ADVANCING LOG ANOMALY DETECTION USING DEEP LEARNING AND TRANSFORMER BASED AI MODELS

SHASHANK BHATNAGAR

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

LIVERPOOL JOHN MOORES UNIVERSITY
School of Computer Science

August 2023

DEDICATION

This research is a heartfelt dedication to my mentors, whose unwavering guidance and expertise have illuminated my academic journey. Their invaluable support and insightful feedback have played a pivotal role in shaping my ideas and refining my work. This dedication serves as a tribute to their profound influence on my intellectual growth. Moreover, I extend my dedication to my parents, whose boundless love and encouragement have been the driving force behind my educational pursuits. Their unwavering belief in me and the sacrifices they have made continue to motivate me. This work is a tribute to them, representing my deep appreciation for the values they've instilled in me and the opportunities they've provided. May this accomplishment reflect the enduring strength of their support and the profound impact of their guidance.

ACKNOWLEDGMENT

I would like to thank my Thesis Supervisor, Aayushi Verma, for her continued support and valuable feedback and suggestions that have helped me prepare this report.
I would also like to thank Dr. Manoj Jayabalan for his guidance in preparing this report.

ADVANCING LOG ANOMALY DETECTION USING DEEP LEARNING AND TRANSFORMER BASED AI MODELS

Abstract

This research study aims to propose a transformer-based method to classify anomalies from system logs that can process logs in real-time and address the challenges associated with extracting meaningful information from unstructured logs. The study also aims to propose a log sequence-based anomaly detection method that fully considers robustness issues caused by updating log message templates, such as concept drift, noise problems, and the challenges associated with processing a huge volume of logs produced by systems. The proposed method aims to detect anomalies in system log files automatically, requiring minimal manual marking of regular expressions by operation engineers, which can be an inefficient process when dealing with a huge volume of logs. The study will evaluate the effectiveness and generalizability of the proposed method using publicly available datasets. The study's findings can have a significant impact on the fields of cybersecurity and system monitoring, enabling organizations to detect abnormal behaviour in system logs, proactively identify and mitigate security threats, reduce downtime, and enhance system reliability. The study employs a comprehensive approach, including data loading, pre-processing, feature engineering, and model building, using multiple models such as log sequence models, transformer-based log classification, and fine-tuned language models.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ALBERT	A Lite Robustly Optimized BERT Pre-training Approach
BERT	Bidirectional Encoder Representations from Transformers
BGL	BlueGene/L
BoW	Bag of Words
CV	Cross Validation
EDA	Exploratory Data Analysis
LSTM	Long Short-Term Memory
NN	Neural Network
Q Learning	Quality Learning
roBERTa	Robustly Optimized BERT Pre-training Approach
ROC	Receiver operating characteristic
RNN	Recurrent neural network
TF-IDF	Term Frequency - Inverse Document Frequency
XLNet	Generalized Auto regressor Extension of Transformer Extra Long Mode

CHAPTER 1

INTRODUCTION

Log anomaly detection is a critical task in cybersecurity, system monitoring, and other domains. It involves identifying unusual patterns in logs that indicate security breaches, system failures, or operational inefficiencies. Traditional methods of log analysis and anomaly detection are labour-intensive and error prone. To address this, machine learning techniques have emerged as effective tools for automated log analysis. Advanced models, such as deep learning and transformer-based architectures, have shown promise in capturing complex patterns and adapting to evolving data distributions. This research aims to propose innovative techniques and models for log anomaly detection, leveraging machine learning advancements. The goal is to develop accurate and robust systems that can promptly detect log anomalies, enhancing cybersecurity, system performance, and operational efficiency. By contributing to the field of log anomaly detection, this research aims to benefit various industries and improve system reliability and security.

1.1 Background

Anomaly detection is an essential task in cybersecurity and system monitoring, which involves identifying unusual patterns in data that indicate potential security threats, system failures, or other abnormal events. Implementing anomaly detection techniques can help organizations detect potential security threats and system failures, improve the reliability and performance of their systems, prevent system downtime, enhance operational efficiency, and reduce costs associated with identifying performance issues and inefficiencies. Additionally, compliance with regulatory requirements can be ensured, avoiding legal issues and penalties for organizations. Overall, anomaly detection can help organizations improve their cybersecurity posture, increase their operational efficiency, and comply with regulatory requirements.

Despite the advantages, traditional methods of anomaly detection may have limitations. They can be inefficient, time-consuming, and lack robustness when handling unstructured log data. Some gaps in the state-of-the-art approaches include robustness and false positive and real-time processing. Machine learning-based approaches have emerged to address these challenges Despite this, machine learning-based methods can lack robustness and treat normal noise logs as abnormal, leading to poor performance. To address these challenges, the study proposes a log sequence LSTM (Zhao et al., n.d.) based anomaly detection method that fully considers robustness issues by updating the log message template, as well as concept drift and noise problems. The proposed method aims to detect anomalies in system log files automatically, requiring minimal manual marking of regular expressions by operation engineers, and will be evaluated using publicly available datasets BGL(Oliner and Stearley, 2007).The objective is to minimize false positive.

The proposed method aims to overcome the challenges associated with processing a large number of logs and proposes a transformer-based (for e.g. BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), roBERTa (Naudé et al., 2022), XLNET (Naudé et al., 2022), ALBERT (Lan et al., 2019; Choi et al., 2020) etc.) method that can process logs in real-time, classify anomalies, and enhance system reliability and security. The proposed research objectives align with addressing these gaps and advancing the state-of-the-art in log anomaly detection. Objectives such as analysing log classification issues, suggesting precise techniques, comparing AI-based approaches, and developing and evaluating the proposed log classification model directly contribute to bridging these gaps.

While the proposed method has potential limitations, such as the requirement for computational resources and data availability due to sensitive information, the research suggests extending the evaluation to other log datasets to validate the effectiveness and generalizability of the proposed method. This further supports the aim of developing an improved log anomaly detection model that can be applied across various industry scenarios.

1.2 Problem Statement

A specific gap in anomaly detection lies in the robustness and accuracy of existing methods when applied to unstructured system logs. Anomaly detection techniques face challenges due to the unstructured nature of log data, the high volume of logs generated, and the potential for concept drift and noise problems. Although there is existing literature on anomaly detection and the use of ML algorithms to improve it, there is often a lack of direct linkage between the two.

The literature on anomaly detection encompasses various techniques, including transformer models such as BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), roBERTa (Naudé et al., 2022), ALBERT (Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020), and DistilBERT (Jiao et al., 2019; Sanh et al., 2019b), log sequence models (Fu et al., 2023), and QL-Log/Q Learning models (Duan et al., 2021a) . These approaches aim to enhance anomaly detection accuracy by leveraging the power of ML and natural language processing (NLP) techniques. However, the literature often fails to adequately address the robustness issues, concept drift, and noise problems that can impact the performance of anomaly detection systems.

This thesis will do the following:

The proposed thesis makes a significant contribution by addressing these gaps in the literature. It proposes a log sequence LSTM (Zhao et al., n.d.; Malaiya et al., 2019) based anomaly detection method that considers robustness issues through the updating of log message templates. Additionally, a transformer-based method, utilizing models like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), roBERTa (Naudé et al., 2022) or XLNET (Naudé et al., 2022), is proposed to optimize real time log processing and improve anomaly classification. By integrating these approaches, the thesis aims to fill the gaps in existing literature by providing a more comprehensive and accurate anomaly detection solution for unstructured system logs

The study contributes by bridging the gap between the existing literature on anomaly detection techniques and the utilization of ML models to enhance their performance. It aims to improve the robustness and accuracy of anomaly detection systems, specifically focusing on unstructured system logs. The proposed methods address the challenges associated with concept drift, noise problems, and real-time log processing. The evaluation of the proposed methods on publicly available datasets, such as BGL (Oliner and Stearley, 2007), will validate their effectiveness and generalizability as the proposed method may require a lot of computational resources, which may not be practical for some real-world applications. Additionally, the unavailability of data due to sensitive log and secure information may limit the evaluation of the proposed method. Nevertheless, the study suggests that the evaluation could be extended to other log datasets to further validate the effectiveness and generalisability of the proposed method

Table 1.1 : Model Methods with approach and Limitations

Method	**Approach**	**Limitations**	**Reference**
RoBERTa	RoBERTa is a transformer-based language model similar to BERT but trained on a larger corpus of data using improved training techniques.	It lacks external knowledge sources, making it less effective for domain-specific tasks	(Naudé et al., 2022)
ALBERT	It reduces model size and training time while maintaining the performance	High computational cost and requirement of large amounts of data for pre-training also it has some limitations in domain specific tasks	(Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020)
DistilBERT	DistilBERT is a faster and smaller compressed version of BERT, which maintains most of its accuracy and performance on natural language processing tasks.	DistilBERT, being a compressed and faster version of BERT, may have lower accuracy and perform sub-optimally on tasks that require high level of precision and detail.	(Jiao et al., 2019; Sanh et al., 2019b)
XLNet	XLNet is a neural language model that utilizes a permutation-based approach to generate context-aware representations of input text, outperforming BERT on many NLP tasks.	XLNet has high computational demands, slow training pace, and intricate structure, making it challenging to fine-tune for downstream tasks.	(Naudé et al., 2022)

Method	Approach	Limitations	Reference
Log Sequence Model LSTM	Log Sequence Model LSTM is a deep learning method that utilizes LSTM neural networks to analyze system logs in sequence, identifying patterns in the log data to detect anomalies.	The Log Sequence Model LSTM has limitations such as high computational demands, hyperparameter selection challenges, and managing large amounts of unstructured data	(Zhao et al., n.d.; Malaiya et al., 2019)
BERT	BERT is a deep learning model that uses a bidirectional transformer to pre-train natural language processing tasks and achieve state-of-the-art results.	Need for large amounts of labeled data, high computational costs	(Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022)
BERT-Log	BERT-Log algorithm uses a pre-trained language model called BERT to learn the semantic representation of normal and anomalous logs	Need for large amounts of labeled data, high computational costs	
QL-Log/Q Learning	Q-learning is a type of machine learning algorithm that enables agents to learn the best actions to take in a particular environment. It does this by updating the expected rewards an agent can receive by performing certain actions in different states, which are called Q-values.	The limitations of Q-learning algorithms include the problem of exploration vs. exploitation, the curse of dimensionality, the requirement of a complete and accurate model, and the difficulty of handling continuous state and action spaces.	(Duan et al., 2021b)

Method	Approach	Limitations	Reference
Precision, Recall, and F1-score	The metrics Precision, Recall, and F1-score are used for evaluating classification models, where Precision is the true positive rate of positive predictions, Recall is the true positive rate of actual positives, and F1-score is the balanced harmonic mean of both metrics.	The limitations include their unsuitability for models that prioritize one metric over the other, their inability to handle imbalanced datasets, and their failure to capture the intricacies of a model's performance on specific data subsets.	(Chen and Liao, 2022b),
ROC AUC	ROC AUC is a metric for evaluating classification model performance that measures the area under the curve of the receiver operating characteristic	The ROC AUC curve has limitations such as limited handling of imbalanced datasets, insensitivity to changes in data distribution, and lack model performance explanation	(Chen and Liao, 2022b),
Accuracy	Accuracy refers to the ratio of correct predictions made by a model to the total number of predictions.	One limitation of accuracy is its potential to be misleading for imbalanced datasets, as a high accuracy score may not necessarily indicate good performance on the minority class due to its significantly lower representation in the dataset	(Chen and Liao, 2022b),

1.3 Aim & Objectives

The main aim of this research on log anomaly detection is to propose a model which can detect anomalies in system and event logs and can indicate system malfunctions, security breaches, or other unexpected behaviour. Identifying anomalies using the well-studied robust models allow us to improve system performance, increase reliability and availability, enhance security, reduce maintenance costs, and prevent potential failures or downtime.

The research objectives stated below are formulated based on the aim of the study are as follows:

- To analyze various issues related to log classification and anomaly detection, including real-time processing, robustness issues, noise problems, efficiency, and generalizability. This analysis will guide the selection and development of appropriate Log Sequence-based/AI Transformer-based techniques and machine learning models.

- To suggest a precise and complete description of the Log Sequence-based/AI Transformer-based techniques and machine learning models that will be utilized. Additionally, to propose a well-structured approach that incorporates pre-processing steps, model training, parameter optimization, and evaluation metrics to thoroughly assess the efficacy of the selected models.

- To compare the performance of different AI-based techniques in identifying anomalies. Through comprehensive experiments and evaluations, the models will be compared based on key performance metrics such as accuracy, precision, recall, and F1 score to determine the best performing model for anomaly detection in system and event logs.

- To develop and evaluate the performance of the proposed log classification model. The proposed model will be implemented, and its effectiveness will be evaluated using suitable datasets and evaluation metrics. The model's performance will be compared against existing state-of-the-art approaches to assess its superiority and potential practical applications.

1.4 Research Questions

Following research questions are formed considering the literature review done in the field of anomaly detection

- How can a new approach using transformer method be identified to classify anomalies from system logs in real-time and address the issue of extracting meaningful information from unstructured logs?

- What are the drawbacks of recent log anomaly detection studies, such as concept drift, noise problems, and how can they be addressed to improve performance and reduce data instability and abnormal misjudgment?

- What are the potential limitations of the proposed method, such as requiring a lot of computational resources and limited evaluation due to sensitive log and secure information, and how can they be overcome or mitigated to improve the effectiveness and generalizability of the proposed method?

1.5 Scope of the Study

Considering the established deadlines for conducting this research, the scope of the study is limited as follows:

1.5.1 In scope

The proposed research aims to develop and evaluate the performance of anomaly detection models using transformers and log sequence models to detect abnormal behavior in system logs in real-time. The research will compare the performance of different models in terms of accuracy(Chen and Liao, 2022b), precision(Chen and Liao, 2022b), recall(Chen and Liao, 2022b), F1-score(Chen and Liao, 2022b), and AUC-ROC(Chen and Liao, 2022b). The study will also investigate the robustness issues, noise problems, efficiency, and generalizability of the proposed models.

1.5.2 Out of scope

The proposed research will not focus on investigating new methods of anomaly detection, developing new technologies to enhance the performance of the anomaly detection models, evaluating the quality and usability of the system logs, conducting a detailed analysis of the impact of the anomalies on the system, identifying ways to improve the performance of the system, and enhancing the cybersecurity posture of the organization.

1.5.3 Reason for defining the scope

Defining the scope is crucial in research as it clarifies the research's objectives and what is not relevant to the study, as with the proposed research on anomaly detection models, defining in-scope activities helps to ensure that the research stays focused on developing and evaluating the models' performance, while defining out-of-scope activities helps prevent distractions and irrelevant investigations.

1.6 Significance of the Study

The expected outcome of this study is to identify the artificial intelligence (AI) based anomaly detection model which will be significant in fields such as cybersecurity, system monitoring, predictive maintenance, and fraud detection, where organizations can identify unusual patterns in data and alert them to potential threats or issues. By detecting such anomalies proactively, organizations can minimize the risk of system downtime, data breaches, or financial losses and improve their operational efficiency and reliability. In cybersecurity, anomaly detection models monitor network traffic, log files, and other data sources in real-time, detecting patterns that deviate from normal behaviour and alerting security teams. In system monitoring, these models analyse system logs and telemetry data to detect patterns that may indicate impending failures, allowing organizations to take preventative measures and minimize downtime. In predictive maintenance, anomaly detection models analyse sensor data to identify equipment that requires maintenance or repair, while in fraud detection, these models analyse transactional data to identify unusual behaviour that may indicate fraudulent activity. By detecting anomalies early, organizations can take corrective action and improve their overall efficiency and reliability. By

incorporating proposed techniques into the anomaly detection process, the proposed model aims to achieve a higher accuracy rate and can potentially improve detection rates by up to 5%.

1.7 Structure of the study

The Proposed Structure of the thesis is as follows : Chapter 1 Introduction In this chapter, the research topic of anomaly detection in system logs and the need for an improved transformer-based log classification approach are introduced (Section 1.1 Background). The problem statement addresses the challenges faced by current anomaly detection methods and proposes the use of the proposed transformer method as a solution (Section 1.2 Problem Statement). The study's aim and objectives are to enhance anomaly detection accuracy by updating log message templates and incorporating efficient log parsing techniques (Section 1.3 Aim and Objectives). The research questions focus on evaluating the proposed method's effectiveness, generalizability, and performance compared to existing techniques (Section 1.4 Research Questions). The scope of the study includes transformer-based models, hyperparameter tuning, and evaluation on a selected dataset, with specified out-of-scope elements (Section 1.5 Scope of the Study). The reason for defining the scope is to ensure feasibility and relevance to the research objectives, considering practical limitations (Section 1.5.3 Reason for Defining the Scope). Additionally, the significance of the study is highlighted, emphasizing its potential contributions to the field of anomaly detection and log classification (Section 1.6 Significance of the Study).

Chapter 2 Literature Review, Section 2.1 provides an overview of existing research on anomaly detection, highlighting recent studies, surveys, journals, and publications. In Section 2.2, different methods for anomaly detection are categorized into traditional, machine learning, and deep learning approaches. The models covered include fuzzy C-means, MLP-based, log-based, Q-learning-based, NLP-based, and pre-trained transformer Bert model for anomaly detection (Subsections 2.2.1 to 2.2.8). Subsection 2.2.12 explores evaluation metrics and performance evaluation methods used in anomaly detection tasks, while Subsection 2.2.13 delves into the process of fine-tuning pre-trained language models for anomaly detection. Subsection 2.2.14 focuses on techniques for optimizing machine learning models and hyperparameter tuning, and Subsection 2.2.15 discusses challenges and limitations in anomaly detection approaches. In Section 2.3, a summary of the literature review highlights identified gaps and research areas that the current study aims to address.

Chapter 3 Research Methodology, Section 3.1 introduces the research methodology, presenting the overall approach and steps taken to achieve the research objectives. In Section 3.2, the research approach encompasses data selection (3.2.1), data pre-processing (3.2.2), feature extraction (3.2.3), model building (3.2.4), hyperparameter tuning (3.2.5), model evaluation (3.2.6), and the proposed hybrid method. Subsection 3.2.1 provides details on the dataset chosen for the study, while Subsection 3.2.2 delves into data pre-processing steps like exploratory data analysis, cleaning, and outlier analysis. Subsection 3.2.3 discusses feature extraction techniques like Bag-of-Words and TF-IDF essential for log classification. In Subsection 3.2.4, different log classification models, including log sequence models and transformer-based models, are explored. The significance of hyperparameter tuning and the techniques used for model optimization are covered in Subsection 3.2.5. Subsection 3.2.6

outlines the model evaluation metrics and methods employed. Additionally, Section 3.3 introduces the proposed hybrid approach, combining LSTM and BERT for improved anomaly detection. The expected research outcomes and deliverables are outlined in Section 3.4 to set clear expectations. Finally, Section 3.5 provides a proposed flow diagram to visualize the research methodology's overall structure and process

Chapter 4 : Analysis ,This chapter Analysis uncovers the results derived from our systematic approach, as outlined in the previous Research Methodology chapter (Chapter 3), for conducting Log Anomaly Detection using the BGL dataset. Our primary focus was on identifying anomalies within the BGL dataset. The initial step involved acquiring a comprehensive comprehension of the dataset to facilitate the development of suitable models. In Section 4.2, we present a comprehensive overview of the BGL dataset, a compilation of system logs from the BlueGene/L supercomputer, outlining its distinctive characteristics as the foundation for our Log Anomaly Detection study. Moving on to Section 4.3, our exploration of the dataset begins, meticulously inspecting its contents and revealing crucial insights such as missing values, duplicates, memory usage, variable types, cardinality, correlations, and class imbalances. Section 4.4 signifies a critical phase in our study, where we prepare the data for anomaly detection by initializing paths and applying log parsing using the Drain Algorithm. This phase includes feature engineering and column transformations to enhance data quality. Additionally, in Section 4.4.1, we delve into the Drain Algorithm's application, focusing on log parsing and standardization. Section 4.4.2 elucidates how specific columns undergo transformations to facilitate subsequent analyses, while Section 4.4.3 introduces log sequencing using sliding windows to maintain data context and event sequence. Finally, in Section 4.5, we provide a comprehensive summary of our dataset description, exploratory analysis, data pre-processing, and feature engineering efforts, setting the stage for the subsequent phases of our Log Anomaly Detection study.

Chapter 5: Results and Discussions , In Section 5.2, our attention shifts towards the core of our research: Model Experimentation. Within this section, we embark on an extensive exploration of various model categories, all crafted to address the challenge of Log Anomaly Detection. We commence with the Baseline Models in Section 5.2.1, diving into essential techniques like PCA (Principal Component Analysis), Isolation Forest, One-Class SVM, and LogClustering. This segment concludes with a brief recap of our discoveries. Transitioning to Section 5.2.2, we investigate Transformer Models, encompassing Bert Base, Tuned Bert, DistilBERT, RoBERTa, Albert, and XLNet models. Each model undergoes evaluation for its effectiveness in anomaly detection, and we consolidate the collective insights. In Section 5.2.3, we delve into LSTM Models, covering the Base LSTM Model, Tuned LSTM Model, and BiLSTM Model, with each model's performance assessed and summarized. Finally, Section 5.2.4 introduces Hybrid Models, uniting LSTM and BERT capabilities, with detailed examinations and summarizations of each hybrid model variant. To conclude our journey, Section 5.3 represents the Model Evaluation and Summary phase, where we thoroughly assess the performance of each model, resulting in an inclusive overview of our Log Anomaly Detection investigation

Chapter 6: Discussion and Conclusion initiates with an Introduction (6.1) that paves the way for summarizing the study's outcomes. The core of this chapter lies in the Discussion and Conclusion (6.2) section, subdivided into several parts. Baseline Models Recommendation (6.2.1) offers detailed guidance on baseline models, highlighting their unique strengths and limitations. Transformer Models Recommendation (6.2.2) discusses transformer-based models, emphasizing their suitability and performance in specific tasks. LSTM Models

Recommendation (6.2.3) explores LSTM models' efficacy in anomaly detection. Meanwhile, Hybrid Models (6.2.4) investigates models combining various techniques, revealing their potential applications. The chapter culminates in the Recommended Model (6.2.5), conclusively suggesting the most suitable model, grounded in empirical evidence and performance metrics. In the Contribution to Knowledge (6.3) section, the study explores critical facets, including challenges in log classification and anomaly detection (6.3.1), explanations of methods and models (6.3.2), performance comparisons of AI-based approaches (6.3.3), and the development and evaluation of the proposed model (6.3.4). The chapter candidly acknowledges Limitations (6.4), covering dataset constraints (6.4.1), inherent limitations in anomaly detection research (6.4.2), model-specific restrictions (6.4.3), challenges in interpreting anomalies and domain adaptation (6.4.4), resource-intensive concerns and scalability hurdles (6.4.5), and strategies for mitigating these limitations and future prospects (6.4.6). Finally, the chapter concludes with Future Recommendations (6.5), outlining pathways for future research, model enhancements, and potential directions to advance the field. This structured approach ensures a comprehensive and insightful conclusion to the research paper.

CHAPTER 2

LITERATURE REVIEW

Log anomaly detection plays a pivotal role in fortifying cybersecurity, network monitoring, troubleshooting, and predictive maintenance. It encompasses the discernment of atypical patterns in log data. In this literature review, we delve into the existing research and techniques employed in log anomaly detection. We engage in a thorough analysis of the literature to identify the various approaches, methodologies, algorithms, and evaluation metrics utilized in this domain. The review also encapsulates the obstacles, constraints, and voids prevalent in the literature. Our objective is to establish a solid groundwork for designing efficacious log anomaly detection methods, thereby enhancing the security and dependability of intricate information systems.

2.1 Introduction

Anomaly detection has received significant attention in fields like cybersecurity, system monitoring, and software applications due to its vital role in enhancing system quality, reliability, and security. As modern systems become more complex and extensive, manual monitoring and analysis of system logs have become impractical, leading to the development of automated anomaly detection methods. This literature review provides a comprehensive overview of existing research in the field of log anomaly detection, highlighting notable contributions and advancements.

In the domain of log-based anomaly detection, automatic log parsers have emerged as crucial tools, particularly heuristic-based parsers, which have demonstrated effectiveness in parsing system logs. The accuracy of log parsing and the number of parsed event templates significantly impact the performance and efficiency of anomaly detection. Furthermore, the adoption of machine learning and deep learning techniques has shown promising outcomes in log anomaly detection.

The Bert Log Paper introduces a proposed approach called BERT-Log (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022) that utilizes pre-trained language models to learn the semantic representation of normal and anomalous logs, achieving high accuracy and generalization ability in anomaly detection. The literature also explores LSTM-based models (Zhao et al., n.d.; Malaiya et al., 2019) for anomaly detection, specifically focusing on character-level log text analysis. However, further research and improvements are necessary to address the limitations of these models. ALBERT (Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020), a parameter-reduced version of BERT, tackles memory limitations and longer training times while achieving state-of-the-art performance with fewer parameters.

Anomaly detection in log files using selected natural language processing methods addresses the challenge of detecting anomalies in system log files and discusses various approaches, including supervised and unsupervised learning methods. The literature proposes an LSTM-based (Zhao et al., n.d.; Malaiya et al., 2019) sequence mining model that simplifies calculation complexity of log text analysis. Nonetheless, further research is needed to overcome specific limitations.

The literature review also covers DistilBERT (Jiao et al., 2019; Sanh et al., 2019b), a distilled version of BERT, which focuses on knowledge distillation to develop lighter and faster models that attain comparable performance in downstream tasks. Additionally, HitAnomaly (Huang et al., 2020a) sheds light on the difficulties of log-based anomaly detection, particularly in analysing unstructured log data. Other proposed methods, such as Log Sequence Anomaly Detection Method and QLLog (Duan et al., 2021b) explore different techniques like contrastive adversarial training, dual feature extraction, and Q-learning algorithms to achieve effective log anomaly detection.

In conclusion, the literature review encompasses a wide range of research endeavours in log anomaly detection, emphasizing the significance of automated methods for analysing system logs. By leveraging machine learning and natural language processing techniques, these methods contribute to enhancing the accuracy and efficiency of anomaly detection, thereby improving system reliability, security, and performance across diverse domains.

2.2 Anomaly Detection: Recent Studies, Surveys, Journals and Publications

To generate in depth understanding of the work done in log anomaly detection domain we went through few surveys, publications, case studies and articles. This help us to understand the approaches which has been used over the years in this domain

Below is the list of approaches that we came across while doing the surveys and challenges which help us to understand the problem area which is in focus in recent years.

2.2.1 Traditional Methods of Anomaly Detection

Traditional Methods of Anomaly Detection includes Rules Based and Statistics based approaches

Rule Based Anomaly Detection

The initial stage in the development of a rule-based anomaly detection system (Hela et al., 2018) involves the generation of a collection of association rules through the utilization of historical data obtained from various sources. These association rules serve to capture the interconnections and dependencies that exists. The generation of these rules is based on the observation of patterns and correlations that are discernible within the historical data. After the rules have been generated, they are assessed against real-time data. The system continuously

monitors the data stream and compares it with the predefined rules to identify any deviations or anomalies. Each incoming event or activity is scrutinized against the set of rules to ascertain if it complies with any rule or contravenes any predefined patterns. If an event or activity deviates significantly from the anticipated patterns, it is flagged as an anomaly.

Statistical Based Anomaly Detection

On Further reviewing literatures on Statistical based Anomaly Detection(Nooribakhsh and Mollamotalebi, 2020; Jasra et al., 2022) we came across Data-driven statistical techniques, like the Local Outlier Factor (LOF) algorithm (Jasra et al., 2022), are used in statistical-based anomaly detection to find outliers or deviations in datasets. Instead of categorising data points as either inliers or outliers, the LOF method assesses how outlier-like each data point is. The goal of statistical-based anomaly detection is to locate data examples that significantly differ from predicted patterns by examining statistical variables including mean, standard deviation, covariance, and distribution characteristics. LOF algorithm makes it possible to recognise unusual data patterns and provide anomaly scores for unsupervised comparison. Additionally, it aids in the long-term monitoring of unusual data behaviour. In general, statistical-based anomaly detection, which includes the LOF algorithm, offers a useful data-driven method for spotting abnormalities and learning about different anomaly types.

After doing survey on traditional approaches, we came across below challenges and limitation

- Both statistical and rules-based approaches to anomaly detection. Context-specific anomalies that differ from accepted statistical norms may be difficult for statistical tools to detect. Rules-based techniques, on the other hand, have difficulty identifying unknown or unexpected anomalies that do not follow specified rules. As new patterns arise, rules must be continuously updated and refined, and these approaches may be less flexible to changing and dynamic data patterns.

2.2.2 Machine Learning for Anomaly Detection

Machine learning-based approaches have emerged to address these challenges faced in traditional methods of anomaly detection. During our comprehensive literature review (Nassif et al., 2021), we identified and classified machine learning (ML) techniques into six distinct groups: classification models, ensemble models, optimization models, rule-based systems, clustering, and regression. These techniques were commonly employed either as standalone models or in hybrid forms, often involving the combination of multiple ML techniques. Notably, support vector machines (SVM) (Nassif et al., 2021) emerged as the most widely used technique, demonstrating its popularity in both independent and hybrid model implementations.

To enhance model precision and computational efficiency by eliminating irrelevant data, researchers extensively explored feature selection and extraction techniques such as Principal Component Analysis (PCA) (Nassif et al., 2021) and Correlation-based Feature Selection (CFS) (Nassif et al., 2021). These techniques aimed to optimize the performance of anomaly detection models by focusing on the most relevant features.

However, our literature review also highlighted several challenges and limitations associated with machine learning-based anomaly detection methods. One-class SVM (Nassif et al., 2021) and Naïve Bayesian models (Nassif et al., 2021), despite their effectiveness in detecting known malicious behaviour or abnormal states, face difficulties when it comes to real-world anomaly detection due to their high false positive rates. K-Means Clustering (Nassif et al., 2021) encounters issues with class dominance and forced assignment, which can hinder its accuracy in certain scenarios. Adaboost (Nassif et al., 2021), although it performs well on less noisy data, exhibits poor performance in the presence of noise. Furthermore, Logistic Regression (LR) (Nassif et al., 2021) and Random Forest (RF) (Nassif et al., 2021) models encounter challenges related to high detection accuracy.

Addressing these challenges requires further research and innovation. Exploring alternative machine learning techniques and hybrid models may also present promising avenues for overcoming the limitations observed in individual methods.

2.2.3 Deep Learning Techniques for Anomaly Detection

To address challenges faced in Machine Learning approaches we have further explored deep learning area. In the literature review of deep learning models with LSTM (Malaiya et al., 2019) for anomaly detection, several important aspects were explored. Researchers focused on various data processing techniques to pre-process log data effectively and enhance the performance of LSTM (Zhao et al., n.d.; Malaiya et al., 2019) models. Different network architectures were investigated which includes Fully Connected Networks (FCNs) (Malaiya et al., 2019), Variational Autoencoder (Malaiya et al., 2019), Sequence-to-Sequence (Seq2Seq) (Malaiya et al., 2019), including variations in LSTM layers, hidden unit sizes, and bidirectional LSTMs, aiming to capture temporal dependencies and detect anomalies more accurately. Parameter adjustment played a crucial role in optimizing model performance, with researchers fine-tuning parameters such as learning rate, batch size, and dropout rate.

A notable contribution was the proposal of a feature extraction mechanism that precisely characterized malicious behaviours. This mechanism enabled the LSTM model to distinguish anomalies from normal log patterns effectively. Comparative evaluations were conducted with popular anomaly detection algorithms like one-class SVM (Zhao et al., n.d.), GMM (Zhao et al., n.d.), and PCA (Zhao et al., n.d.), consistently showing the superiority of the LSTM model in terms of accuracy and efficiency.

Deep learning models, including LSTM, offer several advantages for anomaly detection. They eliminate the need for manual feature extraction, making them suitable for handling complex and unstructured log data. LSTM models excel in capturing temporal dependencies, enabling the detection of subtle abnormalities indicative of anomalous behaviour.

Overall, the literature suggests that deep learning models with LSTM have the potential to improve accuracy and efficiency compared to traditional methods in anomaly detection. However, deep learning approaches also come with challenges. They often require a significant amount of labelled data for training, which may not always be available. Deep learning models are computationally demanding and may require substantial resources. Interpreting their results can be challenging since they operate as black boxes. Ongoing research aims to address the challenges associated with these models and further enhance their performance in detecting anomalies in log data.

2.2.4 Fuzzy C-means and MLP based - Anomaly Detection

The reviewed paper (Farzad and Gulliver, 2022) introduces a novel method for detecting anomalies in log messages. The proposed approach combines radius-based fuzzy C-means and an MLP network to achieve improved results compared to existing methods. The method involves selecting reliable positive and negative logs using cluster centres and a radius and employs class probabilities and expert correction to refine suspect log outputs. The proposed model is evaluated on three well-known datasets: BGL, Open stack, and Thunderbird.

The paper concludes that log messages contain valuable information and automating anomaly detection using these messages can offer significant time and cost savings. The proposed method, utilizing radius-based fuzzy C-means and an MLP network, outperforms existing techniques for detecting anomalies in log messages. The utilization of cluster centres, radius values, and class probabilities, combined with expert correction, enhances the practicality and effectiveness of the model.

However, the paper lacks an explicit discussion of the proposed method's limitations. It should be noted that the evaluation is limited to three datasets, which may limit the generalizability of the approach to other datasets. Additionally, the absence of a comparison with other state-of-the-art methods for log message anomaly detection leaves unanswered questions regarding performance comparisons. Future research should address these limitations to provide a more comprehensive evaluation and comparative analysis.

2.2.5 Log Based- Anomaly Detection

During our extensive literature review, we encountered a research paper (Fu et al., 2023) that thoroughly investigated the impact of different log parsers on the performance of log-based anomaly detection methods, specifically focusing on machine learning-based and deep learning-based approaches. The paper proposed a systematic methodology that encompassed log parsers from various categories, including heuristic-based, frequency-based, clustering-based, and subsequence-based methods. Among the six log-based anomaly detection methods analyzed in the study, heuristic-based log parsers consistently demonstrated remarkable effectiveness and efficiency.

The study highlighted an important finding that achieving high parsing accuracy alone does not guarantee superior anomaly detection performance. It emphasized the significance of considering both the parsing accuracy and the number of parsed event templates when selecting log parsers for effective anomaly detection. Notably, heuristic-based log parsers, such as Drain (He et al., 2017; Fu et al., 2023) and IPLoM (Fu et al., 2023), emerged as the top performers across the evaluated anomaly detection techniques.

These research findings provide valuable insights for researchers and practitioners seeking to optimize their log-based anomaly detection models. By leveraging heuristic-based log parsers, they can enhance the effectiveness and efficiency of their anomaly detection systems, leading to improved anomaly detection performance and overall system reliability.

2.2.6 Q-learning Based- Anomaly Detection

During the literature review, we came across a research paper that proposes QLLog (Duan et al., 2021a), a log anomaly detection method based on Q-learning. QLLog offers several notable features, including the ability to detect various types of system anomalies, rank abnormal events based on severity, and update the detection model. This method performs anomaly detection at both the log sequence and log entry levels, providing a comprehensive approach to identify anomalies. An additional advantage is that QLLog incorporates ops feedback reports, enabling online updates to the anomaly detection model and facilitating adaptability to new log states and sequences.

However, it is important to consider the limitations highlighted in the paper. One limitation is related to the scalability of the Q table and abnormal level table, which can become inefficient to store and learn as the number of log states increases. Furthermore, QLLog may generate a considerable number of false positives, which can impact the efficiency of the overall detection process. The paper also acknowledges the difficulty in quickly and accurately detecting anomalies in certain log datasets, such as the Blue Gene/L supercomputer system dataset, due to their large number of log entries and diverse log types.

To overcome these limitations, future research should focus on improving the efficiency of storing and learning the Q table and abnormal level table, considering approaches like data compression or approximation techniques. Strategies to reduce false positives, such as incorporating more advanced anomaly scoring mechanisms, should also be explored. Additionally, developing specialized techniques to effectively handle complex log datasets is crucial for enhancing the accuracy and timeliness of anomaly detection.

In conclusion, while the QLLog method shows promise in log anomaly detection, addressing its limitations is vital. By addressing scalability concerns, reducing false positives, and developing effective techniques for complex log datasets, the QLLog method can be further refined and optimized for practical anomaly detection applications.

2.2.7 Natural Language Processing Based- Anomaly Detection

The challenges in machine learning-based anomaly detection arise from the reliance on predefined features and patterns, which involve laborious manual feature engineering and the potential oversight of relevant information. However, natural language-based anomaly detection addresses these challenges by automatically extracting significant features from textual data using NLP techniques. This approach captures complex patterns and dependencies, resulting in enhanced accuracy. Moreover, it offers interpretability through human-readable explanations of anomalies, providing valuable insights that can aid in troubleshooting and decision-making processes.

The literature review focuses on a paper (van der Aa et al., 2021) proposing a natural language-based approach for detecting semantic execution anomalies in event logs. The paper utilizes several methods, including automatic extraction of business objects and actions from textual labels, comparison against a process-independent knowledge base, detection of semantically inconsistent execution patterns, and identification of violations related to order, exclusion, and co-occurrence. The proposed approach is evaluated using real-world and synthetic event logs, and a comparison is made with existing frequency-based techniques. The limitations of the approach are discussed, such as reliance on accurate natural language labels, the need for a populated knowledge base, limited evaluation scope, and complementarity to frequency-based techniques.

The paper recommends expanding and refining the proposed approach by incorporating additional knowledge sources and improving anomaly detection algorithms. The applicability of the approach to different scenarios, such as event log privatization and uncertainty in event data, is also suggested for exploration. The impact of noise on the approach's accuracy, the use of machine learning techniques to enhance accuracy, and further evaluations on diverse real-world event logs are highlighted as important areas for future investigation. By addressing these aspects, the field of process mining can advance, and anomaly detection techniques can become more accurate and applicable in various domains.

2.2.8 Pre-trained Transformer BERT Model - Anomaly Detection

Transformer-based models have several advantages over traditional NLP-based anomaly detection methods. They excel at capturing intricate patterns and dependencies in data, making them effective for large-scale datasets. Transformers utilize attention mechanisms to better understand context and identify context-specific anomalies. They are well-suited for analyzing sequential or temporal data, as they can capture temporal dependencies and sequential patterns. Additionally, transformers benefit from transfer learning, leveraging pre-trained representations for better generalization. They offer scalability and parallel processing capabilities, making them suitable for real-time or streaming data. Overall, transformers provide advanced capabilities for accurate and efficient anomaly detection.

In this literature review, the focus is on a paper proposing a novel method called BERT-Log (Guo et al., 2021; Chen and Liao, 2022a) for anomaly detection in system logs. The approach involves utilizing a pre-trained language model, specifically BERT, to learn the semantic representation of both normal and anomalous logs. To detect anomalies, the BERT model is fine-tuned using a fully connected neural network. The paper highlights two future research directions: reducing model training time to enhance real-time log processing capabilities and developing an approach for directly classifying anomalous logs based on event templates.

The proposed methodology involves treating the log sequence as a natural language sequence and leveraging the semantic information captured by the BERT model. By incorporating context and position, the BERT-Log method achieves outstanding performance on datasets such as HDFS and BGL. On the HDFS dataset, it achieves an impressive F1-score of 99.3%, showcasing its effectiveness. Additionally, on the BGL dataset, the proposed log feature extractor, which involves sliding windows with node ID, window size, and step size, demonstrates superior performance compared to LogRobust (Wang et al., 2022) and HitAnomaly (Huang et al., 2020a). In another paper we reviewed how a pre-trained BERT model can be fine tuned with just one additional output layer to create state of the art models which can be used for various range of tasks. Overall, the paper introduces the BERT-Log method as an innovative approach for anomaly detection in system logs. It leverages pre-trained language models and fine-tuning techniques to achieve exceptional performance on different datasets. The future research directions mentioned in the paper further emphasize the need for improving the model's efficiency and exploring alternative approaches for classifying anomalous logs.

Following the same trail in another literature review explores the Prog-BERT-LSTM (Shao et al., 2022) method for log anomaly detection, highlighting its superiority over the BERT-based model (LogBERT) in detecting system faults in log text data. The Prog-BERT-LSTM approach combines BERT for text vectorization and LSTM for sequence feature learning, using a progressive masking strategy to aggregate the text semantic and sequence feature vectors. Results from three public log datasets demonstrate the method's accurate and swift detection of anomalies compared to LogBERT.

The paper emphasizes the importance of log-based anomaly detection in large-scale computer systems. Traditional methods based on feature extraction and classical algorithms face limitations, especially with fixed window sizes for anomaly detection. In response, the paper proposes Prog-BERT-LSTM, an innovative approach that leverages BERT and LSTM's capabilities, effectively identifying system faults from log data.

Another literature review (Devlin et al., n.d.) examines a research paper proposing a framework for anomaly detection in log data using pre-trained language models. The authors introduce a plug-and-play approach that leverages pre-trained language models to obtain semantically aware embeddings for log events. By utilizing a Bi-LSTM (Devlin et al., n.d.) neural network, they effectively exploit contextual properties of log messages for enhanced anomaly detection. The method exhibits robustness against alterations in log messages, which commonly occur due to software updates and system deployments. The experimental evaluation on a cloud dataset, using various language models such as BERT (Devlin et al., n.d.), GPT (Devlin et al., n.d.) demonstrates the framework's high performance and resilience to semantic changes. The research highlights the importance of accurate anomaly detection in complex computer systems, emphasizing the need for timely identification to ensure system reliability, security, and operational efficiency. The proposed framework's success opens up new possibilities for future research in log-based anomaly detection using language models and supports the adoption of AI-driven solutions for automated operational tasks.

The review underscores future research directions, including exploring alternative deep learning models and enhancing the Prog-BERT-LSTM method's efficiency. This research opens opportunities for advancing log anomaly detection using deep learning techniques and optimizing detection methodologies. As a researcher, this review highlights the potential impact of Prog-BERT-LSTM in the log-based anomaly detection field and identifies promising avenues for future investigations

2.2.9 Hierarchical Transformer based Anomaly Detection

While doing the research on BERT-Log, we came across a research paper that introduces HitAnomaly, an innovative log-based anomaly detection model. HitAnomaly (Huang et al., 2020a) utilizes a hierarchical transformer structure to effectively model both log template sequences and parameter values. This approach outperforms existing methods in log-based anomaly detection, showcasing its potential in accurately detecting anomalies within log data. The paper presents compelling experimental results from evaluations conducted on three diverse log datasets, providing strong evidence of the model's effectiveness.

In addressing the challenge of log-based anomaly detection, the authors of the paper propose a hierarchical transformer structure as the foundation of HitAnomaly. This structure enables the model to capture the hierarchical relationships and dependencies among log templates and their associated parameter values, allowing for comprehensive and precise anomaly detection. By leveraging the power of transformers, HitAnomaly demonstrates superior performance compared to existing approaches, signifying its potential in enhancing the accuracy and reliability of anomaly detection systems.

To evaluate the proposed method, the researchers conducted extensive experiments on three distinct log datasets, covering various domains and log entry types. The experimental results clearly indicate that HitAnomaly outperforms existing methods, achieving higher accuracy in detecting anomalies within log sequences. These findings have significant implications for log-based anomaly detection, as they demonstrate the effectiveness of utilizing a hierarchical transformer structure. The success of HitAnomaly highlights the potential for improving anomaly detection in domains where log data plays a crucial role.

2.2.10 Anomaly Detection using Contrastive Learning

During our literature review, we came across a paper (Wang et al., 2022) proposing CL2MLog, a robust log anomaly detection method based on contrastive learning and multi-scale MASS. This method aims to improve accuracy and robustness in log-based anomaly detection. The proposed approach involves log parsing, robust semantic extraction using contrastive learning, log anomaly detection based on MSMASS, and contrast experiments with four baseline methods.

The paper demonstrates that CL2MLog outperforms the baseline methods in terms of accuracy and robustness. Experimental evaluations on stable and unstable log datasets provide evidence of the method's superiority. However, there are some limitations to consider. Firstly, the evaluation is conducted on a limited number of datasets, which may not fully represent diverse log data encountered in real-world scenarios. Further evaluation on more diverse datasets is necessary to validate the method's effectiveness. Secondly, the computational resource requirements of CL2MLog may hinder its practical applicability in real-world applications. Additionally, the paper lacks a detailed analysis of the interpretability of the proposed method, which is crucial for understanding detected anomalies. It is also important to acknowledge that the assumption of correctly pre-processed and parsed log data may not always hold true in real-world scenarios. Finally, the paper would benefit from comparing CL2MLog with state-of-the-art methods in other domains to gain more insights into its effectiveness.

In conclusion, while the CL2MLog method shows promise in log-based anomaly detection, further research is needed to address the identified limitations and validate its effectiveness in real-world scenarios.

2.2.11 Pre-Trained Transformer Models: RoBERTa, ALBERT, DistilBERT & XLNet

While conducting our Literature Review we came across alternative transformers models that can be used in anomaly detection and we can leverage the specific advantages of each model which will include reduced model size, training time , efficient resource utilization and enhanced understanding of contextual dependencies .

RoBERTa, (Naudé et al., 2022) ALBERT, (Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020) DistilBERT (Jiao et al., 2019; Sanh et al., 2019b), and XLNet (Naudé et al., 2022) are alternative models from the transformer family that can be employed for anomaly detection, offering distinct advantages compared to BERT.

RoBERTa, an optimized variant of BERT, addresses its limitations by employing larger batch sizes and training on more data, resulting in improved performance. By leveraging RoBERTa, anomaly detection tasks can benefit from its refined training approach.

ALBERT, on the other hand, focuses on reducing model size and training time while maintaining or enhancing performance. It achieves this through parameter-reduction techniques, making it suitable for larger-scale anomaly detection tasks that require efficient resource utilization.

DistilBERT, a distilled version of BERT, utilizes knowledge distillation to create a smaller and faster model without compromising performance significantly. Its advantage lies in reducing computational requirements, making it ideal for resource-constrained environments or scenarios where faster inference times are desired.

XLNet introduces an autoregressive approach, considering both preceding and following context to predict words in a sequence. This bidirectional nature enhances the understanding of context and dependencies, making it valuable for anomaly detection tasks that rely on capturing complex patterns and dependencies.

2.2.12 Evaluation and Performance Metrics in Anomaly Detection

In this literature review (Kim et al., 2022), we investigate the evaluation and performance metrics commonly used in anomaly detection. The evaluation metrics play a crucial role in assessing the effectiveness of anomaly detection algorithms. We specifically focus on the metrics of accuracy, precision, recall, F1-score, Receiver Operating Characteristic (ROC) curve, and Area Under the Curve (AUC).

Metrics and Analysis:

Accuracy:

Accuracy is a widely used metric that measures the overall correctness of an anomaly detection model. It calculates the ratio of correctly classified instances to the total number of instances. While accuracy provides a general evaluation, it can be misleading in the presence of class imbalance.

Precision and Recall:

Precision and recall are metrics that focus on the performance of anomaly detection in identifying true anomalies. Precision measures the proportion of correctly detected anomalies out of all instances classified as anomalies, while recall quantifies the proportion of correctly detected anomalies out of all actual anomalies. Both metrics have their strengths and limitations, as precision does not consider false negatives, and recall does not account for false positives.

F1-Score:

The F1-score combines precision and recall into a single metric by calculating their harmonic mean. It provides a balanced measure of the model's performance. However, it does not provide insights into the specifics of false positives and false negatives.

ROC Curve:

The ROC curve is a graphical representation of the true positive rate against the false positive rate at different classification thresholds. It helps determine the optimal threshold for anomaly detection. However, the ROC curve does not consider class distribution or the consequences of false positives and false negatives.

AUC (Area Under the Curve):

The AUC summarizes the performance of the model using the ROC curve. It provides an aggregate measure of the model's ability to distinguish between normal and anomalous instances. However, it may not capture the specifics of false positives and false negatives

2.2.13 Fine-Tuning Pretrained Language Models

This literature review (Dodge et al., 2020) delves into the subject of fine-tuning pretrained contextual word embedding models for supervised downstream tasks in natural language processing (NLP). The review is centred around a specific research paper that conducts experiments on the BERT model, exploring the impact of varying weight initialization and data order on performance variability.

The study investigates the sources of randomness that influence fine-tuning performance and reveals that both weight initialization and data order significantly contribute to the variation in out-of-sample performance. Some weight initialization approaches consistently yield favourable results across diverse tasks. Particularly noteworthy is the observation that on small datasets, numerous fine-tuning trials exhibit divergent behaviour during training. To address this, the paper proposes early stopping strategies to optimize the training process and improve overall efficiency.

In a commendable move to foster further research, the authors release all experimental data, encompassing training and validation scores for an extensive 2,100 trials. This generous data sharing serves as a valuable resource for the NLP community, encouraging collaboration and facilitating in-depth analysis of fine-tuning dynamics.

In summary, this research offers valuable insights into the significance of weight initialization and data order in fine-tuning pretrained contextual word embeddings for NLP tasks. The findings underscore the importance of achieving robust model performance, and the suggested early stopping strategies enhance training efficiency. The open data release provides an opportunity for researchers to explore and build upon the study's findings, ultimately advancing the field of NLP. This review highlights the relevance of investigating factors that influence fine-tuning outcomes and emphasizes the adoption of best practices to optimize model performance

2.2.14 Machine Learning Optimization and Hyperparameter Tuning

This review(Moya, 2021) centres around a research paper that explores the optimization of machine learning models by tuning hyperparameters. The paper examines different techniques for hyperparameter tuning in deep learning models, such as grid search, random forest, and Bayesian optimization. The primary goal of the study is to assess and analyse the effectiveness of these methods using a real-world synthetic polymer dataset that encompasses varying parameters and tuning strategies. Additionally, the paper delves into the four key components of hyperparameter optimization: an estimator, a search space, an optimization method, and a performance metric for comparing different hyperparameter configurations.

Regarding the methodology and findings, the paper investigates multiple approaches to hyperparameter tuning, specifically grid search, random forest, and Bayesian optimization. Grid search entails an exhaustive search across predefined ranges of hyperparameters, while random forest employs an ensemble of decision trees to optimize hyperparameters. Bayesian optimization utilizes a probabilistic model to guide the search for optimal hyperparameter values. By conducting empirical evaluations on the synthetic polymer dataset, the paper provides insights into the strengths and limitations of each method.

In conclusion, the paper highlights the critical role of hyperparameter tuning in achieving optimal machine learning parameters. It discusses the advantages and disadvantages of various hyperparameter tuning methods and presents the outcomes of implementing these methods on different machine learning models, measuring their Mean Squared Error (MSE) values. The paper suggests that randomly selected subsets effectively optimize all types of hyperparameters and are representative of the given dataset. It underscores the significance of automating the hyperparameter optimization process due to the computational expenses associated with manual tuning.

2.2.15 Challenges and Limitations in Anomaly Detection

Anomaly detection is a critical task in various domains such as cybersecurity, system monitoring, and fraud detection. While anomaly detection algorithms have shown promising results, they still face several challenges and limitations. This literature review examines three some challenges encountered in anomaly detection: imbalanced datasets and class imbalance, generalization to new and unseen anomalies, and computational complexity and scalability.

Imbalanced Datasets and Class Imbalance:

One of the primary challenges in anomaly detection is dealing with imbalanced datasets, where the number of normal instances outweighs the number of anomalous instances. This class imbalance introduces bias and affects the performance of anomaly detection algorithms. Traditional machine learning algorithms tend to prioritize the majority class, resulting in low sensitivity towards the minority class. To address this challenge, researchers have proposed various techniques, including oversampling, under-sampling, and cost-sensitive learning, to rebalance the dataset and improve the detection of anomalies in the minority class.

Generalization to New and Unseen Anomalies:

Anomaly detection algorithms are typically trained on historical data that contain known anomalies. However, in real-world scenarios, new and unseen anomalies may emerge over time. Traditional methods struggle to generalize to these novel anomalies as they were not encountered during the training phase. The ability to detect unknown anomalies is crucial for maintaining system security and integrity. To tackle this limitation, researchers have explored techniques such as transfer learning. These approaches aim to enhance the generalization capabilities of anomaly detection algorithms, enabling them to adapt to new and unseen anomalies and improve overall detection performance.

Computational Complexity and Scalability:

The increasing volume and complexity of data pose challenges in terms of computational complexity and scalability for anomaly detection algorithms. Traditional approaches may struggle to handle large-scale datasets and complex computations required for effective anomaly detection. Additionally, real-time anomaly detection in dynamic systems demands low-latency and efficient algorithms.

To overcome these challenges, researchers have proposed scalable algorithms, distributed computing frameworks, and parallel processing techniques. These advancements enable efficient anomaly detection in large-scale and dynamic environments, facilitating real-time monitoring and detection.

Complexity of Unstructured Log Data:

Anomaly detection techniques face difficulties due to the lack of a predefined structure in log data. The absence of consistent formatting or irregular log formats makes it challenging to extract meaningful information. Algorithms need to be adaptable to diverse log formats and effectively extract relevant features.

Handling High Volume of Log Data:

Anomaly detection algorithms encounter the challenge of processing and analysing the massive amounts of log data generated by systems and applications. The computational complexity and time required for analysis increase significantly with the scale of log data. Scalable and efficient algorithms are necessary to handle the data deluge while maintaining real-time or near-real-time performance

Concept Drift:

Anomaly detection algorithms must address concept drift, where the underlying patterns and characteristics of data change over time. The behaviour of normal and anomalous instances may evolve or shift, rendering models trained on historical data less effective. Continuous adaptation to changing patterns and regular model updates are crucial to ensure accurate anomaly detection.

Noise Problems:

Noise in log data refers to irrelevant or misleading information that can introduce errors or false positives in the anomaly detection process. Noise can stem from system glitches, measurement errors, or data corruption, making it challenging to identify genuine anomalies. Robust algorithms and noise filtering techniques are required to mitigate the impact of noise and improve detection accuracy.

2.3 Summary

The literature review encompasses a comprehensive investigation of log anomaly detection, exploring various research papers and methods utilized in this domain. Log anomaly detection is crucial for fortifying cybersecurity, network monitoring, troubleshooting, and predictive maintenance. The review identifies a wide array of approaches, methodologies, algorithms, and evaluation metrics utilized in this field. It also addresses the challenges and limitations faced by anomaly detection techniques.

In the realm of traditional methods, rule-based and statistical-based approaches are commonly used. Rule-based systems involve generating association rules based on historical data and then monitoring real-time data for deviations. Statistical-based methods, like the Local Outlier Factor (LOF) algorithm, find outliers in datasets based on statistical variables. However, both approaches face difficulties in detecting context-specific anomalies and handling class imbalance.

To address these limitations, machine learning-based approaches have emerged, classified into six groups: classification models, ensemble models, optimization models, rule-based systems, clustering, and regression. Support vector machines (SVM) are a popular choice, but various models exhibit specific limitations.

Deep learning techniques, especially those using LSTM, show promise in log anomaly detection, outperforming traditional methods in capturing temporal dependencies and handling unstructured log data. However, they face challenges with the availability of labeled data and the interpretability of results.

Pre-trained transformer models like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), roBERTa (Naudé et al., 2022), ALBERT (Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020), and DistilBERT (Jiao et al., 2019; Sanh et al., 2019b) offer advantages in terms of performance, model size, and training time for anomaly detection tasks.

Evaluation metrics such as accuracy, precision, recall, F1-score, ROC curve, and AUC play a crucial role in assessing anomaly detection algorithms. However, each metric has its strengths and limitations, emphasizing the importance of considering multiple metrics.

Fine-tuning pretrained language models for NLP tasks is essential for achieving optimal performance. Techniques like early stopping strategies can improve efficiency and robustness.

Hyperparameter tuning in machine learning models is critical for achieving better performance. Grid search, random forest, and Bayesian optimization are commonly used methods, each with its strengths and limitations.

Log anomaly detection faces challenges such as imbalanced datasets, generalization to new anomalies, computational complexity, unstructured log data, high volume of log data, concept drift, and noise problems. Various techniques and approaches are being explored to overcome these challenges.

In conclusion, the literature review highlights the significance of automated log anomaly detection in enhancing system reliability, security, and performance. By leveraging different machine learning, deep learning, Transformer techniques researchers aim to improve the accuracy and efficiency of anomaly detection, paving the way for more effective cybersecurity and system monitoring solutions.

CHAPTER 3

RESEARCH METHODOLOGY

The sequential flow of steps for model building will be provided in this section.

3.1 Introduction

System logs are critical in monitoring and troubleshooting computer systems, but their unstructured nature and vast amount of data make them challenging to analyze. Machine learning algorithms have shown promise in detecting anomalies, but their robustness issues can lead to poor performance and inaccurate results. To address these challenges, this study aims to propose a transformer method for log classification that addresses the limitations of existing log anomaly detection methods and the robustness issues by updating log message templates. The proposed method also includes a more efficient and automated log parsing method to enable faster and more effective troubleshooting and security monitoring. The effectiveness and generalizability of the proposed method will be evaluated by testing it on various log datasets and comparing its performance with existing methods.

This study focuses on using various data pre-processing techniques to prepare system log data for anomaly detection. The data preparation process involves exploratory data analysis, data cleaning, and outlier analysis, along with feature engineering techniques like converting categorical variables into numerical values and standardization. After preparing the data, the study explores different machine learning models for log classification, such as developing a log sequence model and comparing transformer-based architectures like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022) and RoBERTa (Naudé et al., 2022). The best-performing model will be fine-tuned using a fully connected neural network to classify normal and anomalous log sequences. Finally, the performance of the models will be evaluated on a test set using various metrics like Confusion Matrix, Accuracy, Precision, Recall, F1-score, and AUC-ROC. The goal is to propose a comprehensive and effective framework for log classification that can overcome the limitations of existing methods through extensive experimentation and evaluation.

3.2 Research Approach

3.2.1 Data Selection

This thesis will focus on the BGL(Oliner and Stearley, 2007) dataset is a publicly available collection of system logs that were gathered from a BlueGene/L(Oliner and Stearley, 2007) supercomputer located at the Lawrence Livermore National Labs in Livermore, California (Oliner and Stearley, 2007). The supercomputer had 131,072 processors and 32,768GB memory. The dataset includes both alert and non-alert messages, which are distinguished by alert category tags.

The first column of the log file indicates whether a message is an alert or non-alert message, with "-" representing non-alert messages. This labelling information makes the dataset useful for research on alert detection and prediction. The BGL dataset has been utilized in various studies related to log parsing, anomaly detection, and failure prediction.

Table 3.1 : Data Description - Alert Categories , Counts and Sample Message Body

The system names are accompanied by the total number of alerts. The alert category is indicated by "Cat.", and there are three types of categories: H (Hardware), S (Software), and I (Indeterminate). Alerts categorized as Indeterminate may originate from either hardware or software or have an unknown cause**.**

Alert Type/Cat.	**Raw**	**Example Message Body (Anonymized)**
BG/L	3,48,460	-
H / KERNDTLB	1,52,734	data TLB error interrupt
H / KERNSTOR	63,491	data storage interrupt
S / APPSEV	49,651	ciod: Error reading message prefix after LOGIN MESSAGE on CioStream [...]
S / KERNMNTF	31,531	Lustre mount FAILED : bglio11 : block id : location
S / KERNTERM	23,338	rts: kernel terminated for reason 1004rts: bad message header: [...]
S / KERNREC	6145	Error receiving packet on tree network, expecting type 57 instead of [...]
S / APPREAD	5983	ciod: failed to read message prefix on control stream [...]
S / KERNRTSP	3983	rts panic! - stopping execution
S / APPRES	2370	ciod: Error reading message prefix after LOAD MESSAGE on CioStream [...]
I / APPUNAV	2048	ciod: Error creating node map from file [...]
I / Others	7186	machine check interrupt

Table 3.2 : Data Description - Number of Messages Per Facility

Facility	No. Of Messages
KERNEL	4324651
APP	228536
DISCOVERY	97172
MMCS	88930
HARDWARE	5148
MONITOR	1681
LINKCARD	1170
FATAL	306
CMCS	211
BGLMASTER	145
SERV_NET	3

Table 3.3 - No of Messages per Severity Level

Severity	No. Of Messages
INFO	3735813
FATAL	855195
ERROR	112355
WARNING	23357
SEVERE	19213
FAILURE	1714

3.2.2 Data Pre-processing

Exploratory Data Analysis (EDA) involves using visual and quantitative methods to analyze and summarize complex data sets to uncover underlying patterns, trends, and relationships.

- **Data Cleaning**: This involves identifying and correcting or removing incomplete, inaccurate, irrelevant, or inconsistent data from the dataset.

    - Preprocess log data using a log parser to extract log keys.
    - Define a log sequence as an ordered sequence of log keys.
    - Label the log sequences as normal or anomalous

- **Duplicate Records and Missing Value Analysis**: Identifying and handling duplicate records and missing values to ensure that the dataset is accurate and complete.

- **Skewness Analysis**: This involves detecting skewed data points that may affect the performance of the model.

- **Univariate Analysis - Data Distribution**: Examining the distribution of the dataset, including measures of central tendency, and spread, to gain insights into the data and identify patterns.

- **Bi-variate Analysis with Respect to Target Variable**: Analyzing the relationship between the features and the target variable to identify the most important variables for the model.

- **Multivariate Analysis Using Correlation Metrics to Check Multicollinearity**: This involves analyzing the correlation between features to identify potential multicollinearity issues and to select the most important variables for the model.

- **Feature Engineering**: Creating new features from existing ones to enhance the predictive power of the model.

- **Convert Categorical Variable to Numerical Value**: Converting categorical variables into numerical values to make them compatible with the algorithms used for the analysis.

- **Standardization and Scaling**: This involves transforming the data to a standard scale to ensure that all variables are treated equally, and the model performs optimally.

- **Sentence Tokenization:** Split text into sentences to identify boundaries and treat them as separate units.

- **Word Tokenization:** Split sentences into words or tokens for further analysis and manipulation.

- **Removing Stop Words:** Eliminate common, insignificant words to focus on meaningful content during summarization.

- **Text Normalization:** Clean text by removing special characters, punctuation, and numbers, while ensuring consistency in case, stemming, and lemmatization.

- **Handling Outliers:** Address outliers, such as extremely long or short sentences, which can impact summary quality.

3.2.3 Feature Extraction

Convert preprocessed text into numerical representations (e.g., word embeddings or TF-IDF(Bhanage et al., 2021a) vectors) for machine learning models to process and understand the data

- The Bag-of-Words (BoW) (Bhanage et al., 2021a) Model treats a text as a collection of words, independent of grammar and word order, and assigns each word as a separate feature.

- TF-IDF(Bhanage et al., 2021a) assigns greater importance to terms that are common within a document but rare across other documents, emphasizing the significance of unique terms in distinguishing documents.

- Word2Vec(Chen and Liao, 2022a) is a widely used method in natural language processing, which converts words into dense vectors to capture semantic associations and similarities between words. These word embeddings are learned through a neural network-based language modeling technique, facilitating effective word representations for diverse NLP applications.

3.2.4 Model Building

In the process of building a model, the first step involves partitioning the data into training, testing, and validation sets. This partitioning is crucial as it allows for the evaluation of the model's performance and its ability to generalize to new data. The training set is utilized to train the model on a substantial portion of the data, enabling it to capture the underlying patterns and relationships present. The testing set is then employed to assess the model's performance on unseen data, providing an estimation of its accuracy and effectiveness. Lastly, the validation set is used to fine-tune the model's hyperparameters and make necessary adjustments prior to finalizing the model. By dividing the data into these distinct sets, the model undergoes training, evaluation, and adjustment on separate data samples, ensuring a comprehensive assessment of its performance while mitigating the risk of overfitting.

Below models will be explored and based on the study we will propose the best performing model

Log Sequence Model:

A log sequence model, like LSTM(Zhao et al., n.d.; Malaiya et al., 2019) (Long Short-Term Memory), is an RNN(Zhao et al., n.d.) variant specialized in capturing sequential log data's temporal patterns and dependencies. We will develop a log sequence model (e.g., an LSTM(Zhao et al., n.d.) or Transformer model) to learn the sequence patterns in the log data and detect anomalies. Train the log sequence model on the labelled log sequences and evaluate its performance using cross-validation. Select the best-performing log sequence model based on its classification accuracy.

Transformer-based Log Classification

Transformer-based log classification models leverage transformer architectures to process sequential data and capture long-range dependencies, enabling their high effectiveness in log classification tasks. We will Investigate and compare different transformer-based architectures such as BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), RoBERTa(Naudé et al., 2022), ALBERT ,(Lan et al., 2019; Choi et al., 2020) DistilBERT(Jiao et al., 2019; Sanh et al., 2019b) and XLNet(Naudé et al., 2022) etc. for log classification.

- **BERT** (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022) , which extensively pre-trains on large amounts of unlabelled text data and exhibits exceptional performance across diverse natural language processing tasks, including log classification.

- **RoBERTa** (Naudé et al., 2022) , an optimized variant of BERT, enhances performance by adjusting training methods and hyperparameters, building upon the accomplishments of BERT and refining its capabilities.

- **ALBERT** (Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020), on the other hand, provides a compact alternative to BERT, maintaining high performance while reducing the model size and computational requirements.

- **DistilBERT** (Jiao et al., 2019; Sanh et al., 2019), as the name suggests, distils the knowledge of BERT into a smaller model that achieves comparable performance with significant reductions in size and training time, making it efficient for log classification and other text classification tasks.

- **XLNet** (Naudé et al., 2022), a transformer-based model with a unique permutation-based training approach, captures bidirectional dependencies without relying on masking, achieving strong performance in log classification and other text classification tasks by effectively capturing intricate relationships within log data.

As a part of study will train each transformer model on the log data and evaluate their performance using cross-validation (k-fold cross validation) and will the best-performing transformer model based on its classification accuracy.

Using a Pre-trained Language Model

Fine-tune the selected pre-trained language model (e.g., BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022) , roBERTa (Naudé et al., 2022), XLNET (Naudé et al., 2022)) on the log data to learn the semantic representation of normal and anomalous logs. Evaluate the fine-tuned language model on the validation set to select the best-performing model.

Fine-tuning the BERT Model

Use the best-performing pre-trained language model (e.g., BERT (Dodge et al., 2020)) to detect abnormal logs.Fine-tune the BERT(Bhanage et al., 2021b) model using a fully connected neural network to classify normal and anomalous log sequences. Evaluate the fine-tuned BERT(Huang et al., 2020b) model on the validation set to select the best-performing model.

Fine-tuning the Log Sequence Model

Use the best-performing LSTM(Zhao et al., n.d.; Wang et al., 2022a) model to detect abnormal logs. Fine-tune the LSTM model using a fully connected neural network to classify normal and anomalous log sequences. Evaluate the fine-tuned LSTM(Zhao et al., n.d.) model on the validation set to select the best-performing model.

Hybrid Model:

In this phase we will use the hybrid approach that combines the outputs of LSTM(Zhao et al., n.d.) and Transformer-based models (Devlin et al., n.d.; Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022). This approach aims to capitalize the complementary strength of both architectures, enhancing the overall anomaly detection performance.

3.2.5 Model Controls – Hyperparameter Tuning

Model control hyperparameter tuning details for the transformer-based (Devlin et al., n.d.; Dodge et al., 2020)log classification are as follows

- Learning Rate will help to optimize the step size for parameter updates during training.
- Dropout Rate can we use Fine-tune the probability of dropping out units for regularization.
- Batch Size will determine the number of training examples processed in each iteration.
- Number of Layers can find the right balance of layers for capturing complex patterns.
- Hidden Dimension Size can tune the dimensionality of hidden states for better representation.
- Attention Mechanism will explore hyperparameters related to attention heads and dropout.
- Regularization can help to prevent overfitting and improve generalization.
- Optimization techniques may help to choose the optimal optimization algorithm and related hyperparameters

In addition to Learning Rate, Dropout Rate, No of Layers, Batch Size, Regularization and Optimization Model control hyperparameter tuning details for the log-sequence based LSTM (Zhao et al., n.d.) model are as follows

- Number of LSTM Units, Determine the LSTM unit count per layer for capturing temporal dependencies effectively.
- Sequence Length, Set the input log sequence length for LSTM training.
- Initialization Method, Explore different weight initialization techniques (e.g., uniform, normal distribution) for LSTM weights.

We can use below algorithms find the best combinations of hyperparameters for tuning

Grid Search: Employ grid search to systematically assess various hyperparameter combinations for transformers and LSTM (Zhao et al., n.d.) models in anomaly detection.

Random Search: Utilize random search to randomly select hyperparameter configurations for transformers and LSTM (Zhao et al., n.d.) models, facilitating a wider exploration of the hyperparameter space.

Bayesian Optimization: Apply Bayesian optimization techniques to effectively search and optimize hyperparameters for transformers and LSTM models, leveraging probabilistic models and previous evaluations to guide the search process efficiently

3.2.6 Model Evaluation

Compare the performance of the Transformers-based model, and Log Sequence based with other anomaly detection methods such as logistic regression, decision trees, and Naive Bayes.

We will use performance metrics such as Confusion Matrix, Accuracy, Precision, Recall, F1-score, and AUC-ROC(Bhanage et al., 2021a; Chen and Liao, 2022b) to compare the performance of different methods.

Confusion Matrix(Chen and Liao, 2022b): It is a representation of a 2x2 table of actual vs predicted class in binary classification. It includes four parameters: true positives, false positives, true negatives, and false negatives.

Accuracy(Chen and Liao, 2022b): This measures the proportion of correctly classified observations. It is the simplest evaluation metric and is calculated as the number of correct predictions divided by the total number of predictions.

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

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

Where :

- **TP (True Positives)** represents the instances correctly predicted as positive.
- **TN (True Negatives)** represents the instances correctly predicted as negative.
- **FP (False Positives)** accounts for actual negatives incorrectly classified as positives (Type I error).
- **FN (False Negatives)** indicates actual positives mistakenly classified as negatives (Type II error).

Precision, Recall, Sensitivity, and Specificity(Chen and Liao, 2022b): These are long established evaluation metrics based on business requirements. Sensitivity and recall are similar.

Precision Formula :

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

Where :

- **TP (True Positives)** represents the instances correctly predicted as positive.
- **FP (False Positives)** accounts for actual negatives incorrectly classified as positives (Type I error).

Recall or Sensitivity Formula :

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

- **TP (True Positives)** represents the instances correctly predicted as positive.
- **FN (False Negatives)** indicates actual positives mistakenly classified as negatives (Type II error).

Specificity Formula :

$$\mathbf{Specificity} = \frac{TN}{TN+FP}$$

Where:

- **TN (True Negatives)** represents the instances correctly predicted as negative.
- **FP (False Positives)** accounts for actual negatives incorrectly classified as positives (Type I error).

ROC Curve and AUC(Chen and Liao, 2022b): The ROC (Bhanage et al., 2021a; Chen and Liao, 2022b) curve is used to understand the strength of the model by evaluating its performance at all classification thresholds. The Area Under the Curve (AUC) (Bhanage et al., 2021a; Chen and Liao, 2022b) summarizes the overall performance of the classifier.

Best Threshold: Higher True Positive Rate and lower False Positive Rate(misclassifications) will lead to the best threshold.

Choosing Precision or Recall: Depending on the problem statement, the study will identify which measure is more important: high precision or high recall.

F1-Score(Chen and Liao, 2022b): The F1-score is the harmonic mean of precision and recall values for a classification problem where the requirement is to have the best precision and recall at the same time.

F1-Score Formula :

$$\boldsymbol{F1}_{\boldsymbol{Score}} = \frac{2 * Precision * Recall}{Precision + Recall}$$

Generalizability: Evaluating the generalizability of the proposed method by extending the evaluation to other log datasets. This step could involve cross--validation, transfer learning, and domain adaptation techniques.

3.3 Proposed Method

The proposed approach to detect anomalies in log data involves a hybrid architecture that synergizes the strengths of LSTM (Zhao et al., n.d.)and Transformer models like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), RoBERTa (Naudé et al., 2022) , ALBERT(Lan et al., 2019; Choi et al., 2020), XLNet (Naudé et al., 2022) with log parsing techniques like Drain (He et al., 2017; Fu et al., 2023) and IPLoM (Fu et al., 2023), utilized for preprocessing. This strategy empowers the system to effectively identify both sequential patterns and contextual information from log messages, thereby enhancing its anomaly detection capabilities.

Step 1: Exploratory Data Analysis (EDA):

Firstly, researchers conduct exploratory data analysis on the raw log data. EDA involves using visual and quantitative methods to analyse and summarize complex data sets, uncovering underlying patterns, trends, and relationships. This step provides insights into data distribution, identifies potential data quality issues, and guides decisions on data pre-processing and feature engineering.

Step 2: Data Preprocessing:

Next, the raw log data undergoes preprocessing to prepare it for further analysis. Data pre-processing includes various sub-steps, such as data cleaning, outlier detection, and handling missing values. Since log messages can contain irrelevant or inconsistent data impacting anomaly detection models' accuracy, preprocessing ensures the data is clean and consistent, enhancing subsequent analyses' quality.

Step 3: Data Cleaning:

Data cleaning is a critical aspect of the preprocessing phase. It involves identifying and correcting or removing incomplete, inaccurate, irrelevant, or inconsistent data from the dataset. Log messages undergo thorough examination, filtering out noisy or erroneous entries. This step ensures the log data's accuracy, rendering it ready for further processing.

Step 4: Log Parsing using Drain or IPLoM:

The initial step revolves around log parsing, where unstructured log data generated by systems or applications undergoes transformation into well-structured log sequences. To achieve this, the hybrid approach employs Drain (He et al., 2017; Fu et al., 2023) or IPLoM (Fu et al., 2023), as log parsers, extracting meaningful log keys from the raw log messages. These log keys contain vital information that aids in the identification of patterns and anomalies within the log data.

Step 5: Representing Log Sequences for LSTM:

After parsing, the log sequences act as input for the LSTM (Zhao et al., n.d.; Malaiya et al., 2019) model, which belongs to the family of recurrent neural networks specialized in learning sequential dependencies. By processing the log sequences, LSTM captures temporal patterns ingrained in the data. The LSTM output may be a sequence of hidden states or a final hidden state, summarizing the information from the log sequence.

Step 6: Contextual Log Key Representations with Transformer based Models:

Concurrently, the log sequences obtained from log parsing are fed into the Transformer models like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), roBERTa (Naudé et al., 2022), ALBERT (Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020), DistilBERT (Jiao et al., 2019; Sanh et al., 2019b) and XLNet (Naudé et al., 2022) that generates contextual embeddings for each input token. In this context, each log key serves as an input token, and Transformer models learns contextual representations for them. The output could be individual embeddings for each log key or a fixed-size embedding representing the entire log sequence.

Step 7 Integrating the Hybrid Model:

By combining the representations from LSTM and Transformer models, the hybrid model amalgamates their strengths to make the final prediction. This integration can occur through various means, such as concatenating the LSTM and Transformer representations, additional layers, or employing attention mechanisms to weigh their importance. As a result, the hybrid model learns to leverage both local sequential patterns from LSTM and global context from Transformer models like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022), roBERTa (Naudé et al., 2022), ALBERT (Lan et al., 2019; Sanh et al., 2019a; Choi et al., 2020), and DistilBERT (Jiao et al., 2019; Sanh et al., 2019b), XLNet (Naudé et al., 2022) enhancing its ability to make accurate predictions.

Step 8: Anomaly Detection:

The ultimate outcome of the hybrid model is a binary classification for each log sequence, indicating whether it is normal or anomalous. Through training, the model becomes adept at discerning normal behaviour patterns from anomalies based on the representations learned by LSTM and Transformer Models like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022) representations and XLNet (Naudé et al., 2022).

Step 9: Feature Extraction:

Feature extraction is a crucial phase in anomaly detection, especially when dealing with intricate log data. In addition to utilizing LSTM (Zhao et al., n.d.) and Transformer models representations, we may extract additional domain-specific features from the log sequences. These features can encompass statistical measures, domain-specific keywords, or aggregate metrics derived from the log data. Feature extraction enhances the model's capacity to capture relevant information and improves anomaly detection accuracy.

Step 10: Overcoming Noise and Ensuring Robustness:

To address noise in log data, the hybrid architecture incorporates resilient preprocessing techniques. During log parsing, Drain (He et al., 2017; Fu et al., 2023) or IPLoM (Fu et al., 2023), are tailored to handle irregular log formats, extracting pertinent log keys while filtering out noisy or irrelevant data. Moreover, during training, the hybrid model learns to prioritize essential patterns while disregarding noise, leveraging local sequential information from LSTM and global context from Transformer Models like BERT (Devlin et al., n.d.; Huang et al., 2020a; Guo et al., 2021; Chen and Liao, 2022a; Shao et al., 2022) representations and XLNet (Naudé et al., 2022).. This combination effectively distinguishes genuine anomalies from noise.

Step 11: Real-time Processing Efficiency:

Efficient real-time anomaly detection demands timely processing to respond promptly to potential threats or issues. The hybrid architecture optimizes real-time processing by utilizing lightweight log parsing techniques like Drain (He et al., 2017; Fu et al., 2023) or IPLoM (Fu et al., 2023), efficiently transforming log sequences into structured data. The usage of well-optimized and efficient architectures enables rapid sequential and contextual analysis. Additionally, the hybrid model can leverage hardware accelerators and parallel processing for further real-time efficiency enhancement.

Step 12: Generalizability to New and Unseen Anomalies:

The hybrid architecture ensures generalizability by capitalizing on the strengths of LSTM and BERT model. LSTM's capability to capture temporal patterns makes it well-suited for detecting anomalies conforming to known patterns. In contrast, BERT's contextual embeddings enable the model to recognize novel and previously unseen anomalies, comprehending the semantics and context of log keys. Throughout the training process, the hybrid model encounters various anomalies and normal behaviour patterns, enhancing its ability to generalize to new and unseen anomalies during inference

Incorporating Drain (He et al., 2017; Fu et al., 2023) and IPLoM (Fu et al., 2023), during log parsing ensures robust preprocessing, effectively addressing noise and extracting relevant log keys. The hybrid architecture's efficiency enables real-time processing, ensuring timely responses to potential threats or issues. Additionally, the model's generalizability to new and unseen anomalies is enhanced by leveraging LSTM's capture of temporal patterns and BERT's contextual embeddings. As a result, the hybrid architecture becomes a powerful and reliable solution for anomaly detection across various domains, including cybersecurity, system monitoring, and fraud detection.

3.4 Expected Outcomes and Deliverables

Expected outcome from this research are as follows :

- Thorough analysis of log classification and anomaly detection, covering aspects such as real-time processing, robustness, noise handling, efficiency, and generalizability.
- Detailed and precise description of the techniques and models employed, accompanied by a well-structured evaluation methodology to gauge their efficacy.
- Comparative examination of various AI-based techniques to identify the most efficient model for anomaly detection.
- Creation and evaluation of a LSTM and Transformer based models to assess its performance and accuracy in detecting anomalies.

3.5 Proposed Flow Diagram

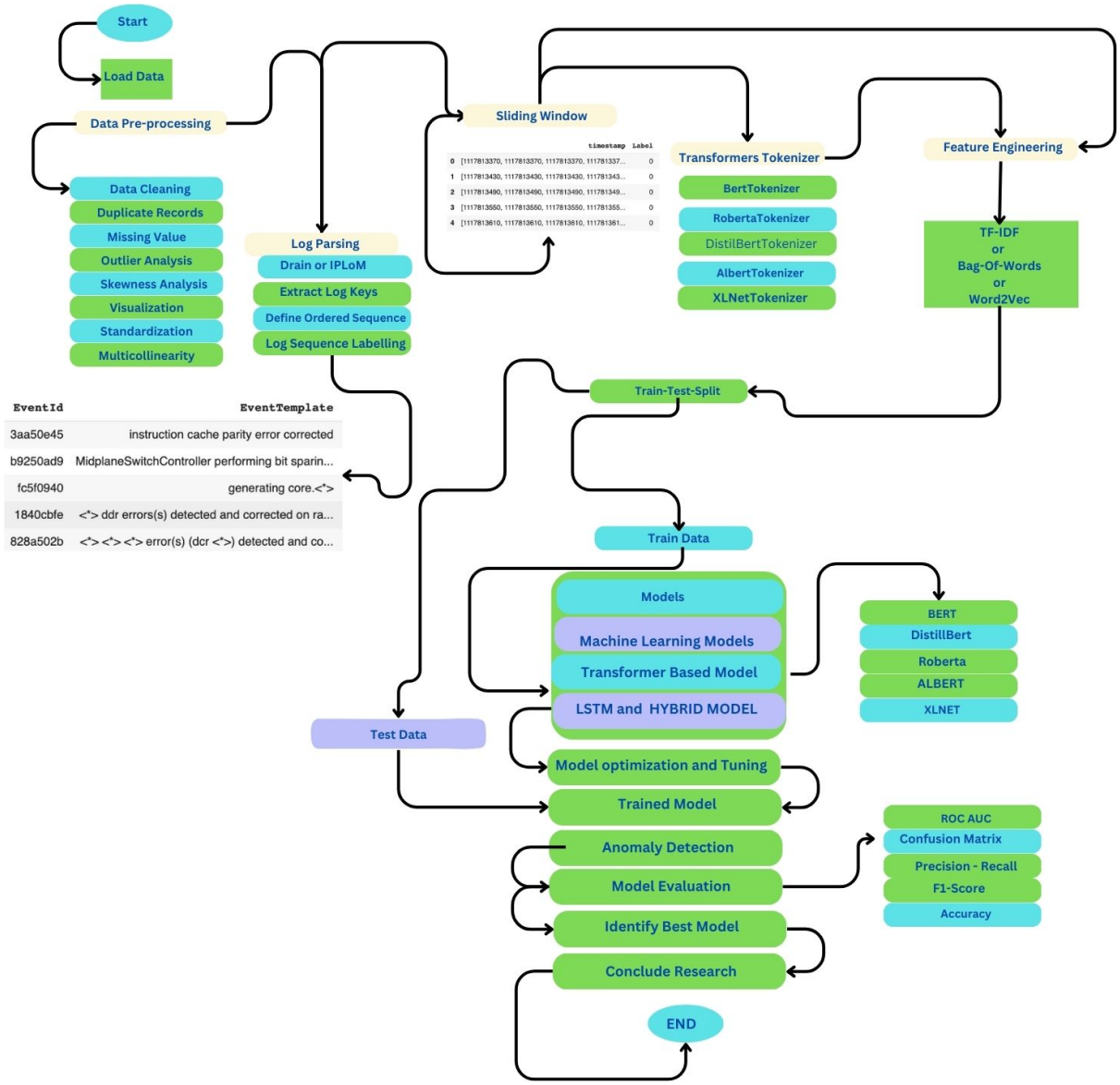


Figure 3.1 : Model Building Flow Chart

Summary:

The section 3.1 started with the introduction which aims to tackle anomaly detection challenges in system logs by proposing a transformer-based log classification method. It addresses robustness issues and updates log message templates for accurate results. An efficient log parsing approach enables faster troubleshooting and security monitoring. Evaluation on various log datasets and comparison with existing methods ensures effectiveness and generalizability. Under Section 3.2 we covers data selection, data pre-processing, exploring machine learning models, and model evaluation. Data cleaning, outlier analysis, and feature engineering prepare log data for analysis. Different models, including log sequence and transformer-based, are fine-tuned using a neural network for log classification. Model performance is evaluated using various metrics. In Section 3.3 we proposed the hybrid architecture combines LSTM (Zhao et al., n.d.) and BERT (Chen and Liao, 2022a)with Drain (He et al., 2017; Fu et al., 2023) and IPLoM (Fu et al., 2023), log parsing techniques. It overcomes noise, ensures real-time processing, and generalizes to new anomalies, providing an effective solution for anomaly detection in various domains. Section 3.4 will cover the expected outcomes and deliverables. Section 3.5 will contain proposed flow diagram.

CHAPTER 4

ANALYSIS

4.1 Introduction

This chapter uncovers the results derived from our systematic approach, as outlined in the previous Research Methodology chapter (Chapter 3), for conducting Log Anomaly Detection using the BGL dataset. Our primary focus was on identifying anomalies within the BGL dataset. The initial step involved acquiring a comprehensive comprehension of the dataset to facilitate the development of suitable models. In Section 4.2, we present a comprehensive overview of the BGL dataset, a compilation of system logs from the BlueGene/L supercomputer, outlining its distinctive characteristics as the foundation for our Log Anomaly Detection study. Moving on to Section 4.3, our exploration of the dataset begins, meticulously inspecting its contents and revealing crucial insights such as missing values, duplicates, memory usage, variable types, cardinality, correlations, and class imbalances. Section 4.4 signifies a critical phase in our study, where we prepare the data for anomaly detection by initializing paths and applying log parsing using the Drain Algorithm. This phase includes feature engineering and column transformations to enhance data quality. Additionally, in Section 4.4.1, we delve into the Drain Algorithm's application, focusing on log parsing and standardization. Section 4.4.2 elucidates how specific columns undergo transformations to facilitate subsequent analyses, while Section 4.4.3 introduces log sequencing using sliding windows to maintain data context and event sequence. Finally, in Section 4.5, we provide a comprehensive summary of our dataset description, exploratory analysis, data pre-processing, and feature engineering efforts, setting the stage for the subsequent phases of our Log Anomaly Detection study.

4.2 Dataset Description

The BGL dataset, compiled by Oliner and Stearley in 2007, constitutes a publicly accessible compilation of system logs collected from the BlueGene/L supercomputer positioned at the Lawrence Livermore National Labs in Livermore, California. This supercomputer was outfitted with 131,072 processors and 32,768GB memory. Within this dataset, both alert and non-alert messages are included, classified by distinct alert category tags.The initial column of the log file designates whether a message falls under the category of an alert or a non-alert message, with "-" signifying non-alert messages. This categorization facet enhances the dataset's suitability for studies in alert detection and forecasting. The BGL dataset has found utility across diverse investigations related to log parsing, anomaly detection, and failure prediction. The dataset comprises a staggering 4,747,963 rows, representing millions of entries with 708.8 MB log file size

4.3 Exploratory Data Analysis and Visualization

The dataset consists of a total of 10 variables, encompassing a substantial number of 4,747,963 observations. Remarkably, there are no missing values present in the dataset, rendering a missing cell percentage of 0.0%. The occurrence of duplicate rows is quite minimal,

constituting less than 0.1% of the entire dataset. In terms of memory consumption, the dataset occupies 3.2 GiB of space, with an average memory footprint of 716.5 bytes per record. The dataset variable types comprise 9 categorical variables and 1 numeric variable.. The dataset has a minimal number of duplicate rows (less than 0.1%). Certain columns like "Date," "Code1," "Time," "Code2," and "Content" exhibit high cardinality, meaning they have a large number of distinct values. Moreover, columns such as "Id," "Label," "Component1," "Component2," and "Level" demonstrate strong overall correlations with each other. Additionally, columns like "Label," "Component1," "Component2," and "Level" are characterized by significant class imbalance, indicating that certain values are much more prevalent than others. Lastly, the "Time" column displays a uniform distribution, signifying that its values are evenly spread out.

4.3.1 Missing Value

Based on the Analysis no missing value present in the dataset

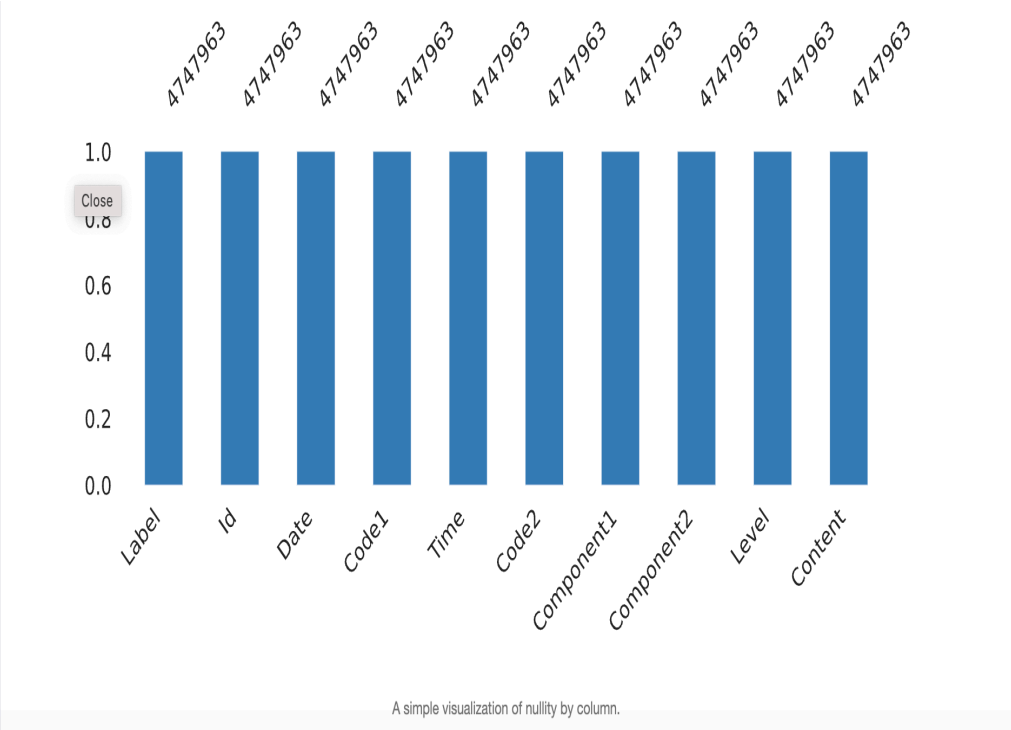


Figure 4.3.1 : Missing Values Visualization

4.3.2 Correlation Heatmap:

Correlation heatmaps find their primary utility in illustrating the associations between pairs of variables within a dataset.

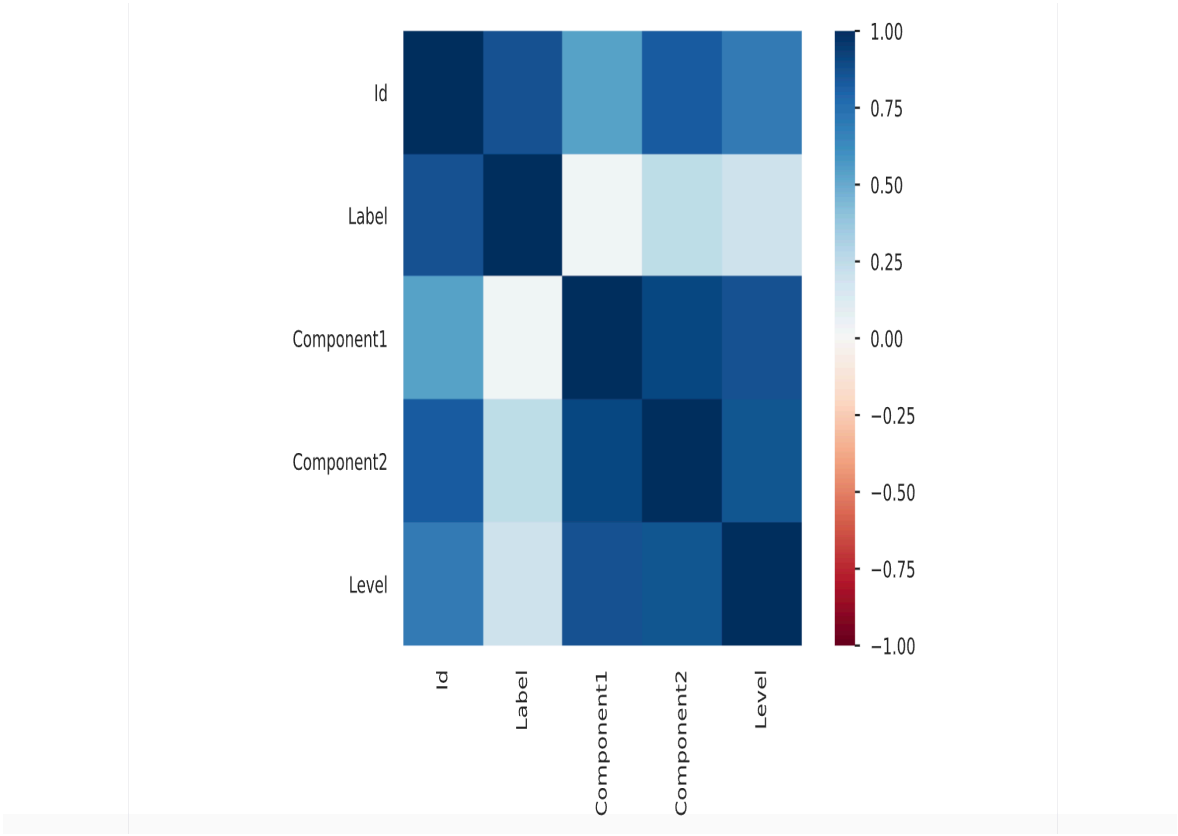


Figure 4.3.2 : Correlation Heatmap

4.3.3 Univariate Analysis

Label Column:

The "Label" column indicates different categories for log entries, with corresponding counts and percentage frequencies. Logs labeled as "-" constitute the majority, accounting for approximately 92.7% of the dataset, representing normal logs. Other categories include "KERNDTLB," "KERNSTOR," "APPSEV," and more, each with varying proportions indicating anomalies in the system. Below is the list of other categories

Value	Count	Frequency (%)
-	4399503	92.7%
KERNDTLB	152734	3.2%
KERNSTOR	63491	1.3%
APPSEV	49651	1.0%
KERNMNTF	31531	0.7%
KERNTERM	23338	0.5%
KERNREC	6145	0.1%
APPREAD	5983	0.1%
KERNRTSP	3983	0.1%
APPRES	2370	< 0.1%
Other values (32)	9234	0.2%

Figure 4.3.3.1 : Label Categories Count and Frequency

Date Column:

The "Date" column consists of categorical values indicating dates. There are a total of 215 distinct dates in the dataset, each making up less than 0.1% of the total. No missing values are present in this column. The memory size allocated for storing these dates is around 303.4 MiB. The dates span from 2005.06.14 to 2005.12.01, with varying frequencies. The length of each date is consistent at 10 characters. The characters used are from a limited set of 11 distinct characters. Overall, the "Date" column provides essential temporal information for the dataset.

Code1 Column:

The "Code1" column plays a crucial role in categorizing and distinguishing different codes within the dataset. The uniqueness of the codes is observed, with 96 unique values, contributing to less than 0.1% of the total.

Table 4.3.3.1 : Code1 Feature Data Sample

1st row	R02-M1-N0-C:J12-U11
2nd row	R02-M1-N0-C:J12-U11
3rd row	R02-M1-N0-C:J12-U11
4th row	R02-M1-N0-C:J12-U11
5th row	R02-M1-N0-C:J12-U11

Time Column :

The "Time" column is categorized as categorical and exhibits high cardinality with a uniform distribution. It encompasses over 4.7 million distinct values, accounting for over 99.9% of the total entries. Remarkably, there are no instances of missing data in this column. The storage space required for this information is approximately 375.8 MiB. Each timestamp incorporates both date and time details, and specific timestamps have unique counts. For example, the timestamp "2005-09-20-12.06.14.325292" is duplicated twice in the dataset. This multitude of unique timestamps emphasizes the temporal aspect of the dataset, with more than 99.9% of the values being distinct.

Table 4.3.3.2 : Time Feature Data Sample

1st row	2005-06-03-15.42.50.363779
2nd row	2005-06-03-15.42.50.527847
3rd row	2005-06-03-15.42.50.675872
4th row	2005-06-03-15.42.50.823719
5th row	2005-06-03-15.42.50.982731

Code 2 Column :

The "Code2" column exhibits distinct values totaling 69,258, which constitutes approximately 1.5% of the entire dataset. There are no missing entries in this column. The memory space occupied by this information is approximately 342.0 MiB. Additionally, this column possesses a unique count of 96, signifying its diverse nature within the dataset. This unique count accounts for less than 0.1% of the total values.

Table 4.3.3.3 : Code2 Feature Data Sample

1st row	R02-M1-N0-C:J12-U11
2nd row	R02-M1-N0-C:J12-U11
3rd row	R02-M1-N0-C:J12-U11
4th row	R02-M1-N0-C:J12-U11
5th row	R02-M1-N0-C:J12-U11

Component 1 Column :

The "Component1" column is characterized by its categorical nature, featuring a distinct count of 7 values, constituting less than 0.1% of the entire dataset. Notably, there are no missing entries in this column. The memory occupied by this information is approximately 271.8 MiB. Among these values, "RAS" appears most frequently with a count of 4,643,432, followed by "NULL" with 104,215 occurrences, and "KERNEL" with 306 instances. The column also includes terms such as "for" and "interrupts," each appearing 4 times. This column demonstrates a unique count of 2, accounting for less than 0.1% of the total values. Additionally, "Component1" exhibits high correlation and imbalance characteristics within the dataset.

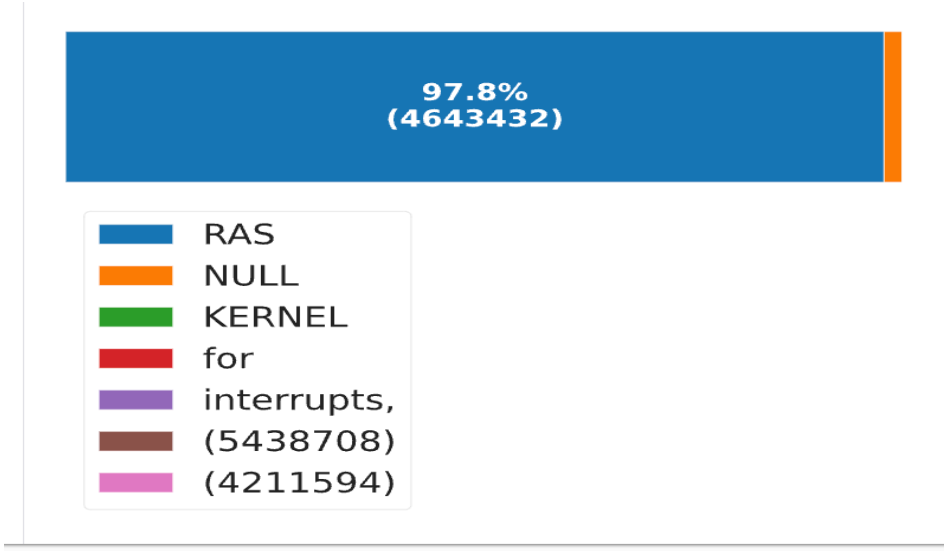


Figure 4.3.3.2 : Component Feature Common Value Plot

Table 4.3.3.4 : Component1 Feature Data Sample

1st row	RAS
2nd row	RAS
3rd row	RAS
4th row	RAS
5th row	RAS

Component 2 Column :

The "Component2" column is characterized by categorical data, encompassing a distinct count of 14 values, which accounts for less than 0.1% of the dataset. There are no missing entries within this column. The memory footprint of this information is approximately 284.7 MiB. The value "KERNEL" stands out as the most frequent, appearing 4,324,651 times, followed by "APP" which is present 228,536 times, "DISCOVERY" occurring 97,172 times, "MMCS" noted 88,930 times, and "HARDWARE" with 5,148 occurrences. The column also encompasses 9 other distinct values, totaling 3,526. Moreover, it's noteworthy that the uniqueness of this column is 0, representing 0.0% uniqueness.

Table 4.3.3.5 : Component2 Feature Data Sample

1st row	KERNEL
2nd row	KERNEL
3rd row	KERNEL
4th row	KERNEL
5th row	KERNEL

Level Column :

The "Level" column is characterized by categorical data with high correlation and imbalance. It holds 10 distinct values, constituting less than 0.1% of the dataset, with no missing entries. The memory usage for this column is approximately 277.2 MiB. The term "INFO" appears as the most frequent value, occurring 3,735,813 times, followed by "FATAL" with 855,195 instances and "ERROR" occurring 112,355 times. Additionally, "WARNING" appears 23,357 times and "SEVERE" 19,213 times. The column also encompasses 5 other distinct values

Table 4.3.3.6 : Level Feature Data Sample

1st row	INFO
2nd row	INFO
3rd row	INFO
4th row	INFO
5th row	INFO

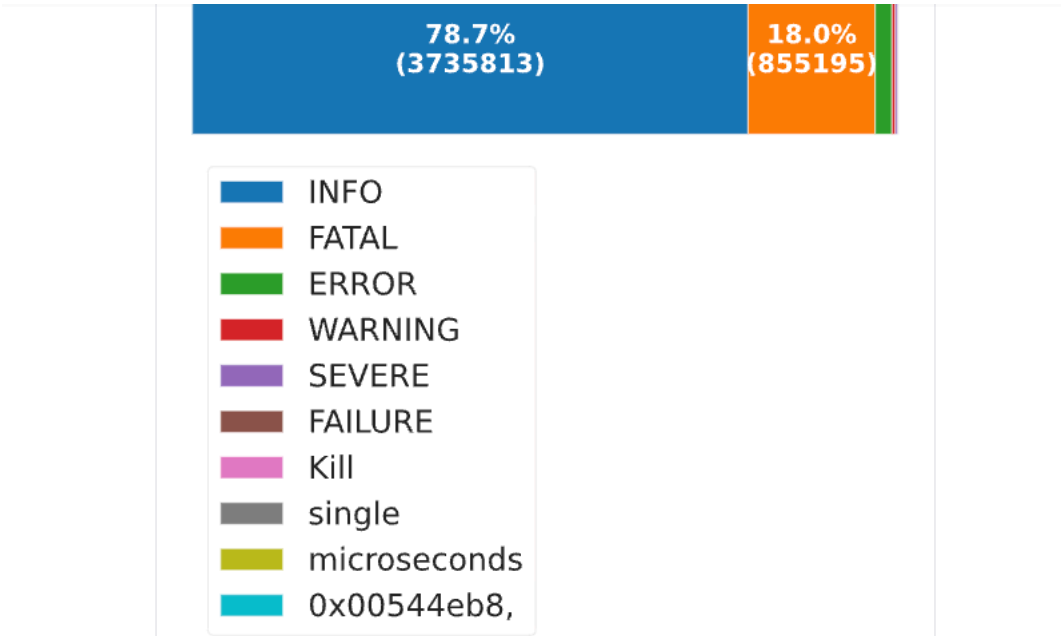


Figure 4.3.3.3 :Common value plot : Level

Content Column:

The "Content" column is of categorical nature and exhibits a distinct count of 358,357, representing approximately 7.5% of the dataset, with no missing values present. The memory space occupied by this column is approximately 479.3 MiB. Additionally, there are 242,471 unique values, accounting for around 5.1% of the total values in the column.

Figure 4.3.3.4 : Most occurring categories based on words: Content Column

Table 4.3.3.7 : Content Feature Data Sample

1st row	instruction cache parity error corrected
2nd row	instruction cache parity error corrected
3rd row	instruction cache parity error corrected
4th row	instruction cache parity error corrected
5th row	instruction cache parity error corrected

4.4 Data Cleaning, Pre Processing and Feature Engineering

We start the data pre-processing steps by setting the paths for input data and output directory. It establishes the necessary file paths for loading and saving data.

4.4.1 Log Parsing using Drain Algorithm

As a component of Feature Engineering, this data pre-process involves Log Parsing using heuristic parsing algorithm which is Drain, The Drain algorithm's objective is to extract structured log templates from unprocessed logs, aiding log analysis and troubleshooting. The process involves constructing a prefix tree that captures hierarchical relationships between log tokens, facilitating quick matching and clustering. The algorithm pre-processes logs, generates templates, and outputs structured logs. The Drain algorithm improves log readability by standardizing log patterns while retaining crucial details systematically applying various regular expression patterns. For each pattern in content column, it replaces corresponding occurrences in the provided log line with a general symbol <*>. This initial step aims to streamline log entries by concealing specific elements like hexadecimal values, IP addresses, and numerical sequences. This procedure not only standardizes the log content but also filters out potentially sensitive or less relevant information from the dataset.

As a part of this process we have generated BGL Structured Log templates csv file which will have all the details regarding common occurring EventID, EventTemplate and Occurrences.

```
df_templates.head()
```

	EventId	EventTemplate	Occurrences
0	3aa50e45	instruction cache parity error corrected	105924
1	b9250ad9	MidplaneSwitchController performing bit sparin...	964
2	fc5f0940	generating core.<*>	1706751
3	1840cbfe	<*> ddr errors(s) detected and corrected on ra...	33939
4	828a502b	<*> <*> <*> error(s) (dcr <*>) detected and co...	5040

Figure 4.4.1.1 : Drain Log Parsing Template Results

Below Visualization identifies top 5 eventid's which occurs frequently

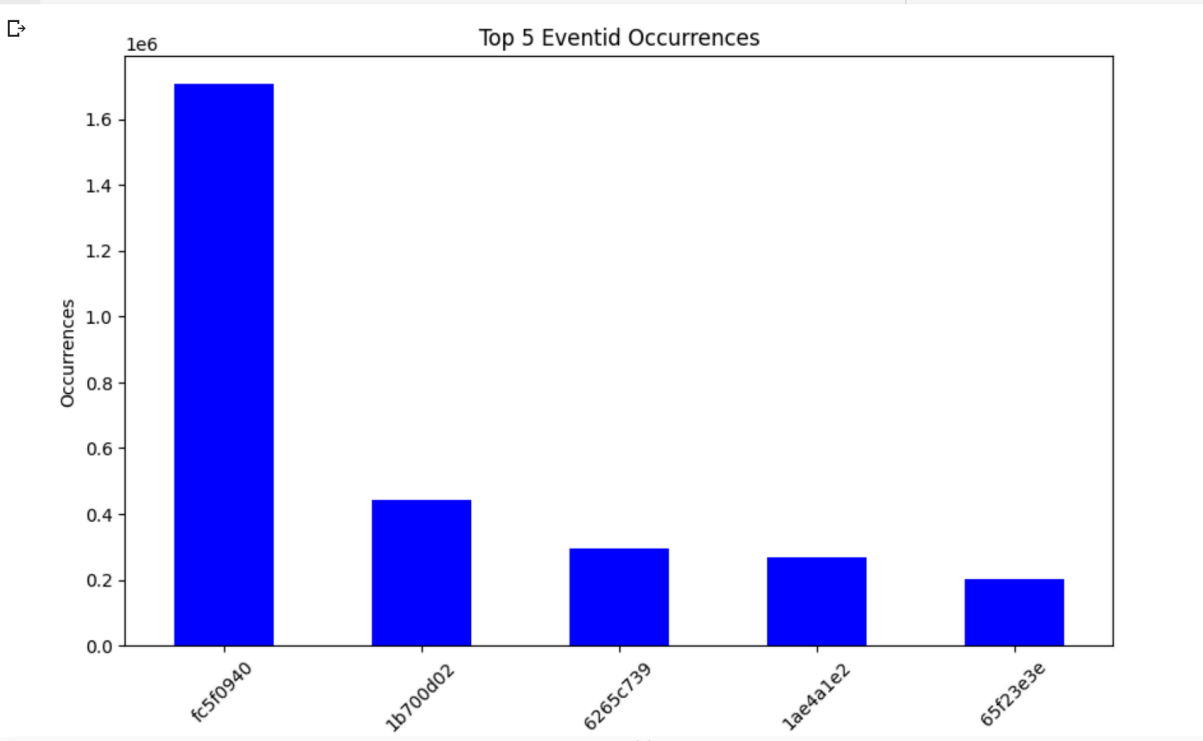


Figure 4.4.1.2 : Top 5 Event ID Occurrences

We have added two Feature Engineered Column EventId and EventTemplate to the data frame each log will be mapped to its structured EventId and EventTemplate.

```
df.head()
```

	LineId	Label	Id	Date	Code1	Time	Code2	Component1	Component2	Level	Content	EventId	EventTemplate
0	1	-	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.363779	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected
1	2	-	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.527847	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected
2	3	-	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	3aa50e45	instruction cache parity error corrected
3	4	-	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.823719	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected
4	5	-	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.982731	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected

Figure 4.4.1.3 : DataFrame head – Feature Engineered Column

The top event IDs for both the 'Normal' and 'Abnormal' labels along with their occurrences. For EventID fc5f0940 we have maximum number of Normal Logs in the dataset

Label		Label	EventId	Occurrences
Label				
-	1265	-	fc5f0940	1706751
APPALLOC	1282	APPALLOC	4496b375	144
APPBUSY	1283	APPBUSY	0fae9ddc	512
APPCHILD	1284	APPCHILD	4496b375	320
APPOUT	1285	APPOUT	083912c3	816
APPREAD	1286	APPREAD	183ff96a	5983
APPRES	1288	APPRES	8fab64d7	2256
APPSEV	1290	APPSEV	8fab64d7	26546
APPTO	1291	APPTO	5eeb7c0e	1991
APPTORUS	1292	APPTORUS	89c713e5	10
APPUNAV	1293	APPUNAV	4496b375	2048
KERNBIT	1294	KERNBIT	e60185c5	1
KERNCON	1295	KERNCON	f331f27a	16
KERNDTLB	1296	KERNDTLB	fd2fdc47	152734
KERNEXT	1297	KERNEXT	810c7f78	1
KERNFLOAT	1298	KERNFLOAT	1bff92a7	3

Figure 4.4.1.4 : Top Event Id / Normal & Abnormal Log Occurrences

4.4.2 : Column Transformation

During the data Pre-processing and cleaning phase, several transformations are applied to the dataset. The 'Time' column is converted to a datetime format, leading to the creation of a new 'datetime' column. The 'Label' column is modified to binary values (1 or 0 where 0 is normal

and 1 is abnormal logs) based on whether it differs from "-". Additionally, the 'timestamp' column is generated from the 'datetime' values, representing Unix timestamps in seconds. Calculating the time difference between consecutive rows results in the 'deltaT' column, which captures temporal intervals. To handle missing values in 'deltaT', they are replaced with 0, ensuring uniformity and accuracy in the dataset's temporal aspects.

```
df.head()
```

	LineId	Label	Id	Date	Code1	Time	Code2	Component1	Component2	Level	Content	EventId	EventTemplate	datetime	timestamp	deltaT
0	1	0	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.363779	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected	2005-06-03 15:42:50.363779	1117813370	0.000000
1	2	0	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.527847	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected	2005-06-03 15:42:50.527847	1117813370	0.164068
2	3	0	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	3aa50e45	instruction cache parity error corrected	2005-06-03 15:42:50.675872	1117813370	0.148025
3	4	0	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.823719	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected	2005-06-03 15:42:50.823719	1117813370	0.147847
4	5	0	1117838570	2005.06.03	R02-M1-N0-C:J12-U11	2005-06-03-15.42.50.982731	R02-M1-N0-C:J12-U11	RAS	KERNEL	INFO	instruction cache parity error corrected	3aa50e45	instruction cache parity error corrected	2005-06-03 15:42:50.982731	1117813370	0.159012

Figure 4.4.2.1 : Pre-processed and cleaned dataset

4.4.3 Log sequencing using sliding window

Further on the cleaned dataset we have applied the sliding window technique where a fixed - size window moves sequentially over the dataset. When applying a sliding window, the window size and step size are crucial parameters. The window size influences the amount of data considered for analysis, affecting the granularity of insights. The step size controls the overlap between consecutive windows, impacting how much information is shared between them. The timestamp and deltaT assist in aligning the sliding window with the temporal aspects of the data, ensuring accurate analysis and maintaining a contextually meaningful sequence of events.we have considered window_size = 300 and step_size = 60 for log sequencing.

```
there are 37315 instances (sliding windows) in this dataset
```

	timestamp	Label	EventId	deltaT
0	[1117813370, 1117813370, 1117813370, 111781337...	0	[3aa50e45, 3aa50e45, 3aa50e45, 3aa50e45, 3aa50...	[0.0, 0.164068, 0.148025, 0.147847, 0.159012, ...
1	[1117813430, 1117813430, 1117813430, 111781343...	0	[3aa50e45, 3aa50e45, 3aa50e45, 3aa50e45, 3aa50...	[0.0, 0.154454, 0.15088, 0.137283, 0.165617, 6...
2	[1117813490, 1117813490, 1117813490, 111781349...	0	[3aa50e45, 3aa50e45, 3aa50e45, 3aa50e45, 3aa50...	[0.0, 0.136003, 0.169535, 0.158248, 0.141224, ...
3	[1117813550, 1117813550, 1117813550, 111781355...	0	[3aa50e45, 3aa50e45, 3aa50e45, 3aa50e45, 3aa50...	[0.0, 0.158966, 0.136016, 0.160414, 0.148952, ...
4	[1117813610, 1117813610, 1117813610, 111781361...	0	[3aa50e45, 3aa50e45, 3aa50e45, 3aa50e45, 3aa50...	[0.0, 0.286636, 0.148804, 0.307791, 0.290853, ...

Figure 4.4.3.1 : Data Frame after sliding window execution

4.4.4 Feature Scaling :

This process is utilized when dealing with categorical data, like the “Event Id” column, and the goal is to represent it numerically. It is a crucial transformation since numerous machine learning algorithms require numerical data for effective processing. This conversion enables the utilization of such algorithms with datasets containing categorical information.

4.4.5 Data Splitting : Splitting a dataset into two distinct parts, known as train-test splitting, is a crucial stage in the development of machine learning models. This process entails the creation of a training subset and a testing subset. The training portion is utilized to teach the machine learning model, enabling it to grasp data patterns and relationships effectively. Meanwhile, the testing subset remains isolated and serves the purpose of appraising the model's performance. This segregation allows us to gauge the model's ability to adapt to new, unseen data, providing us with insights into its real-world capabilities. We have spitted the data for Training and Testing and applied Feature Extraction to process and transform log sequences into a structured data format that can be more suitable for machine learning algorithms or other analytical tasks.

4.5 Summary :

In this section, we present the findings of our systematic investigation into Log Anomaly Detection using the BGL dataset. The primary objective was to detect anomalies within this dataset, which contains a mixture of alert and non-alert messages. The dataset's characteristics, including its size with millions of entries and significant log file size, are detailed.

Our Exploratory Data Analysis (EDA) reveals essential insights about the dataset. Notably, there are no missing values, and instances of duplicate data are minimal. The dataset consumes substantial memory space and comprises a combination of categorical and numeric variables. Several columns exhibit high cardinality, correlations, and class imbalances. We proceed with a closer look at individual columns, such as "Label," "Date," "Code1," "Time," "Code2," "Component1," "Component2," "Level," and "Content." Each column's distinct values, frequencies, and memory usage are thoroughly examined.

The data undergoes a comprehensive cleaning and pre-processing phase. The Drain algorithm is employed for log parsing, which extracts structured log templates, standardizes log patterns, and removes sensitive information. Feature engineering introduces new columns like EventId and Event Template. Log sequencing is applied using sliding windows, and feature scaling is employed to represent categorical data numerically. The data is then split into training and testing sets, and feature extraction prepares log sequences for machine learning models. This thorough analysis and pre-processing lay the foundation for building anomaly detection models in subsequent sections.

Chapter 5

Results and Discussions

5.1 Introduction :

In Section 5.2, our attention shifts towards the core of our research: Model Experimentation. Within this section, we embark on an extensive exploration of various model categories, all crafted to address the challenge of Log Anomaly Detection. We commence with the Baseline Models in Section 5.2.1, diving into essential techniques like PCA (Principal Component Analysis), Isolation Forest, One-Class SVM, and LogClustering. This segment concludes with a brief recap of our discoveries. Transitioning to Section 5.2.2, we investigate Transformer Models, encompassing Bert Base, Tuned Bert, DistilBERT, RoBERTa, Albert, and XLNet models. Each model undergoes evaluation for its effectiveness in anomaly detection, and we consolidate the collective insights. In Section 5.2.3, we delve into LSTM Models, covering the Base LSTM Model, Tuned LSTM Model, and BiLSTM Model, with each model's performance assessed and summarized. Finally, Section 5.2.4 introduces Hybrid Models, uniting LSTM and BERT capabilities, with detailed examinations and summarizations of each hybrid model variant. To conclude our journey, Section 5.3 represents the Model Evaluation and Summary phase, where we thoroughly assess the performance of each model, resulting in an inclusive overview of our Log Anomaly Detection investigation.

5.2 Model Experimentation

In this research thesis, we extensively appraised a range of machine learning models tailored for anomaly detection. We evaluated the efficacy of each model concerning accuracy, precision, recall, and the F1 score, with a primary focus on their adeptness in proficiently identifying anomalies.
As a foundational benchmarking approach, we established baseline using the below listed models. These models served as the initial reference point for evaluating the performance of subsequent approaches.

5.2.1. Baseline Models :

5.2.1.1 PCA (Principal Component Analysis) Model

Model Description:

Principal Component Analysis (PCA) is a technique employed for reducing the dimensionality of data and enhancing visualization. It accomplishes this by projecting data from a high-dimensional space into a lower-dimensional one, with a focus on retaining maximum variance.

Input Parameters:

- n_components=0.8: This parameter specifies the number of principal components to keep, ensuring they account for 80% of data variance.
- threshold=1: Data points exceeding this threshold are classified as anomalies.
- c_alpha=1.9600: A constant linked to the Chi-squared distribution's critical value, used to compute the Mahalanobis distance threshold for anomaly detection.

Output:

The outcome of the PCA model is a transformed dataset where features are projected onto a fresh principal component subspace. This transformed data proves valuable for downstream tasks, visualization, and additional analyses, particularly anomaly detection.

Performance Metrics:

Table 5.2.1.1.1 : PCA Model Evaluation Matrix

Metric	Value
Confusion Matrix	
TP (True Positives)	1194
FP (False Positives)	11868
TN (True Negatives)	1851
FN (False Negatives)	14
Precision	9.141%
Recall	98.841%
F1-Measure	16.734%
Accuracy Score	0.20399276478863804

Interpretation:

The PCA model demonstrated an accuracy of 20.40%, suggesting a correct classification of around 20% of instances. Nonetheless, its precision was notably low (9.141), indicating a considerable proportion of instances labelled as anomalies were not truly anomalous. Impressively, the model exhibited a high recall value (98.8411), effectively capturing a significant portion of actual anomalies. The F1 score (16.7344) serves as a trade-off between precision and recall, portraying the model's comprehensive performance. It’s worth noting that while PCA isn't typically used for anomaly detection, the provided parameters and outcomes shed light on its performance within this specific context. The diminished precision and F1 score emphasize that PCA might not be the most fitting technique for this precise anomaly detection task.

5.2.1.2 Isolation Forest Model

Model Description:

The Isolation Forest algorithm serves as an effective tool for anomaly detection by isolating anomalies from normal instances. It achieves this through the construction of a tree structure, where anomalies are distinguished by requiring fewer splits for isolation, facilitating their distinct identification.

Input Parameters:

- n_estimators=100: The forest consists of a specified count of trees or estimators.
- max_samples='auto': The selection of samples to build individual trees is governed by this parameter, set to 'auto' for default behaviour.
- contamination='auto': This parameter denotes the expected proportion of anomalies in the dataset, with 'auto' signifying the use of default values
- random_state=19: Ensuring reproducibility, this parameter fixes the random seed.

Output:

The Isolation Forest model categorizes data instances into anomalies or normal based on their isolation scores, assigned during the construction of the tree structure. Anomalies are characterized by lower isolation scores owing to their swift isolation in the tree layout.

Performance Metrics:

Table 5.2.1.2.1 : Isolation Forest Evaluation Matrix

Metric	Value
Confusion Matrix	
True Positives (TP)	170
False Positives (FP)	0
True Negatives (TN)	13,719
False Negatives (FN)	1,038
Precision	100.000%
Recall	14.073%
F1-measure	24.673%
Accuracy Score	93.05%

Interpretation:

The Isolation Forest model exhibits a noteworthy 93.05% accuracy, effectively classifying the majority of instances. Notably, an immaculate precision of 100.0 implies that flagged anomalies indeed correspond to actual anomalies, reflecting an absence of false positives. However, the recall value of 14.0728 indicates that the model captures only a limited fraction of true anomalies, consequently resulting in a relatively modest F1 score of 24.6734. Although the Isolation Forest excels in precision, further enhancements are essential to enhance its capacity in recognizing a larger subset of genuine anomalies. This underscores the model's competence in precise anomaly detection while emphasizing the need to expand its coverage of actual anomalies.

5.2.1.3 One-Class SVM Model

Approach Description:

The One-Class Support Vector Machine (SVM) is an algorithm employed to detect novel instances and outliers in data. It operates by constructing a hyperplane that encloses a subset of data points, aiming to separate them from the bulk of the dataset.

Input Parameters:

- kernel='rbf': The radial basis function (RBF) kernel is utilized to transform input data into a higher-dimensional space, aiding in distinguishing anomalies from normal instances.

- degree=3: The degree parameter applies to polynomial kernels and influences the degree of the polynomial function integrated into the kernel.

- gamma='scale': The gamma parameter shapes the RBF kernel's characteristics. 'Scale' implies that gamma is calculated as 1 / (n_features * X.var()), with X being the training data.

- nu=0.5: The nu parameter governs the ratio of training errors and the upper limit of margin errors.

- max_iter=-1: This parameter sets the maximum number of iterations for algorithm convergence. A value of -1 signifies no iteration limit.

Output:

The One-Class SVM model furnishes a binary classification outcome, categorizing instances as anomalies or normal data points.

Performance Metrics:

Table 5.2.1.3.1 : One-Class SVM Evaluation Matrix

Metric	Value
Confusion Matrix	
True Positives (TP)	144
False Positives (FP)	13,719
True Negatives (TN)	0
False Negatives (FN)	1,064
Precision	1.039
Recall	11.921
F1-Measure	1.911
Accuracy Score	0.0096

Interpretation:

The achieved accuracy of the One-Class SVM model was 0.009647, indicating a notably low rate of accurate classifications. Precision registered at 1.0387, implying a modest capacity to accurately identify authentic anomalies but permitting a considerable number of false positives. A recall score of 11.9205 signifies the model's partial success in capturing real anomalies, albeit not comprehensively. The F1 score (1.911) strikes a balance between precision and recall, underscoring the model's intricacies within this specific anomaly detection context.

It's important to acknowledge that the One-Class SVM is designed for binary classification scenarios, distinguishing normal instances from anomalies. The outcomes suggest potential challenges in the model's capability to effectively identify anomalies, given the specified parameters and data distribution. Further parameter fine-tuning and exploration of alternative models could be essential for enhanced anomaly detection performance.

5.2.1.4 LogClustering Model

Model Description:

LogClustering is a methodology aimed at detecting anomalies in log data. It accomplishes this by organizing similar log events into clusters and identifying anomalies based on deviations from established patterns.

Input Parameters:

- max_dist=0.3: This parameter establishes a threshold that guides the clustering process. Clusters are formed until the distance between log events surpasses this defined threshold.

- anomaly_threshold=0.3: The anomaly detection threshold defines the degree to which deviations from clustered patterns are flagged as anomalies.

- mode='online': The mode parameter specifies whether the clustering process is conducted online or offline.

- num_bootstrap_samples=1000: This parameter governs the number of bootstrap samples employed for generating confidence intervals in anomaly detection.

Output:

The LogClustering model yields both the identification of formed clusters and the categorization of anomalies based on deviations from established patterns.

Performance Metrics:

Table 5.2.1.4.1 : LogClustering Model Evaluation Matrix

Metric	Value
Confusion Matrix	
True Positives	798
False Positives	29
True Negatives	13,690
False Negatives	410
Precision	96.493%
Recall	66.060%
F1-measure	78.427%
Accuracy Score	97.059%

Interpretation:

The LogClustering model leverages clustering to group log events sharing similar patterns. It subsequently pinpoints anomalies by identifying deviations from these established patterns. As deduced from the evaluation summary, the model displayed commendable performance.
A precision score of 96.493 indicates a noteworthy proportion of accurate predictions for anomalies within the total predicted anomalies. With a recall value of 66.060, the model adeptly captures a significant segment of actual anomalies. The F1-measure of 78.427 strikes an equilibrium between precision and recall, showcasing the model's comprehensive performance. Furthermore, the robust accuracy score of 97.06% underscores the model's overall proficiency in anomaly detection, based on the given parameters and data characteristics. Notably, the success of LogClustering is contingent upon the specific log data characteristics and parameter choices. Fine-tuning and exploration of model parameters can prove advantageous across diverse data scenarios.

5.2.1.5 Baseline Models Summary:

The baseline evaluation of various models revealed intriguing insights. While the PCA model achieved an accuracy of 20.40%, its notably low precision and F1 score highlighted limitations in distinguishing genuine anomalies. Conversely, the Isolation Forest displayed strong accuracy and precision, but its restricted recall affected its overall anomaly capture. The One-Class SVM model encountered challenges in both accurate classifications and comprehensive anomaly detection. In contrast, the LogClustering model, with high precision and respectable recall, showcased adeptness in capturing real anomalies. Notably, the LogClustering models stood out with commendable performance, emphasizing the importance of precision-recall balance also reliant on specific data characteristics and parameter choices, warranting further exploration for enhanced performance across diverse data scenarios.

5.2.2 Transformers Models

In addition to the baseline model results, we integrated Transformers models into our approach. Utilizing the renowned Transformers framework, celebrated for its excellence in natural language processing and sequence-to-sequence tasks, we expanded its application to anomaly detection. We deployed both pre-trained and fine-tuned versions of well-known Transformers architectures, including BERT, DistilBERT, RoBERTa, ALBERT, and XLNet. These models consistently demonstrated promising performance, yielding accuracy, precision, recall, and F1 scores.

5.2.2.1 Bert Base Model

Model Description:

The Bert_base_uncased model is a variant of BERT, a widely-used transformer-based model pre-trained on lowercase text. Its application is tailored to text classification tasks, where the objective is to predict the appropriate label or category for a given text input.

Input Parameters:

- **batch_size = 16:** This parameter governs the number of training examples processed within each iteration.
- **num_labels = 2**: Denoting the count of distinct labels or categories within the classification task.
- **criterion = nn.CrossEntropyLoss():** Employed as the chosen loss function during model training.
- **optimizer = torch.optim.Adam(bert_base_model.parameters(), lr=1e-5):** The Adam optimizer is employed for parameter updates during training, utilizing the specified learning rate.
- **max_seq_length = 512**: The maximum sequence length allowed for input text, guiding the handling of lengthier text.

Training Process:

The model undergoes training across 5 epochs, during which it progressively refines its understanding of data patterns. The CrossEntropyLoss function serves as the guiding criterion for learning, while the Adam optimizer fine-tunes model parameters. The learning rate is fixed at 1e-5.

Performance Metrics:

Table 5.2.2.1.1 : Bert Base Model Evaluation Matrix

Metric	Score
Test Average Accuracy	0.9800
Test Average Precision	0.9900
Test Average Recall	0.8000
Test Average F1-Score	0.8800

Interpretation:

The Bert_base_uncased model exhibits robust performance across diverse evaluation metrics:

- An average accuracy of 98.00% indicates that the model effectively predicts labels for approximately 98% of the test instances.
- The average precision of 99.00% underscores the model's proficiency in minimizing false positive predictions.
- Averaging at 80.00%, the recall metric demonstrates the model's ability to correctly identify 80% of the actual positive instances.

- Balancing precision and recall, the average F1-score of 88.00% provides an overall gauge of the model's equilibrium between the two metrics.

5.2.2.2 Tuned Bert Model

Model Description:

The Tuned Bert model is a personalized iteration of BERT, optimized for text classification assignments. It integrates a customized classifier design to enhance its predictive capabilities.

Input Parameters:

- **batch_size = 16:** Dictates the quantity of training examples processed per iteration.
- **num_labels = 2:** Denotes the number of distinct labels or categories within the classification task.
- **criterion = nn.CrossEntropyLoss():** Adopted as the loss function during the model's training phase.
- **optimizer = torch.optim.Adam(bert_base_model.parameters(), lr=1e-5):** The Adam optimizer orchestrates parameter updates during training, with a learning rate set at 1e-5.
- **max_seq_length = 512:** Specifies the maximum permissible sequence length for input text, accommodating varying text sizes.

Model Architecture:

The Tuned Bert model employs a personalized classifier layered atop BERT's contextual embeddings. This classifier comprises sequential components:

- A fully connected layer encompassing 256 units, succeeded by a ReLU activation.
- A dropout layer with a dropout rate of 20%.
- A final fully connected layer, designed to map to the designated number of labels.

Training Process:

Training is conducted over 5 epochs. The gradual decline in loss values across epochs signifies the model's learning convergence. The adopted CrossEntropyLoss function guides the model's learning trajectory, while parameter fine-tuning is facilitated by the Adam optimizer.

Performance Metrics:

Table 5.2.2.2.1 : Tuned Bert Base Model Evaluation Matrix

Metric	Value
Training Loss	0.0756
Test Accuracy	98.00%
Test Precision	97.00%
Test Recall	82.00%
Test F1-Score	88.00%

Interpretation:

The Tuned Bert model underscores its prowess through diverse evaluation metrics:

- An average accuracy of 98.00% underscores the model's precision in labeling around 98% of the test instances correctly.
- The average precision, standing at 97.00%, highlights the model's adeptness in curtailing false positive predictions.
- With an average recall of 82.00%, the model showcases its efficiency in capturing 82% of actual positive instances.
- Striking a balance between precision and recall, the average F1-score of 88.00% encapsulates the model's comprehensive performance.

5.2.2.3 DistilBERT Model

Model Description:

The DistilBERT model represents a distilled iteration of the BERT model, tailored for efficient text comprehension and classification tasks. It employs a streamlined architecture while retaining BERT's performance capabilities..

Input Parameters:

- **batch_size = 16**: Dictates the number of training examples processed in each iteration.

- **num_labels = 2**: Specifies the count of distinct labels or categories within the classification assignment.

- **criterion = nn.CrossEntropyLoss():** Adopted as the loss function for guiding model training.

- **optimizer = torch.optim.Adam(bert_base_model.parameters(), lr=1e-5):** The Adam optimizer is employed to update model parameters during training, utilizing a learning rate of 1e-5.

- **max_seq_length = 512**: Establishes the maximum allowable sequence length for input text, accommodating varying text lengths.

Model Structure:

The DistilBERT model showcases a refined classifier superimposed on DistilBERT's contextual embeddings. This classifier integrates sequential components:

- An initial fully connected layer featuring 256 units, followed by a ReLU activation.
- A dropout layer with a 20% dropout rate.
- A final fully connected layer, mapped to the specified number of labels.

Training Progression:

The training phase spans 5 epochs. The progressive reduction in loss values over epochs reflects the model's learning advancement. The CrossEntropyLoss function is employed to steer the model's learning trajectory, while the Adam optimizer fine-tunes parameters.

Performance Metrics:

Table 5.2.2.3.1 : DistilBERT Model Evaluation Matrix

Metric	Value
Training Loss	0.0752
Test Accuracy	98.00 %
Test Precision	99.00 %
Test Recall	81.00 %
Test F1-Score	88.00 %

Interpretation:

The DistilBERT model excels in diverse evaluation metrics:

- An average accuracy of 98.00% underscores the model's adeptness in labeling around 98% of the test instances accurately.
- Averaging at 99.00%, the precision metric attests to the model's ability to minimize false positive predictions effectively.
- An average recall of 81.00% signifies the model's capability to capture 81% of actual positive instances.
- Striking a balance between precision and recall, the average F1-score of 88.00% encapsulates the model's comprehensive performance.

5.2.2.4 RoBERTa Model

Model Approach:

The RoBERTa model represents an advanced iteration of the BERT framework, tailored for adept text understanding and classification tasks. Leveraging the capabilities of RoBERTa, this model achieves precise and reliable classification outcomes.

Input Parameters:

- **batch_size = 16:** Determines the quantity of training examples processed per iteration.

- **num_labels = 2**: Specifies the count of distinct labels or categories within the classification context.

- **criterion = nn.CrossEntropyLoss**(): Selected as the loss function guiding model training.

- **optimizer = torch.optim.Adam(bert_base_model.parameters(), lr=1e-**5): The Adam optimizer is employed for updating model parameters during training, with a learning rate set at 1e-5.

- **max_seq_length = 512**: Establishes the maximum allowable sequence length for input text, accommodating a diverse range of text lengths.

Model Composition:

The RoBERTa model features a specialized classifier architecture layered on top of RoBERTa's contextual embeddings. This classifier consists of a sequence of elements:

- An initial fully connected layer comprising 256 units, followed by a ReLU activation.
- A dropout layer introducing a 20% dropout rate.
- A final fully connected layer, designed to map to the specified number of labels.

Training Progression:

Training unfolds across 5 epochs. The gradual decrease in loss values over epochs reflects the model's iterative learning process. The CrossEntropyLoss function guides the model's learning trajectory, while the Adam optimizer optimizes model parameters.

Performance Metrics:

Table 5.2.2.4.1 : RoBERTa Model Evaluation Matrix

Metric	Value
Training Loss	0.0710
Test Accuracy	98.00 %
Test Precision	94.00 %
Test Recall	77.00 %
Test F1-Score	84.00 %

Interpretation:

The RoBERTa model showcases robust performance across diverse evaluation metrics:

- An average accuracy of 98.00% underscores the model's precision in accurately classifying approximately 98% of the test instances.
- Averaging at 94.00%, the precision metric highlights the model's ability to minimize false positive predictions adeptly.
- With an average recall of 77.00%, the model excels in capturing 77% of actual positive instances.
- Striking a balance between precision and recall, the average F1-score of 84.00% provides a comprehensive measure of the model's effectiveness.

5.2.2.5 Albert Model:

Model Approach:

The Albert model represents an advanced neural architecture tailored for proficiently comprehending and classifying text. Harnessing the capabilities of Albert, this model achieves precise classification outcomes.

Input Parameters:

- **batch_size = 16**: Governs the number of training examples processed per iteration.
- **num_labels = 2:** Specifies the count of distinct labels or categories within the classification context.

- **criterion = nn.CrossEntropyLoss():** Selected as the loss function guiding model training.
- **optimizer = torch.optim.Adam(bert_base_model.parameters(), lr=1e-5):** The Adam optimizer orchestrates parameter updates during training, with a learning rate set at 1e-5.
- **max_seq_length = 512**: Sets the maximum allowable sequence length for input text, accommodating varying text lengths.

Model Composition:

The Albert model integrates a specialized classifier structure atop Albert's contextual embeddings. This classifier includes sequential elements:

- An initial fully connected layer comprising 256 units, followed by a ReLU activation.
- A dropout layer with a 20% dropout rate.
- A final fully connected layer, designed to map to the specified number of labels.

Training Progression:

Training spans 5 epochs. The gradual reduction in loss values across epochs reflects the model's iterative learning. The CrossEntropyLoss function steers the model's learning trajectory, while the Adam optimizer optimizes parameters.

Performance Metrics:

Table 5.2.2.5.1 : ALBERT Model Evaluation Matrix

Metric	Value
Training Loss	0.0609
Test Accuracy	98.00 %
Test Precision	98.00 %
Test Recall	81.00 %
Test F1-Score	88.00 %

Interpretation:

The Albert model demonstrates robust performance across various evaluation metrics:

- Averaging an accuracy of 98.00%, the model precisely classifies about 98% of the test instances.
- With an average precision of 98.00%, the model effectively minimizes false positive predictions.

- An average recall of 81.00% indicates the model's proficiency in capturing 81% of actual positive instances.
- Striking a balance, the average F1-score of 88.00% encapsulates the model's comprehensive performance.

5.2.2.6 XLNet Model

Model Approach:

The XLNet model embodies a sophisticated neural architecture meticulously designed for adeptly comprehending and classifying text. By harnessing the strengths inherent to XLNet, this model achieves precise and resilient classification outcomes.

Input Parameters:

- **batch_size = 16**: Governs the number of training examples processed during each iteration.

- **num_labels = 2**: Specifies the count of unique labels or categories inherent to the classification context.

- **criterion = nn.CrossEntropyLoss():** Chosen as the guiding loss function for the model's training process.

- **optimizer = torch.optim.Adam(bert_base_model.parameters(), lr=1e-5):** The Adam optimizer orchestrates parameter updates during training, utilizing a learning rate set at 1e-5.

- **max_seq_length = 512**: Determines the maximum allowable sequence length for input text, accommodating diverse text lengths.

Model Configuration:

The XLNet model integrates a purpose-built classifier structure superimposed upon XLNet's contextual embeddings. This classifier encompasses a sequence of elements:

- An initial fully connected layer consisting of 256 units, followed by a ReLU activation.
- A dropout layer introducing a dropout rate of 20%.
- A concluding fully connected layer, mapped to the designated number of labels.

Training Progression:
The training phase spans 5 epochs. The gradual decrease in loss values across epochs signifies the model's iterative learning progression. The CrossEntropyLoss function guides the model's learning trajectory, while the Adam optimizer fine-tunes parameters.

Performance Metrics:

Table 5.2.2.6.1 : XLNet Model Evaluation Matrix

Metric	Value
Training Loss	0.0668
Test Accuracy	98.00 %
Test Precision	99.00 %
Test Recall	79.00 %
Test F1-Score	87.00 %

Interpretation:

The XLNet model showcases robust performance across a range of evaluation metrics:

- An average accuracy of 98.00% underscores the model's precision in correctly classifying approximately 98% of test instances.
- Averaging at 99.00%, the precision metric underscores the model's proficiency in minimizing false positive predictions.
- An average recall of 79.00% signifies the model's capacity to capture 79% of actual positive instances.
- Achieving an average F1-score of 87.00%, the model adeptly balances precision and recall.

5.2.2.7 Transformer Models Summary

Above Transformer-based models, which encompasses Bert_base_uncased, Tuned Bert, DistilBERT, RoBERTa, Albert, and XLNet, has impressively demonstrated their prowess in forecasting anomalies. Amidst them, the DistilBERT model emerges as an exceptional standout, showcasing robust performance across a spectrum of assessment metrics. Boasting an average accuracy of 98.00%, the model adeptly labels roughly 98% of test instances with precision. Its precision score of 99.00% underscores its adeptness in curbing false positives, while an average recall of 81.00% underscores its ability to identify 81% of genuine positive cases. By harmonizing precision and recall, the model achieves an average F1-score of 88.00%, signifying its holistic efficacy. DistilBERT's expertise designates it as a formidable contender for anomaly detection tasks, where precision, recall, and overall effectiveness bear significance. This model's proficiency in precisely pinpointing anomalies while minimizing false positives positions it as a valuable asset for real-world scenarios necessitating dependable

anomaly predictions. Additionally, it is well-suited for applications entailing real-time or resource-constrained processing.

5.2.3 LSTM Models

5.2.3.1 Base LSTM Model

Model Architecture:

The LSTM model is tailored for sequential data processing and classification. Comprising an LSTM layer followed by a fully connected layer, this design effectively captures sequential patterns for accurate predictions.

Input Parameters:

- **input_size = 1241**: Aligned with input data specifications.
- **hidden_size = 256**: Determines the dimensionality of hidden states within the LSTM.
- **num_layers = 2:** Specifies the count of LSTM layers.
- **num_classes = 2**: Defines the number of distinct classes for classification.
- **batch_size = 32**: Governs simultaneous data processing during training.
- **learning_rate = 0.001**: Sets the rate at which model parameters are adjusted.
- **num_epochs = 10**: Determines the training iteration count.

Model Structure:

Sequential data undergoes processing in the LSTM layer, leveraging hidden states to capture temporal dependencies. Subsequently, the fully connected layer translates LSTM outputs into class predictions.

Training Strategy:

The training phase spans 10 epochs. The CrossEntropyLoss function serves as the guiding criterion, augmented by class weights to address class imbalance. The Adam optimizer handles parameter updates with the designated learning rate.

Performance Evaluation:

Table 5.2.3.1.1 : LSTM Model Evaluation Matrix

Metric	Value
Training Loss	0.0382
Test Accuracy	99.07 %
Test Precision	94.06 %
Test Recall	94.45 %
Test F1-Score	94.26 %

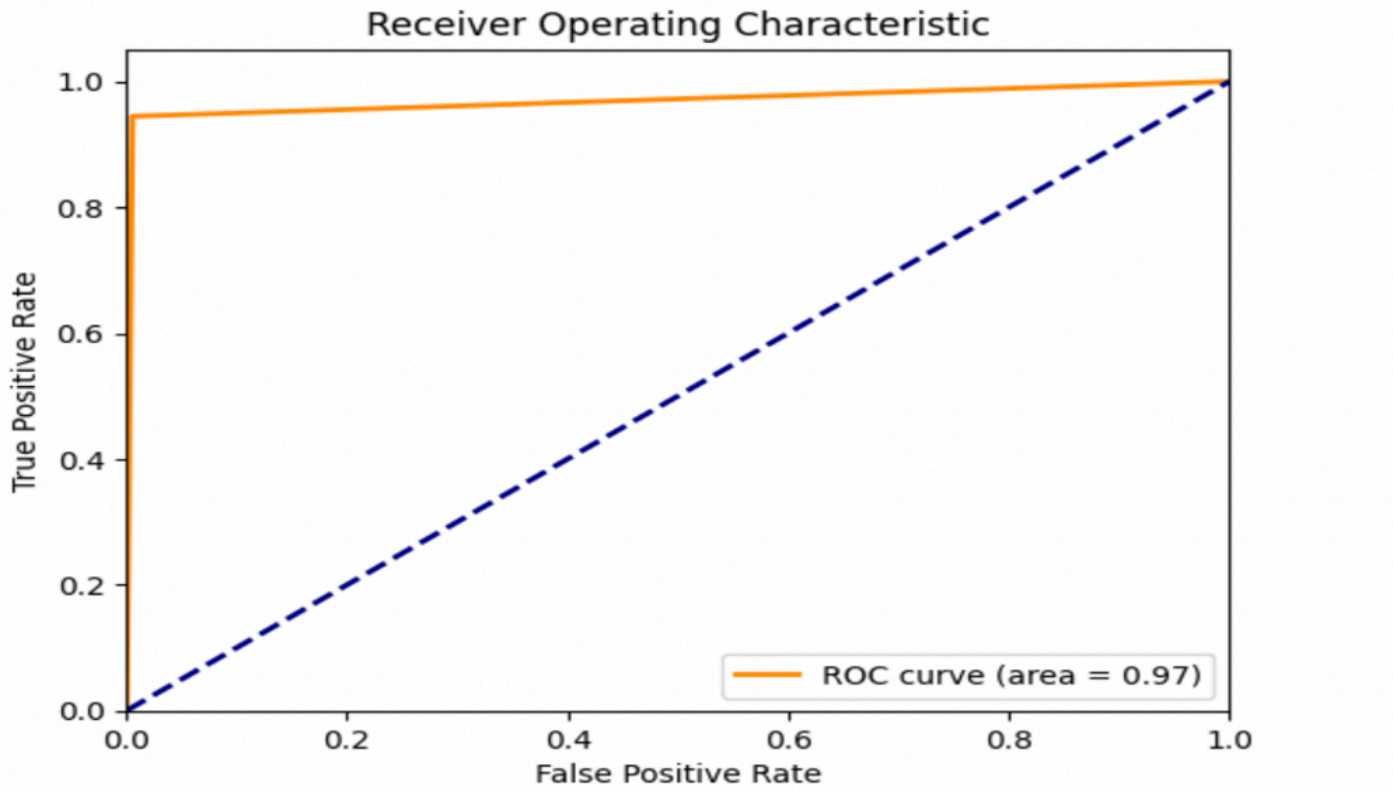


Figure 5.2.3.1.1 : LSTM Model ROC AUC CURVE

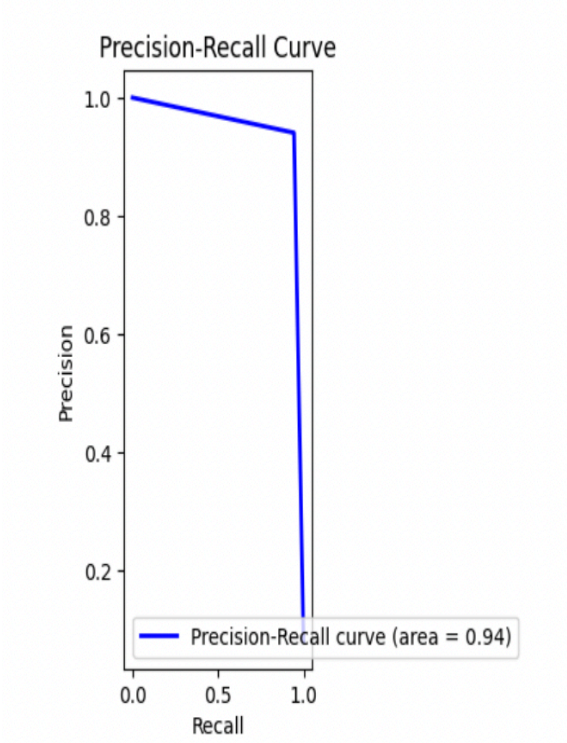


Figure 5.2.3.1.2 : LSTM Precision-Recall

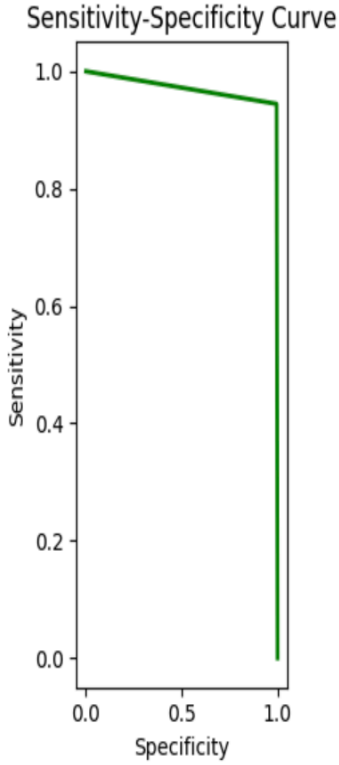


Figure 5.2.3.1.3: LSTM Sensitivity - Specificity

Interpretation:

The LSTM model demonstrates robust performance across a spectrum of evaluation metrics:

- Exceptional test accuracy of 99.07% reflects precise classification.
- Averaging at 94.06%, the precision metric underscores proficient false positive management.
- An average recall of 94.45% underscores the model's adeptness in identifying actual positive instances.
- Balancing precision and recall, the average F1-score of 94.26% encapsulates holistic performance.
- The ROC curve, with an area of 0.97, illustrates how well the model balances true positive rate (recall) and false positive rate. The Precision-Recall curve's area of 0.94 signifies the model's skill in maintaining precision and recall harmony. Additionally, the model's specificity showcases its competence in accurately labeling negative instances, collectively indicating its strong predictive capacity in achieving accurate, balanced, and reliable classifications.

5.2.3.2 Tuned LSTM Model

Model Architecture:

The Tuned LSTM model is purposefully designed for sequential data classification. It integrates an LSTM layer enhanced with dropout regularization and pairs it with a fully connected layer for generating class predictions.

Input Parameters:

- **input_size = 1241:** Tailored to match the data's input characteristics.
- **hidden_size = 256:** Governs the dimensionality of LSTM hidden states.
- **num_layers = 2:** Specifies the count of stacked LSTM layers.
- **num_classes = 2:** Defines the number of classes for classification.
- **dropout = 0.2**: Applies dropout for regularization during training.
- **batch_size = 32:** Manages data processing in batches.
- **learning_rate = 0.001:** Sets the learning rate for optimizing model parameters.
- **num_epochs = 20**: Determines the training epoch count.

Model Workflow:

Sequential data undergoes processing in the LSTM layer, where incorporated dropout enhances temporal pattern capture. The ultimate LSTM output is then fed into the fully connected layer for predicting classes.

Training Strategy:

Training extends over 20 epochs. Leveraging the CrossEntropyLoss function, which accounts for class imbalances through weight adjustment, the model refines its performance. The Adam optimizer steers parameter updates based on the learning rate specified.

Performance Evaluation:

Table 5.2.3.2.1 : LSTM Tuned Model Evaluation

Metric	Value
Training Loss	0.0367
Test Accuracy	99.09 %
Test Precision	95.04 %
Test Recall	93.63 %
Test F1-Score	94.33 %

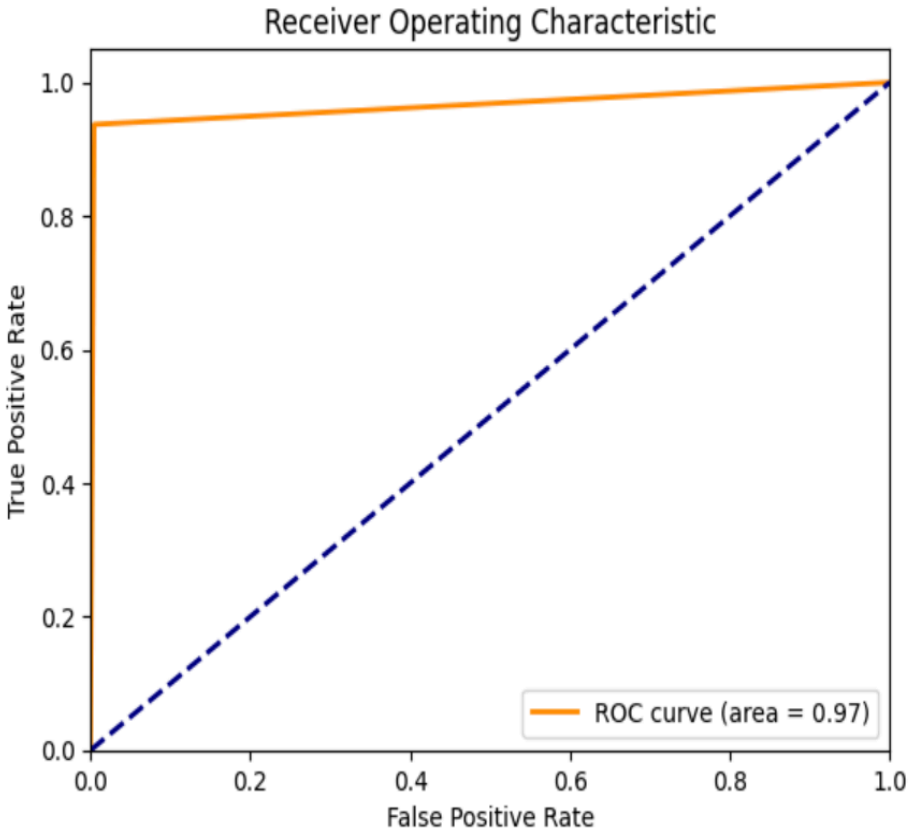
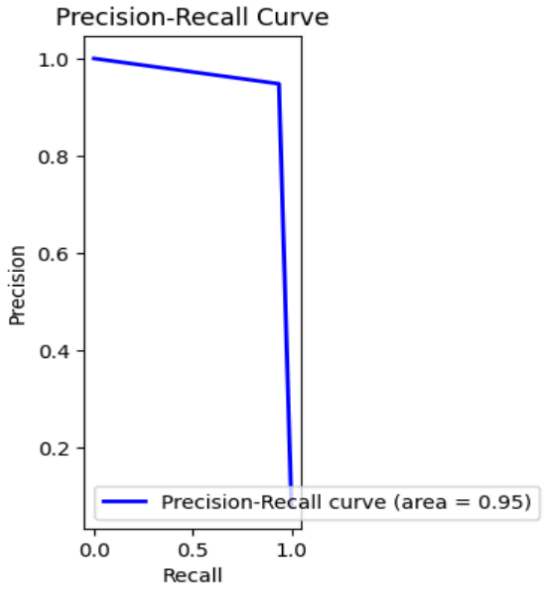


Figure 5.2.3.2.1: LSTM Tuned Model ROC AUC CURVE



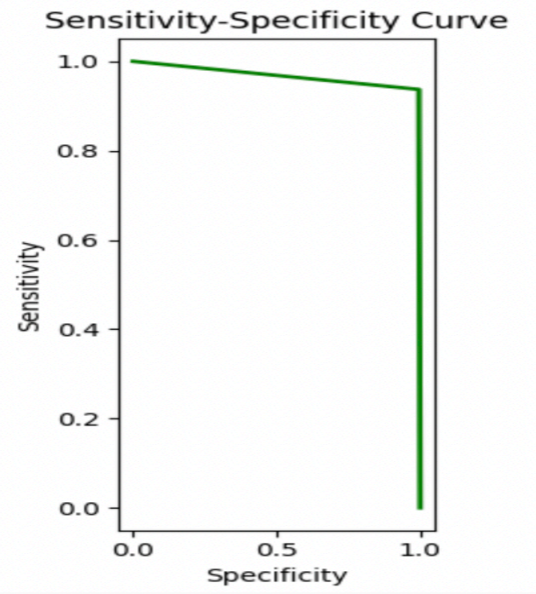


Figure 5.2.3.2.2 :LSTM Tuned Precision-Recall

Figure 5.2.3.2.3: LSTM Tuned Sensitivity - Specificity

Interpretation:

The Tuned LSTM model showcases exceptional performance, substantiated by diverse evaluation metrics:

- Impressive test accuracy of 99.09% signifies precise classification.
- Averaging at 95.04%, the precision metric underscores effective management of false positives.
- An average recall of 93.63% demonstrates the model's adeptness in identifying genuine positive instances.
- With an average F1-score of 94.33%, the model adeptly balances precision and recall, showcasing comprehensive proficiency.
- The ROC curve, with an area of 0.97, illustrates how well the model balances true positive rate (recall) and false positive rate. The Precision-Recall curve's area of 0.95 signifies the model's skill in maintaining precision and recall harmony. Additionally, the model's specificity showcases its competence in accurately labeling negative instances, collectively indicating its strong predictive capacity in achieving accurate, balanced, and reliable classifications.

These outcomes underscore the model's effectiveness in tasks involving sequential data classification. The model's benefits stem from dropout-based regularization and meticulous training. Notably, the model's performance hinges on intricacies within the data and task requirements

5.2.3.3 BiLSTM Model

Model Architecture:

The BiLSTM model is structured to effectively process sequential data for classification tasks. It employs a bidirectional Long Short-Term Memory (BiLSTM) layer, followed by a fully connected layer for generating class predictions.

Input Parameters:

- **input_size = 1241**: Customized to align with input data characteristics.
- **hidden_size = 256:** Governs the dimensionality of hidden states within the BiLSTM.
- **num_layers = 2:** Specifies the count of stacked BiLSTM layers.
- **num_classes = 2:** Defines the number of classes for classification.
- **dropout = 0.2:** Incorporates dropout for regularization during training.
- **batch_size = 32:** Organizes data processing in batch format.
- **learning_rate = 0.001:** Sets the learning rate for parameter updates.
- **num_epochs = 20:** Determines the duration of training epochs.

Model Operation:

The sequential data undergoes processing through the bidirectional LSTM layer, which captures patterns from both directions. The final BiLSTM output is directed to the fully connected layer to generate class predictions.

Training Strategy:

Training spans 20 epochs. The CrossEntropyLoss function is utilized, with adjustable class weights to accommodate class imbalance. The Adam optimizer orchestrates parameter updates based on the specified learning rate.

Performance Metrics:

Table 5.2.3.3.1 : BiLSTM Model Evaluation

Metric	Value
Training Loss	0.0347
Test Accuracy	99.33 %
Test Precision	97.68 %
Test Recall	93.96 %
Test F1-Score	95.78 %

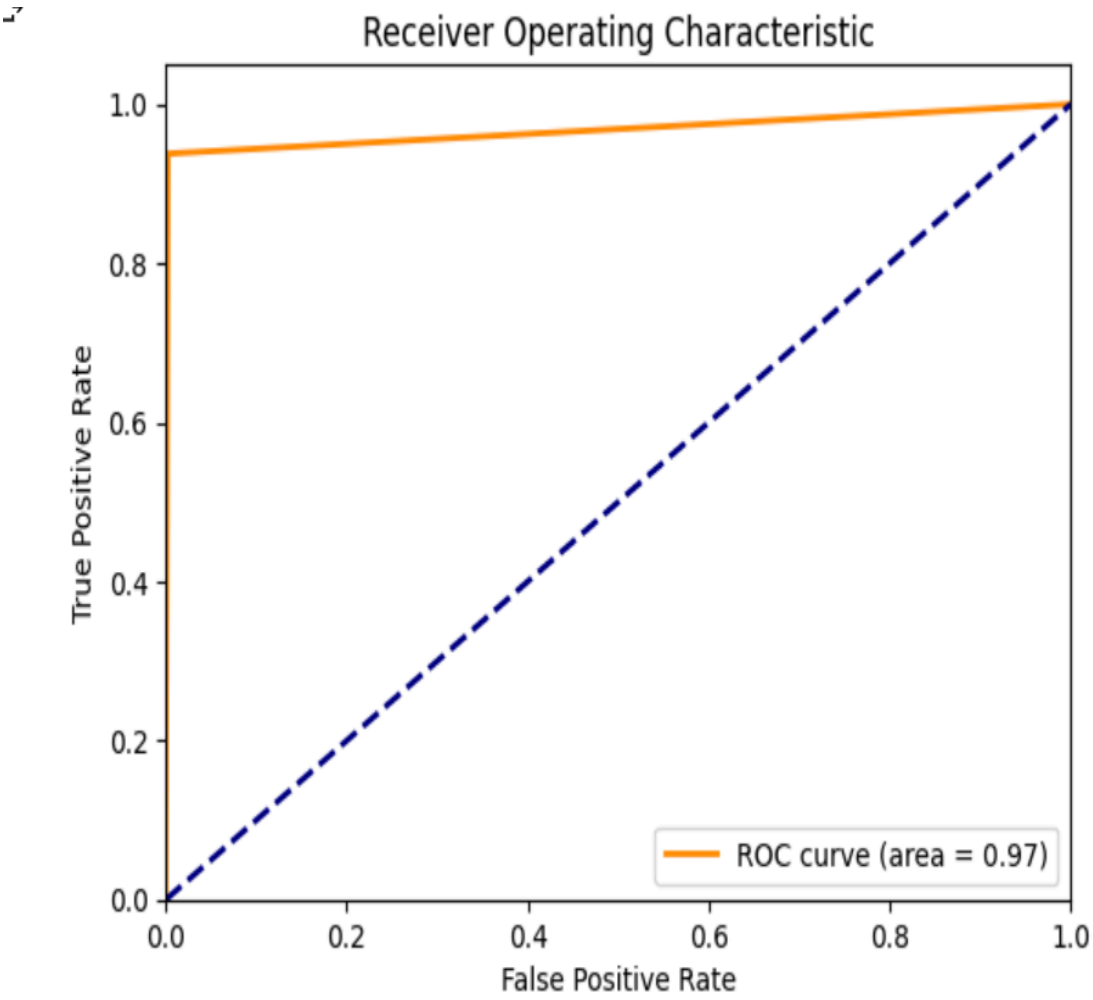
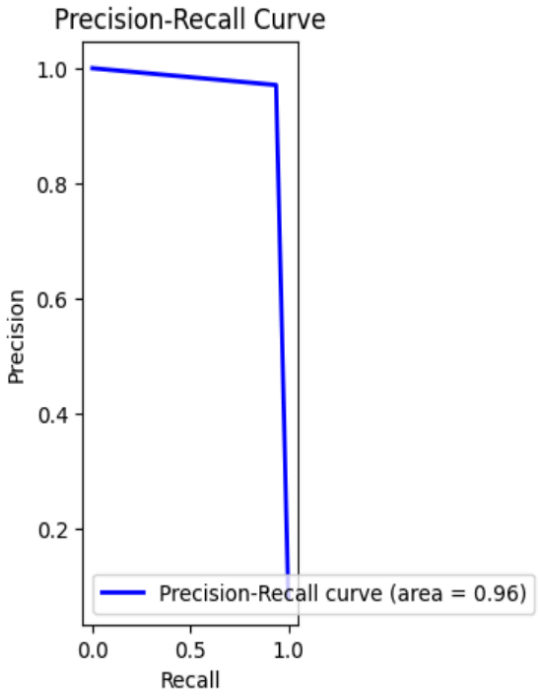


Figure 5.2.3.3.1 : BiLSTM Model ROC AUC CURVE



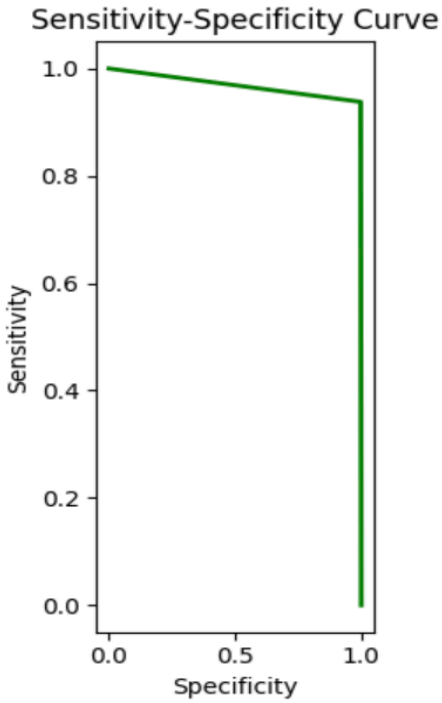


Figure 5.2.3.3.2 : BiLSTM Precision-Recall

Figure 5.2.3.3.3 : BiLSTM Sensitivity - Specificity

Interpretation:

The BiLSTM model excels across diverse evaluation metrics:

- Impressive test accuracy of 99.33% signifies precise classification.
- With an average precision of 97.68%, the model aptly manages false positive instances.
- An average recall of 93.96% highlights the model's efficiency in capturing genuine positive instances.
- The average F1-score of 95.78% adeptly balances precision and recall, indicating comprehensive performance.

The BiLSTM model's proficiency in processing sequential data and capturing bidirectional dependencies positions it as a powerful tool for tasks demanding nuanced pattern recognition. The success stems from its regularization and meticulous training. Notably, the model's performance is influenced by data intricacies and task requirements

5.2.3.4 LSTM Models Summary :

The LSTM model demonstrates strong performance across a variety of assessment metrics, including an outstanding test accuracy of 99.07% that signifies accurate classification. It maintains an average precision of 94.06%, emphasizing efficient management of false positives, and achieves an average recall of 94.45%, highlighting its capability in recognizing genuine positive instances. The model achieves an overall balanced performance with an

average F1-score of 94.26%. Within the evaluated models, the Tuned LSTM model emerges as the frontrunner, exhibiting exceptional outcomes in key metrics. It achieves a noteworthy test accuracy of 99.09%, ensuring precise classification, and maintains an average precision of 95.04% for effective control of false positives. Additionally, it attains an average recall of 93.63%, showcasing its adeptness in identifying true positive instances. The model adeptly balances precision and recall with an average F1-score of 94.33%. Furthermore, the BiLSTM model demonstrates remarkable results with a notable test accuracy of 99.33%, substantial precision at 97.68%, and adept recall at 93.96%. Most notably, it achieves a remarkable average F1-score of 95.78%, showcasing robustness across various evaluation metrics. The BiLSTM model's comprehensive capabilities make it a prime candidate for prediction tasks that demand both accuracy and well-rounded performance.

5.2.4 Hybrid Models :

5.2.4.1 Hybrid Model 1 - LSTM + BERT

Model Architecture:

The Hybrid Model seamlessly combines the strengths of BERT and LSTM to enhance anomaly detection. The architecture integrates BERT for contextual embeddings, followed by an LSTM layer to capture temporal patterns, and a fully connected layer for classification.

Input Parameters:

- The chosen BERT model is 'bert-base-uncased' for leveraging pre-trained capabilities.
- The LSTM input size is tailored to fit the data at hand (1241 in this case).
- A hidden size of 256 is selected to determine the LSTM's hidden state dimensions.
- Two LSTM layers are stacked to optimize the temporal learning process.
- The classification task involves two classes.
- Data is processed in batches of 100 for efficiency.
- Learning occurs with a rate of 0.001.
- The model undergoes training over 5 epochs.

Model Operation:

The model's operation begins with BERT's contextual embeddings, followed by a linear layer for dimension alignment. The LSTM component captures temporal patterns from these embeddings. Lastly, a fully connected layer generates predictions.

Training Strategy:

The training plan spans 5 epochs and employs the CrossEntropyLoss function, while accommodating class imbalance using adjustable class weights. The Adam optimizer manages parameter updates via the specified learning rate.

Performance Metrics:

Table 5.2.4.1.1 : Hybrid Model 1 Evaluation Matrix

Metric	Value
Training Loss	0.4045
Test Accuracy	93.92 %
Test Precision	88.57 %
Test Recall	25.40 %
Test F1-Score	38.12 %

Interpretation:

The Hybrid Model presents an integrated approach, combining BERT's contextual understanding with LSTM's sequential learning, yielding a balanced strategy for anomaly detection. Key performance indicators include a precise average accuracy of 93.92%. The model maintains a high average precision of 88.57%, effectively minimizing false positives. However, it exhibits room for improvement in capturing true positives, evident in the average recall of 25.40%. The F1-Score, averaging at 38.12%, offers a holistic view of model effectiveness. This versatile hybrid approach is particularly advantageous for tasks requiring temporal dependencies and a nuanced balance between precision and recall. Tailoring the model to specific task nuances and data attributes is recommended for optimal outcomes.

5.2.4.2 Hybrid Model 2 - LSTM + BERT

Upon decreasing the learning rate to 0.0001 while maintaining the aforementioned parameters, the following results were obtained:

Performance Metrics:

Table 5.2.4.2.1 : Hybrid Model 2 Evaluation Matrix

Metric	Value
Training Loss	0.3987
Test Accuracy	93.66 %
Test Precision	85.14 %
Test Recall	22.51 %
Test F1-Score	34.29 %

5.2.4.3 Hybrid Model Summary :

The Hybrid Model synergistically blends BERT's contextual understanding with LSTM's sequential learning, providing a balanced strategy for anomaly detection. It demonstrates a precise average accuracy of 93.92% and maintains a robust average precision of 88.57%, efficiently minimizing false positives. However, improvements are needed in capturing true positives, as indicated by the average recall of 25.40%. The comprehensive view of model effectiveness comes from the average F1-Score at 38.12%. This versatile hybrid approach is particularly valuable for tasks involving temporal dependencies and a delicate balance between precision and recall. Customization based on task specifics and data attributes and hyperparameter tuning is advised for optimal performance.

5.3 Model Evaluation and Summary

The comprehensive evaluation of various models serves as a pivotal step in pinpointing the most suitable anomaly detection solution.

```
df_metrics # Below Table values are in Percentage
```

	Model	Accuracy	Precision	Recall	F1 Score
0	PCA	20.3993	9.141	98.8411	16.7344
1	Isolation Forest	93.0462	100.0	14.0728	24.6734
2	OneClassSVM	0.9647	1.0387	11.9205	1.911
3	LogClustering	97.0590	96.4933	66.0596	78.4275
4	bert_base_model	98.0000	99.0000	80.0000	88.0000
5	bert_tuned_model	98.0000	97.0000	82.0000	88.0000
6	distillbert_tuned_model	98.0000	99.0000	81.0000	88.0000
7	roberta_tuned_model	98.0000	94.0000	77.0000	84.0000
8	albert_tuned_model	98.0000	98.0000	81.0000	88.0000
9	xlnet_tuned_model	98.0000	99.0000	79.0000	87.0000
10	lstm_model	99.0688	94.0643	94.4536	94.2586
11	lstm_tuned_model	99.0889	95.0420	93.6258	94.3286
12	bilstm_model	99.3301	97.6764	93.9570	95.7806
13	hybrid_model	93.9153	88.5667	25.3977	38.1159
14	tuned_hybrid_model	93.6553	85.1444	22.5141	34.2869

Figure 5.3.1 : Model Metrics Evaluation Comparison

- Comparative analysis of all experimented models accuracy score

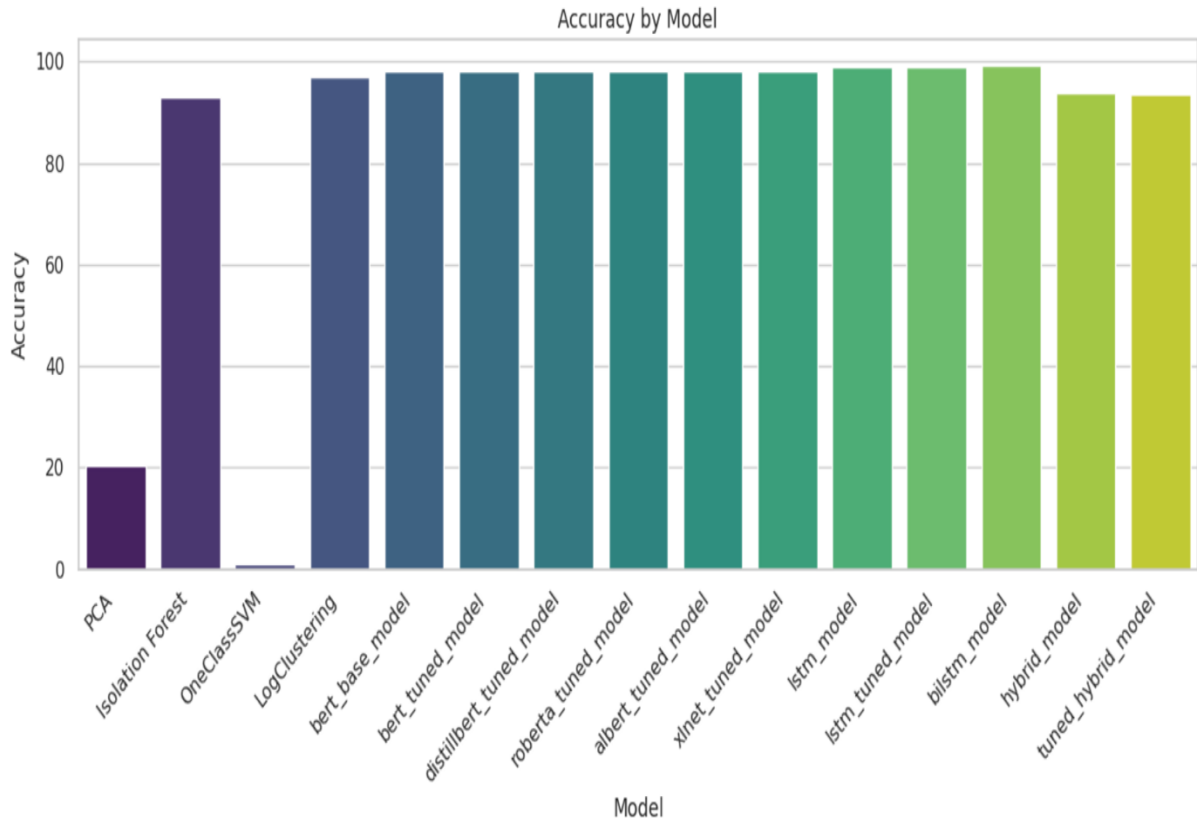


Figure 5.3.2 : Accuracy Metrics Visualization

- Comparative analysis of all experimented models Precision score

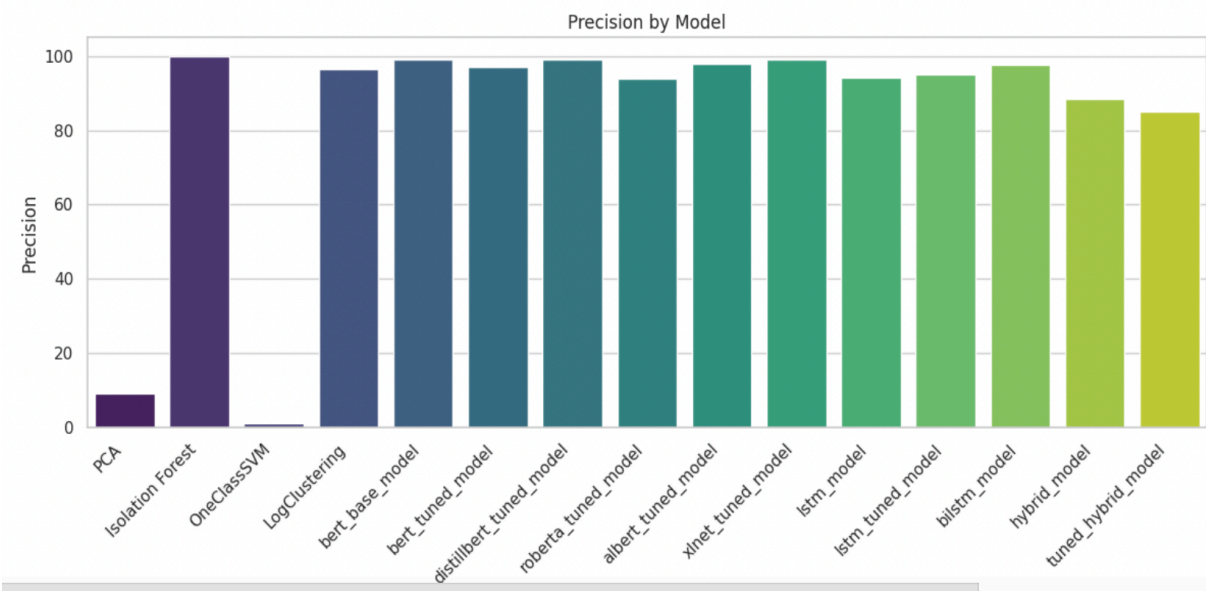


Figure 5.3.3 : Precision Metrics Visualization

- Comparative analysis of all experimented models Recall score

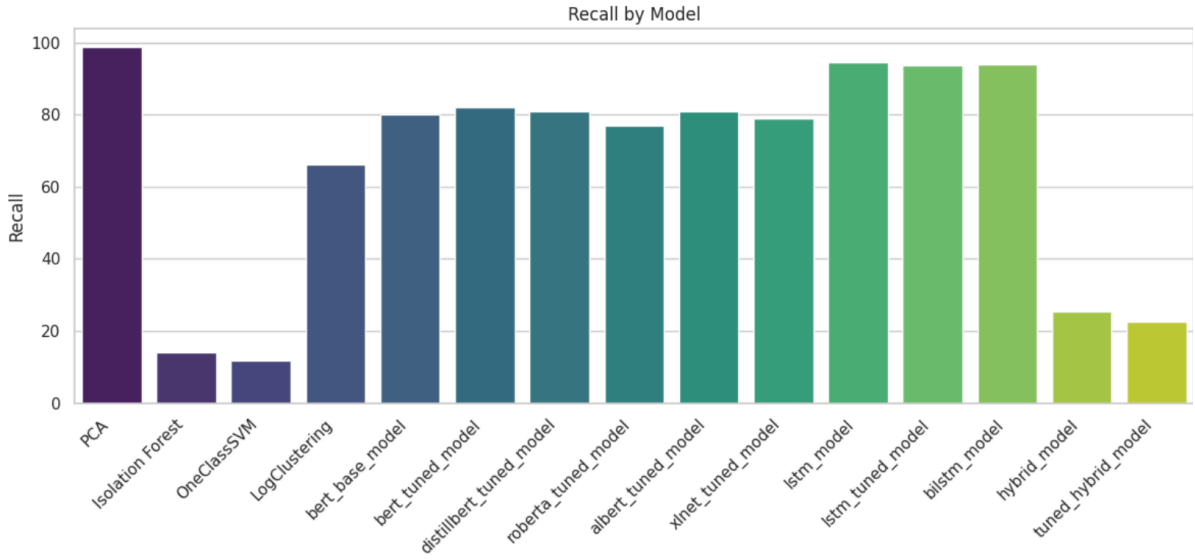


Figure 5.3.4 : Recall Metrics Visualization

- Comparative analysis of all experimented models F1 score

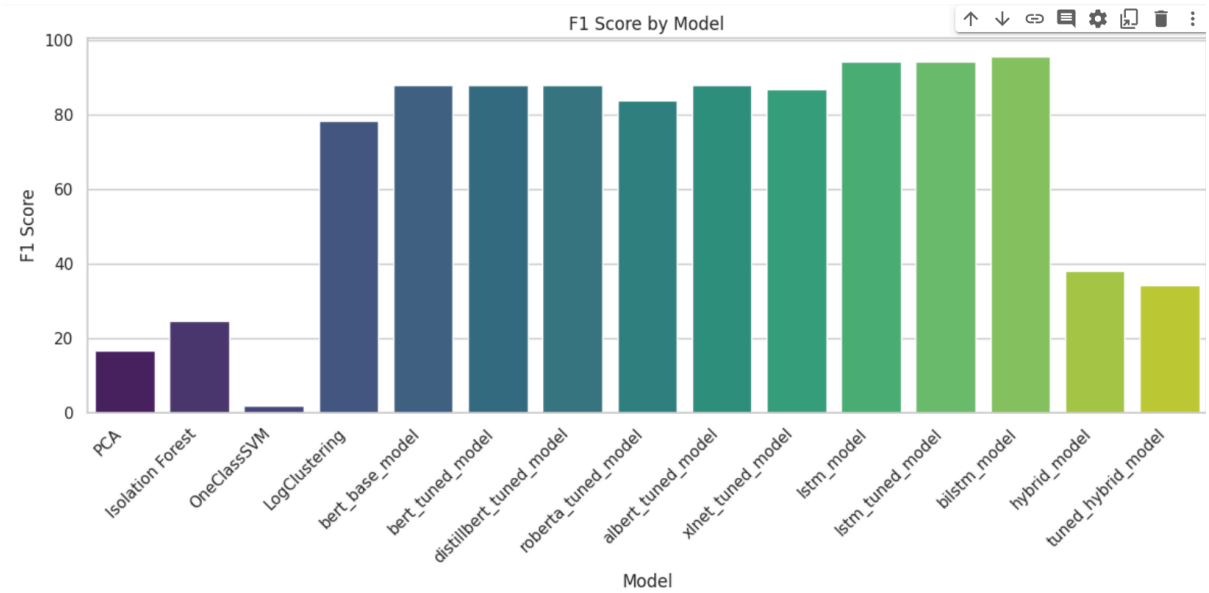


Figure 5.3.5 : F1 - Score Metrics Visualization

The baseline models, encompassing PCA, Isolation Forest, OneClassSVM, and LogClustering, offer valuable insights into their distinct merits and limitations. PCA struggles to achieve precise classification, while Isolation Forest excels in accuracy and precision, yet falters in recall. OneClassSVM, on the other hand, grapples with both accurate classification and effective anomaly detection. In contrast, the LogClustering model emerges as a standout performer with its noteworthy precision and commendable recall, effectively capturing genuine

anomalies. Importantly, LogClustering's performance underscores the delicate balance between precision and recall, emphasizing the significance of optimal parameter selection.

Transformer models, exemplified by Bert_base_uncased, Tuned Bert, DistilBERT, RoBERTa, Albert, and XLNet, exhibit remarkable predictive capabilities in anomaly detection. Among these, the DistilBERT model distinctly shines, demonstrating consistent and robust performance across an array of metrics. Its impressive accuracy, precision, and F1-score designate it as a prime contender for tasks demanding dependable anomaly predictions. The model's distinct strength lies in its precision in identifying anomalies while concurrently minimizing false positives, rendering it highly suitable for real-world scenarios, particularly those with real-time constraints.

LSTM models, noteworthy for their sequential learning capabilities, manifest exceptional performance. The Tuned LSTM model, distinguished by its elevated accuracy, precision, recall, and F1-score, emerges as a front-runner. Likewise, the BiLSTM model showcases remarkable proficiency across various evaluation criteria, illustrating its resilience in diverse scenarios. Both models exhibit significant promise for predictive tasks necessitating an equilibrium between accuracy and comprehensive performance.

The Hybrid Model epitomizes the fusion of BERT's contextual understanding and LSTM's sequential learning, culminating in a harmonious approach to anomaly detection. It attains notable heights in accuracy and precision, deftly mitigating the issue of false positives. While strides have been made in these areas, there exists an opportunity for enhancement in capturing true positives, as denoted by recall metrics. The model's inherent versatility renders it especially conducive to tasks where temporal intricacies are at play, affording a judicious balance between precision and recall. It is incumbent upon users to tailor the model's parameters in tandem with the specifics of the task and unique data attributes, thereby unlocking its maximum potential. Emphasizing the pivotal role of hyperparameter tuning in refining performance further enhances its viability.

Considering the metrics and the model's equilibrium in performance, computational time ,the BiLSTM model emerges as the memory efficient and quintessential choice for predictive endeavours. Sporting exceptional accuracy, precision, recall, and F1-score, it encapsulates the multifaceted capacities essential for real-world applications. This model's acumen in both precise classification and accurate detection of genuine anomalies positions it as an apt solution for tasks necessitating both meticulous accuracy and comprehensive overall performance.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

In this research study, an exhaustive exploration of diverse anomaly detection models was conducted, encompassing baseline models including PCA, Isolation Forest, OneClassSVM, and LogClustering, alongside advanced Transformer-based models like Bert_base_uncased, Tuned Bert, DistilBERT, RoBERTa, Albert, and XLNet, in addition to LSTM models and the Hybrid Model. A comprehensive analysis was undertaken to unveil the unique strengths and weaknesses inherent in each model category, spanning a range of evaluation metrics. While LogClustering showcased exceptional precision and recall, the DistilBERT model consistently displayed robust performance. Similarly, the Tuned LSTM and BiLSTM models exhibited remarkable accuracy and overall efficacy, setting them apart as promising frontrunners. The Hybrid Model's amalgamation of BERT's and LSTM's capabilities rendered it a balanced contender. Remarkably, the BiLSTM model emerged as a standout choice for predictive tasks demanding both accuracy and comprehensive overall performance.

6.2 Discussion and Conclusion

Based on the meticulous evaluation and analysis of the anomaly detection models, tailored recommendations were formulated for each model category:

6.2.1 Baseline Models Recommendation

LogClustering Model: Among the baseline models, the LogClustering model stands out as a dependable choice. Its exceptional precision and recall performance highlight its proficiency in identifying genuine anomalies. The model's capacity to strike a harmonious balance between precise anomaly identification and minimized false positives is particularly notable. Thus, the LogClustering model is recommended for scenarios where precision and recall equilibrium is crucial.

6.2.2 Transformer Models Recommendation

DistilBERT Model: Within the Transformer-based models, the DistilBERT model emerged as a standout performer. Its consistent and robust performance across diverse metrics underscores its reliability. Notably, the model's accuracy, precision, and F1-score excellence positions it as a prime candidate for tasks demanding dependable anomaly predictions. The high precision score underscores its skill in reducing false positives, which is pivotal in applications tolerating minimal errors. Additionally, the commendable recall score signifies its ability to capture true positive instances effectively. Given its balanced performance, the DistilBERT model is strongly recommended for anomaly detection tasks requiring precision, recall, and comprehensive overall effectiveness.

6.2.3 LSTM Models Recommendation

Tuned LSTM Model: Among LSTM models, the Tuned LSTM model emerges as a frontrunner. Its exceptional outcomes in key metrics render it an appealing choice for predictive tasks. Featuring remarkable test accuracy, high precision, and commendable recall, the Tuned LSTM model maintains a delicate equilibrium between accurate classification and genuine anomaly identification. The model's ability to maintain high precision and recall levels underscores its comprehensive performance. Thus, the Tuned LSTM model is recommended for applications necessitating a thorough and accurate approach to anomaly detection.

BiLSTM Model: Similarly, the BiLSTM model displays remarkable proficiency across various evaluation criteria. Its impressive accuracy, substantial precision, and adept recall highlight its robustness across diverse scenarios. Most notably, the BiLSTM model achieves a remarkable average F1-score, signifying its all-encompassing capabilities in anomaly detection. Given its balanced performance, the BiLSTM model is highly recommended for predictive tasks requiring both accuracy and a holistic approach.

6.2.4 Hybrid Models

Hybrid Model (BERT with LSTM): The Hybrid Model, blending BERT's contextual understanding with LSTM's sequential learning, presents a unique strategy for anomaly detection. The model attains notable heights in accuracy and precision, effectively mitigating the challenge of false positives. While there is potential for improvement in capturing true positives, its versatility and equilibrium between precision and recall stand as valuable attributes. The Hybrid Model is recommended for tasks entailing temporal dependencies, where a harmonious fusion of precision and recall is pivotal. Customization based on task specifics and data attributes, coupled with hyperparameter tuning, is advised to unlock its maximum potential.

6.2.5 Recommended Model

In summary, each recommended model brings distinct strengths to the table, catering to diverse anomaly detection requirements. The choice among these models hinges on task-specific characteristics, the desired precision-recall equilibrium, and available resources. Considering the metrics and the model's performance balance, the BiLSTM model emerges as the quintessential choice for predictive tasks. With its exceptional accuracy, precision, recall, and F1-score, it embodies the multifaceted capacities crucial for real-world applications, positioning it as an ideal solution for tasks necessitating both meticulous accuracy and comprehensive overall performance.

6.3 Contribution to Knowledge

This research paper contributes by not only identifying and dissecting the challenges in log classification and anomaly detection but also by providing a comprehensive framework for method and model descriptions, empirical evidence for model performance, and the introduction of a novel model. These contributions collectively advance the knowledge and practical capabilities of the field, empowering researchers and practitioners to make informed decisions and develop effective solutions in the domain of log analysis and anomaly detection.

6.3.1 Challenges in Log Classification and Anomaly Detection

The analysis delved into various obstacles encountered in log classification and anomaly detection. These challenges encompassed aspects like real-time processing, issues related to robustness, dealing with noise, enhancing efficiency, and ensuring the ability to generalize findings. This thorough examination aimed to guide the selection and development of suitable techniques and models.

6.3.2 Explanations of Methods and Models

The study proposes a detailed and accurate description of techniques based on Log Sequences and AI Transformers, along with machine learning models. This comprehensive portrayal includes a well-structured approach that covers activities such as pre-processing steps, model training, parameter tuning, and the use of evaluation metrics to thoroughly gauge the effectiveness of the chosen models.

6.3.3 Comparing the Performance of AI-Based Approaches:

The research conducted a thorough evaluation of various AI-based methods for the detection of anomalies. Through extensive experiments and assessments, these models were scrutinized based on pivotal performance indicators, namely accuracy, precision, recall, and the F1 score. The primary objective was to identify the model with the most exceptional performance in spotting anomalies within system and event logs.

6.3.4 Development and Evaluation of the Proposed Model:

The research undertook the creation and assessment of a suggested log classification model. This model was put into practice and thoroughly scrutinized using appropriate datasets and evaluation criteria. Furthermore, its performance was benchmarked against existing state-of-the-art approaches to gauge its superiority and practical utility.

6.4 Limitations

6.4.1 Acknowledging Dataset Limitations:

The utilization of the BGL dataset prompts a crucial acknowledgment of the dataset's inherent limitations. Due to its specific nature, the conclusions drawn may not necessarily extend

effectively to analogous databases found in diverse domains. It is worth noting, however, that our approach to model selection takes into account this limitation in generalization.

6.4.2 Inherent Limitations of Anomaly Detection Research

The exploration of anomaly detection research, employing both Transformers and LSTM models, brings to light several inherent limitations that warrant meticulous examination. Particularly noteworthy is the token constraint present in BERT models, exemplified by the 512-token limit in BERT-base. This constraint poses significant challenges when dealing with lengthy textual content, potentially resulting in the inadvertent loss of essential information due to truncation or the use of unconventional text handling techniques. Additionally, BERT's memory-intensive architecture during both training and inference introduces concerns related to memory limitations, especially when handling voluminous datasets or introducing substantial modifications. The selection of various BERT variants, ranging from BERT-base to BERT-large, entails a complex balancing act between performance and computational resources. While opting for larger variants may yield improved outcomes, it comes at the cost of heightened hardware requirements.

6.4.3 Model-Specific Limitations

Upon closer analysis of individual models, specific constraints become apparent. For instance, the execution time required for a single epoch of BERT models on GPUs like V100 is substantial, varying from 12.5 minutes for BERT to 20 minutes for Roberta, and even 25-26 minutes for XLNet. Hybrid Models, which integrate BERT with LSTM, demand a time investment of 25 minutes per epoch on V100 GPUs. This hybrid methodology involves adapting BERT outputs to match LSTM's input size using a linear layer, effectively bridging the gap in sequence length. However, this adaptation introduces intricate complexities that could potentially compromise information integrity while adding computational overhead. Moreover, the fine-tuning process of Transformer models entails intricate hyperparameter optimization, which consumes a considerable amount of resources and time.

6.4.4 Interpreting Detected Anomalies and Domain Adaptation Challenges

Despite the robust modelling capabilities of both Transformers and LSTMs, their inherent nature as black-box models hinders the generation of interpretable explanations for identified anomalies. Achieving a delicate balance between the complexity of the model and its interpretability is of paramount importance for practical applications. Moreover, the challenge of domain adaptation arises as anomalies can manifest differently across various datasets, demanding extensive fine-tuning, data augmentation, or domain-specific models to ensure consistent and dependable performance.

6.4.5 Resource-Intensive Considerations and Scalability

Larger Transformer variants, owing to their resource-intensive nature during both training and inference, present a challenge to scalability and widespread adoption, especially for entities with limited access to robust hardware resources. In dynamic environments characterised by the evolution of anomaly patterns, adapting pre-trained models to new anomalies necessitates the implementation of advanced techniques such as continual learning, which might pose challenges in effective execution.

6.4.6 Addressing Limitations and Future Prospects

Considering the substantial size of these models, characterised by millions of parameters, the need for potent hardware resources, including GPUs and high-speed RAMs, becomes apparent. Constraints arising from GPU availability and RAM capacity have implications for training times and the feasibility of utilising larger model versions. As a result, the significance of prudent parameter selection, effective adaptation strategies, and a comprehensive understanding of model behaviour cannot be overstated in ensuring the success of anomaly detection initiatives.

6.5 Future Recommendations

Enhanced Generalization: Given the dataset's inherent limitations, future research should explore strategies to enhance model generalization across various domains. Techniques like data augmentation and transfer learning could broaden the models' applicability.

Optimized Memory Management: Addressing the memory-intensive nature of BERT models with innovative memory-efficient techniques would enhance scalability and accessibility to these models.

Interpretable Insights: Developing methods to interpret complex models like Transformers and LSTMs will enhance practical utility and trust by providing insights into anomaly detection decisions.

Dynamic Domain Adaptation: As anomalies evolve over time, models should adapt to new patterns. Techniques like continual learning can facilitate effective adaptation, ensuring accurate detection of evolving anomalies.

Accessible Hardware: Overcoming challenges posed by resource-intensive Transformer models will promote scalability and democratize access, benefiting researchers and organizations with varying levels of resources.

Advanced Hyperparameter Tuning: Employing advanced hyperparameter tuning techniques will fine-tune model performance, adaptability, and efficacy across diverse tasks and datasets, ensuring optimal outcomes in different scenarios.

References :

van der Aa, H., Rebmann, A. and Leopold, H., (2021) Natural language-based detection of semantic execution anomalies in event logs. *Information Systems*, 102.

Bhanage, D.A., Pawar, A.V. and Kotecha, K., (2021a) *IT Infrastructure Anomaly Detection and Failure Handling: A Systematic Literature Review Focusing on Datasets, Log Preprocessing, Machine Deep Learning Approaches and Automated Tool*. *IEEE Access*, .

Bhanage, D.A., Pawar, A.V. and Kotecha, K., (2021b) *IT Infrastructure Anomaly Detection and Failure Handling: A Systematic Literature Review Focusing on Datasets, Log Preprocessing, Machine Deep Learning Approaches and Automated Tool*. *IEEE Access*, .

Chen, S. and Liao, H., (2022a) BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model. *Applied Artificial Intelligence*, 361.

Chen, S. and Liao, H., (2022b) BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model. *Applied Artificial Intelligence*, 361.

Choi, H., Kim, J., Joe, S. and Gwon, Y., (2020) Evaluation of BERT and Albert sentence embedding performance on downstream NLP tasks. In: *Proceedings - International Conference on Pattern Recognition*. Institute of Electrical and Electronics Engineers Inc., pp.5482–5487.

Devlin, J., Chang, M.-W., Lee, K., Google, K.T. and Language, A.I., (n.d.) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. [online] Available at: https://github.com/tensorflow/tensor2tensor.

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H. and Smith, N., (2020) Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. [online] Available at: http://arxiv.org/abs/2002.06305.

Duan, X., Ying, S., Yuan, W., Cheng, H. and Yin, X., (2021a) QLLog: A log anomaly detection method based on Q-learning algorithm. *Information Processing and Management*, 583.

Duan, X., Ying, S., Yuan, W., Cheng, H. and Yin, X., (2021b) QLLog: A log anomaly detection method based on Q-learning algorithm. *Information Processing and Management*, 583.

Farzad, A. and Gulliver, T.A., (2022) Log message anomaly detection with fuzzy C-means and MLP. *Applied Intelligence*, 5215, pp.17708–17717.

Fu, Y., Yan, M., Xu, Z., Xia, X., Zhang, X. and Yang, D., (2023) An empirical study of the impact of log parsers on the performance of log-based anomaly detection. *Empirical Software Engineering*, 281.

Guo, H., Yuan, S. and Wu, X., (2021) LogBERT: Log Anomaly Detection via BERT. *2021 International Joint Conference on Neural Networks (IJCNN)*, [online] null, pp.1–8. Available at: https://www.semanticscholar.org/paper/38c0543aa72b68d1ded4237e8cc5333b165ea249.

He, P., Zhu, J., Zheng, Z. and Lyu, M.R., (2017) Drain: An Online Log Parsing Approach with Fixed Depth Tree. In: *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*. Institute of Electrical and Electronics Engineers Inc., pp.33–40.

Hela, S., Amel, B. and Badran, R., (2018) Early anomaly detection in smart home: A causal association rule-based approach. *Artificial Intelligence in Medicine*, 91, pp.57–71.

Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., Yang, H. and Luan, Z., (2020a) HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log. *IEEE Transactions on Network and Service Management*, 174, pp.2064–2076.

Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., Yang, H. and Luan, Z., (2020b) HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log. *IEEE Transactions on Network and Service Management*, 174, pp.2064–2076.

Jasra, S.K., Valentino, G., Muscat, A. and Camilleri, R., (2022) Hybrid Machine Learning–Statistical Method for Anomaly Detection in Flight Data. *Applied Sciences (Switzerland)*, 1220.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F. and Liu, Q., (2019) TinyBERT: Distilling BERT for Natural Language Understanding. [online] Available at: http://arxiv.org/abs/1909.10351.

Kim, G.Y., Lim, S.M. and Euom, I.C., (2022) A Study on Performance Metrics for Anomaly Detection Based on Industrial Control System Operation Data. *Electronics (Switzerland)*, 118.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R., (2019) ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. [online] Available at: http://arxiv.org/abs/1909.11942.

Malaiya, R.K., Kwon, D., Suh, S.C., Kim, H., Kim, I. and Kim, J., (2019) An Empirical Evaluation of Deep Learning for Network Anomaly Detection. *IEEE Access*, 7, pp.140806–140817.

Moya, H., (2021) *Machine learning model optimization with hyper-parameter tuning approach*. [online] Available at: https://www.researchgate.net/publication/354495368.
Nassif, A.B., Talib, M.A., Nasir, Q. and Dakalbab, F.M., (2021) *Machine Learning for Anomaly Detection: A Systematic Review*. *IEEE Access*, .

Naudé, M., Adebayo, K.J. and Nanda, R., (2022) A machine learning approach to detecting fraudulent job types. *AI and Society*.

Nooribakhsh, M. and Mollamotalebi, M., (2020) *A review on statistical approaches for anomaly detection in DDoS attacks*. *Information Security Journal*, .
Oliner, A. and Stearley, J., (2007) *What Supercomputers Say: A Study of Five System Logs*.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T., (2019a) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. [online] Available at: http://arxiv.org/abs/1910.01108.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T., (2019b) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. [online] Available at: http://arxiv.org/abs/1910.01108.

Shao, Y., Zhang, W., Liu, P., Huyue, R., Tang, R., Yin, Q. and Li, Q., (2022) Log Anomaly Detection method based on BERT model optimization. In: *2022 7th International Conference on Cloud Computing and Big Data Analytics, ICCCBDA 2022*. Institute of Electrical and Electronics Engineers Inc., pp.161–166.

Wang, X., Cao, Q., Wang, Q., Cao, Z., Zhang, X. and Wang, P., (2022) Robust log anomaly detection based on contrastive learning and multi-scale MASS. *Journal of Supercomputing*, 7816, pp.17491–17512.

Zhao, Z., Xu, C. and Li, B., (n.d.) A LSTM-Based Anomaly Detection Model for Log Analysis. [online] Available at: https://doi.org/10.1007/s11265-021-01644-4.

Zhao, Z., Xu, C. and Li, B., (n.d.) A LSTM-Based Anomaly Detection Model for Log Analysis. [online] Available at: https://doi.org/10.1007/s11265-021-01644-4.

LEVERAGING NLP TECHNIQUES AND TRANSFORMER-BASED AI MODELS FOR IMPROVED LOG ANOMALY

SHASHANK BHATNAGAR

Research Proposal

MAY 2023

Abstract

This research study aims to propose a transformer-based method to classify anomalies from system logs that can process logs in real-time and address the challenges associated with extracting meaningful information from unstructured logs. The study also aims to propose a log sequence based anomaly detection method that fully considers robustness issues caused by updating log message templates, such as concept drift, noise problems, and the challenges associated with processing a huge volume of logs produced by systems. The proposed method aims to detect anomalies in system log files automatically, requiring minimal manual marking of regular expressions by operation engineers, which can be an inefficient process when dealing with a huge volume of logs. The study will evaluate the effectiveness and generalizability of the proposed method using publicly available datasets. The study's findings can have a significant impact on the fields of cybersecurity and system monitoring, enabling organizations to detect abnormal behaviour in system logs, proactively identify and mitigate security threats, reduce downtime, and enhance system reliability. The study employs a comprehensive approach, including data loading, pre-processing, feature engineering, and model building, using multiple models such as log sequence models, transformer-based log classification, and fine-tuned language models.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ALBERT	A Lite Robustly Optimized BERT Pre-training Approach
BERT	Bidirectional Encoder Representations from Transformers
BGL	BlueGene/L
BoW	Bag of Words
CV	Cross Validation
EDA	Exploratory Data Analysis
LSTM	Long Short-Term Memory
NN	Neural Network
Q Learning	Quality Learning
roBERTa	Robustly Optimized BERT Pre-training Approach
ROC	Receiver operating characteristic
RNN	Recurrent neural network
TF-IDF	Term Frequency - Inverse Document Frequency
XLNet	Generalized Auto regressor Extension of Transformer Extra Long Mode

1. Background

Anomaly detection is an essential task in cybersecurity and system monitoring, which involves identifying unusual patterns in data that indicate potential security threats, system failures, or other abnormal events. Implementing anomaly detection techniques can help organizations detect potential security threats and system failures, improve the reliability and performance of their systems, prevent system downtime, enhance operational efficiency, and reduce costs associated with identifying performance issues and inefficiencies. Additionally, compliance with regulatory requirements can be ensured, avoiding legal issues and penalties for organizations. Overall, anomaly detection can help organizations improve their cybersecurity posture, increase their operational efficiency, and comply with regulatory requirements.

However, traditional methods of anomaly detection can be inefficient and time-consuming, necessitating the need for machine learning-based approaches to detect anomalies from unstructured log data. Despite this, machine learning-based methods can lack robustness and treat normal noise logs as abnormal, leading to poor performance. To address these challenges, the study proposes a log sequence LSTM (Zhao et al., n.d.) based anomaly detection method that fully considers robustness issues by updating the log message template, as well as concept drift and noise problems. The proposed method aims to detect anomalies in system log files automatically, requiring minimal manual marking of regular expressions by operation engineers, and will be evaluated using publicly available datasets BGL(Oliner and Stearley, 2007).

The proposed method aims to overcome the challenges associated with processing a large number of logs and proposes a transformer-based (for e.g. BERT (Chen and Liao, 2022), roBERTa (Naudé et al., 2022), XLNET (Naudé et al., 2022) etc.) method that can process logs in real-time, classify anomalies, and enhance system reliability and security. However, the proposed method has potential limitations, including the requirement for significant computational resources and the unavailability of data due to sensitive log and secure information. Nevertheless, the study suggests that the evaluation could be extended to other log datasets to validate the effectiveness and generalizability of the proposed method.

2. Related Research

The primary focus of this study is to classify anomalies from system logs, which can be challenging due to their unstructured nature and the sheer amount of data they contain. To address this issue, the study proposes a transformer method(for e.g. BERT (Chen and Liao, 2022), roBERTa (Naudé et al., 2022), XLNET (Naudé et al., 2022) etc.) for log classification that can process logs optimally in real-time. While machine learning algorithms for anomaly detection can achieve high accuracy, they often lack robustness, which is why the study aims to propose a log sequence LSTM (Zhao et al., n.d.) based anomaly detection method that fully considers the robustness issues by updating the log message template. Moreover, the study aims to address the drawbacks of recent log anomaly detection studies, such as concept drift and noise problems that cause instability in data and unexpected misjudgement, causing performance degradation.

The study also emphasizes the challenges associated with processing a large number of logs produced by security devices, which is beyond the processing speed of human beings. Therefore, the proposed method aims to detect anomalies in system log files automatically, requiring minimal manual marking of regular expressions by operation engineers, making it an efficient process when dealing with a large number of logs. However, the proposed method may require a lot of computational resources, which may not be practical for some real-world applications. Additionally, the unavailability of data due to sensitive log and secure information may limit the evaluation of the proposed method. Nevertheless, the study suggests that the evaluation could be extended to other log datasets to further validate the effectiveness and generalisability of the proposed method. In conclusion, the study aims to propose a transformer method that can classify anomalies in system logs in real-time while addressing the robustness issues, drawbacks of recent log anomaly detection studies, and challenges associated with processing a large number of logs produced by security devices, making the log parsing method more efficient and automated. There is much research around among various transformer BERT (Chen and Liao, 2022), roBERTa (Naudé et al., 2022), ALBERT (Lan et al., 2019; Choi et al., 2020) , DistilBERT(Jiao et al., 2019; Sanh et al., 2019) ,log sequence models(Zhao et al., n.d.) and QL-Log/Q Learning models (Duan et al., 2021) in the last few years and this research has explored few important methods.

Table 1 : Model Methods with approach and Limitations

Method	Approach	Limitations	Reference
RoBERTa	RoBERTa is a transformer-based language model similar to BERT but trained on a larger corpus of data using improved training techniques.	It lacks external knowledge sources, making it less effective for domain-specific tasks	(Naudé et al., 2022)

Table 1 : Model Methods with approach and Limitations

Method	Approach	Limitations	Reference
ALBERT	It reduces model size and training time while maintaining the performance	High computational cost and requirement of large amounts of data for pre-training also it has some limitations in domain specific tasks	(Lan et al., 2019; Choi et al., 2020)
DistilBERT	DistilBERT is a faster and smaller compressed version of BERT, which maintains most of its accuracy and performance on natural language processing tasks.	DistilBERT, being a compressed and faster version of BERT, may have lower accuracy and perform sub-optimally on tasks that require high level of precision and detail.	DistilBERT(Jiao et al., 2019; Sanh et al., 2019)
XLNet	XLNet is a neural language model that utilizes a permutation-based approach to generate context-aware representations of input text, outperforming BERT on many NLP tasks.	XLNet has high computational demands, slow training pace, and intricate structure, making it challenging to fine-tune for downstream tasks.	(Naudé et al., 2022)
Log Sequence Model LSTM	Log Sequence Model LSTM is a deep learning method that utilizes LSTM neural networks to analyze system logs in sequence, identifying patterns in the log data to detect anomalies.	The Log Sequence Model LSTM has limitations such as high computational demands, hyperparameter selection challenges, and managing large amounts of unstructured data	(Zhao et al., n.d.)
BERT	BERT is a deep learning model that uses a bidirectional transformer to pre-train natural language processing tasks and achieve state-of-the-art results.	Need for large amounts of labeled data, high computational costs	(Chen and Liao, 2022)
BERT-Log	BERT-Log algorithm uses a pre-trained language model called BERT to learn the semantic representation of normal and anomalous logs	Need for large amounts of labeled data, high computational costs	

Table 1 : Model Methods with approach and Limitations			
Method	**Approach**	**Limitations**	**Reference**
QL-Log/Q Learning	Q-learning is a type of machine learning algorithm that enables agents to learn the best actions to take in a particular environment. It does this by updating the expected rewards an agent can receive by performing certain actions in different states, which are called Q-values.	The limitations of Q-learning algorithms include the problem of exploration vs. exploitation, the curse of dimensionality, the requirement of a complete and accurate model, and the difficulty of handling continuous state and action spaces.	(Duan et al., 2021)
Precision, Recall, and F1-score	The metrics Precision, Recall, and F1-score are used for evaluating classification models, where Precision is the true positive rate of positive predictions, Recall is the true positive rate of actual positives, and F1-score is the balanced harmonic mean of both metrics.	The limitations include their unsuitability for models that prioritize one metric over the other, their inability to handle imbalanced datasets, and their failure to capture the intricacies of a model's performance on specific data subsets.	(Chen and Liao, 2022),
ROC AUC	ROC AUC is a metric for evaluating classification model performance that measures the area under the curve of the receiver operating characteristic	The ROC AUC curve has limitations such as limited handling of imbalanced datasets, insensitivity to changes in data distribution, and lack model performance explanation	(Chen and Liao, 2022),
Accuracy	Accuracy refers to the ratio of correct predictions made by a model to the total number of predictions.	One limitation of accuracy is its potential to be misleading for imbalanced datasets, as a high accuracy score may not necessarily indicate good performance on the minority class due to its significantly lower representation in the dataset	(Chen and Liao, 2022),

3. Research Questions

Following research questions are formed considering the literature review done in the field of anomaly detection

- How can a new approach using transformer method be identified to classify anomalies from system logs in real-time and address the issue of extracting meaningful information from unstructured logs?

- What are the drawbacks of recent log anomaly detection studies, such as concept drift, noise problems, and how can they be addressed to improve performance and reduce data instability and abnormal misjudgment?

- What are the potential limitations of the proposed method, such as requiring a lot of computational resources and limited evaluation due to sensitive log and secure information, and how can they be overcome or mitigated to improve the effectiveness and generalizability of the proposed method?

4. Aim & Objectives

The main aim of this research on log anomaly detection is to propose a model which can detect anomalies in system and event logs and can indicate system malfunctions, security breaches, or other unexpected behaviour. Identifying anomalies using the well-studied robust models allow us to improve system performance, increase reliability and availability, enhance security, reduce maintenance costs, and prevent potential failures or downtime.

The research objectives stated below are formulated based on the aim of the study are as follows:

- To analyze various issues related to log classification and anomaly detection, including real-time processing, robustness issues, noise problems, efficiency, and generalizability.

- To suggest precise and complete description of the Log Sequence based / AI Transformer based techniques and machine learning models that will be utilized, along with a well-structured approach to evaluate their efficacy.

- To compare the different AI based techniques to identify the best performing model in identifying the anomalies.

- To develop and evaluate the performance of the proposed log classification model.

5. Significance of the Study

The expected outcome of this study is to identify the artificial intelligence (AI) based anomaly detection model which will be significant in fields such as cybersecurity, system monitoring, predictive maintenance, and fraud detection, where organizations can identify unusual patterns in data and alert them to potential threats or issues. By detecting such anomalies proactively, organizations can minimize the risk of system downtime, data breaches, or financial losses and improve their operational efficiency and reliability. In cybersecurity, anomaly detection models monitor network traffic, log files, and other data sources in real-time, detecting patterns that deviate from normal behaviour and alerting security teams. In system monitoring, these models analyse system logs and telemetry data to detect patterns that may indicate impending failures, allowing organizations to take preventative measures and minimize downtime. In predictive maintenance, anomaly detection models analyse sensor data to identify equipment that requires maintenance or repair, while in fraud detection, these models analyse transactional data to identify unusual behaviour that may indicate fraudulent activity. By detecting anomalies early, organizations can take corrective action and improve their overall efficiency and reliability.

6. Scope of the Study

Considering the established deadlines for conducting this research, the scope of the study is limited as follows:

6.1 In scope

The proposed research aims to develop and evaluate the performance of anomaly detection models using transformers and log sequence models to detect abnormal behavior in system logs in real-time. The research will compare the performance of different models in terms of accuracy(Chen and Liao, 2022), precision(Chen and Liao, 2022), recall(Chen and Liao, 2022), F1-score(Chen and Liao, 2022), and AUC-ROC(Chen and Liao, 2022). The study will also investigate the robustness issues, noise problems, efficiency, and generalizability of the proposed models.

6.2 Out of scope

The proposed research will not focus on investigating new methods of anomaly detection, developing new technologies to enhance the performance of the anomaly detection models, evaluating the quality and usability of the system logs, conducting a detailed analysis of the impact of the anomalies on the system, identifying ways to improve the performance of the system, and enhancing the cybersecurity posture of the organization.

6.3 Reason for defining the scope

Defining the scope is crucial in research as it clarifies the research's objectives and what is not relevant to the study, as with the proposed research on anomaly detection models, defining in-scope activities helps to ensure that the research stays focused on developing and evaluating the models' performance, while defining out-of-scope activities helps prevent distractions and irrelevant investigations.

7. Research Methodology

The sequential flow of steps for model building will be provided in this section

7.1 Introduction

System logs are critical in monitoring and troubleshooting computer systems, but their unstructured nature and vast amount of data make them challenging to analyze. Machine learning algorithms have shown promise in detecting anomalies, but their robustness issues can lead to poor performance and inaccurate results. To address these challenges, this study aims to propose a transformer method for log classification that addresses the limitations of existing log anomaly detection methods and the robustness issues by updating log message templates. The proposed method also includes a more efficient and automated log parsing method to enable faster and more effective troubleshooting and security monitoring. The effectiveness and generalizability of the proposed method will be evaluated by testing it on various log datasets and comparing its performance with existing methods.

This study focuses on using various data pre-processing techniques to prepare system log data for anomaly detection. The data preparation process involves exploratory data analysis, data cleaning, and outlier analysis, along with feature engineering techniques like converting categorical variables into numerical values and standardization. After preparing the data, the study explores different machine learning models for log classification, such as developing a log sequence model and comparing transformer-based architectures like BERT(Chen and Liao, 2022) and RoBERTa(Naudé et al., 2022).The best-performing model will be fine-tuned using a fully connected neural network to classify normal and anomalous log sequences. Finally, the performance of the models will be evaluated on a test set using various metrics like Confusion Matrix(Chen and Liao, 2022), Accuracy(Chen and Liao, 2022), Precision(Chen and Liao, 2022), Recall(Chen and Liao, 2022), F1-score,(Chen and Liao, 2022) and AUC-ROC(Chen and Liao, 2022). The goal is to propose a comprehensive and effective framework for log classification that can overcome the limitations of existing methods through extensive experimentation and evaluation.

7.2 Data Description

The BGL(Oliner and Stearley, 2007) dataset is a publicly available collection of system logs that were gathered from a BlueGene/L(Oliner and Stearley, 2007) supercomputer located at the Lawrence Livermore National Labs in Livermore, California (Oliner and Stearley, 2007). The supercomputer had 131,072 processors and 32,768GB memory. The dataset includes both alert and non-alert messages, which are distinguished by alert category tags.

The first column of the log file indicates whether a message is an alert or non-alert message, with "-" representing non-alert messages. This labelling information makes the dataset useful for research on alert detection and prediction. The BGL dataset has been utilized in various studies related to log parsing, anomaly detection, and failure prediction.

Table 2 : Data Description - Alert Categories , Counts and Sample Message Body

The system names are accompanied by the total number of alerts. The alert category is indicated by "Cat.", and there are three types of categories: H (Hardware), S (Software), and I (Indeterminate). Alerts categorized as Indeterminate may originate from either hardware or software or have an unknown cause**.**

Alert Type/Cat.	Raw	Example Message Body (Anonymized)
BG/L	3,48,460	-
H / KERNDTLB	1,52,734	data TLB error interrupt
H / KERNSTOR	63,491	data storage interrupt
S / APPSEV	49,651	ciod: Error reading message prefix after LOGIN MESSAGE on CioStream [...]
S / KERNMNTF	31,531	Lustre mount FAILED : bglio11 : block id : location
S / KERNTERM	23,338	rts: kernel terminated for reason 1004rts: bad message header: [...]
S / KERNREC	6145	Error receiving packet on tree network, expecting type 57 instead of [...]
S / APPREAD	5983	ciod: failed to read message prefix on control stream [...]

S / KERNRTSP	3983	rts panic! - stopping execution
S / APPRES	2370	ciod: Error reading message prefix after LOAD MESSAGE on CioStream [...]
I / APPUNAV	2048	ciod: Error creating node map from file [...]
I / Others	7186	machine check interrupt

Table 3 : Data Description - Number of Messages Per Facility

Facility	**No. Of Messages**
KERNEL	4324651
APP	228536
DISCOVERY	97172
MMCS	88930
HARDWARE	5148
MONITOR	1681
LINKCARD	1170
FATAL	306
CMCS	211
BGLMASTER	145
SERV_NET	3

Table 4 - No of Messages per Severity Level

Severity	**No. Of Messages**
INFO	3735813
FATAL	855195
ERROR	112355
WARNING	23357
SEVERE	19213
FAILURE	1714

7.3 Data Pre-processing

Exploratory Data Analysis (EDA) involves using visual and quantitative methods to analyze and summarize complex data sets to uncover underlying patterns, trends, and relationships.

- **Data Cleaning**: This involves identifying and correcting or removing incomplete, inaccurate, irrelevant, or inconsistent data from the dataset.
  - Preprocess log data using a log parser to extract log keys.
  - Define a log sequence as an ordered sequence of log keys.
  - Label the log sequences as normal or anomalous

- **Duplicate Records and Missing Value Analysis**: Identifying and handling duplicate records and missing values to ensure that the dataset is accurate and complete.

- **Skewness Analysis**: This involves detecting skewed data points that may affect the performance of the model.

- **Univariate Analysis - Data Distribution**: Examining the distribution of the dataset, including measures of central tendency, and spread, to gain insights into the data and identify patterns.

- **Bi-variate Analysis with Respect to Target Variable**: Analyzing the relationship between the features and the target variable to identify the most important variables for the model.

- **Multivariate Analysis Using Correlation Metrics to Check Multicollinearity**: This involves analyzing the correlation between features to identify potential multicollinearity issues and to select the most important variables for the model.

- **Feature Engineering**: Creating new features from existing ones to enhance the predictive power of the model.

- **Convert Categorical Variable to Numerical Value**: Converting categorical variables into numerical values to make them compatible with the algorithms used for the analysis.

- **Standardization and Scaling**: This involves transforming the data to a standard scale to ensure that all variables are treated equally, and the model performs optimally.

- **Sentence Tokenization:** Split text into sentences to identify boundaries and treat them as separate units.

- **Word Tokenization:** Split sentences into words or tokens for further analysis and manipulation.

- **Removing Stop Words:** Eliminate common, insignificant words to focus on meaningful content during summarization.

- **Text Normalization:** Clean text by removing special characters, punctuation, and numbers, while ensuring consistency in case, stemming, and lemmatization.

- **Handling Outliers:** Address outliers, such as extremely long or short sentences, which can impact summary quality.

7.4 Feature Extraction

Convert preprocessed text into numerical representations (e.g., word embeddings or TF-IDF(Bhanage et al., 2021a) vectors) for machine learning models to process and understand the data

- The Bag-of-Words (BoW) (Bhanage et al., 2021a)Model treats a text as a collection of words, independent of grammar and word order, and assigns each word as a separate feature.

- TF-IDF(Bhanage et al., 2021a) assigns greater importance to terms that are common within a document but rare across other documents, emphasizing the significance of unique terms in distinguishing documents.

7.5 Model Building

In the process of building a model, the first step involves partitioning the data into training, testing, and validation sets. This partitioning is crucial as it allows for the evaluation of the model's performance and its ability to generalize to new data. The training set is utilized to train the model on a substantial portion of the data, enabling it to capture the underlying patterns and relationships present. The testing set is then employed to assess the model's performance on unseen data, providing an estimation of its accuracy and effectiveness. Lastly, the validation set is used to fine-tune the model's hyperparameters and make necessary adjustments prior to finalizing the model. By dividing the data into these distinct sets, the model undergoes training, evaluation, and adjustment on separate data samples, ensuring a comprehensive assessment of its performance while mitigating the risk of overfitting.

Below models will be explored and based on the study we will propose the best performing model

Log Sequence Model:

A log sequence model, like LSTM(Zhao et al., n.d.) (Long Short-Term Memory), is an RNN(Zhao et al., n.d.) variant specialized in capturing sequential log data's temporal patterns and dependencies. We will develop a log sequence model (e.g., an LSTM(Zhao et al., n.d.) or Transformer model) to learn the sequence patterns in the log data and detect anomalies. Train the log sequence model on the labeled log sequences and evaluate its performance using cross-validation. Select the best-performing log sequence model based on its classification accuracy.

Transformer-based Log Classification

Transformer-based log classification models leverage transformer architectures to process sequential data and capture long-range dependencies, enabling their high effectiveness in log classification tasks. We will Investigate and compare different transformer-based architectures such as BERT(Chen and Liao, 2022), RoBERTa(Naudé et al., 2022), ALBERT(Lan et al., 2019; Choi et al., 2020), DistilBERT(Jiao et al., 2019; Sanh et al., 2019) and XLNet(Naudé et al., 2022) etc. for log classification.

- **BERT**(Chen and Liao, 2022), which extensively pre-trains on large amounts of unlabelled text data and exhibits exceptional performance across diverse natural language processing tasks, including log classification.

- **RoBERTa**, (Naudé et al., 2022) an optimized variant of BERT, enhances performance by adjusting training methods and hyperparameters, building upon the accomplishments of BERT and refining its capabilities.

- **ALBERT**(Lan et al., 2019; Choi et al., 2020), on the other hand, provides a compact alternative to BERT, maintaining high performance while reducing the model size and computational requirements.

- **DistilBERT**(Jiao et al., 2019; Sanh et al., 2019), as the name suggests, distils the knowledge of BERT into a smaller model that achieves comparable performance with significant reductions in size and training time, making it efficient for log classification and other text classification tasks.

- **XLNet** (Naudé et al., 2022), a transformer-based model with a unique permutation-based training approach, captures bidirectional dependencies without relying on masking, achieving strong performance in log classification and other text classification tasks by effectively capturing intricate relationships within log data.

As a part of study will train each transformer model on the log data and evaluate their performance using cross-validation (k-fold cross validation) and will the best-performing transformer model based on its classification accuracy.

Using a Pre-trained Language Model

Fine-tune the selected pre-trained language model (e.g., BERT(Chen and Liao, 2022), roBERTa(Naudé et al., 2022), XLNET (Naudé et al., 2022)) on the log data to learn the semantic representation of normal and anomalous logs.Evaluate the fine-tuned language model on the validation set to select the best-performing model.

Fine-tuning the BERT Model

Use the best-performing pre-trained language model (e.g., BERT(Ryciak et al., 2022)) to detect abnormal logs.

Fine-tune the BERT(Bhanage et al., 2021b) model using a fully connected neural network to classify normal and anomalous log sequences. Evaluate the fine-tuned BERT(Huang et al., 2020) model on the validation set to select the best-performing model.

Fine-tuning the Log Sequence Model

Use the best-performing LSTM(Zhao et al., n.d.; Wang et al., 2022) model to detect abnormal logs. Fine-tune the LSTM(Zhao et al., n.d.) model using a fully connected neural network to classify normal and anomalous log sequences. Evaluate the fine-tuned LSTM(Zhao et al., n.d.) model on the validation set to select the best-performing model.

7.6 Model Controls – Hyperparameter Tuning

Model control hyperparameter tuning details for the transformer-based log classification are as follows

- Learning Rate will help to optimize the step size for parameter updates during training.
- Dropout Rate can we use Fine-tune the probability of dropping out units for regularization.
- Batch Size will determine the number of training examples processed in each iteration.
- Number of Layers can find the right balance of layers for capturing complex patterns.
- Hidden Dimension Size can tune the dimensionality of hidden states for better representation.
- Attention Mechanism will explore hyperparameters related to attention heads and dropout.
- Regularization can help to prevent overfitting and improve generalization.
- Optimization techniques may help to choose the optimal optimization algorithm and related hyperparameters

In addition to Learning Rate, Dropout Rate, No of Layers, Batch Size, Regularization and Optimization Model control hyperparameter tuning details for the log-sequence based LSTM (Zhao et al., n.d.) model are as follows

- Number of LSTM Units, Determine the LSTM unit count per layer for capturing temporal dependencies effectively.
- Sequence Length, Set the input log sequence length for LSTM training.
- Initialization Method, Explore different weight initialization techniques (e.g., uniform, normal distribution) for LSTM weights.

We can use below algorithms find the best combinations of hyperparameters for tuning

Grid Search: Employ grid search to systematically assess various hyperparameter combinations for transformers and LSTM (Zhao et al., n.d.) models in anomaly detection.

Random Search: Utilize random search to randomly select hyperparameter configurations for transformers and LSTM (Zhao et al., n.d.) models, facilitating a wider exploration of the hyperparameter space.

Bayesian Optimization: Apply Bayesian optimization techniques to effectively search and optimize hyperparameters for transformers and LSTM models, leveraging probabilistic models and previous evaluations to guide the search process efficiently

7.7 Model Evaluation

Compare the performance of the Transformers-based model, and Log Sequence based with other anomaly detection methods such as logistic regression, decision trees, and Naive Bayes.

We will use performance metrics such as Confusion Matrix, Accuracy, Precision, Recall, F1-score, and AUC-ROC(Bhanage et al., 2021a; Chen and Liao, 2022) to compare the performance of different methods.

Accuracy(Chen and Liao, 2022): This measures the proportion of correctly classified observations. It is the simplest evaluation metric and is calculated as the number of correct predictions divided by the total number of predictions.

Confusion Matrix(Chen and Liao, 2022): It is a representation of a 2x2 table of actual vs predicted class in binary classification. It includes four parameters: true positives, false positives, true negatives, and false negatives.

Precision, Recall, Sensitivity, and Specificity(Chen and Liao, 2022): These are log established evaluation metrics based on business requirements. Sensitivity and recall are similar.

ROC Curve and AUC(Chen and Liao, 2022): The ROC (Bhanage et al., 2021a; Chen and Liao, 2022) curve is used to understand the strength of the model by evaluating its performance at all classification thresholds. The Area Under the Curve (AUC) (Bhanage et al., 2021a; Chen and Liao, 2022) summarizes the overall performance of the classifier.

Best Threshold: Higher True Positive Rate and lower False Positive Rate(misclassifications) will lead to the best threshold.

Choosing Precision or Recall: Depending on the problem statement, the study will identify which measure is more important: high precision or high recall.

F1-Score(Chen and Liao, 2022): The F1-score is the harmonic mean of precision and recall values for a classification problem where the requirement is to have the best precision and recall at the same time. It is calculated as 2*(precision*recall)/(precision+recall).

Generalizability: Evaluating the generalizability of the proposed method by extending the evaluation to other log datasets. This step could involve cross--validation, transfer learning, and domain adaptation techniques.

7.8 Proposed Flow Diagram

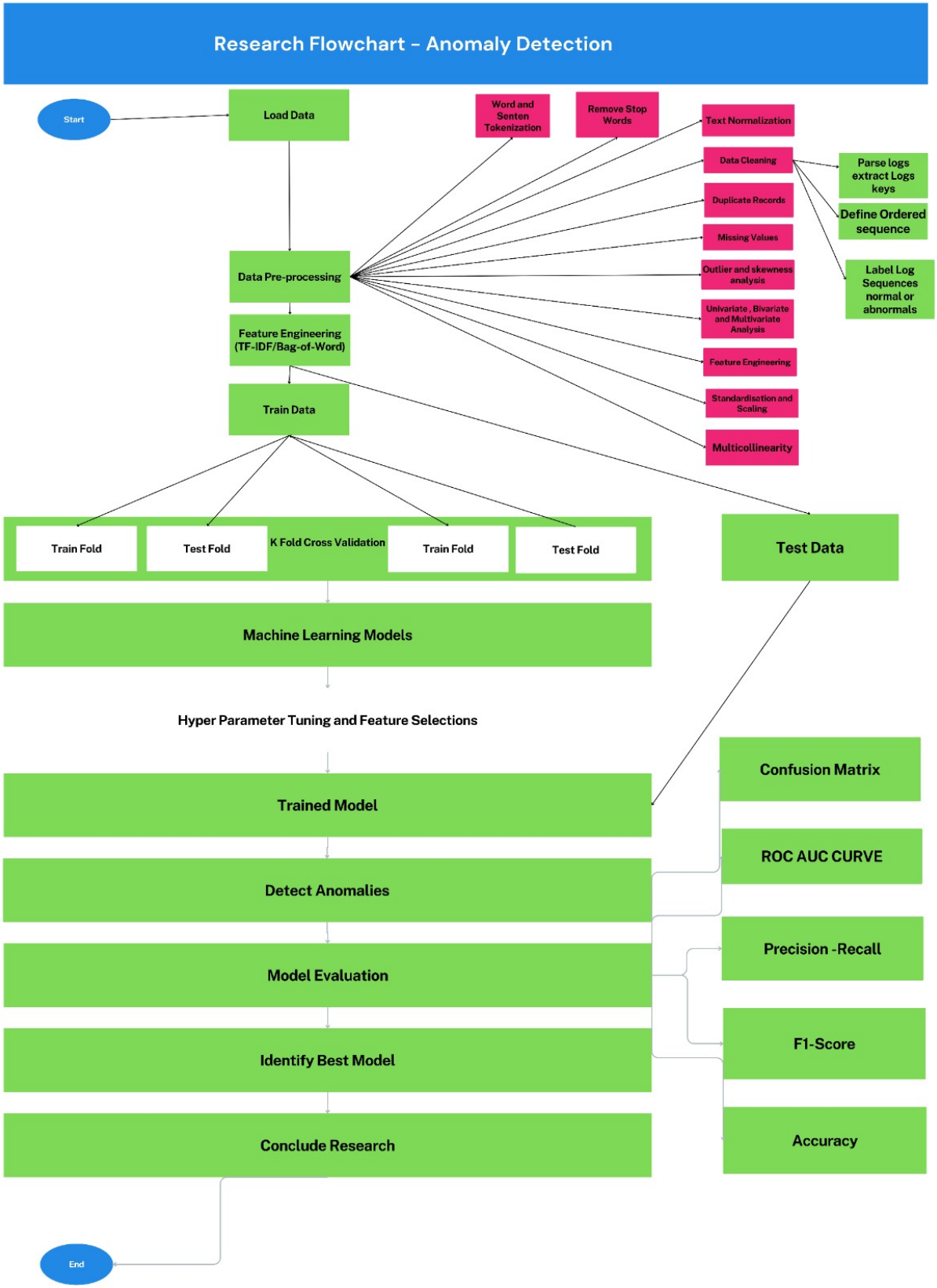


Figure 2 : Model Building Flow Chart

7.9 Expected Outcomes and Deliverables

Expected outcome from this research are as follows :

- Thorough analysis of log classification and anomaly detection, covering aspects such as real-time processing, robustness, noise handling, efficiency, and generalizability.
- Detailed and precise description of the techniques and machine learning models employed, accompanied by a well-structured evaluation methodology to gauge their efficacy.
- Comparative examination of various AI-based techniques to identify the most efficient model for anomaly detection.
- Creation and evaluation of a log classification model to assess its performance and accuracy in detecting anomalies

8. Requirements / Resources

Below resources will be needed to carry out the research during the implementation.

8.1 Software Requirements

- Python 3.6.X: Programming language for data analysis and machine learning tasks.
- Anaconda Navigator 1.9.12: Package manager for managing Python libraries and environments.
- Jupyter Notebook 6.1+: Interactive computing environment for Python.
- JupyterLab 3.x: Advanced interactive development environment for Python.
- TensorFlow 2.4+: Open-source library for machine learning and deep learning tasks.
- PyTorch 1.x or 1.9+: Open-source machine learning framework.
- Hugging Face Transformers 4.x: Library for state-of-the-art natural language processing models.
- Scikit-learn 0.24+: Machine learning library for Python.
- Keras 2.4+: Open-source neural network library for Python.
- Matplotlib 3.3+: Plotting library for Python.
- Seaborn 0.11+: Data visualization library for Python.
- Pandas (latest): Data analysis library for Python.
- Numpy (latest): Numerical computing library for Python.
- Tableau 2021.1+ or PowerBI 2021.1+: Data visualization and analysis tools.
- Git 2.29+: Version control system for tracking and managing code and data changes.

Packages and Libraries:

- H2O (latest): Open-source software for data analysis and machine learning.
- PyCaret (latest): Low-code machine learning library.
- MLFlow (latest): Open-source platform for managing the machine learning lifecycle.
- AirFlow (latest): Platform for programmatically authoring, scheduling, and monitoring workflows.
- Pandas: Data processing library for Python.
- Numpy: Numerical computing library for Python.
- Matplotlib: Data visualization library for Python.
- Seaborn: Data visualization library for Python.
- Scikit-learn: Machine learning library for data pre-processing, predictive modeling, and model evaluation.

Apart from above listed software's, packages and libraries based on the requirement we may use other tools and libraries as well.

8.2 Hardware Requirements

Hardware requirement for carrying out the task are as follows

- Processor: Intel Core i5 or AMD Ryzen 5 (or higher)
- RAM: 8 GB (or higher)
- Storage: 100 GB (or higher) of available disk space
- Graphics Card: NVIDIA GeForce GTX 1060 (or equivalent) with 4 GB VRAM (or higher)
- Operating System: Windows 10, macOS, or Linux
- Internet Connection: High-speed internet for downloading datasets, libraries, and updates
- Additional Hardware: SSD (Solid State Drive) for faster data access and model training

9. Research Plan

Below is the Gantt chart visualization for the detailed research plan on Anomaly Log Detection.

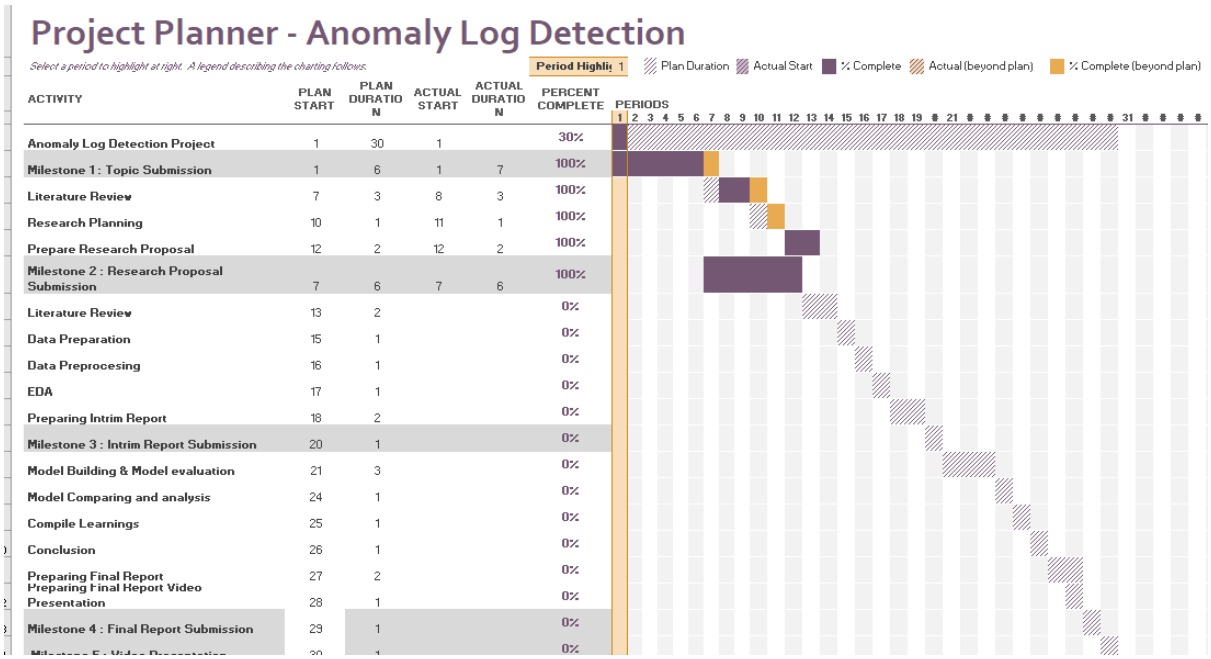


Figure 3 : Gantt chart Research Plan

10. Risk & Contingency Plan

Table 5 : Risk & Contingency Plan

Risk	Contingency Plan
Computational Limitations	Optimize code efficiency and consider using cloud or high-performance computing resources.
Model Performance	Fine-tune models, experiment with different architectures.
Time Management	Review project milestones, allocate additional resources, or adjust research objectives to manage time.
Document Change Loss	Maintain multiple copies of the research report in different locations to prevent accidental changes.
Unforeseen Circumstances	Allocate a buffer in the research timeline and start working on the research early to handle unforeseen events.
IDE Crash	Create modular copies of code and store module outputs to recover progress in case of platform crash.
System Crash	Regularly backup coding files using reliable platforms to minimize the risk of losing work due to system failure.
Research Plan Deviation	Maintain regular communication with the thesis supervisor to track progress and align with the original plan.

11. References :

Bhanage, D.A., Pawar, A.V. and Kotecha, K., (2021a) *IT Infrastructure Anomaly Detection and Failure Handling: A Systematic Literature Review Focusing on Datasets, Log Preprocessing, Machine Deep Learning Approaches and Automated Tool*. *IEEE Access*, .

Bhanage, D.A., Pawar, A.V. and Kotecha, K., (2021b) *IT Infrastructure Anomaly Detection and Failure Handling: A Systematic Literature Review Focusing on Datasets, Log Preprocessing, Machine Deep Learning Approaches and Automated Tool*. *IEEE Access*, .

Chen, S. and Liao, H., (2022) BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model. *Applied Artificial Intelligence*, 361.

Choi, H., Kim, J., Joe, S. and Gwon, Y., (2020) Evaluation of BERT and Albert sentence embedding performance on downstream NLP tasks. In: *Proceedings - International Conference on Pattern Recognition*. Institute of Electrical and Electronics Engineers Inc., pp.5482–5487.

Duan, X., Ying, S., Yuan, W., Cheng, H. and Yin, X., (2021) QLLog: A log anomaly detection method based on Q-learning algorithm. *Information Processing and Management*, 583.

Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., Yang, H. and Luan, Z., (2020) HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log. *IEEE Transactions on Network and Service Management*, 174, pp.2064–2076.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F. and Liu, Q., (2019) TinyBERT: Distilling BERT for Natural Language Understanding. [online] Available at: http://arxiv.org/abs/1909.10351.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R., (2019) ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. [online] Available at: http://arxiv.org/abs/1909.11942.

Naudé, M., Adebayo, K.J. and Nanda, R., (2022) A machine learning approach to detecting fraudulent job types. *AI and Society*.

Oliner, A. and Stearley, J., (2007) *What Supercomputers Say: A Study of Five System Logs*.
Ryciak, P., Wasielewska, K. and Janicki, A., (2022) Anomaly Detection in Log Files Using Selected Natural Language Processing Methods. *Applied Sciences (Switzerland)*, 1210.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T., (2019) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. [online] Available at: http://arxiv.org/abs/1910.01108.

Wang, Q., Zhang, X., Wang, X. and Cao, Z., (2022) Log Sequence Anomaly Detection Method Based on Contrastive Adversarial Training and Dual Feature Extraction. *Entropy*, 241.

Zhao, Z., Xu, C. and Li, B., (n.d.) A LSTM-Based Anomaly Detection Model for Log Analysis. [online] Available at: https://doi.org/10.1007/s11265-021-01644-4.