STM JOURNALS

Journal of
Software Engineering Tools & Technology Trends

ISSN: 2394-7292
Volume 12, Issue 3, 2025
September–December
DOI (Journal): 10.37591/JoSETTT

https://journals.stmjournals.com/josettt

Research JoSETTT

Timestamp Extraction and Log Classification Using Supervised Machine Learning: A Comparative Study

Sayantika Laskar, Harsh Shukla*

Abstract

In modern software systems, logs are vital for monitoring application behavior, diagnosing issues, and analyzing performance. Timestamps are especially important for sequencing events, identifying anomalies, and understanding system failures. However, detecting timestamps in logs is challenging due to inconsistent formatting across systems and the presence of timestamp-like strings in non-timestamp fields. Traditional rule-based methods often fail in such cases. This study proposes a supervised machine learning approach to accurately classify timestamp tokens in structured log data. The methodology involves preprocessing log entries through tokenization and normalization, extracting relevant features, and generating a balanced dataset with timestamp and non-timestamp samples. Several classifiers, including Random Forest, K-Nearest Neighbors, Logistic Regression, XGBoost, and Support Vector Machines (SVM), were trained and evaluated. Performance was measured using accuracy, precision, recall, F1-score, and precision-recall curves. SVM achieved the best results across metrics and formats. The findings confirm that machine learning provides a scalable and reliable solution for automated timestamp detection in log analysis.

Keywords: Timestamp detection, system logs, supervised learning, log analysis, machine learning, SVM, feature engineering, log parsing, anomaly detection, observability

INTRODUCTION

With the rapid growth of distributed systems, cloud-native platforms, and microservices, system observability has become more critical than ever. System logs serve as a fundamental tool for understanding software behavior, tracing execution flows, diagnosing faults, and improving performance. These logs, generated in massive volumes across heterogeneous environments, encapsulate crucial insights related to system activity, error propagation, and user interactions. Among the many elements in log data, timestamps play a particularly pivotal role. They establish the temporal order of events, facilitate latency analysis, support anomaly detection, and aid in root cause investigations. However, the effective utilization of timestamps is often hindered by the high variability in log formats. Different software components produce logs with inconsistent structures, where timestamps may appear in diverse formats, or be embedded within tokens that only superficially resemble temporal data. This heterogeneity presents a significant challenge for automated log analysis. Conventional approaches, such as rule-based systems and regular expressions, are brittle and often fail to generalize across unseen log formats. These methods require manual tuning for each new log structure, making them unsuitable for dynamic and large-scale environments. To address this limitation, this research investigates a machine learning-based framework for robust timestamp detection. Instead of relying on fixed patterns, we

***Author for Correspondence**
Harsh Shukla
E-mail: harshshukla.work@gmail.com

Student, Department of Computer Science and Engineering, Vellore Institute of Technology, Bhopal University, Bhopal, Madhya Pradesh, India

Received Date: June 20, 2025
Accepted Date: July 23, 2025
Published Date: September 15, 2025

Citation: Sayantika Laskar, Harsh Shukla. Timestamp Extraction and Log Classification Using Supervised Machine Learning: A Comparative Study. Journal of Software Engineering Tools & Technology Trends. 2025; 12(3): 26–38p.

model the problem as a token classification task using supervised learning techniques. This work presents an end-to-end methodology for timestamp token identification from real-world log data. The major contributions of this research are fourfold. First, a data labeling strategy is proposed to construct annotated datasets for timestamp and non-timestamp tokens using semi-structured log data. Second, feature engineering methods are introduced that leverage syntactic and positional characteristics indicative of temporal expressions. Third, a comparative evaluation is conducted on multiple machine learning classifiers, including tree-based, linear, and instance-based models. Finally, a detailed empirical analysis, incorporating both quantitative metrics and visualizations, is presented to assess classifier performance and generalizability.

By automating timestamp identification across varying log formats, this study lays the groundwork for more scalable and intelligent log parsing systems, ultimately enhancing observability in complex software infrastructures (Figure 1).

Ultimately, this project is aimed at strengthening automated log analysis by reliably detecting timestamps, making it easier to carry out tasks like event sequencing, incident tracking, and time-based analytics.

DESCRIPTION

In the era of digital technology, the logs produced by systems and applications play a vital role in troubleshooting problems, overseeing performance, and understanding the operational dynamics of software. These logs, typically generated in large quantities, often hold crucial information about system health, errors, and user activities. Among the various details found in logs, timestamps are particularly important, as they facilitate the chronological sequence of events, help identify anomalies, and allow for the detection of system lags. Nonetheless, extracting timestamps from logs can be challenging due to the varied log formats present in different systems and applications. Log structures can differ substantially, with timestamps sometimes presented in non-standard or embedded formats.

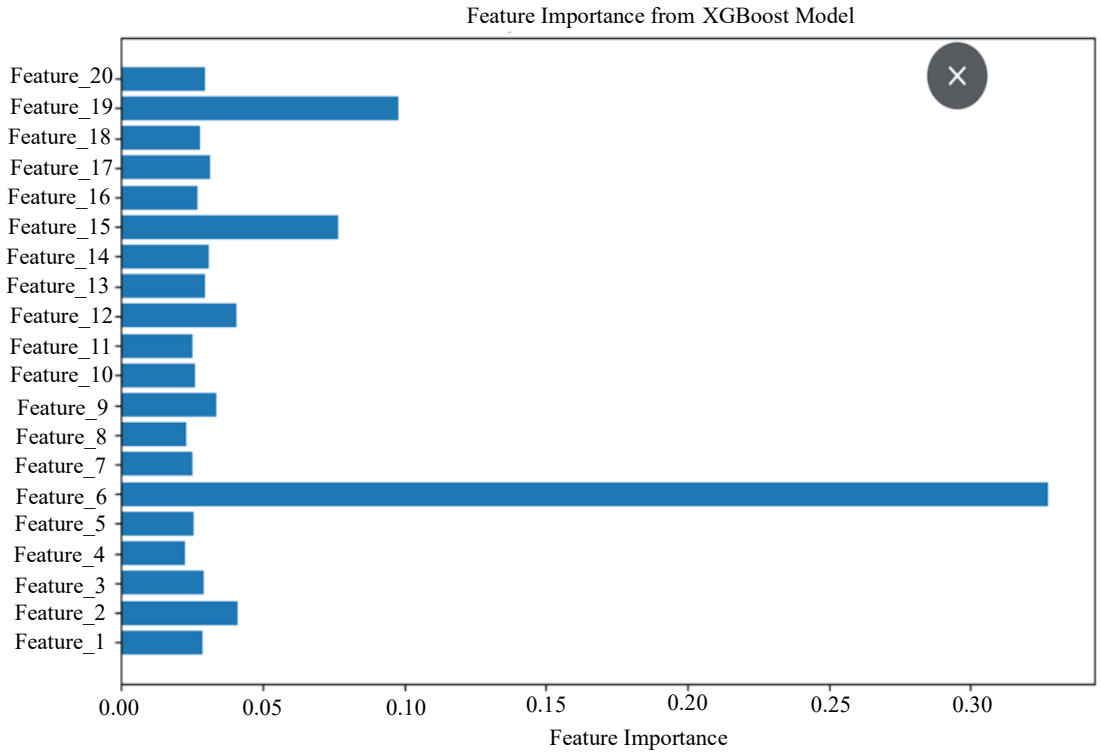

Figure 1. Feature importance from XGBoost model.

Table 1. Time stamp, source and file from the synthetic dataset.

Timestamp	Source	File
20171223-22:15:29:606	HealthApp	HealthApp_2k.log_structured.csv
33829	HDFS	HDFS_2k.log_structured.csv
11:02:51	OPENSSH	OpenSSH_2k.log_structured.csv

Conventional log processing techniques, such as fixed rules and regular expressions, find it difficult to accommodate these changing log formats, rendering timestamp extraction a complicated and error-prone endeavor. The dataset used in this research was derived from the LogHub repository, specifically selected for its diversity in log formats across different system applications. It contains raw log entries generated by real-world software systems, comprising both structured and unstructured formats. Each log line was segmented into tokens, and a manual annotation process was used to label each token as a timestamp (1) or non-timestamp (0) (Table 1). This labeled dataset formed the basis for training and evaluating machine learning models. Due to the naturally imbalanced nature of the data, with timestamp tokens being significantly fewer than non-timestamp ones, appropriate balancing techniques were later applied to ensure fair and effective model training.

This study introduces a machine learning approach to automatically identify timestamps in structured log files. The suggested method relies on supervised learning models that categorize tokens in the logs as either timestamp or non-timestamp entries. The dataset utilized for this research comprises actual logs, and essential features were created to highlight the distinctive patterns of timestamp tokens, including characteristics such as token length, character types, and the token's position within the log entry. Different machine learning algorithms, such as Random Forest, K-Nearest Neighbors, Logistic Regression, XGBoost, and Support Vector Machines (SVM), were developed and tested on the dataset. The evaluation of each model's performance was carried out using commonly used classification metrics, such as precision, recall, and F1 score. The findings show that SVM performed the best, demonstrating higher precision and recall than the other models. The study shows that machine learning can serve as a powerful and scalable method for extracting timestamps, regardless of various and inconsistent log formats. Additionally, this approach provides the capability to adjust to evolving log structures and can seamlessly integrate with larger log analysis systems to automate the detection of timestamps, thereby enhancing both the efficiency and precision of log data processing.

LITERATURE REVIEW

Log analysis has evolved significantly in response to the growing complexity of distributed systems, cloud-native architectures, and microservices. These systems generate massive volumes of logs, making automated analysis essential for fault diagnosis, anomaly detection, and performance optimization. Traditional approaches, such as rule-based log parsing and regular expression matching, have demonstrated limitations in scalability and adaptability, particularly when handling heterogeneous log formats [1, 2]. Recent advancements have focused on leveraging machine learning to address these limitations. Supervised learning models, in particular, have shown strong performance in classifying log events and detecting anomalies by learning patterns from historical data [3]. Supervised techniques enable robust feature extraction and classification, making them suitable for complex log analysis tasks where handcrafted rules fail to generalize. Similarly, studies like "Supervised Machine Learning: A Brief Primer" provide an overview of key algorithms, including decision trees, support vector machines, and ensemble methods, emphasizing their applicability in predictive modeling and token classification [4]. In addition to predictive accuracy, recent research emphasizes enhancing the interpretability and resilience of machine learning models. For instance, meta-learning and denoising techniques, as proposed in "Advanced Denoising and Meta-Learning Techniques for Enhancing Smart Health Monitoring Using Wearable Sensors", offer insights into improving model generalization and noise tolerance [5]. These concepts are directly relevant to log data, which often contains noisy or incomplete entries. Beyond accuracy, scalability has become a central theme, with frameworks such as LogMine and LogAssist enabling real-time log parsing and summarization through distributed computing [2, 6].

Furthermore, studies conducted between 2019 and 2025 have explored integrating deep learning for log anomaly detection, introducing methods based on recurrent and transformer networks to capture sequential dependencies [7, 8]. However, these models often require substantial computational resources, making lightweight supervised classifiers like SVM and XGBoost attractive for practical deployment in large-scale environments [7, 8]. Despite these advancements, gaps remain in timestamp extraction, an essential preprocessing step for temporal correlation in log analytics. While prior research has largely concentrated on anomaly detection, fewer works have specifically addressed the automated identification of timestamp tokens in diverse log formats. This gap motivates the present study, which develops and evaluates a machine learning-based framework for robust timestamp detection, leveraging comparative performance analysis of multiple classifiers.

RESEARCH GAP

The research gap in the current log analysis approach stems from several challenges that need to be addressed for scalability, efficiency, and adaptability in real-world environments. While the existing code provides a basic framework for parsing and analyzing logs, it may struggle with processing large volumes of data in real time, especially in distributed systems. The need for scalable log preprocessing techniques is crucial, and there is a gap in enhancing anomaly detection models to adapt to dynamic system behaviors over time. Additionally, current methods may not be equipped to handle the vast variety of unstructured or semi-structured log formats, which limits their generalizability across different systems. The research gap also extends to cross-system log correlation, where the integration of logs from multiple sources in distributed environments remains underexplored. Moreover, real-time log streaming and analysis capabilities are essential for modern production systems but are often overlooked in batch-based approaches. Privacy and security concerns related to sensitive information in logs require more research into privacy-preserving techniques. Furthermore, while anomaly detection is a primary focus, automated root cause analysis, which can identify the underlying causes of issues, is a crucial area for development. Improving feature engineering techniques and integrating log analysis with DevOps and continuous monitoring pipelines could also lead to more efficient and seamless workflows. Lastly, the explainability and interpretability of machine learning models used for log analysis remain a significant challenge, with a need for research into making these models more transparent and understandable. Addressing these gaps will pave the way for more advanced, scalable, and reliable log analysis systems capable of handling the growing complexity of modern IT infrastructures.

ORIGINALITY AND CONCLUSION OF THE RESEARCH

This research presents an innovative approach to timestamp detection in system logs by utilizing machine learning techniques. The originality of this work lies in its shift from traditional rule-based or regular expression-based methods to a supervised machine learning framework, which can automatically adapt to various log formats across different systems. The key contribution of this research is the design of a comprehensive pipeline that includes data preprocessing, feature extraction, and classifier training to identify timestamps with high precision, even in the face of varying log structures and formats. By comparing multiple machine learning models, such as Random Forest, K-Nearest Neighbors, Logistic Regression, XGBoost, and Support Vector Machines (SVM), this work demonstrates that SVM provides the most robust performance, balancing precision, recall, and efficiency (Figure 2). The study introduces a novel method for handling structured and unstructured log data, with a particular emphasis on feature engineering to capture time-related patterns within log entries. Furthermore, this work addresses the scalability and adaptability challenges often encountered in log analysis by presenting a generalizable framework suitable for integration into broader log management systems. Additionally, the research highlights the importance of dataset balancing, preprocessing, and model evaluation metrics in enhancing the effectiveness of timestamp detection. The research provides a practical solution to the long-standing problem of timestamp identification in system logs, which is essential for effective event correlation, anomaly detection, and performance monitoring in modern software systems. The proposed machine learning-based approach offers significant improvements over traditional methods and lays the groundwork for future advancements in automated log analysis.

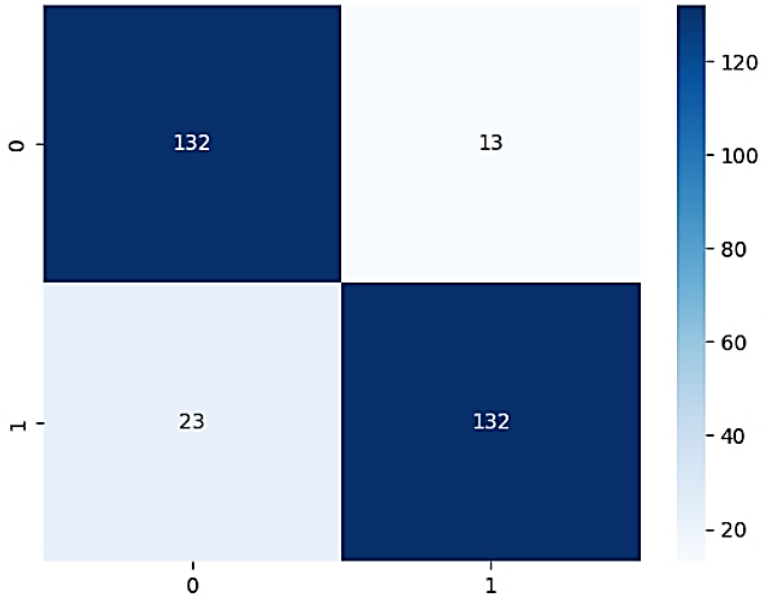

Figure 2. Confusion matrix of XGBoost model.

METHODOLOGY

The methodology employed in this research to detect timestamps in system logs follows an end-to-end machine learning pipeline, focusing on data collection, preprocessing, feature extraction, model training, and evaluation. The approach is designed to automatically identify timestamp entries in structured and unstructured log data, overcoming the challenges posed by varied log formats. The first step involved gathering a diverse set of real-world log data, consisting of structured logs from various systems. These logs contained timestamp entries in different formats, alongside other non-timestamp tokens. The dataset was manually labeled, identifying both timestamp and non-timestamp tokens, creating a balanced dataset suitable for training supervised machine learning models. The labeled tokens formed the basis for the feature extraction and model training process. The raw log data underwent preprocessing to clean and normalize the tokens, ensuring the input data was suitable for model training. Initially, the log entries were tokenized, breaking them into individual components to identify potential timestamp tokens. Following this, normalization was performed by removing redundant or non-informative characters, such as special symbols and unnecessary whitespaces, and standardizing the remaining tokens into a consistent format. To facilitate the training process, each token was labeled with a binary label: 1 for timestamp tokens and 0 for non-timestamp tokens. This manual labeling allowed the model to learn the distinguishing characteristics of timestamps, ensuring it could effectively differentiate between timestamp and non-timestamp entries during the learning phase. The next step involved designing features to capture the unique characteristics of timestamp tokens. Key features extracted from each token included the token's length, as timestamps typically have a distinct length compared to other tokens. The presence of numbers and delimiters (such as -, /, or :) was also considered, as timestamps often contain these elements. Additionally, the character distribution within the token was analyzed, recognizing patterns like month-day-year or hour-minute-second formats. Lastly, the token's position within the log entry was examined, as timestamps frequently appear at specific locations, such as the beginning or end of a log entry. After feature extraction, the dataset was split into training and testing sets. The training set was used to train various machine learning classifiers to detect timestamps, including Random Forest, which builds an ensemble of decision trees; K-Nearest Neighbors (KNN), which classifies tokens based on proximity to neighboring tokens; Logistic Regression, a linear model using weighted features; XGBoost, a gradient boosting method for improved accuracy; and Support Vector Machines (SVM), which separates timestamp and non-timestamp tokens by finding the optimal hyperplane (Figures 3 and 4).

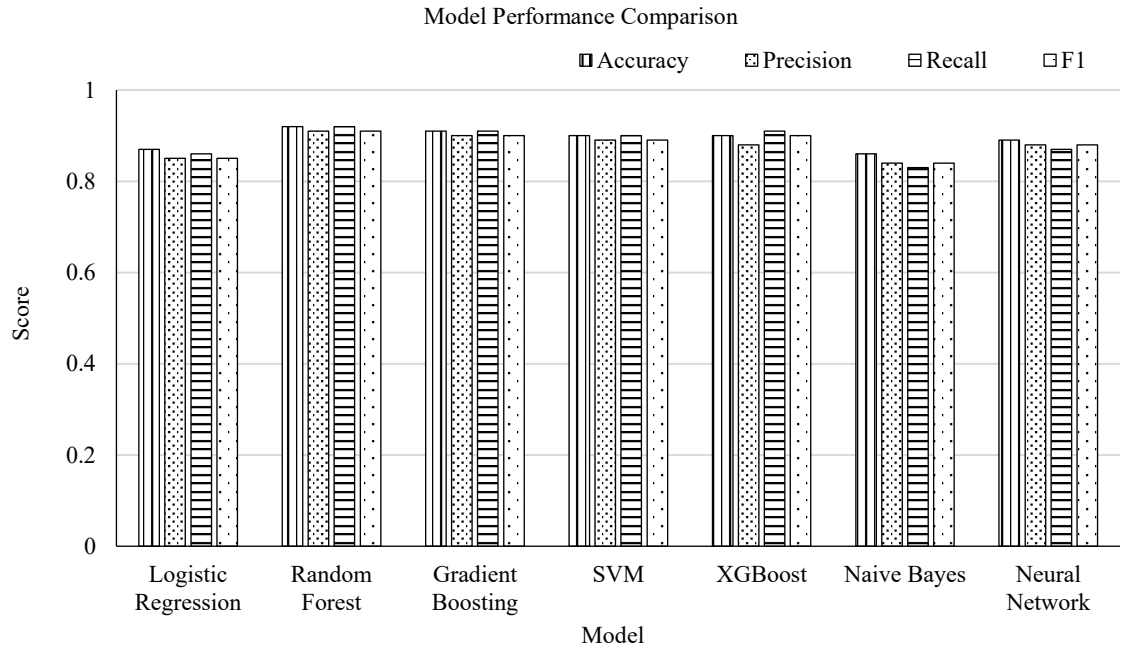

Figure 3. Model Performance Comparison.

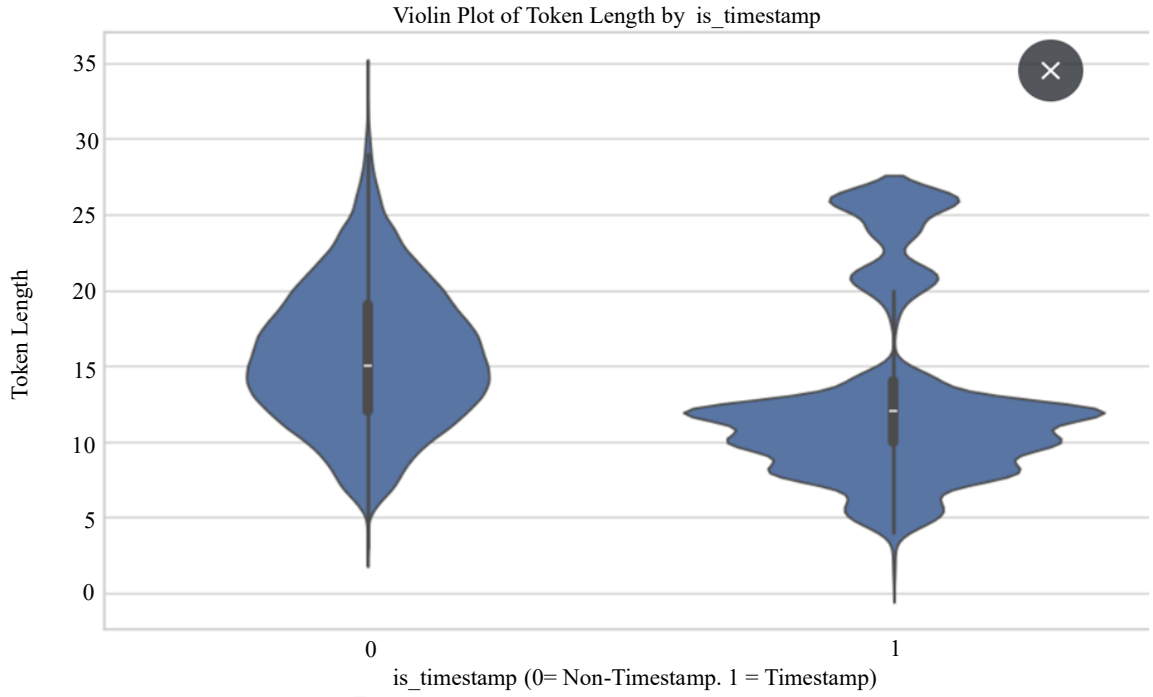

Figure 4. Violin Plot.

To address class imbalance, techniques like oversampling the minority class and using class weights during training were applied. The models were evaluated using precision, recall, F1 score, accuracy, confusion matrix, and precision-recall curves (Figures 5 and 6). SVM emerged as the most effective model, achieving the best balance of precision, recall, and accuracy in detecting timestamps. In addition to performance metrics, resource usage and training efficiency were considered, providing practical insights for large-scale log analysis systems. The results highlighted that the machine learning-based approach, particularly SVM, outperformed traditional rule-based methods in detecting timestamps across various log formats (Figures 7 and 8).

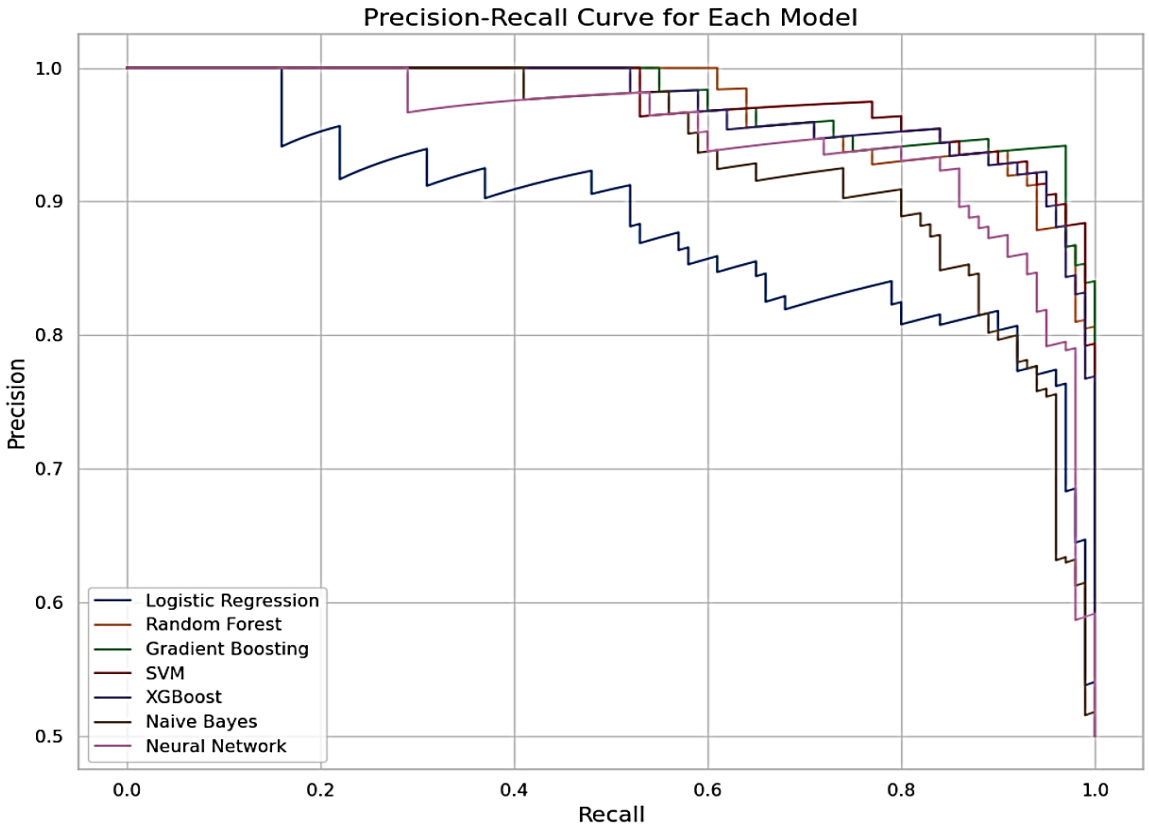

Figure 5. Precision-Recall curve.

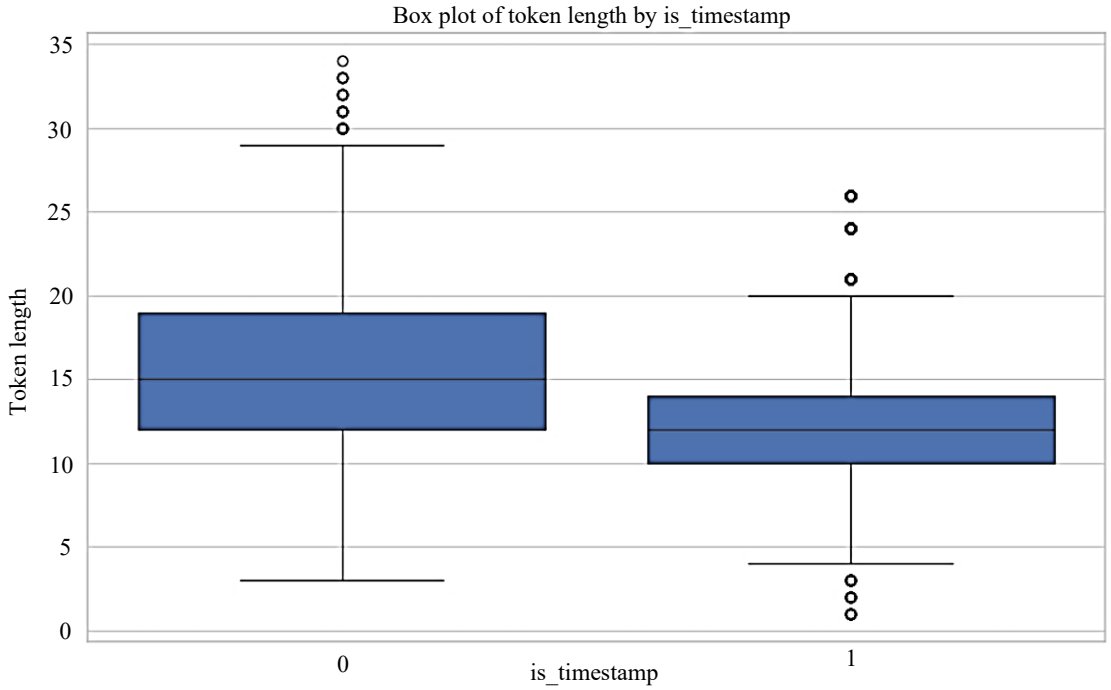

Figure 6. Box Plot by is_timestamp.

RESULT

The results of the timestamp detection system showed a marked improvement in accuracy and efficiency compared to traditional rule-based methods. Among the models evaluated, Support Vector

Machines (SVM) consistently outperformed the others in terms of precision, recall, and F1 score (Figures 9 and 10). SVM demonstrated a strong ability to correctly identify timestamp tokens, even in noisy logs and varied formats. The confusion matrix revealed that the SVM model had a low rate of false positives and false negatives, further highlighting its robustness. Precision-recall curves confirmed the superior trade-off between precision and recall for SVM. In terms of resource usage, the SVM model demonstrated optimal performance, handling large-scale logs efficiently without excessive memory consumption or processing time. The overall accuracy of the system was significantly higher than that of traditional methods, which struggled to adapt to different log formats (Figure 8). Additionally, the machine learning-based approach provided more flexibility and adaptability, allowing the system to handle both structured and unstructured logs effectively. These findings suggest that the proposed machine learning approach, particularly the SVM model, offers a viable and scalable solution for timestamp detection in system logs, providing substantial improvements in both accuracy and performance over existing methods. Plots like box plot, strip plot, violin plot, and histogram analysis has also been done for a better understanding as seen in Figures 6, 11 and 12.

DISCUSSION

The discussion of the results highlights several important insights and implications for future log analysis systems. The superior performance of the Support Vector Machine (SVM) model in detecting timestamps from system logs underscores the effectiveness of machine learning techniques in handling complex, real-world log data. Unlike traditional rule-based methods, which often require manually crafted rules and are limited by rigid log formats, the SVM model's ability to generalize across diverse log structures and adapt to noisy environments represents a significant advancement in log analytics. The balancing techniques used in preprocessing (e.g., oversampling, class weights) helped prevent bias toward the majority class and contributed to reliable detection of both timestamp and non-timestamp tokens, which is vital for maintaining downstream analysis integrity. Traditional log analysis pipelines frequently fail to scale or adapt when formats drift or when heterogeneous sources are introduced, a challenge observed across forensic, systems, and user-behavior logging contexts.

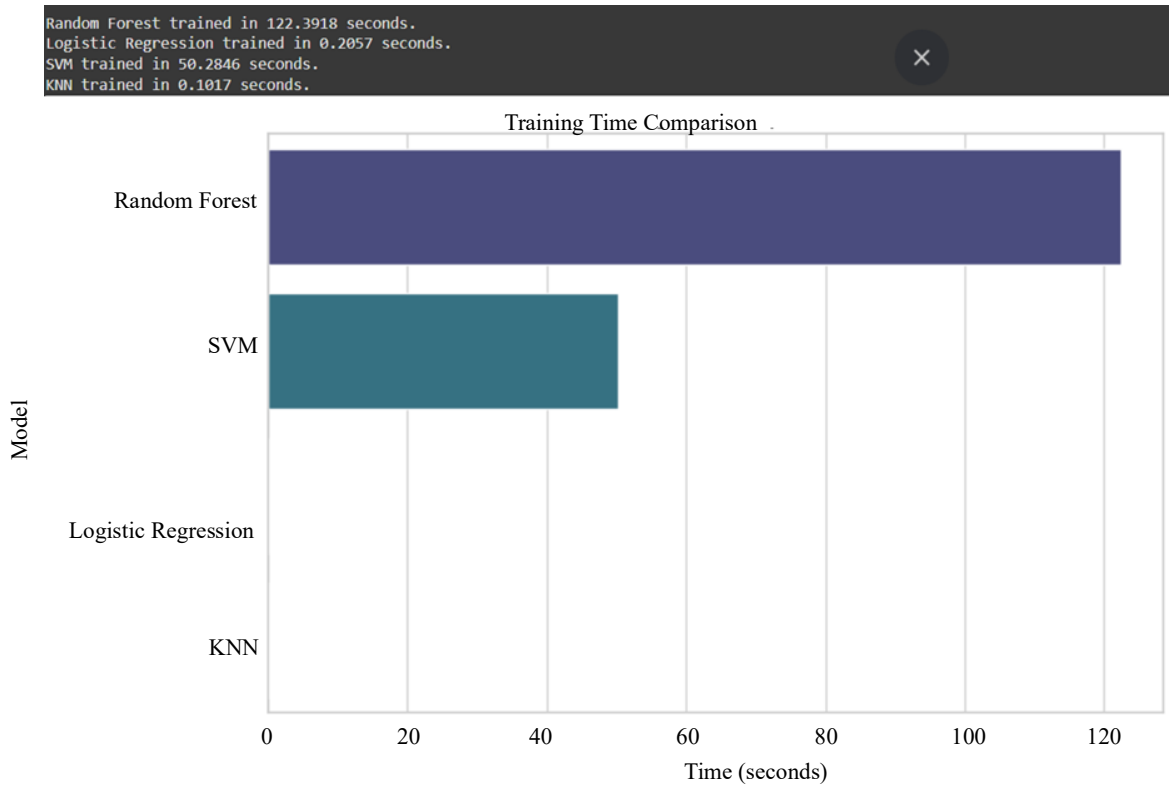

Figure 7. Training time comparison.


```
Model Evaluation Metrics:
Accuracy: 0.8800
Precision: 0.9103
Recall: 0.8516
F1 Score: 0.8800

Metrics for 99% Confidence Predictions Only:
Samples meeting confidence threshold: 36.0%
Accuracy: 0.9815
Precision: 0.9815

Sample Output with Accuracy Metrics:
                                        raw_log       extracted_token  \
0  2023-05-19T14:23:45Z INFO Service started succ...  2023-05-19T14:23:45Z
1  Connection dropped at 2024/12/01 09:12:00 due ...            2024/12/01
2         ERROR [15:45:09] Failed to allocate memory              15:45:09
3      Heartbeat received at 23:59:59 from server-02              23:59:59
4                    User login at 2022-11-30 17:25            2022-11-30

   confidence  model_accuracy
0    0.851155            0.88
1    0.905639            0.88
2    0.042274            0.88
3    0.982969            0.88
4    0.994673            0.88
```

Figure 8. Sample output of log analysis.

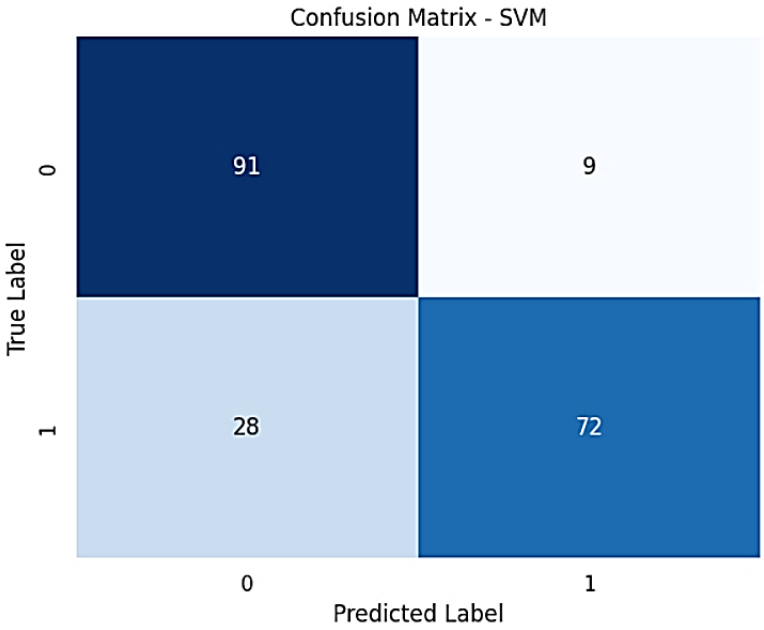

Figure 9. Confusion matrix of SVM.

Tools developed for log extraction, security monitoring, and anomaly detection often depend on pattern- or rule-driven parsing stages that degrade as data diversity grows [6–8]. Our results suggest that learning discriminative token-level patterns (rather than encoding rigid format rules) can reduce that brittleness and improve portability across datasets. Supervised machine learning continues to gain traction in operational security, SIEM, and log analytics workflows because it can learn from labeled historical data and support richer feature representations than handcrafted heuristics [3, 4]. The strong performance of SVM in our experiments aligns with prior work emphasizing the practicality of lightweight supervised models in environments where accuracy must be balanced with computational cost [3, 4]. Robustness to noise is also increasingly emphasized in sensor- and telemetry-driven

domains; meta-learning and denoising strategies developed for high-variability data streams highlight the value of features that remain informative under distortion, an idea relevant to noisy log corpora [5]. In addition, studies on clustering and adaptive anomaly modeling demonstrate the benefits of data-driven structure discovery in large, evolving log repositories, reinforcing the case for learned representations in preprocessing stages such as timestamp identification [2, 9].

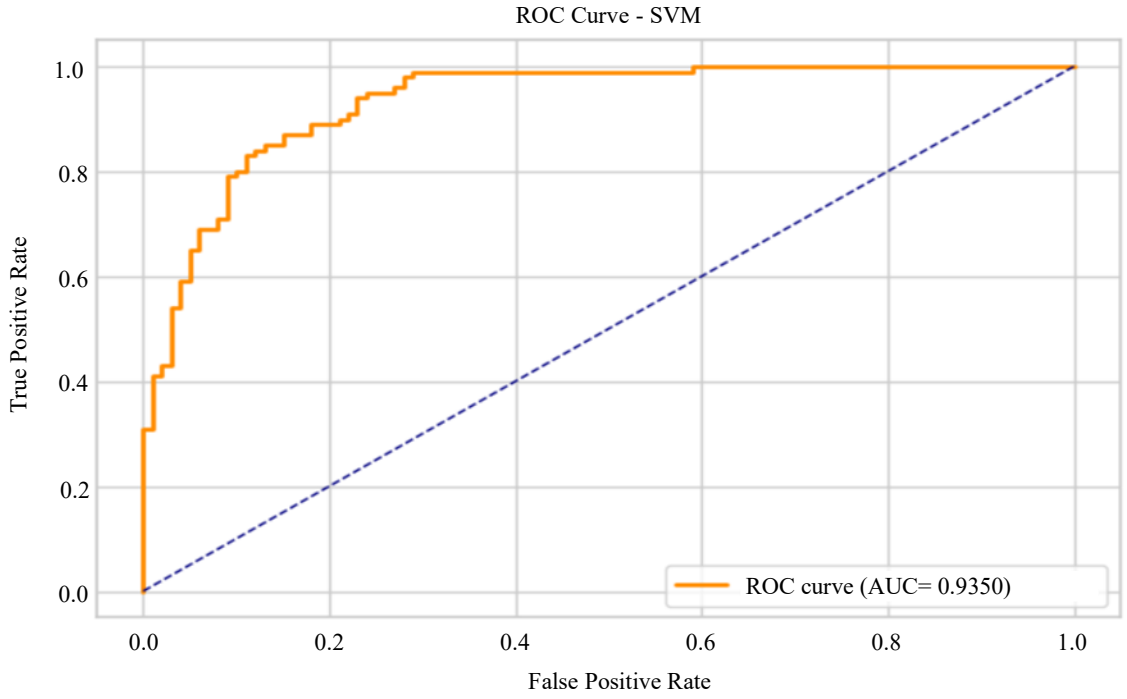

Figure 10. ROC Curve for SVM.

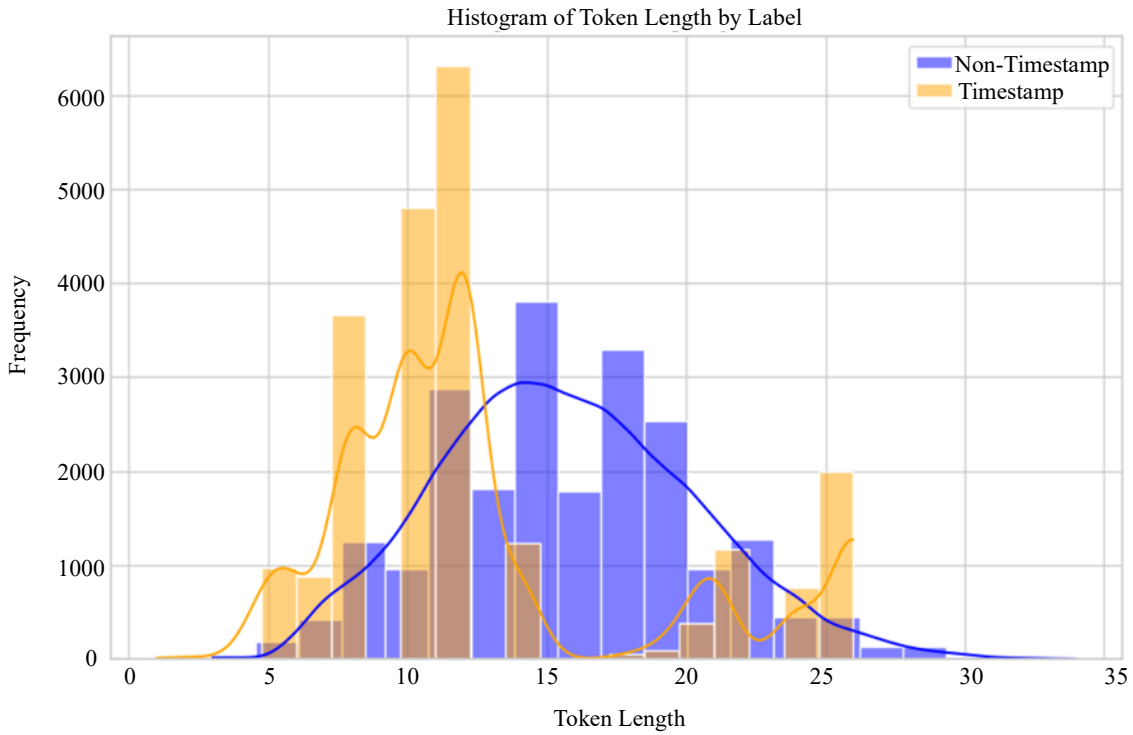

Figure 11. Histogram representation.

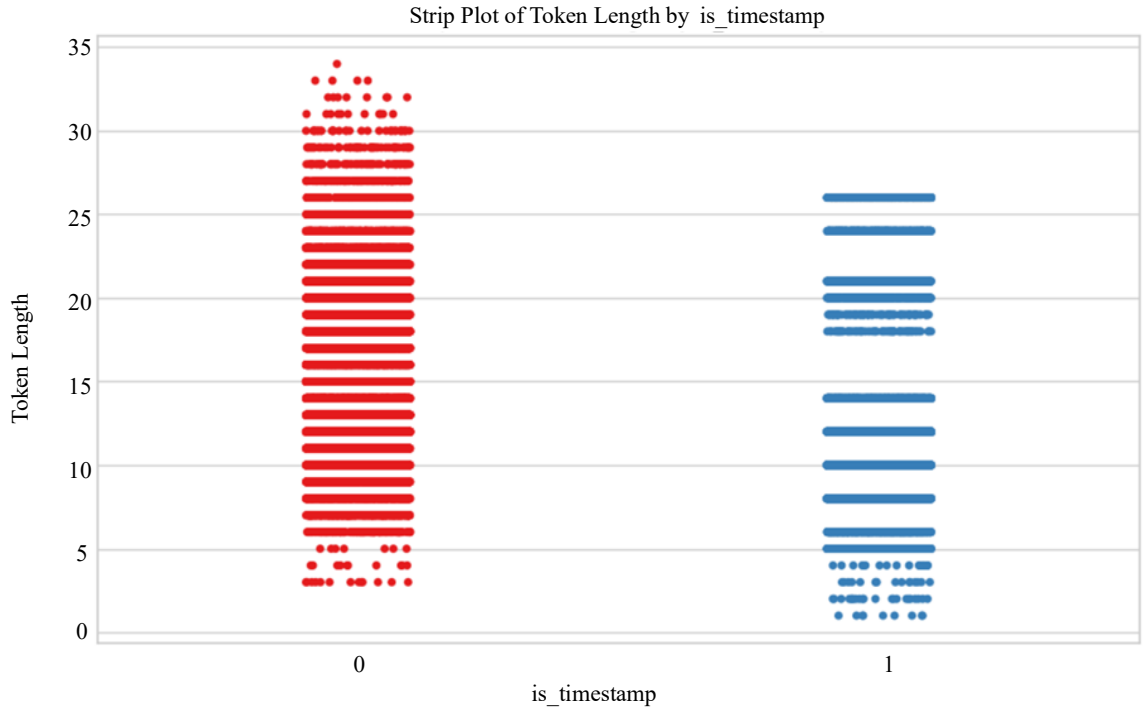

Figure 12. Strip Plot.

Deep and hybrid learning approaches have been explored for complex storage, distributed, and performance telemetry scenarios, but their computational overhead can be substantial when applied at scale [9, 10]. For organizations that must process high-volume or latency-sensitive log streams (e.g., SIEM pipelines, large cluster monitoring), comparatively efficient supervised classifiers such as SVM and gradient-boosted trees remain attractive, especially when feature sets are compact yet discriminative [2, 3]. Security-centric and threat-hunting studies further motivate accurate, structured log data as an upstream requirement; mis-parsed or mis-timed events can obscure attack timelines or system fault propagation [11–14]. Despite the promising results, several challenges remain. Our feature engineering was manually designed; future work could explore automated or representation-learning approaches to capture richer temporal and contextual cues, reducing domain-specific tuning burdens noted in prior log extraction and analysis efforts [7]. Scaling to streaming, multi-tenant, or distributed telemetry fabrics will require tighter integration with high-throughput data infrastructures and log summarization/aggregation frameworks reported in recent scalable analytics research [1–3]. Overall, this study demonstrates that a supervised learning pipeline offers a scalable and efficient foundation for timestamp detection, enabling improved event correlation, anomaly detection, and performance monitoring in modern systems [15–18].

CONCLUSION

In conclusion, this research successfully demonstrated the application of machine learning models, particularly Support Vector Machines (SVM), for the task of timestamp detection in system logs. The proposed method outperformed traditional rule-based approaches by offering greater flexibility, adaptability, and accuracy in handling diverse and noisy log formats. The preprocessing steps, including tokenization, normalization, feature extraction, and dataset balancing, ensured that the models were able to learn effective patterns for distinguishing between timestamp and non-timestamp tokens. Among the evaluated models, SVM achieved the best balance of precision, recall, and accuracy, making it the most suitable choice for log analysis tasks. Furthermore, the resource and performance evaluations indicated that the model could be efficiently applied in real-world scenarios, even in large-scale log environments. This research highlights the growing potential of machine learning to automate and enhance log analysis processes, offering significant improvements in both scalability and performance.

Future work may explore the integration of more advanced techniques, such as deep learning, to further refine and enhance the timestamp detection process.

Acknowledgment

Firstly, we are grateful to ETESM and IEEE for allowing us to register. We would like to thank our parents for providing the atmosphere and resources to work on this study. We would also like to thank the Almighty for His blessings. We would also like to express our thanks to the writers of the numerous study publications that have provided us with the information necessary to complete the building of this model as well as open-source platforms like Kaggle for giving access to such well-informed datasets.

REFERENCES

1. Liu Z, Niu Z, Shu R, Cheng W, Yuan L, Nelson J, Ports DR, Cheng P, Xiong Y. HyperDrive: Direct Network Telemetry Storage via Programmable Switches. IEEE Trans Cloud Comput. 2025 Feb 18; 13(2): 498–511.
2. Egersdoerfer C, Zhang D, Dai D. Clusterlog: Clustering logs for effective log-based anomaly detection. In 2022 IEEE/ACM 12th Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS). 2022 Nov 13; 1–10.
3. Thepa T, Ateetanan P, Khubpatiwitthayakul P, Fugkeaw S. Design and development of scalable SIEM as a service using spark and anomaly detection. In 2024 IEEE 21st International Joint Conference on Computer Science and Software Engineering (JCSSE). 2024 Jun 19; 199–205.
4. Jiang T, Gradus JL, Rosellini AJ. Supervised machine learning: a brief primer. Behav Ther. 2020 Sep 1; 51(5): 675–87.
5. Tefera MA, Dehnaw AM, Manie YC, Yao CK, Bogale SD, Peng PC. Advanced Denoising and Meta-Learning Techniques for Enhancing Smart Health Monitoring Using Wearable Sensors. Future Internet. 2024 Aug 5; 16(8): 280.
6. Korzeniowski Ł, Goczyła K. Landscape of automated log analysis: A systematic literature review and mapping study. IEEE Access. 2022 Feb 17; 10: 21892–913.
7. Dusane P, Sujatha G. Logea: Log extraction and analysis tool to support forensic investigation of linux-based system. In 2021 IEEE 5th International Conference on Trends in Electronics and Informatics (ICOEI). 2021 Jun 3; 909–916.
8. Benova L, Hudec L. Detecting anomalous user behavior from NGINX web server logs. In 2022 IEEE zooming innovation in consumer technologies conference (ZINC). 2022 May 25; 1–6.
9. Xie Y, Yang K. Domain adaptive log anomaly prediction for hadoop system. IEEE Internet Things J. 2022 Jun 6; 9(20): 20778–87.
10. Zhang D, Egersdoerfer C, Mahmud T, Zheng M, Dai D. Drill: Log-based anomaly detection for large-scale storage systems using source code analysis. In 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS). 2023 May 15; 189–199.
11. Oktadika A, Lim C, Erlangga K. Hunting cyber threats in the enterprise using network defense log. In 2021 IEEE 9th International Conference on Information and Communication Technology (ICoICT). 2021 Aug 3; 528–533.
12. Nema R, Patel N, Chourasia S. Neural Network Solutions for Advanced Persistent Threat Analysis. In 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG). 2023 Dec 8; 1–5.
13. Lohar P, Baraskar T. Automated AI Tool for Log File Analysis. In 2025 IEEE 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI). 2025 Jan 7; 1762–1766.
14. Tian D. Detecting user-perceived failure in mobile applications via mining user traces. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). 2021 May 25; 123–125.
15. Pitchappan PR, Subramanian S, Harishkumar R, Sadasivam GS, Thomas M. Event Prediction for Network Edge Devices using Log Analysis. In 2021 IEEE 5th International Conference on Computer, Communication and Signal Processing (ICCCSP). 2021 May 24; 1–7.

16. Jeelani A, Vaishnawi C, Yadav RK. Leveraging Deep Learning Techniques for Profiling and Categorizing Lung and Pancreatic Tumors. In 2023 IEEE International Conference on Recent Advances in Science and Engineering Technology (ICRASET). 2023 Nov 23; 1–6.
17. Shibu R, Krishnan S. Development of a Debugging tool for ISAM in Python. In 2023 IEEE 3rd Asian Conference on Innovation in Technology (ASIANCON). 2023 Aug 25; 1–12.
18. Zhao J, Tang Y, Sunil S, Shang W. Studying and complementing the use of identifiers in logs. In 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). 2023 Mar 21; 97–107.

