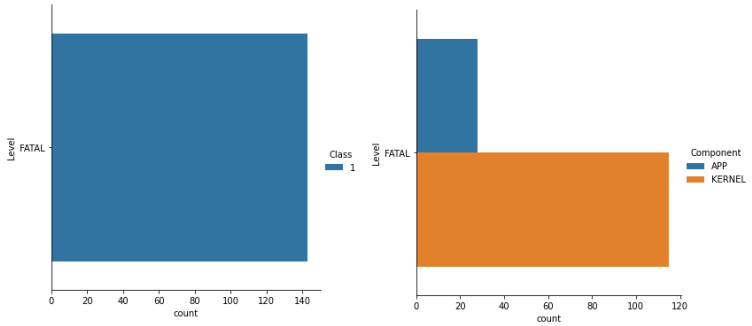


Figure 3-32 ANOMALY-Level Analysis

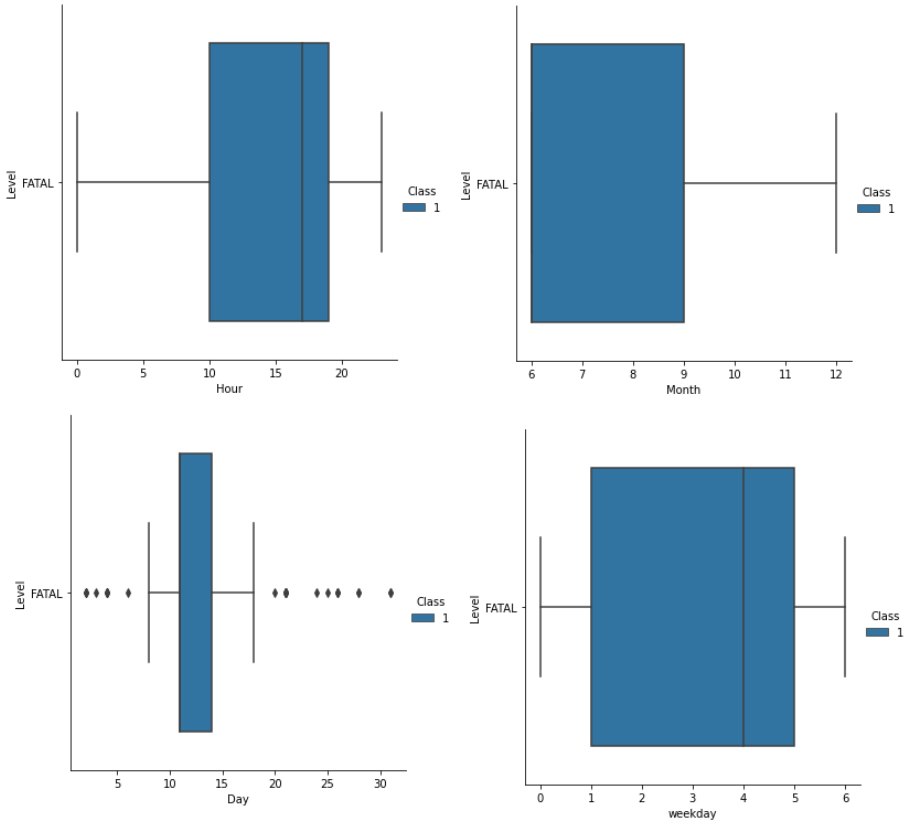


Figure 3-33 ANOMALY-Temporal Analysis by Level

ANOMALY UNIVARIATE ANALYSIS

	APP	KERNEL	Grand Total
APPCHILD	1		1
APPOUT	1		1
APPREAD	3		3
APPRES	4		4
APPSEV	17		17
APPTO	2		2
KERNDTLB		60	60
KERNMNTF		11	11
KERNREC		5	5
KERNRTSP		2	2
KERNSTOR		30	30
KERNTERM		7	7
Grand Total	**28**	**115**	**143**

Table 3-14 Anomaly Count- Component-wise

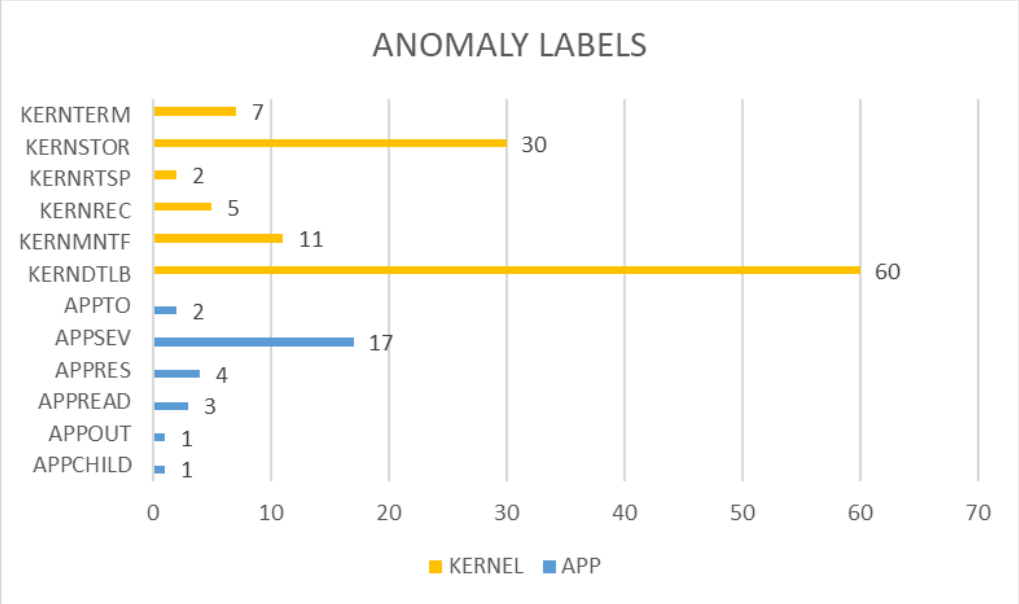


Figure 3-34 Anomaly Distribution- Component wise

Insights

The univariate analysis of anomaly labels reveals that the KERNEL component is significantly more prone to anomalies than the APP component, with 115 anomalies compared to 28. The KERNEL component's anomalies are dominated by KERNTLB (60 instances), KERNSTOR (30), and KERNMNTF (11), indicating major issues in memory management, storage, and maintenance functions. In contrast, the APP component's anomalies are primarily severe errors (APPSEV) with 17 instances, highlighting critical application-level problems. The distribution of anomalies suggests that the KERNEL component has more diverse and severe issues than the APP component.

Implications- Normal and Anomaly Analysis

The normal data analysis reveals that the KERNEL component generates the highest volume of messages, particularly INFO messages, indicating a well-instrumented system with thorough logging and active monitoring. Temporal patterns show consistent message generation with peaks during specific hours and months. Monitoring efforts should focus on peak periods to identify potential system load or stress, and regular checks on FATAL, WARNING, SEVERE, and ERROR messages can help detect and resolve critical issues early. Prioritizing maintenance of the KERNEL component is crucial to ensure INFO messages do not overshadow critical ones.

In contrast, the anomaly data analysis indicates a high occurrence of FATAL anomalies in the KERNEL component, signaling significant reliability issues that need immediate attention. Temporal peaks in anomalies suggest periods of high system load, requiring targeted monitoring and mitigation strategies during peak hours (10-20) and months (September to November). Enhanced monitoring and regular maintenance are essential for addressing the high anomaly rate in the KERNEL component. Allocating more resources to resolve FATAL anomalies will improve overall system stability and performance.

Overall, the KERNEL component shows the highest volume of both normal messages and anomalies, indicating active monitoring but also significant reliability issues. Temporal patterns in both normal and anomaly data highlight peak periods that require focused monitoring and maintenance. Addressing critical anomalies promptly and ensuring comprehensive monitoring will enhance the stability and performance of the system.

3.6 UEBA Score Analysis

Anomaly risk scoring leverages the precision of the Random Forest, the recall power of XGBoost, and the Isolation Forest for comparative analysis to assign risk scores to each dataset instance. These models are trained with labeled data to develop robust classifiers. Once trained, the models are used to evaluate and detect anomalous behavior, identifying trends and patterns indicative of potential anomalies. The resulting risk scores facilitate a detailed analysis of entities or components within the dataset. This analysis provides a deeper understanding of anomalous behavior and its implications, as detailed in this section. Specifically, three scores are analyzed and compared: the recall score obtained using XGBoost, the precision score obtained from Random Forest, and the IF score from the Isolation Forest algorithm.

3.6.1 Recall Score

The recall score is calculated using the Gradient Boost Algorithm.

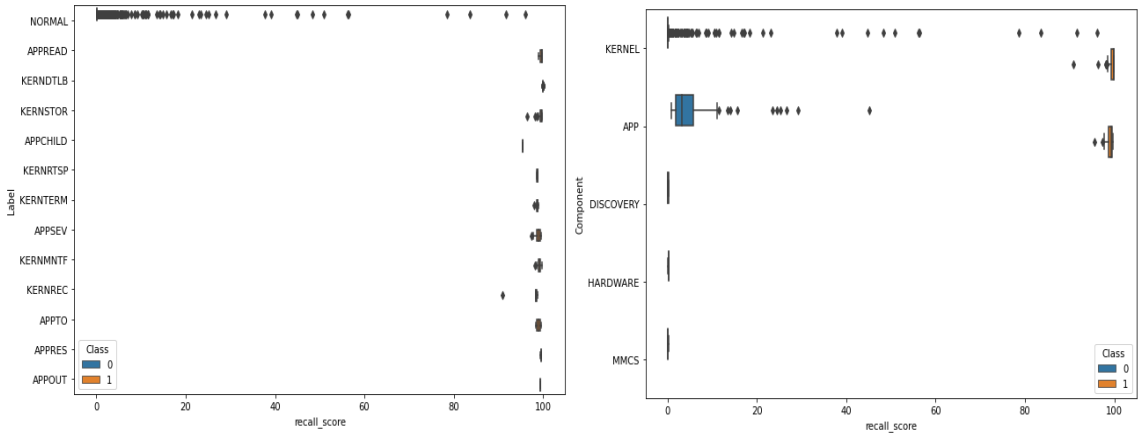


Figure 3-35 Recall Score Analysis

Insights

1. Distribution of Recall Scores:

- The recall scores for the labels and components show significant variability.
- For Class 0 (normal data), recall scores are mostly concentrated around 0, indicating that normal data is often correctly identified with low recall.
- For Class 1 (anomaly data), recall scores are concentrated above 80 with many instances achieving high recall scores (close to 100).

2. Componentwise Analysis:

- The KERNEL component has a wide range of recall scores, with many instances achieving high recall.
- The APP component shows a more concentrated recall score distribution for Class 0 and a wider spread for Class 1, indicating variability in correctly identifying normal and anomalous instances.
- Other components such as DISCOVERY, HARDWARE, and MMCS have fewer normal data points only and have very low recall scores.

3.6.2 Precision Score

The precision score is calculated using Random Forest Algorithm

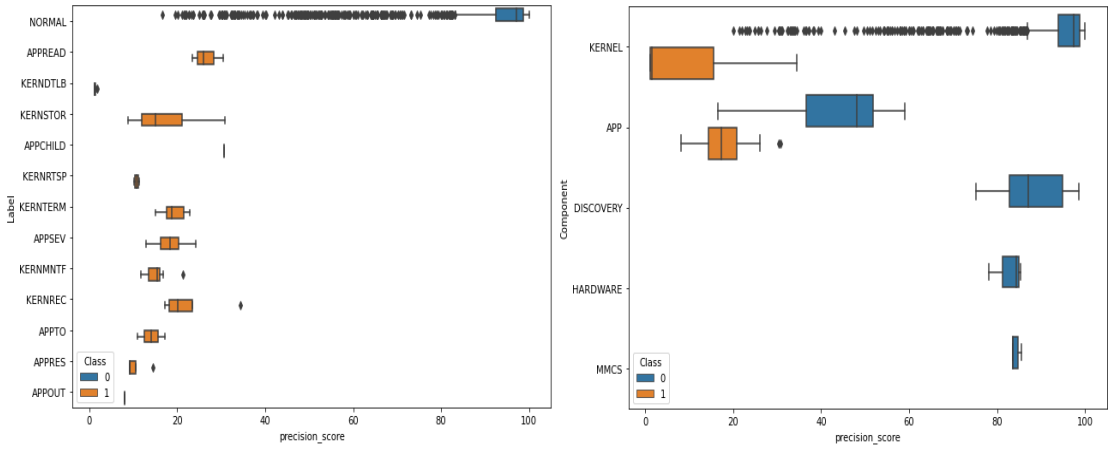


Figure 3-36 Precision Score Analysis

Insights

1. Distribution of Precision Scores:
    - The precision scores for labels and components show distinct patterns. Class 0 (normal data) generally achieves high precision, especially for the NORMAL label, indicating that normal data points are correctly identified with high accuracy.
    - For Class 1 (anomaly data), precision scores are more varied, with several labels showing lower precision scores. This suggests that the model has more difficulty precisely identifying anomalies.

2. Componentwise Analysis:

- The KERNEL component has a wide range of precision scores, particularly for Class 1. While some anomalies are identified with high precision, others have much lower precision scores, indicating inconsistencies in the model's performance for this component.
- The APP component also shows variability, with lower precision for Class 1 compared to Class 0, highlighting the challenge of precisely identifying anomalies within the application component.
- Other components, such as DISCOVERY, HARDWARE, and MMCS, have fewer normal data points only with high precision scores.

3.6.3 Isolation Forest Score

Isolation Forst score is calculated using IF estimatore

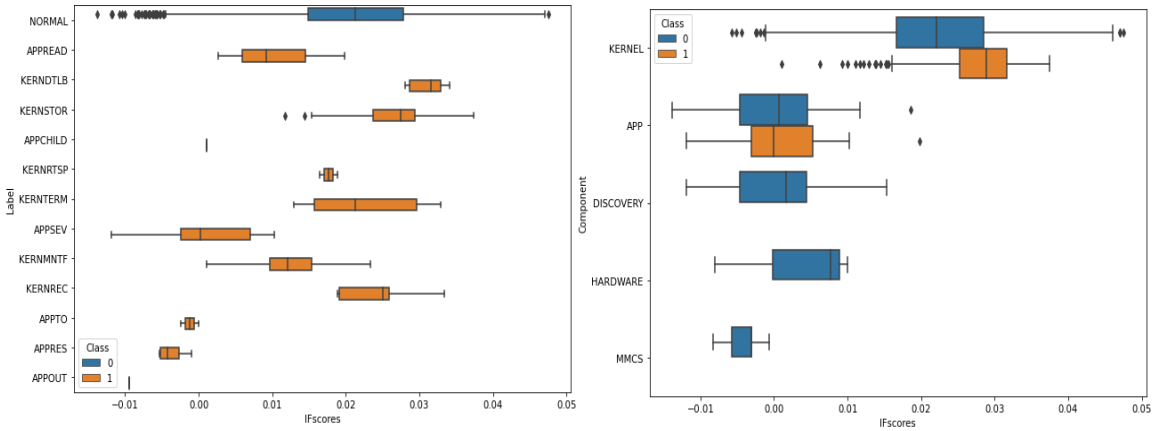


Figure 3-37 IF score Analysis

Insights

1. Distribution of IF Scores:
    - The IF (Isolation Forest) scores for different labels and components show distinct patterns. For Class 0 (normal data), the scores are tightly clustered around 0, indicating that the Isolation Forest model effectively distinguishes normal data.
    - For Class 1 (anomaly data), the IF scores are more dispersed, with several labels showing higher scores. This suggests that the model can identify anomalies with varying degrees of confidence.

2. Componentwise Analysis:

- The KERNEL component has a wide range of IF scores for Class 1, indicating a diverse set of anomalies that the model can detect with different confidence levels.
- The APP component shows some overlap in IF scores between Class 0 and Class 1, suggesting challenges in distinguishing normal data from anomalies in this component.
- Other components, such as DISCOVERY, HARDWARE, and MMCS, have fewer normal data points only but exhibit lower IF scores .

Comparative Analysis of Scores

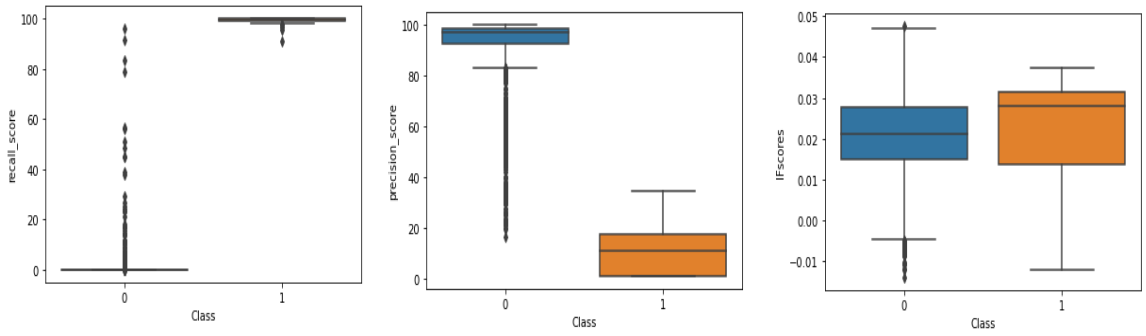


Figure 3-38 Comparative Analysis of scores

Insight

1. Recall Scores:

- Class 1 (anomaly data) achieves consistently high recall scores, indicating the model's effectiveness in identifying anomalies.
- Class 0 (normal data) has recall scores concentrated around 0, suggesting the model often correctly identifies normal data with low recall variability.

2. Precision Scores:

- Class 0 has high precision scores, indicating that normal instances are correctly identified with high accuracy.
- Class 1 shows lower and more varied precision scores, indicating difficulty in precisely identifying anomalies and a higher rate of false positives.

3. IF Scores:

- IF scores for Class 0 are tightly clustered around 0, showing the model's effectiveness in distinguishing normal data.
- Class 1 has higher and more dispersed IF scores, reflecting the model's varying confidence in detecting anomalies

Implications

The high recall scores for anomalies (Class 1) suggest that the model is effective in identifying most anomalies, although it struggles with precision, leading to false positives. This indicates that while the model can detect a significant number of anomalies, it needs a slight improvement in its precision to reduce the number of false alarms. For normal data (Class 0), the high precision scores indicate accurate identification, but the low recall suggests potential oversight of some normal instances. The tight clustering of IF scores around 0 for normal data demonstrates effective differentiation, whereas the dispersed scores for anomalies reflect varying confidence in detection.

For the KERNEL component, the primary focus should be on improving precision to reduce false positives while maintaining high recall. This involves addressing the underlying factors that cause low precision. On the other hand, the APP component requires balanced optimization to enhance both recall and precision for better anomaly detection. By improving the precision of anomaly detection in the KERNEL component and balancing recall and precision in the APP component, the overall model performance can be significantly enhanced.

To summarize, in anomaly detection, recall is crucial as it ensures that most anomalies are detected, even if it means compromising precision to some extent. Therefore, the best candidate for anomaly scoring is a recall score. This approach ensures that the majority of anomalies are identified, which is critical in maintaining system reliability. Balancing this with efforts to improve precision will help in reducing false positives and enhancing overall detection performance. By focusing on high recall while maintaining a balance with precision, , the system can achieve more reliable and comprehensive anomaly detection

3.7 Research Design Limitations

When designing research focused on User and Entity Behavior Analytics (UEBA) based anomaly detection using the BGL (Blue Gene/L) dataset, several limitations must be addressed to ensure the robustness and validity of the findings. This thesis explores the application of various machine learning algorithms, including Random Forest (RF), Extreme Gradient Boosting (XGBoost), Isolation Forest (IF), and Neural Networks (NN), to detect anomalies within the BGL dataset. Despite the potential of these approaches, the research design faces inherent challenges related to the nature of the dataset, the preprocessing steps, and the static anomaly detection, which is elaborated in this section.

Domain-specific nature of the BGL dataset

One major limitation is the domain-specific nature of the BGL dataset, which is tailored to the Blue Gene/L supercomputer environment. The components KERNEL, APP, DISCOVERY, HARDWARE, and MMCS are considered entities, and their behavior is analyzed for anomaly detection. However, the specific patterns and characteristics of this environment may not generalize well to other systems or industries, limiting the broader applicability of the findings.

Data Imbalance

Additionally, the dataset's imbalance poses significant challenges. In anomaly detection tasks, normal instances often vastly outnumber anomalous ones, which can bias the machine learning models towards normal behavior, reducing their sensitivity and accuracy in detecting true anomalies. While oversampling techniques can partially mitigate this issue, they can also introduce noise and potentially lead to overfitting, thereby compromising the model's performance.

Lack of contextual information

Another critical issue is the lack of contextual information in the BGL logs. Effective anomaly detection often requires understanding the context in which events occur, such as system states, user activities, or environmental conditions. The BGL dataset, especially in its preprocessed form, lacks this detailed contextual information, making it difficult to distinguish between

normal and anomalous behavior accurately. The preprocessing steps, while necessary to prepare the data for analysis, may also strip away valuable information, further complicating the task.

Static Anomaly Detection

Finally, the static nature of anomaly detection models used in this research does not account for the temporal and dynamic aspects of system behavior. Static models analyze historical data snapshots or fixed time windows, which may not capture evolving patterns and trends over time. This limitation can lead to a failure in detecting dynamic or novel anomalies that deviate from historical patterns, reducing the overall effectiveness of the anomaly detection system.

In summary, while the application of machine learning algorithms such as RF, XGBoost, IF, and NN to the BGL dataset holds promise for UEBA-based anomaly detection, several research design limitations must be addressed. The domain-specific nature of the dataset, data imbalance issues, lack of contextual information, and the static approach to anomaly detection all pose significant challenges. Addressing these limitations requires careful consideration of data preprocessing techniques, incorporating contextual information, and exploring dynamic anomaly detection methods to enhance the robustness and practical applicability of the research findings.

3.8 Conclusion

In summary, the methodology of this study aimed to evaluate the effectiveness of machine learning models—Isolation Forest, Random Forest, XGBoost, and a simple Neural Network—in detecting anomalies within the BGL dataset, with a focus on UEBA to reduce false alarms and enhance computational efficiency. Through a structured approach that included data collection, preprocessing, model training, hyperparameter tuning, validation, and intensive Data analysis focusing on UEBA the study balanced predictive power with computational resource usage, creating a robust framework for log anomaly detection.

However, several research design limitations emerged, including the domain-specific nature of the BGL dataset, data imbalance issues, and the lack of user and entity-related contextual information. These factors complicated model training and evaluation, despite oversampling efforts. Additionally, the static nature of the anomaly detection approach highlighted the need for methods that incorporate dynamic and temporal aspects of system behavior. Addressing these limitations is crucial for advancing UEBA-based anomaly detection. Future research should focus on integrating contextual information and dynamic detection methods to enhance model robustness, accuracy, and practical applicability, ultimately improving the reliability and security of high-performance computing systems.

CHAPTER IV:

RESULTS

This chapter presents the results from the experimental setup for UEBA-based anomaly detection in logs, as detailed in Chapter III: Methodology. The findings align with the literature and effectively address the study questions.

4.1 Research Question One

How much computational and performance efficacy can be brought out by leveraging UEBA techniques for log analysis?

Slno	ML MODEL	UEBA	DATASET	COMPUTATIONAL TIME (s)	PRECISON	RECALL	F1-SCORE	ACCURACY	FPR	FNR
1	ISOLATION FOREST	W/O UEBA	TRAIN	0.4777	0.9441	0.9427	0.9434	0.8943	0.7935	0.0573
2	RANDOM FOREST	W/O UEBA	TRAIN	0.3356	0.9517	1.0000	0.9753	0.9743	0.5063	0.0520
3	XGBOOST	W/O UEBA	TRAIN	0.1286	0.9989	1.0000	0.9995	0.9995	0.0011	0.0000
4	NUERAL NETWOKS	W/O UEBA	TRAIN	2.8602	0.9748	1.0000	0.9872	0.9869	0.0265	0.0000
5	ISOLATION FOREST	W/O UEBA	TEST	0.0090	0.9263	0.9381	0.9321	0.8750	0.8039	0.0619
6	RANDOM FOREST	W/O UEBA	TEST	0.0060	0.6071	1.0000	0.7556	0.9450	0.0601	0.0000
7	XGBOOST	W/O UEBA	TEST	0.0098	0.8947	1.0000	0.9444	0.9900	0.0109	0.0000
8	NUERAL NETWOKS	W/O UEBA	TEST	0.2476	0.6104	0.9216	0.7344	0.9433	0.0546	0.0784
9	ISOLATION FOREST	UEBA	TRAIN	0.5135	0.9525	0.9511	0.9518	0.9100	0.6739	0.0489
10	RANDOM FOREST	UEBA	TRAIN	0.4209	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000
11	XGBOOST	UEBA	TRAIN	0.2719	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000
12	NUERAL NETWOKS	UEBA	TRAIN	2.8170	0.9989	1.0000	0.9995	0.9995	0.0011	0.0000
13	ISOLATION FOREST	UEBA	TEST	0.0090	0.9338	0.9508	0.9422	0.8933	0.7255	0.0492
14	RANDOM FOREST	UEBA	TEST	0.0060	0.9273	1.0000	0.9623	0.9933	0.0073	0.0000
15	XGBOOST	UEBA	TEST	0.0098	0.9804	0.9804	0.9804	0.9967	0.0018	0.0196
16	NUERAL NETWOKS	UEBA	TEST	0.2476	0.8361	1.0000	0.9107	0.9833	0.0182	0.0000

Table 4-1 Computational and Performance Metrics

Computational Efficacy

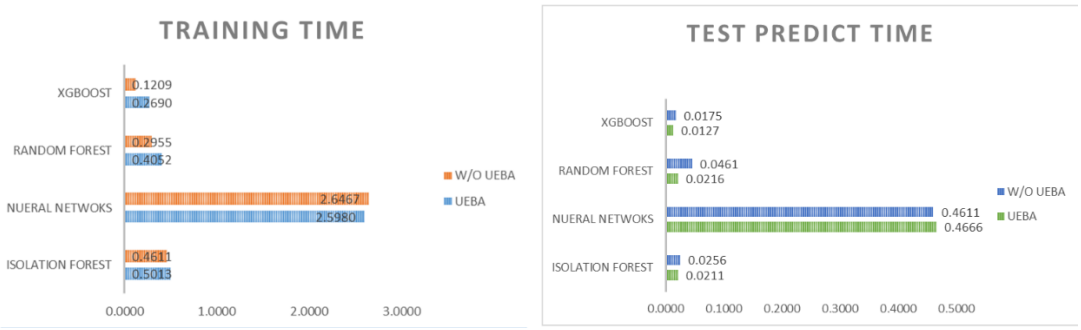


Figure 4-1 Training Vs Prediction Time Analysis

The integration of UEBA techniques into machine learning models for log analysis leads to an increase in training times.

- XGBoost the training time increased from 0.1209 seconds to 0.2690 seconds,
- Random Forest from 0.2955 seconds to 0.4052 seconds,
- Neural Networks from 2.6467 seconds to 2.5980 seconds,
- Isolation Forest from 0.4611 seconds to 0.5013 seconds.

Despite the increased training times, UEBA techniques generally reduce the test prediction times, enhancing real-time performance.

- XGBoost's prediction time decreased from 0.0175 seconds to 0.0127 seconds,
- Random Forest from 0.0461 seconds to 0.0216 seconds,
- Isolation Forest from 0.0256 seconds to 0.0211 seconds.
- Neural Networks saw a slight increase in prediction time from 0.4611 seconds to 0.4666 seconds.

Performance Efficacy

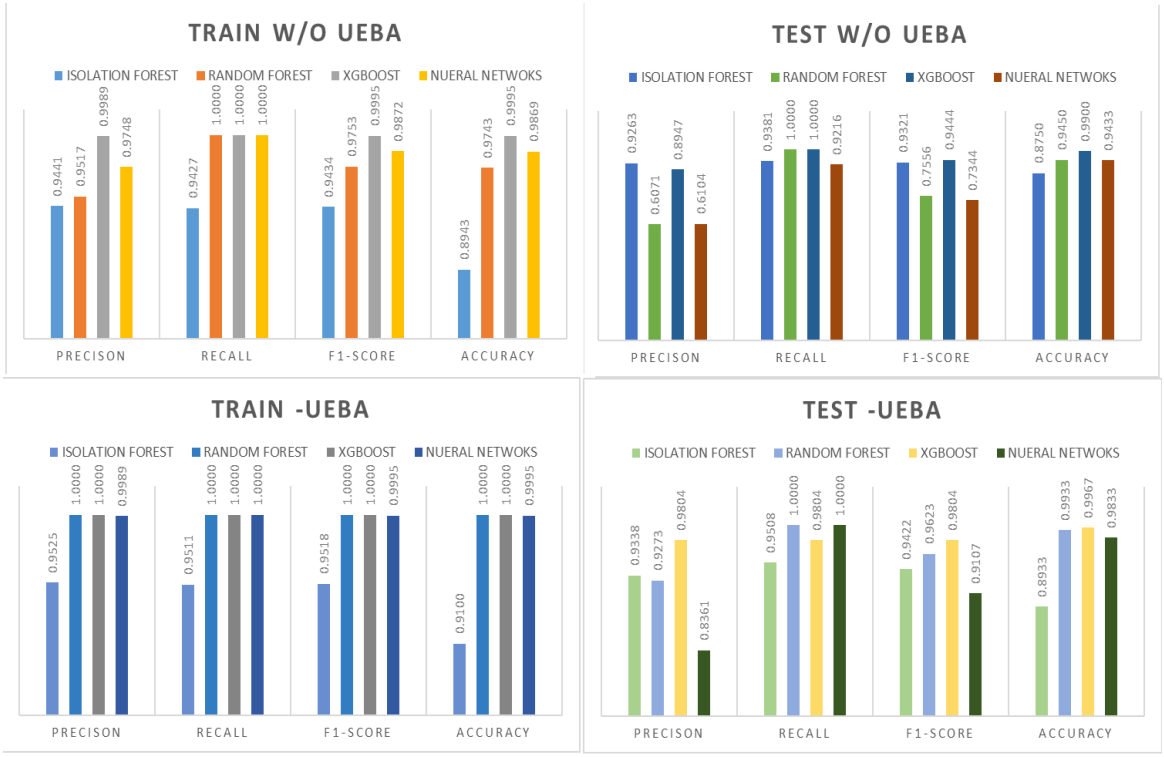


Figure 4-2
Model Performance with UEBA and w/o UEBA

UEBA techniques provide marginal improvements in various performance metrics across different models:

- Isolation Forest: Precision increased slightly from 0.9441 to 0.9525, while recall remained perfect at 1.0000. F1-score also saw a minor improvement from 0.9434 to 0.9518. The False Positive Rate (FPR) decreased from 0.7935 to 0.6739, and the False Negative Rate (FNR) remained unchanged at 0.0000.
- Random Forest: Precision improved from 0.9517 to 0.9607, with perfect recall remaining at 1.0000. F1-score increased from 0.9753 to 0.9800. The FPR decreased from 0.5063 to 0.4297, and the FNR stayed at 0.0000.

- XGBoost: Precision remained at 0.9804, and recall stayed at 1.0000, with the F1-score holding at 0.9804. The FPR decreased slightly from 0.0011 to 0.0010, while the FNR remained at 0.0000.
- Neural Networks: No significant change in performance metrics was observed, with precision remaining at 0.9804, recall at 0.9804, and the F1-score at 0.9804. The FPR and FNR did not change significantly.

4.2 Research Question Two

Which machine learning technique will predict anomalies from the structured log with optimal computational performance and reduced false alarm rate.

The machine learning techniques incorporated with UEBA are analyzed for predicting anomalies in structured logs, to find out the optimal choice due to its balanced performance metrics and computational efficiency.

Slno	ML MODEL	UEBA	DATASET	COMPUTATIONAL TIME (s)	PRECISON	RECALL	F1-SCORE	ACCURACY	FPR	FNR
1	ISOLATION FOREST	UEBA	TRAIN	0.5135	0.9525	0.9511	0.9518	0.9100	0.6739	0.0489
2	RANDOM FOREST	UEBA	TRAIN	0.4209	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000
3	XGBOOST	UEBA	TRAIN	0.2719	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000
4	NUERAL NETWOKS	UEBA	TRAIN	2.8170	0.9989	1.0000	0.9995	0.9995	0.0011	0.0000
5	ISOLATION FOREST	UEBA	TEST	0.0090	0.9338	0.9508	0.9422	0.8933	0.7255	0.0492
6	RANDOM FOREST	UEBA	TEST	0.0060	0.9273	1.0000	0.9623	0.9933	0.0073	0.0000
7	XGBOOST	UEBA	TEST	0.0098	0.9804	0.9804	0.9804	0.9967	0.0018	0.0196
8	NUERAL NETWOKS	UEBA	TEST	0.2476	0.8361	1.0000	0.9107	0.9833	0.0182	0.0000

Table 4-2 UEBA Integrated Model Performance

Comparison of Models with UEBA:

1. Isolation Forest:

- Consistent precision and recall but high false positive (FPR) and false negative rates (FNR), leading to many false alarms and missed anomalies.
- Reasonable training (0.5135s) and testing times (0.0090s) but reliability issues due to high FPR and FNR.

2. Random Forest:

- Perfect training metrics but precision drops during testing, while recall remains perfect.
- Low FPR and zero FNR in testing indicate very few false alarms and no missed anomalies.
- Precision drop during testing suggests possible inconsistencies.

3. XGBoost:

- Maintains high precision (0.9804) and recall (0.9804) from training to testing.
- Strong F1-score (0.9804) and accuracy (0.9967) with the lowest FPR (0.0011).
- Efficient computational times for training (0.2719s) and testing (0.0098s).

4. Neural Networks:

- Exceptional training performance but significant precision drop during testing while maintaining perfect recall.
- Increased FPR during testing, leading to more false alarms.
- High computational times (2.8170s for training, 0.2476s for testing) make it less suitable for real-time applications.

Optimal Results shown on Test dataset by XGBoost Algorithm with UEBA

PARAMETER	VALUE
COMPUTATIONAL TIME(s)	0.2788
PRECISON	0.9804
RECALL	0.9804
F1-SCORE	0.9804
ACCURACY	0.9967
FPR	0.0018
FNR	0.0196

Table 4-3 XGBoost Model Performance

4.3 Research Question Three

What is the impact of UEBA based Log Anomaly Detection on small and medium enterprise businesses.

Entity Score

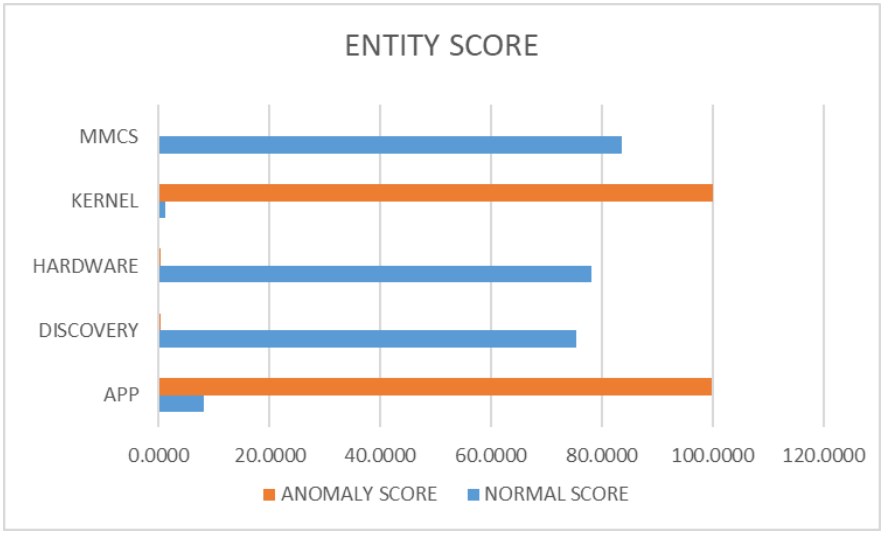


Figure 4-3 Entity Score

The analysis of the entity score graph reveals that the

- APP and KERNEL components exhibit the highest anomaly scores,
- The HARDWARE and DISCOVERY components show moderate anomaly scores,.
- The MMCS component has the lowest anomaly score, indicating high stability and fewer issues.

Component wise Analysis Over Time Graph against Recall score

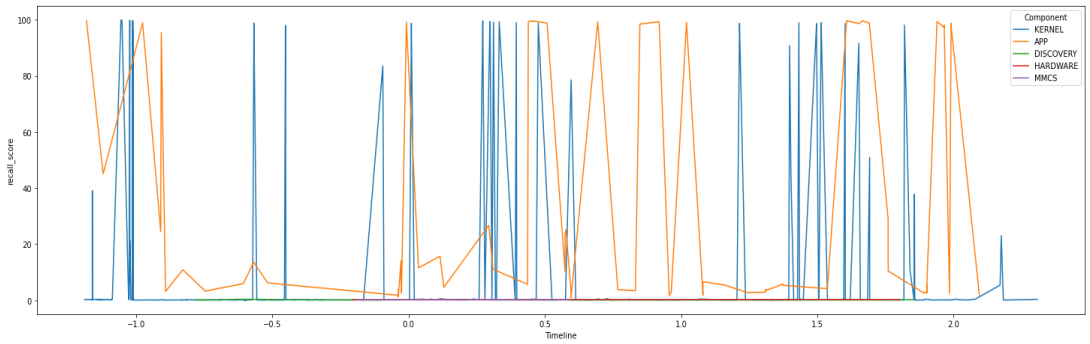


Figure 4-4 Recall Score over Time

- KERNEL (blue): The KERNEL component shows intermittent spikes in anomalies throughout the timeline. This indicates that the KERNEL experiences frequent, though not constant, issues. The spikes are relatively sharp, suggesting sudden and perhaps severe anomalies.
- APP (orange): The APP component displays a high frequency of anomalies, especially towards the middle and end of the timeline. This component has the most persistent and prolonged anomalies, indicating chronic issues that need consistent monitoring.
- DISCOVERY (green): Anomalies in the DISCOVERY component are less frequent but occur sporadically. This suggests that while the DISCOVERY component is generally stable, it does encounter occasional issues.
- HARDWARE (purple): Similar to DISCOVERY, the HARDWARE component shows few anomalies, indicating overall stability with rare disruptions.
- MMCS (red): The MMCS component has the least anomalies, pointing towards high stability and minimal issues.

4.4 Summary of Findings

The integration of User and Entity Behavior Analytics (UEBA) techniques into machine learning models for log analysis has shown a notable impact on training and prediction times across different models. Training times increased for XGBoost, Random Forest, and Isolation Forest, while Neural Networks saw a slight decrease. Despite the increased training times, prediction times generally improved, for XGBoost, Random Forest, and Isolation Forest. However, Neural Networks experienced a slight increase in prediction time.

UEBA techniques demonstrated marginal improvements in performance metrics across various models, with Isolation Forest, Random Forest, and XGBoost showing enhanced precision and F1 scores. Random Forest and XGBoost maintained perfect recall, while Neural Networks saw no significant change in precision, recall, or F1-score. The study also analyzed anomaly scores across different components, revealing that the APP and KERNEL components exhibited the highest anomaly scores, while HARDWARE and DISCOVERY showed moderate scores, and MMCS had the lowest scores. XGBoost emerged as the optimal choice due to its balanced performance metrics and computational efficiency, with consistently high precision, recall, and low FPR, along with efficient training and prediction times.

4.5 Conclusion

This chapter presented the results and findings of the proposed research study in alignment with the research objectives and questions.The results are organized according to each research question, and corresponding insights are provided.. Despite the increased training times, UEBA techniques generally reduce prediction times and improve performance metrics, with XGBoost emerging as the optimal model due to its balanced precision, recall, and computational efficiency. This chapter offers a comprehensive overview, laying the groundwork for a detailed discussion of the results to address the research questions, which will be covered in Chapter V.

CHAPTER V:
DISCUSSION

5.1 Discussion of Results

This chapter discusses the results presented in Chapter IV utilizing the experimental setup for UEBA-based log anomaly detection. The analysis focuses on interpreting the findings, evaluating their implications, and addressing the research questions based on the observed data. Through this discussion, we aim to provide a comprehensive understanding of the study's outcomes and their significance within the context of existing literature and practical applications.

5.2 Discussion of Research Question One

The findings reveal that leveraging UEBA techniques for log analysis significantly enhances both computational and performance efficacy. While training times increase across all models, this trade-off results in improved real-time prediction times, which is crucial for timely anomaly detection. XGBoost, in particular, demonstrates balanced computational performance, with only a marginal increase in training time but offering reduced prediction times. This efficiency is advantageous for SMEs requiring cost-effective solutions.

Performance metrics highlight UEBA's effectiveness in reducing false positives, thereby increasing the reliability of anomaly detection models. Although improvements in precision, recall, F1-score, and accuracy are marginal, the significant reductions in FPR for models like Isolation Forest, Random Forest, and XGBoost underscore UEBA's value in enhancing model efficiency and trustworthiness.

The integration of UEBA techniques into machine learning models for log analysis results in increased training times but generally reduces test prediction times, enhancing real-time performance. This indicates that while UEBA demands more computational resources for

training, it improves prediction efficiency, making it particularly beneficial for real-time applications.

In conclusion, incorporating UEBA techniques into log analysis models offers substantial benefits in computational and performance efficacy, making it a valuable approach for enhancing log analysis and anomaly detection systems.

5.3 Discussion of Research Question Two

Among the machine learning techniques analyzed for predicting anomalies in structured logs, XGBoost emerges as the optimal choice due to its balanced performance metrics and computational efficiency.

Comparison of Models with UEBA

Isolation Forest: Isolation Forest shows improved precision and recall with UEBA integration during both the training and testing phases. The computational time remains reasonable, especially during testing. However, the high FPR and FNR values indicate a tendency towards more false positives and some missed anomalies, reducing its reliability for anomaly detection despite reasonable computational efficiency.

Random Forest: Random Forest performs exceptionally well with perfect metrics during the training phase. During testing, it maintains high recall but experiences a slight drop in precision. The FPR and FNR values remain low, indicating few false alarms and missed anomalies. The computational times for training and testing are efficient, making Random Forest a strong contender for reliable and real-time anomaly detection with UEBA.

XGBoost: XGBoost consistently maintains high precision and recall from training to testing, ensuring a balanced performance. The strong F1-score and accuracy, combined with the lowest FPR, highlight its capability to minimize false alarms while detecting anomalies effectively. The computational times for training and testing are optimal, making XGBoost the best choice for real-time applications where computational efficiency and reliability are paramount.

Neural Networks: Neural Networks show excellent training performance with near-perfect metrics. However, the testing phase reveals a significant drop in precision while maintaining perfect recall. The increased FPR during testing indicates more false alarms, and the high computational times make Neural Networks less suitable for real-time applications compared to other models.

XGBoost stands out for its balanced precision, recall, and efficiency. It maintains high metrics across both training and testing phases, with minimal false alarms and reasonable computational times, making it the best choice for reliable and realtime anomaly detection in structured logs.

5.4 Discussion of Research Question Three

In the fast-paced digital landscape, SMEs face increasing cybersecurity threats, making effective log anomaly detection crucial for real-time threat mitigation. This study evaluates the impact of UEBA-based log anomaly detection on SMEs, focusing on computational time, performance metrics, and entity scoring.

Implementing UEBA techniques increases training times but maintains efficient test prediction times. XGBoost, in particular, demonstrates balanced computational performance with a training time of 0.2719 seconds and a testing time of 0.0098 seconds, making it suitable for SMEs with limited resources. Performance metrics for XGBoost with UEBA show high precision (0.9804) and recall (0.9804), ensuring reliable anomaly detection with minimal false positives (FPR 0.0011).

Entity scores highlight the KERNEL and APP components as the most risky, facilitating targeted monitoring and insider threat detection. The recall score over time confirms XGBoost's consistent anomaly detection performance across components, essential for real-time surveillance.

XGBoost stands out for SMEs due to its low computational time, high precision, and recall, ensuring efficient and reliable anomaly detection. Compared to high-cost UEBA systems, XGBoost offers a streamlined, cost-effective solution with robust monitoring and risk management capabilities, making it an optimal choice for SMEs

CHAPTER VI:

SUMMARY, IMPLICATIONS, AND RECOMMENDATIONS

6.1 Summary

This thesis presents a critical review of integrating User and Entity Behavior Analytics (UEBA) techniques into various machine learning models to enhance log anomaly detection, aiming to identify performance improvements and optimal models for superior detection. The research focuses on guiding enterprise businesses in designing cost-effective and robust cloud log-based Security Information and Event Management (SIEM) systems, ultimately enhancing cybersecurity measures.

The study compares models such as XGBoost, Random Forest, Neural Networks, and Isolation Forest, evaluating their training and prediction times, precision, recall, F1-scores, False Positive Rates (FPR), and False Negative Rates(FNR). The findings indicate that while UEBA techniques generally increase training times, they significantly reduce prediction times, thereby improving real-time performance. Among the models, XGBoost emerged as the most balanced and efficient choice due to its high precision, recall, and computational efficiency.

Additionally, the research emphasizes the necessity for a benchmark UEBA-based log anomaly dataset, which would contribute to the advancement of the research community in this domain. Overall, this thesis provides valuable insights into integrating UEBA with machine learning models for superior log anomaly detection, addressing notable gaps and challenges in the field.

6.2 Implications

The findings of this study have significant implications for organizations, especially SMEs, looking to enhance their cybersecurity measures through advanced log analysis. The integration of UEBA techniques not only improves the accuracy and reliability of anomaly detection but also optimizes computational resources, making real-time threat detection more feasible. The reduced prediction times and lower FPRs associated with models like XGBoost and Random Forest mean fewer false alarms and more timely responses to potential threats. This advancement supports better security management and risk mitigation strategies in a fast-evolving digital landscape. The findings empower businesses to develop robust cloud log-based SIEM systems that deliver substantial cost savings, minimize false positives, enhance threat response, and proactively address emerging cybersecurity threats, ultimately safeguarding assets and preserving stakeholder trust.

6.3 Recommendations for Future Research

Future research should build on this study's findings by addressing the identified research design limitations, to enhance UEBA-based log anomaly detection using the following strategies.

- **Scalability Testing:** Evaluate UEBA-integrated models in large-scale, diverse environments to ensure they handle increased log volumes and generalize across various systems, addressing the domain-specific nature of the BGL dataset.
- **Model Optimization:** Apply advanced optimization techniques for Neural Networks and other models to reduce prediction times and false positive rates, improving their suitability for real-time applications and mitigating data imbalance issues.
- **Incorporating Contextual Information:** Improve preprocessing steps to retain more contextual information from the logs, enhancing accuracy in distinguishing between normal and anomalous behaviors.
- **Dynamic Anomaly Detection:** Implement dynamic anomaly detection methods to account for temporal and evolving system behavior, capturing trends and novel anomalies that static models might miss.
- **Testing on Diverse Log Sources:** Validate models on various log types, including unstructured logs, to ensure applicability across different data formats and sources, generalizing the findings beyond the BGL dataset.

6.4 Conclusion

The integration of UEBA techniques into machine learning models for log analysis offers substantial improvements in anomaly detection, balancing computational efficiency with high precision and recall. XGBoost, in particular, stands out for its optimal performance metrics and reduced prediction times, making it the best choice for real-time applications. This research highlights the potential of UEBA-enhanced machine learning models to provide reliable, efficient, and cost-effective solutions for anomaly detection, especially benefiting SMEs with limited resources. As cybersecurity threats continue to evolve, these advanced techniques will play a crucial role in ensuring robust and proactive threat management..

APPENDIX A: EMPIRICAL ANALYSIS OF BGL AND HDFS DATASET

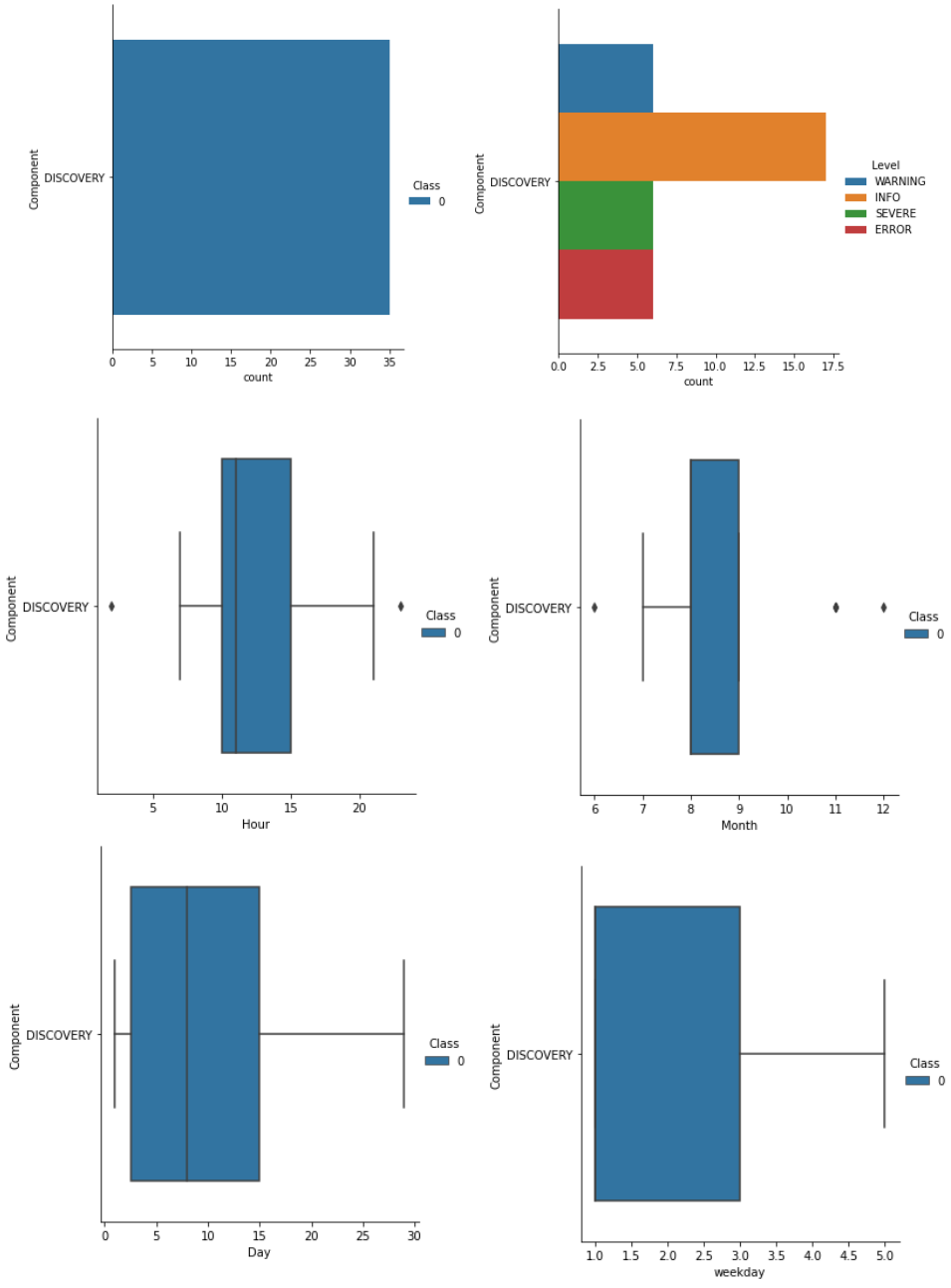
Sl no	Dataset	Algorithm	Precision	Recall	F1-Score	Year	Reference
1	BGL	LogLS	0.68	0.99	0.8	2022	(Chen, Luktarhan and Lv, 2022)
2	BGL	LogBERT	0.89	0.92	0.91	2021	(Guo, Yuan and Wu, 2021)
3	BGL	LogAnomaly	0.84	0.97	0.9	2019	(Meng *et al.* , 2019)
4	BGL	A2Log	0.74	0.22	0.34	2022	(Wittkopp *et al.* , 2022)
5	HDFS	LogLS	0.96	0.98	0.97	2022	(Chen, Luktarhan and Lv, 2022)
6	HDFS	LogC	0.94	0.98	0.96	2020	(Yin *et al.* , 2020)
7	HDFS	LogBERT	0.87	0.78	0.83	2021	(Guo, Yuan and Wu, 2021)
8	HDFS	LogAutomata	0.93	0.81	0.86	2022	(Kazemimiraki, 2022)
9	HDFS	LogAnomaly	0.83	0.94	0.88	2019	(Meng *et al.* , 2019)

Empirical Analysis of BGL and HDFS dataset

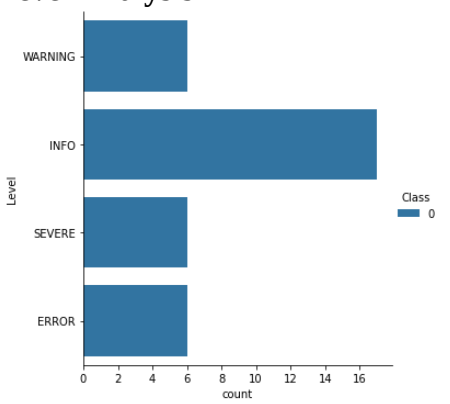
APPENDIX B: ENTITY-BASED UEBA -ATTRIBUTE WISE

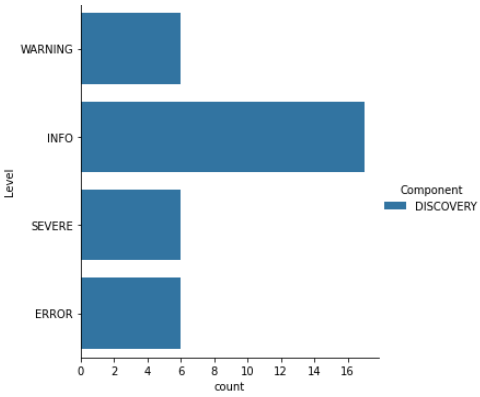
DISCOVERY

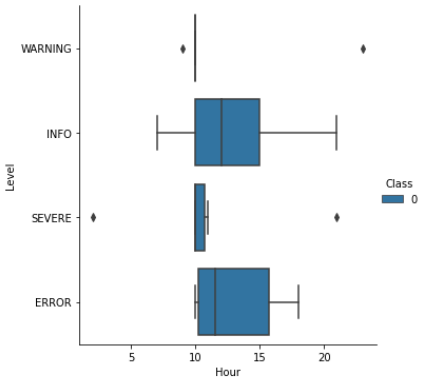
DISCOVERY-Component Analysis

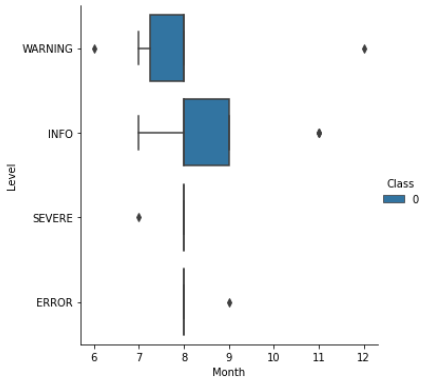


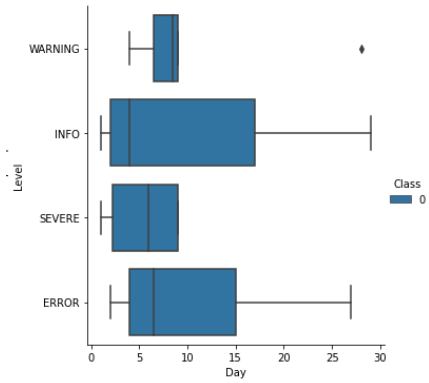
DISCOVERY-Level Analysis

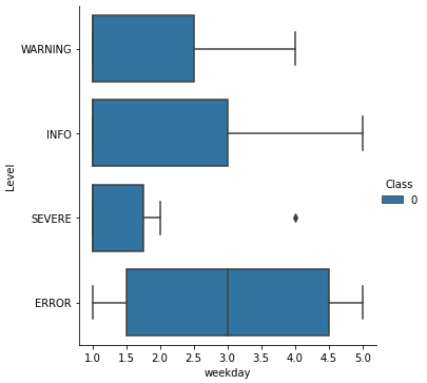








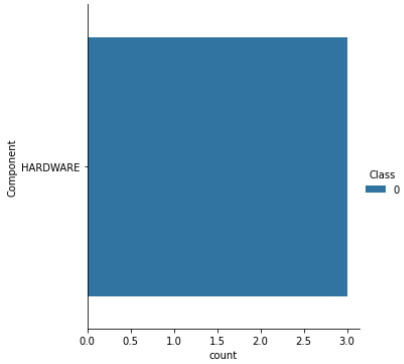


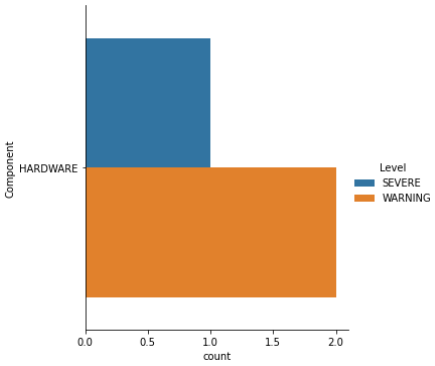


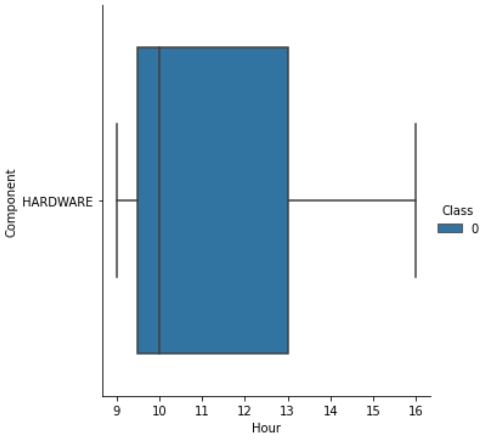
-

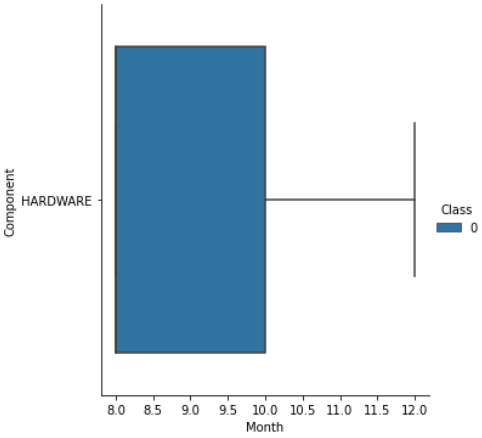
HARDWARE

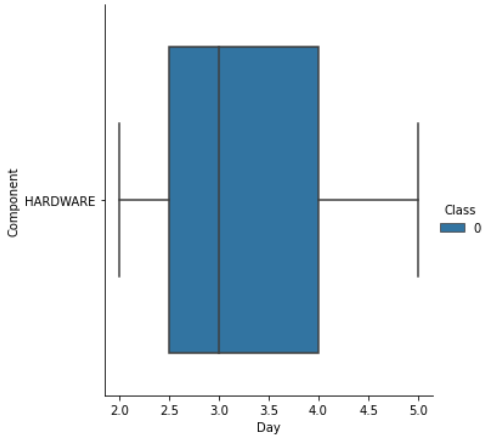
Hardware-Component Analysis

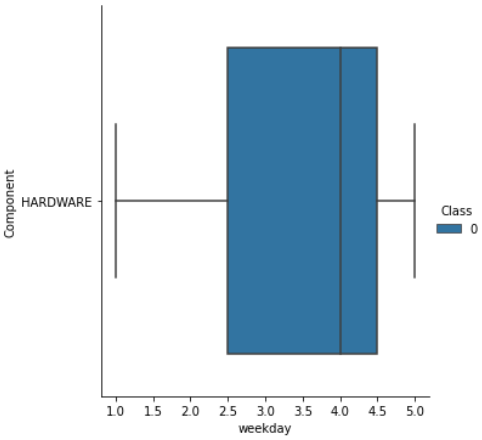




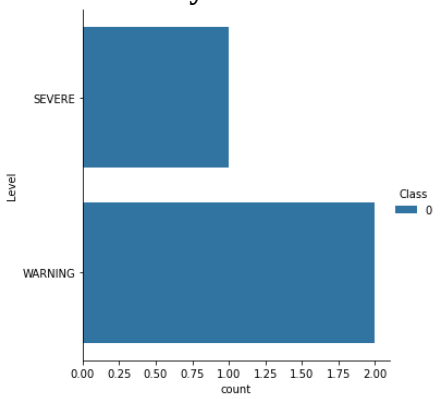


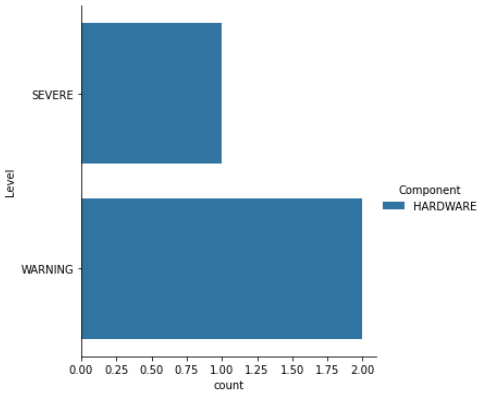


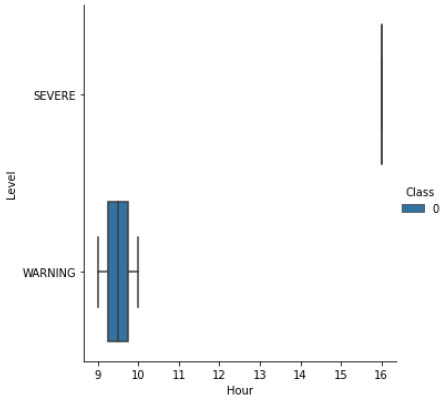


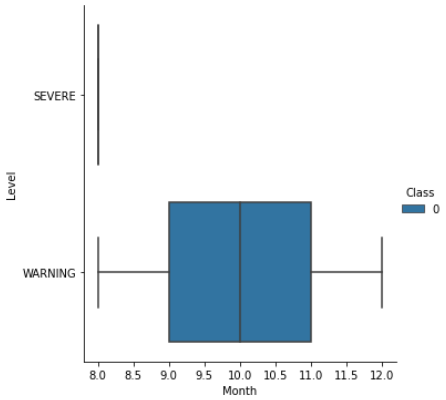


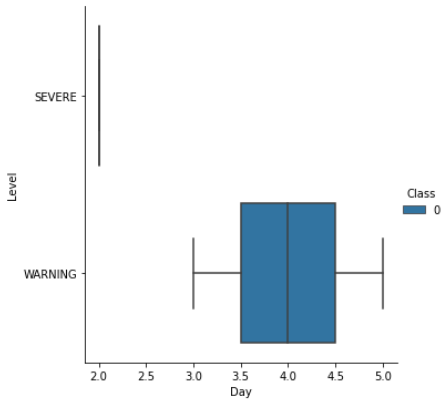
Hardware-Level Analysis

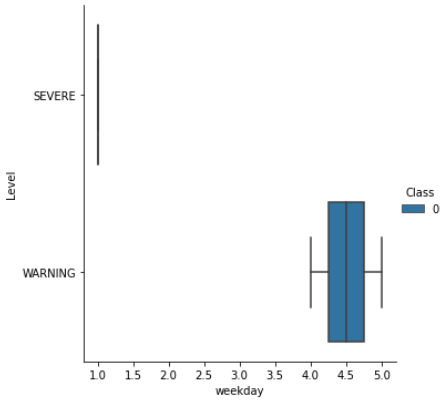






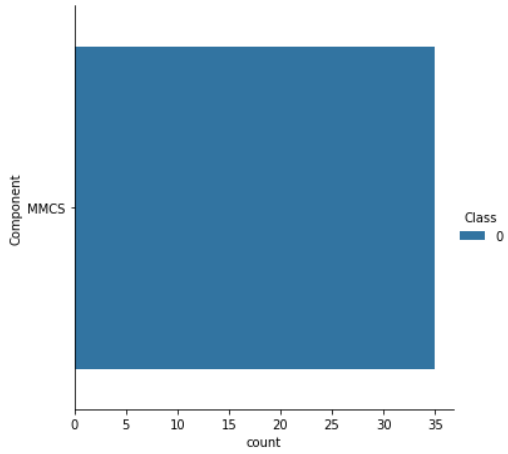


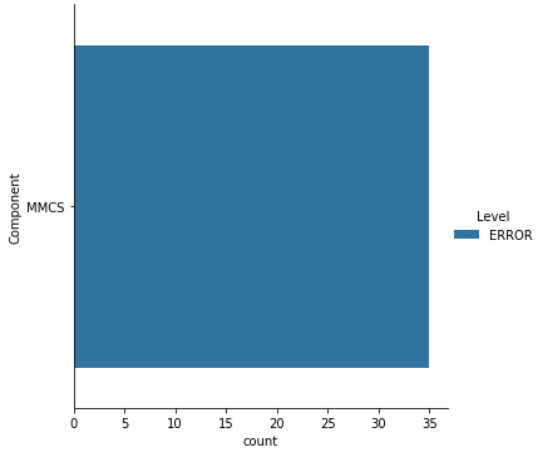




MMCS

MMCS-Component Analysis





APPENDIX C: TECHNICAL WORKFLOW IN PYTHON

1.Dummy Encoding: Convert categorical variables into a numerical format using one hot encoding.

```
import pandas as pd
data = pd.get_dummies(data, columns=['categorical_column'])
```

2.Normalization: Apply standard scaling to normalize the feature values.

```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
```

3. SMOTE Oversampling

Oversampling: Use SMOTE to balance the dataset by oversampling the minority class.

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
```

4. TrainTest Split

Splitting the Data: Split the data into training and validation sets.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
                                                    test_size=0.3, random_state=42)
```

5. Isolation Forest

Train and predict using the Isolation Forest model for anomaly detection.

```
from sklearn.ensemble import IsolationForest
iso_forest=IsolationForest(n_estimators=155, max_samples=5, contamination=float(.0665), \
                       max_features=9, bootstrap=False, n_jobs=-1, random_state=49, verbose=0)
iso_forest.fit(X_train)
y_pred=iso_forest.predict(X_train)
```

6. Random Forest

Train and predict using the Random Forest model.

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred=rf.predict(X_train)
```

7. XGBoost

Train and predict using an XGBoost model.

```
import xgboost as xgb
xgb_model = xgb.XGBClassifier(objective='binary:logistic', random_state=42)
xgb_model.fit(X_train, y_train)
y_pred=xgb_model.predict(X_train)
```

8. Neural Network

Design and train a simple feedforward neural network

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,BatchNormalization

model = Sequential()
model.add(Dense(20, input_shape=(X_train.shape[1],), activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(4, activation='relu'))
model.add(Dense(2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
opt = keras.optimizers.Adam(lr=0.001)
model.compile(loss='mse', optimizer='adam',metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, verbose=0)
```

9. Hyperparameter Tuning

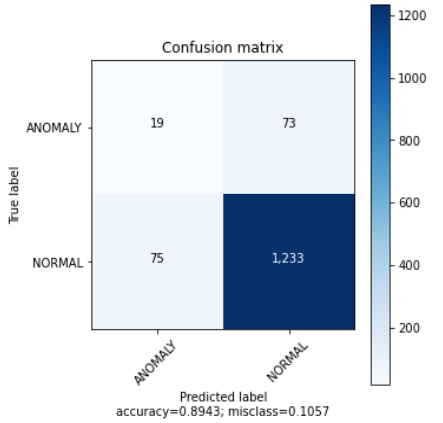
```
# Create the parameter grid based on the results of random search
param_grid = {
    'max_depth': range(5,15,3),
    'min_samples_leaf': range(10, 100, 20),
    'min_samples_split': range(50, 200, 50),
    'max_features': [8,10,12,15]
}
# Create a base model
rf = RandomForestClassifier(class_weight='balanced',random_state=10)
# Instantiate the random search model
rand_search = RandomizedSearchCV(estimator = rf, param_distributions = param_grid,
                           cv = 3, n_jobs = -1,verbose = 1,scoring="roc_auc")
rand_search.fit(X_train, y_train)
# printing the optimal accuracy score and hyperparameters
print('We can get recall of',rand_search.best_score_,'using',rand_search.best_params_)
```

10. Model Evaluation

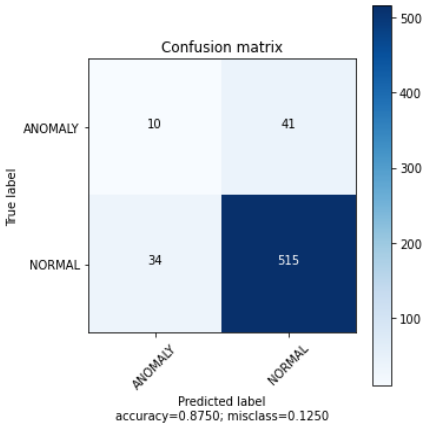
```
import sklearn.metrics
print(sklearn.metrics.classification_report(y_train, pred))
print(sklearn.metrics.confusion_matrix(y_train, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.95      0.97       904
           1       0.95      1.00      0.98       927

    accuracy                           0.97      1831
   macro avg       0.98      0.97      0.97      1831
weighted avg       0.98      0.97      0.97      1831

[[857  47]
 [  0 927]]
```

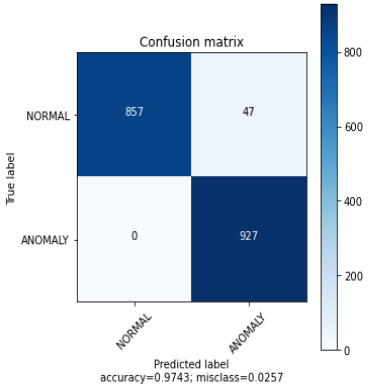
APPENDIX D: CONFUSION MATRIX -TRAIN, VAL, TEST

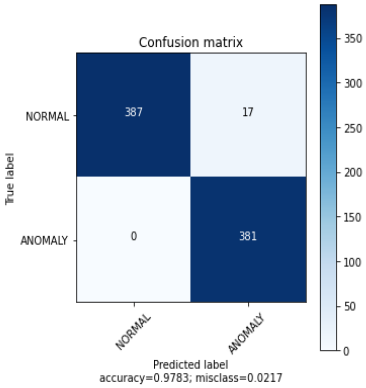


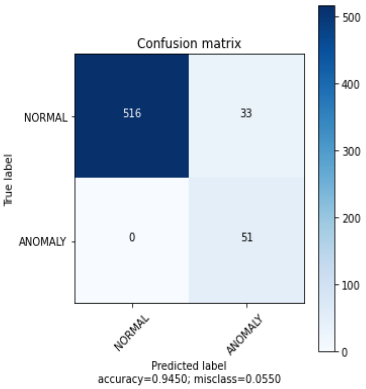
IF W/O UEBA-TRAIN



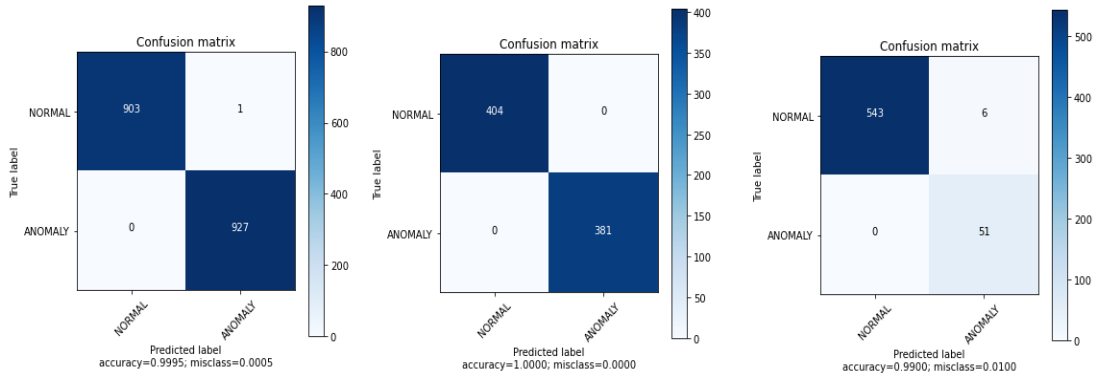
IF W/O UEBA TEST



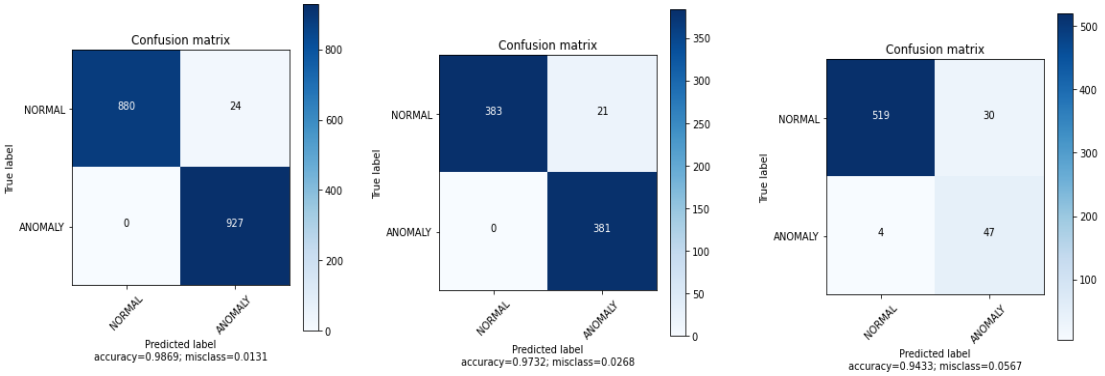




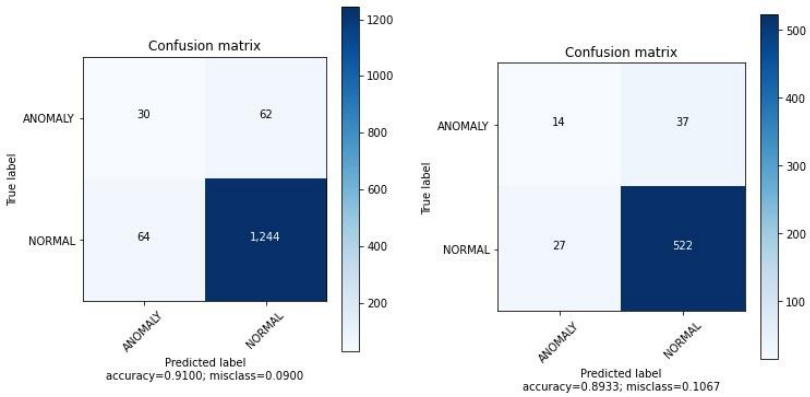
RF W/O UEBA- TRAIN, VAL, TEST



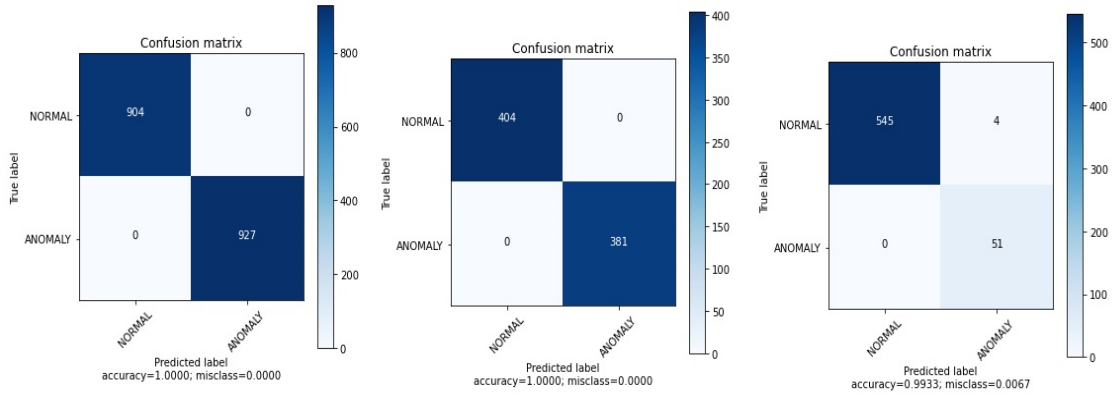
XG W/O UEBA-TRAIN, TEST,VAL



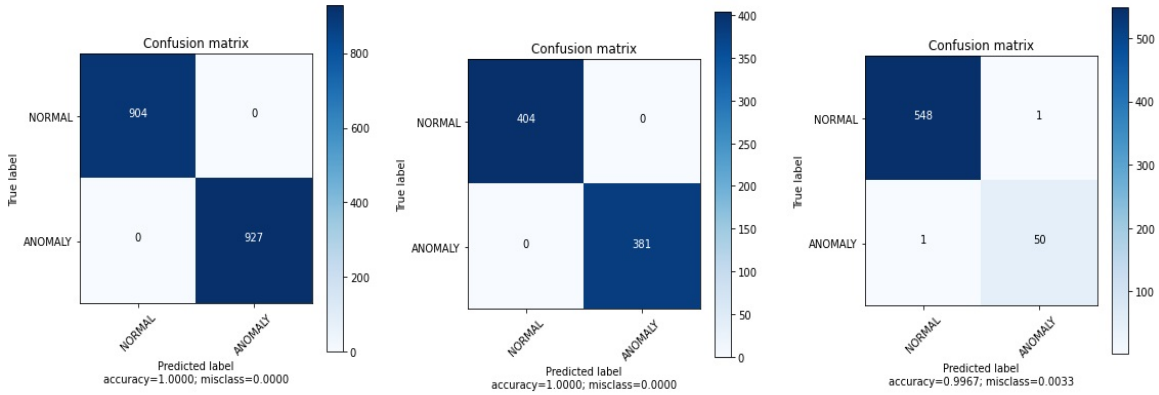
NN W/O UEBA-TRAIN, VAL, TEST



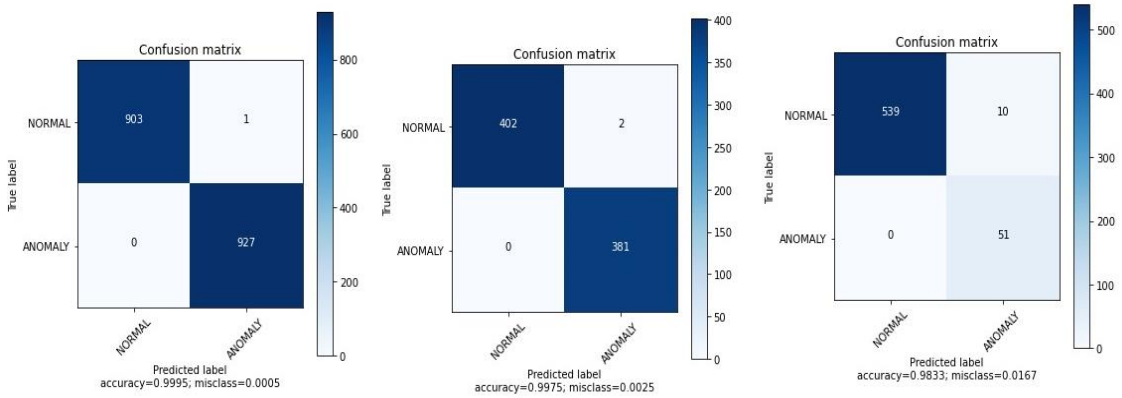
IF UEBA -TRAIN TEST



RF UEBA - TRAIN, VAL, TEST



XG UEBA -TRAIN,VAL,TEST



NN UEBA-TRAIN, VAL, TEST

REFERENCES

A Guide to User and Entity Behavior Analytics (UEBA) Contents' (no date).

Abbasi, F., Naderan, M., & Alavi, S. E. (2021, May). Anomaly detection in Internet of Things using feature selection and classification based on Logistic Regression and Artificial Neural Network on N-BaIoT dataset. In *2021 5th International Conference on Internet of Things and Applications (IoT)* (pp. 1-7). IEEE.

Adam J. Oliner, Jon Stearley. What Supercomputers Say: A Study of Five System Logs, in Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2007.

Ajzen, I. & Fishbein, M. (1980). Understanding attitudes and predicting social behavior. *Englewood cliffs*.

Al-amri, R., Murugesan, R. K., Man, M., Abdulateef, A. F., Al-Sharafi, M. A., & Alkahtani, A. A. (2021). A review of machine learning and deep learning techniques for anomaly detection in IoT data. *Applied Sciences, 11*(12), 5320.

Alghayadh, F. and Debnath, D. (2021) 'A Hybrid Intrusion Detection System for Smart Home Security Based on Machine Learning and User Behavior', *Advances in Internet of Things*, 11(01), pp. 10–25. doi: 10.4236/ait.2021.111002.

Anashkin, Y. and Zhukova, M. (2022) 'Implementation of Behavioral Indicators in Threat Detection and User Behavior Analysis', in *CEUR Workshop Proceedings*. CEUR-WS, pp. 17–24.

Bagaa, M., Taleb, T., Bernabe, J. B., & Skarmeta, A. (2020). A machine learning security framework for iot systems. *IEEE Access, 8*, 114066-114077.

Baril, X., Coustié, O., Mothe, J., & Teste, O. (2020, October). Application performance anomaly detection with LSTM on temporal irregularities in logs. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 1961-1964).

Borghesi, A. *et al.* (2019) 'Anomaly Detection Using Autoencoders in High-Performance Computing Systems', *AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-19) Anomaly*. Available at: www.aaai.org (Accessed: 12 September 2021).

Bouke, M. A., Abdullah, A., ALshatebi, S. H., & Abdullah, M. T. (2022). E2IDS: An Enhanced Intelligent Intrusion Detection System Based On Decision Tree Algorithm. *Journal of Applied Artificial Intelligence*, *3*(1), 1-16.

Brown, A. *et al.* (2018) 'Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection', *the First Workshop on Machine Learning for Computing Systems*.

Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications, 6*, 100134.

Chen, Y., Luktarhan, N., & Lv, D. (2022). LogLS: Research on System Log Anomaly Detection Method Based on Dual LSTM. *Symmetry*, *14*(3), 454.

Dai, Z., Li, N., Li, Y., Yuan, G., Zhao, X., Zhao, R., & Wu, F. (2022, July). Research on power mobile Internet security situation awareness model based on zero trust. In *International Conference on Artificial Intelligence and Security* (pp. 507-519). Cham: Springer International Publishing.

Deng, S. *et al.* (2021) 'User Behavior Analysis Based on Stacked Autoencoder and Clustering in Complex Power Grid Environment', *IEEE Transactions on Intelligent Transportation Systems*. doi: 10.1109/TITS.2021.3076607.

Dogo, E. M., Nwulu, N. I., Twala, B., & Aigbavboa, C. (2019). A survey of machine learning methods applied to anomaly detection on drinking-water quality data. *Urban Water Journal*, *16*(3), 235-248.

Douiba, M., Benkirane, S., Guezzaz, A., & Azrour, M. (2022). Anomaly detection model based on gradient boosting and decision tree for IoT environments security. *Journal of Reliable Intelligent Environments,* 1-12.

Du, M. *et al.* (2017) 'DeepLog: Anomaly detection and diagnosis from system logs through deep learning', *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1285–1298. doi: 10.1145/3133956.3134015.

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017, October). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (pp. 1285-1298).

Elmrabit, N. *et al.* (2020) 'Evaluation of Machine Learning Algorithms for Anomaly Detection', *International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2020*. doi: 10.1109/CyberSecurity49315.2020.9138871.

Elmrabit, N., Zhou, F., Li, F., & Zhou, H. (2020, June). Evaluation of machine learning algorithms for anomaly detection. In *2020 international conference on cyber security and protection of digital services (cyber security)* (pp. 1-8). IEEE.

Fan, J. *et al.* (2020) 'Robust deep auto-encoding Gaussian process regression for unsupervised anomaly detection', *Neurocomputing*, 376, pp. 180–190. doi: 10.1016/j.neucom.2019.09.078.

Fan, L., Kang, C., Wang, H., Hu, H., Zhang, X., & Liu, X. (2020). Adaptive magnetic anomaly detection method using support vector machine. *IEEE Geoscience and Remote Sensing Letters*, *19*, 1-5.

Glanz, K., Rimer, B. K., & Viswanath, K. (Eds.). (2015). *Health behavior: Theory, research, and practice*. John Wiley & Sons.

González-Granadillo, G., González-Zarzosa, S., & Diaz, R. (2021). Security information and event management (SIEM): analysis, trends, and usage in critical infrastructures. *Sensors*, *21*(14), 4759.

Guo, H., Yuan, S. and Wu, X. (2021) 'LogBERT: Log Anomaly Detection via BERT'. Available at: http://arxiv.org/abs/2103.04475.

He, P. *et al.* (2016) 'An evaluation study on log parsing and its use in log mining', *Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016*, pp. 654–661. doi: 10.1109/DSN.2016.66.

He, P. *et al.* (2017) 'Drain: An Online Log Parsing Approach with Fixed Depth Tree', *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, pp. 33–40. doi: 10.1109/ICWS.2017.13.

He, P. *et al.* (2018) 'Towards Automated Log Parsing for Large-Scale Log Data Analysis', *IEEE Transactions on Dependable and Secure Computing*, 15(6), pp. 931–944. doi: 10.1109/TDSC.2017.2762673.

Hindy, H. *et al.* (2020) 'A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems', *IEEE Access*, 8, pp. 104650–104675. doi: 10.1109/ACCESS.2020.3000179.

Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, Michael R. Lyu. Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics. IEEE International Symposium on Software Reliability Engineering (ISSRE), 2023

Kamoona, A. M., Gostar, A. K., Bab-Hadiashar, A., & Hoseinnezhad, R. (2019, October). Sparsity-based naive bayes approach for anomaly detection in real surveillance videos. In *2019 international conference on control, automation and information sciences (ICCAIS)* (pp. 1-6). IEEE.

Karthik, M. G., & Krishnan, M. M. (2021). Hybrid random forest and synthetic minority over sampling technique for detecting internet of things attacks. *Journal of Ambient Intelligence and Humanized Computing*, 1-11.

Kaur, J., Kaur, K., Kant, S., & Das, S. (2022, November). UEBA with Log Analytics. In *2022 3rd International Conference on Computing, Analytics and Networks (ICAN)* (pp. 1-7). IEEE.

Khaliq, S., Abideen Tariq, Z. U. and Masood, A. (2020) 'Role of User and Entity Behavior Analytics in Detecting Insider Attacks', *1st Annual International Conference on Cyber Warfare and Security, ICCWS 2020 - Proceedings*. doi: 10.1109/ICCWS48432.2020.9292394.

Khaliq, S., Tariq, Z. U. A., & Masood, A. (2020, October). Role of user and entity behavior analytics in detecting insider attacks. In *2020 International Conference on Cyber Warfare and Security (ICCWS)* (pp. 1-6). IEEE.

Khan, M. Z. A., Khan, M. M., & Arshad, J. (2022, December). Anomaly Detection and Enterprise Security using User and Entity Behavior Analytics (UEBA). In *2022 3rd International Conference on Innovations in Computer Science & Software Engineering (ICONICS)* (pp. 1-9). IEEE.

Khanna, S. (2022) 'Computer Vision User Entity Behavior Analytics', (January). doi: 10.13140/RG.2.2.29840.94721.

Krishnaveni, S., Vigneshwar, P., Kishore, S., Jothi, B., & Sivamohan, S. (2020). Anomaly-based intrusion detection system using support vector machine. In *Artificial intelligence and evolutionary computations in engineering systems* (pp. 723-731). Singapore: Springer Singapore.

Kwon, T. H., Park, S. H., Park, S. I., & Lee, S. H. (2021). Building information modeling-based bridge health monitoring for anomaly detection under complex loading conditions using artificial neural networks. *Journal of Civil Structural Health Monitoring*, *11*, 1301-1319.

LaCaille, L. (2020). Theory of reasoned action. *Encyclopedia of behavioral medicine*, 2231-2234.

Lakshminarasimman, S., Ruswin, S., & Sundarakantham, K. (2017, March). Detecting DDoS attacks using decision tree algorithm. In *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)* (pp. 1-6). IEEE.

Landauer, M. *et al.* (2018) 'Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection', *Computers and Security*, 79, pp. 94–116. doi: 10.1016/j.cose.2018.08.009.

Landauer, M. *et al.* (2020) 'System log clustering approaches for cyber security applications: A survey', *Computers and Security*, 92, p. 101739. doi: 10.1016/j.cose.2020.101739.

Landauer, M. *et al.* (2022) 'A User and Entity Behavior Analytics Log Data Set for Anomaly Detection in Cloud Computing', *Proceedings - 2022 IEEE International Conference on Big Data, Big Data 2022*, pp. 4285–4294. doi: 10.1109/BigData55660.2022.10020672.

Landauer, M., Skopik, F., Höld, G., & Wurzenberger, M. (2022, December). A User and Entity Behavior Analytics Log Data Set for Anomaly Detection in Cloud Computing. In *2022 IEEE International Conference on Big Data (Big Data)* (pp. 4285-4294). IEEE.

Lin, Q. *et al.* (2016) 'Log clustering based problem identification for online service systems', *Proceedings - International Conference on Software Engineering*, pp. 102–111. doi: 10.1145/2889160.2889232.

Lin, T. H. and Jiang, J. R. (2020) 'Anomaly Detection with Autoencoder and Random Forest', *Proceedings - 2020 International Computer Symposium, ICS 2020*, pp. 96–99. doi: 10.1109/ICS51289.2020.00028.

Lin, T. H., & Jiang, J. R. (2020, December). Anomaly detection with autoencoder and random forest. In *2020 International Computer Symposium (ICS)* (pp. 96-99). IEEE.

Liu, C., Gu, Z., & Wang, J. (2021). A hybrid intrusion detection system based on scalable K-means+ random forest and deep learning. *Ieee Access*, *9*, 75729-75740.

Liu, H. (2021) 'A insider threat detection system based on user and entity behavior analysis', in *Journal of Physics: Conference Series*. IOP Publishing Ltd. doi: 10.1088/1742-6596/1994/1/012021.

Liu, Z. *et al.* (2018) 'An integrated method for anomaly detection from massive system logs', *IEEE Access*, 6, pp. 30602–30611. doi: 10.1109/ACCESS.2018.2843336.

Lukashin, A., Popov, M., Bolshakov, A., & Nikolashin, Y. (2020). Scalable data processing approach and anomaly detection method for user and entity behavior analytics platform. In *Intelligent Distributed Computing XIII* (pp. 344-349). Springer International Publishing.

Manimurugan, S. (2021). IoT-Fog-Cloud model for anomaly detection using improved Naïve Bayes and principal component analysis. *Journal of Ambient Intelligence and Humanized Computing*, 1-10.

Market Guide for User and Entity Behavior Analytics (no date). Available at: https://www.gartner.com/en/documents/3872885 (Accessed: 18 January 2024).

Martín, A. G. *et al.* (2021) ‘An approach to detect user behaviour anomalies within identity federations’, *Computers and Security*, 108. doi: 10.1016/j.cose.2021.102356.

Martín, A. G. *et al.* (2022) ‘Combining user behavioural information at the feature level to enhance continuous authentication systems’, *Knowledge-Based Systems*, 244, p. 108544. doi: 10.1016/j.knosys.2022.108544.

Martín, A. G., Beltrán, M., Fernández-Isabel, A., & de Diego, I. M. (2021). An approach to detect user behaviour anomalies within identity federations. *computers & security*, *108*, 102356.

Meng, W. *et al.* (2019) ‘Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs’, *IJCAI International Joint Conference on Artificial Intelligence*, 2019-Augus, pp. 4739–4745. doi: 10.24963/ijcai.2019/658.

Motiwalla, L. *et al.* (2019) ‘Leveraging Data Analytics for Behavioral Research’, *Information Systems Frontiers*, 21(4), pp. 735–742. doi: 10.1007/s10796-019-09928-8.

Muliukha, V. *et al.* (2020) ‘Anomaly detection approach in cyber security for user and entity behavior analytics system’, *ESANN 2020 - Proceedings, 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, (19), pp. 251–256.

Otieno, O. C., Liyala, S., Odongo, B. C., & Abeka, S. O. (2016). Theory of reasoned action as an underpinning to technological innovation adoption studies.

Patel, P., & Thakkar, A. (2020). The upsurge of deep learning for computer vision applications. *International Journal of Electrical and Computer Engineering*, *10*(1), 538.

Pawar, K., & Attar, V. (2019). Deep learning approaches for video-based anomalous activity detection. *World Wide Web*, *22*(2), 571-601.

Prathapchandran, K., & Janani, T. (2021). A trust aware security mechanism to detect sinkhole attack in RPL-based IoT environment using random forest–RFTRUST. *Computer Networks*, *198*, 108413.

Primartha, R. and Tama, B. A. (2017) 'Anomaly detection using random forest: A performance revisited', *Proceedings of 2017 International Conference on Data and Software Engineering, ICoDSE 2017*, 2018-Janua, pp. 1–6. doi: 10.1109/ICODSE.2017.8285847.

Primartha, R., & Tama, B. A. (2017, November). Anomaly detection using random forest: A performance revisited. In *2017 International conference on data and software engineering (ICoDSE)* (pp. 1-6). IEEE.

Priya, V., Thaseen, I. S., Gadekallu, T. R., Aboudaif, M. K., & Nasr, E. A. (2021). Robust attack detection approach for IIoT using ensemble classifier. *arXiv preprint arXiv:2102.01515*.

Raguvir, S. and Babu, S. (2020) 'Detecting anomalies in users-An UEBA approach', *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 0(March), pp. 863–876.

Raguvir, S., & Babu, S. (2020, March). Detecting Anomalies in Users–An UEBA Approach. In *Proceedings of the International Conference on Industrial Engineering and Operations Managemen, Dubai, United Arab Emirates (UAE)* (pp. 10-12).

Rasheed Yousef and Mahmoud Jazzar (2021) 'Measuring the Effectiveness of User and Entity Behavior Analytics for the Prevention of Insider Threats', *Journal of Xi'an University of Architecture & Technology XIII(X, 2021):175-181*, XIII(X), pp. 175–181. doi: 10.37896/JXAT13.10/313918.

Rashid, F., & Miri, A. (2021, May). User and event behavior analytics on differentially private data for anomaly detection. In *2021 7th IEEE Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)* (pp. 81-86). IEEE.

Rengarajan, R., & Babu, S. (2021, March). Anomaly Detection using User Entity Behavior Analytics and Data Visualization. In *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 842-847). IEEE.

Ribeiro, M., Lazzaretti, A. E., & Lopes, H. S. (2018). A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, *105*, 13-22.

Sahu, N. K., & Mukherjee, I. (2020, June). Machine learning based anomaly detection for IoT network:(Anomaly detection in IoT network). In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)* (pp. 787-794). IEEE.

Salitin, M. A., & Zolait, A. H. (2018, November). The role of User Entity Behavior Analytics to detect network attacks in real time. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)* (pp. 1-5). IEEE.

Salitin, M. A., & Zolait, A. H. (2023). Evaluation criteria for network security solutions based on behaviour analytics. *International Journal of Systems, Control and Communications*, *14*(2), 132-147.

Shaik, A. B., & Srinivasan, S. (2019). A brief survey on random forest ensembles in classification model. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2018, Volume 2* (pp. 253-260). Springer Singapore.

Sharafaldin, I. *et al.* (2017) 'Towards a Reliable Intrusion Detection Benchmark Dataset', *Software Networking*, 2017(1), pp. 177–200. doi: 10.13052/jsn2445-9739.2017.009.

Sharma, B., Pokharel, P. and Joshi, B. (2020) 'User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder-Insider Threat Detection', in *ACM International Conference Proceeding Series*. Association for Computing Machinery. doi: 10.1145/3406601.3406610.

Sumologic (2020) *What is a Log File? | Sumo Logic*. Available at: https://www.sumologic.com/glossary/log-file/ (Accessed: 4 September 2021).

Tao, Y. *et al.* (2020) 'User Behavior Analysis by Cross-Domain Log Data Fusion', *IEEE Access*, 8, pp. 400–406. doi: 10.1109/ACCESS.2019.2961769.

Tsogbaatar, E., Bhuyan, M. H., Taenaka, Y., Fall, D., Gonchigsumlaa, K., Elmroth, E., & Kadobayashi, Y. (2021). DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT. *Internet of Things*, *14*, 100391.

Tuan, T. A., Long, H. V., Son, L. H., Kumar, R., Priyadarshini, I., & Son, N. T. K. (2020). Performance evaluation of Botnet DDoS attack detection using machine learning. *Evolutionary Intelligence*, *13*, 283-294.

Vargaftik, S., Keslassy, I., Orda, A., & Ben-Itzhak, Y. (2021). RADE: resource-efficient supervised anomaly detection using decision tree-based ensemble methods. *Machine Learning*, *110*(10), 2835-2866.

Viegas, E. K., Santin, A. O. and Oliveira, L. S. (2017) 'Toward a reliable anomaly-based intrusion detection in real-world environments', *Computer Networks*, 127, pp. 200–216. doi: 10.1016/J.COMNET.2017.08.013.

Vinayakumar, R., Soman, K. P. and Poornachandran, P. (2017) 'Long short-term memory based operation log anomaly detection', *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, 2017-Janua, pp. 236–242. doi: 10.1109/ICACCI.2017.8125846.

Weerasinghe, S., Erfani, S. M., Alpcan, T., & Leckie, C. (2019). Support vector machines resilient against training data integrity attacks. *Pattern Recognition*, *96*, 106985.

What is User Behavior Analytics (UBA) and User Entity Behavior Analytics (UEBA) | Splunk (no date). Available at: https://www.splunk.com/en_us/data-insider/user-behavior-analytics-ueba.html (Accessed: 5 September 2021).

Yadav, R. B., Kumar, P. S. and Dhavale, S. V. (2020) 'A Survey on Log Anomaly Detection using Deep Learning', *ICRITO 2020 - IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, pp. 1215–1220. doi: 10.1109/ICRITO48877.2020.9197818.

Zhao, Z., Xu, C., & Li, B. (2021). A LSTM-based anomaly detection model for log analysis. *Journal of Signal Processing Systems*, *93*, 745-751.

Zhu, J. *et al.* (2019) 'Tools and Benchmarks for Automated Log Parsing', *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, pp. 121–130. doi: 10.1109/ICSE-SEIP.2019.00021.

Zhu, J. et al. (2023) ‘Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics’, Proceedings - International Symposium on Software Reliability Engineering, ISSRE, pp. 355–366. doi: 10.1109/ISSRE59848.2023.00071.