International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

Efficient Log File Classification Using Template Mining and Machine Learning with Event ID Optimization

K.Baby Ramya[1], K.Pavani [2],M.Sreevalli[3]

#1 Assistant Professor in the Department of MCA,SRK Institute of Technology Vijayawada
#2 Assistant Professor & Head of Department of MCA, SRK Institute of Technology, Vijayawada

#3 Student in the Department of MCA, SRK Institute of Technology, Vijayawada

Abstract: Log files generated from large-scale systems are critical for monitoring performance and detecting anomalies, but their massive size makes analysis challenging. This paper proposes an efficient log classification approach using template mining by converting log messages into compact event IDs, reducing data size and improving processing speed. Machine learning algorithms including Random Forest, Decision Tree, XGBoost, and LightGBM are applied, with SMOTE used to address class imbalance. Experimental results demonstrate that the proposed method achieves high classification accuracy (up to 99%) while minimizing training time and computational cost.

Index terms - *— Log File Classification, Template Mining, Event ID Conversion, Machine Learning, Random Forest, XGBoost, LightGBM, SMOTE, TF-IDF, Anomaly Detection*

1. INTRODUCTION

Log files play a vital role in modern software systems by recording runtime activities, system events, and error information. They are widely used for system monitoring, performance evaluation, anomaly detection, and security analysis. However, with the rapid growth of large-scale distributed systems, the volume of log data has increased significantly, making manual analysis inefficient and time-consuming. Traditional log analysis techniques often struggle with scalability

and fail to provide accurate insights in real-time environments.

To address these challenges, automated log classification using machine learning has emerged as an effective solution. Existing approaches rely on raw log messages, which are often large and complex, leading to increased computational cost and longer training time. Moreover, handling class imbalance and extracting meaningful features from unstructured log data remain critical issues in achieving high classification performance.

This paper proposes an efficient log classification approach using template mining, where log messages are converted into compact event IDs to reduce data size and improve processing efficiency. Advanced machine learning algorithms such as Random Forest, Decision Tree, XGBoost, and LightGBM are employed for accurate classification, while SMOTE is used to address class imbalance. The proposed system enhances classification accuracy, reduces training time, and provides a scalable solution for real-time log analysis in large-scale systems.

2. LITERATURE SURVEY

a) Adversarial Networks and Machine Learning for File Classification:

An essential component of a forensic inquiry is accurately determining the kind of file being examined. The embedded content, such as an image, video, document, spreadsheet, etc., can be inferred from the file type alone. We suggest utilizing an adversarially-trained machine learning neural network to ascertain a file's real type even if the extension or file header is obfuscated to make its discovery more difficult, in situations where a system owner would wish to keep their files unreadable or their file type hidden. Our semi-supervised generative adversarial network (SGAN) classified 11 distinct categories of data with 97.6% accuracy. Additionally, we contrasted our network with three additional machine learning techniques including a conventional standalone neural network. The most accurate file classifier was the adversarially-trained network, particularly when there were few accessible supervised data.

b) A Survey on Automated Log Analysis for Reliability Engineering:

Logging statements in software source code produce semi-structured text called logs. Since software logs are sometimes the only data available that records program runtime

information, they have become essential to the reliability assurance mechanism of many software systems in recent decades. The number of logs has significantly expanded as contemporary software develops on a massive scale. Numerous research on automated log analysis have been carried out in order to facilitate the effective and efficient use of contemporary software logs in reliability engineering. This survey provides a thorough overview of automated log analysis research, covering topics such as automating and supporting the creation of logging statements, compressing logs, parsing logs into structured event templates, and using logs to identify anomalies, forecast failures, and aid in diagnosis. We also review research that makes open-source datasets and toolkits available. We offer a number of possible future approaches for next-generation automated log analysis and real-world applications based on the discussion of recent developments..

c) Anomaly Detection for Web Log Data Analysis: A Review:

Numerous techniques have been created to defend web servers from intrusions. Generic user models and application behavior are the foundation of anomaly detection techniques, which interpret deviations from the predefined pattern as signs of potentially hazardous activity. In order to avoid and detect online attacks, we conducted a comprehensive evaluation of anomaly detection techniques in this study. Specifically, we used Kitchenham's standard approach to conduct a systematic analysis of computer science literature. 8041 peer-reviewed articles have been published in prestigious journals. This method is used on 88 articles. The methods used to conduct this systematic review are described on this page, along with the results and conclusions. Most logs are used for recording system runtime data and anonymous detection. In the past, developers (or operators) would manually search logs for terms and matching criteria. However, the amount of logs increases exponentially with the size and complexity of modern systems, rendering manual testing impractical. To reduce human labor, several anomaly detection methods for automated log analysis have been proposed. However, engineers may still choose which detection methods should not be employed because there aren't enough analyses and comparisons of different anomaly detection strategies. Furthermore, re-implementation will take a lifetime, even if engineers employ a novel detection method. To address these issues, we provide

a thorough study and assessment of six current log-based detection methods, including three monitored and three unchecked modes, along with an open toolbox that enables easy reuse. Two publicly available production log datasets containing 15,923,592 log messages and 365,298 anomaly instances were used to test these methods. We believe that the testing outcomes and related findings, together with our work, might serve as a basis for future study and as guidance for implementing these tactics.

d) AutoLog: Anomaly detection by deep autoencoding of system logs:

Since the early days of computers, system logs have been used to identify and resolve abnormalities in production systems. Despite the advancements in the field, analyzing log files from real-world systems presents a number of unique difficulties. Modern technologies, such log management and Security Information and Event Management (SIEM) packages, take use of common data formats, logging protocols, and threat signature dictionaries that barely fit industrial and custom system logs. The study of computer system logs with an emphasis on anomaly identification is the subject of this paper. The suggested method, called AutoLog, involves calculating numerical scores by sampling the logs at regular intervals. A semi-supervised deep autoencoder is trained using scores obtained under normative operations, providing a baseline for future score classification. The method does not need anomalies during training and is not limited by the structure of the underlying logs. The recall of AutoLog is between 0.96 and 0.99, while the accuracy is between 0.93 and 0.98, according to the findings obtained in identifying anomalies of two industrial systems and the public BG/L and Hadoop datasets commonly used as benchmarks. To prove the proposal's validity, a comparison with isolation forest, one-class SVM, decision tree, vanilla autoencoder, and variational autoencoder is carried out.

e) DIP: a log parser based on "disagreement index token" conditions::

Sequences of parsed log messages including the message tokens that are part of the message template are needed as inputs for some types of log analytical models, such as those for log anomaly detection. Because of this, a log parser—a software that finds the template of every message in a log file—is frequently used in such a paradigm. It has been demonstrated that even the most

precise log parsers found in the literature are unable to identify message templates from the log files of some systems with high accuracy. DIP, a tree-based log parser, is presented in this study. The way by which DIP determines whether pairs of extremely similar messages share the same template is its main methodological breakthrough. When determining whether two similar messages have the same template, many existing parsers only take into account the percentage of matching tokens between them. However, DIP also takes into account the actual tokens at which the two messages disagree, declaring a pair of similar messages to have the same template if and only if each of those tokens satisfies one of three conditions. According to our experimental findings, DIP can outperform all 13 parsers examined in a 2019 survey research on log parsers in terms of average accuracy. Additionally, we demonstrate that it attains this high accuracy without sacrificing runtime.

3. METHODOLOGY

i) Proposed Work:

The proposed system introduces an efficient log file classification approach using template mining, where raw log messages are transformed into structured event IDs. This conversion significantly reduces the size and complexity of log data, enabling faster processing and improved scalability. The system applies TF-IDF vectorization to extract meaningful features from the log data, followed by the use of SMOTE to handle class imbalance and ensure better representation of minority classes.

Multiple machine learning algorithms, including K-Nearest Neighbors (KNN), Random Forest, AdaBoost, and Decision Tree, are employed for classification. Additionally, advanced boosting techniques such as XGBoost and LightGBM are integrated to enhance performance and accuracy. The proposed approach improves classification efficiency, reduces training time, and achieves high accuracy, making it suitable for real-time log analysis and system monitoring in large-scale environments.

ii) System Architecture:

The system architecture begins with the collection of log datasets from multiple sources, which are then passed to the preprocessing stage. In this stage, the data undergoes visualization and transformation using TF-IDF vectorization to convert

textual log messages into numerical features. To address class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) is applied, ensuring balanced data distribution for effective model training. The processed data is then split into training and testing sets to facilitate model evaluation and validation.

The training data is used to build multiple machine learning models, including KNN, Random Forest, AdaBoost, and Decision Tree, along with advanced extensions such as XGBoost and LightGBM. These trained models are then evaluated using the testing data, and their performance is measured using metrics such as accuracy, precision, recall, and F1-score. The architecture ensures efficient data flow from preprocessing to model evaluation, enabling accurate and scalable log classification for real-time system monitoring.

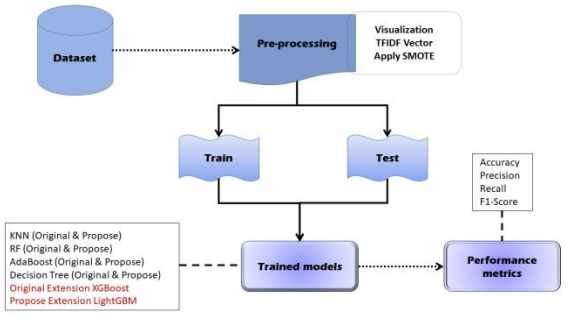

Fig1 proposed architecture

iii) Modules:

1. Dataset Loading

Loads the Real Log Classification dataset (HDFS, OpenSSH, BGL, HPC, Proxifier, Health App) into the system for processing and analysis.

2. Data Visualization

Visualizes log data distribution to understand patterns and detect class imbalance, helping in better preprocessing decisions.

3. TF-IDF Vectorization

Converts textual log messages into numerical feature vectors, capturing the importance of terms for machine learning models.

4. SMOTE (Data Balancing)

Applies Synthetic Minority Over-sampling Technique to generate synthetic samples for minority classes and balance the dataset.

5. Train-Test Splitting

Divides the dataset into training (80%) and testing (20%) sets to evaluate model performance effectively.

6. Model Training

Builds classification models using KNN, Random Forest, AdaBoost, Decision Tree, XGBoost, and LightGBM to learn patterns from log data.

7. Prediction & Classification

Uses trained models to classify new log data and predict system behavior or anomalies.

8. Performance Evaluation

Evaluates model performance using metrics such as accuracy, precision, recall, and F1-score.

9. Admin Interface

Provides a user interface for login, log upload, classification results, and system interaction through a web-based platform (Flask).

iv) ALGORITHMS:

a. K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) is a simple yet effective instance-based learning algorithm used for log classification. It works by identifying the 'k' closest data points (neighbors) to a given input based on distance metrics such as Euclidean distance. In this system, log messages converted into event IDs and TF-IDF vectors are compared with existing labeled data, and the majority class among the nearest neighbors is assigned as the output. KNN does not require a training phase, making it easy to implement, but it can be computationally expensive for large datasets due to distance calculations.

b. Random Forest (RF):

Random Forest is an ensemble learning algorithm that improves classification performance by constructing multiple decision trees during training and combining their outputs. Each tree is trained on a random subset of the data and features, which helps reduce overfitting and improves generalization. In log classification, Random Forest effectively handles high-dimensional TF-IDF features and event ID representations, making it robust against noise and variations in log data. It provides high accuracy and reliability, especially when dealing with large-scale datasets.

c. AdaBoost:

AdaBoost (Adaptive Boosting) is a boosting algorithm that combines multiple weak classifiers to form a strong classifier. It works by assigning higher weights to incorrectly classified samples in each iteration, forcing the model to focus more on difficult cases. In this system, AdaBoost improves the classification of log data by iteratively refining predictions and minimizing errors. It is particularly useful for improving performance when base classifiers alone are not sufficiently accurate, though it may be sensitive to noisy data.

d. Decision Tree:

Decision Tree is a supervised learning algorithm that uses a tree-like structure to make decisions based on feature conditions. It splits the dataset into subsets based on the most significant features, creating branches

that lead to classification outcomes. In log classification, Decision Trees use event ID-based features to generate interpretable rules, making it easier to understand how decisions are made. While it provides good accuracy and simplicity, it may suffer from overfitting if not properly controlled.

e. XGBoost (Extreme Gradient Boosting):

XGBoost is a highly efficient and scalable implementation of gradient boosting that enhances classification performance by sequentially building models to correct previous errors. It incorporates regularization techniques to prevent overfitting and handles large-scale datasets effectively. In this proposed system, XGBoost significantly improves log classification accuracy by learning complex patterns from event ID features and TF-IDF vectors. Its speed and performance make it one of the best-performing algorithms in the study.

f. LightGBM (Light Gradient Boosting Machine):

LightGBM is an advanced gradient boosting framework designed for high efficiency and fast training. It uses a leaf-wise tree growth strategy, which allows it to achieve higher accuracy with lower computational cost compared to traditional boosting methods. In this system, LightGBM is applied to the proposed dataset to optimize classification performance and reduce training time. It is particularly suitable for handling large-scale log data due to its low memory consumption and faster processing speed.

4. EXPERIMENTAL RESULTS

The proposed log classification system is evaluated using the Real Log Classification dataset, which includes log data from multiple sources such as HDFS, OpenSSH, BGL, HPC, Proxifier, and Health App. The dataset is preprocessed using TF-IDF vectorization, and SMOTE is applied to handle class imbalance effectively. The data is then split into training and testing sets to validate the performance of various machine learning models, including KNN, Random Forest, AdaBoost, Decision Tree, XGBoost, and LightGBM. Performance is measured using standard evaluation metrics such as accuracy, precision, recall, and F1-score.

The experimental results demonstrate that ensemble-based algorithms outperform traditional methods in log classification tasks. Random Forest and Decision Tree show strong baseline performance, while advanced boosting algorithms such as XGBoost and LightGBM achieve superior results, reaching accuracy levels of up to 99%. Additionally, the use of event ID conversion significantly reduces data size

and training time without compromising classification performance. The integration of SMOTE further enhances model reliability by improving classification accuracy for minority classes, making the proposed system efficient and scalable for real-time applications.

Accuracy: A test's accuracy is determined by its capacity to distinguish between healthy and ill cases. To gauge the accuracy of the test, find the percentage of examined instances that had true positives and true negatives. According to the computations:

Accuracy = TP + TN /(TP + TN + FP + FN)

$$Accuracy = \frac{(TN + TP)}{T}$$

Precision: Precision is the number of affirmative cases or the classification's accuracy rate. The following formula is applied to assess accuracy:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\Pr e\,cision = \frac{TP}{(TP + FP)}$$

Recall: A model's ability to recognise every instance of a pertinent machine learning class is measured by its recall. The ratio of accurately predicted positive observations to the total number of positives indicates how well a model can identify class instances.

$$Recall = \frac{TP}{(FN + TP)}$$

mAP: Mean Average Precision is one ranking quality metric (MAP). It considers the number of relevant recommendations and their position on the list. MAP at K is calculated as the arithmetic mean of the Average Precision (AP) at K for each user or query.

$$\boldsymbol{mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k}$$

$$\boldsymbol{AP_k = \text{the AP of class } k}$$

$$\boldsymbol{n = \text{the number of classes}}$$

F1-Score: An accurate machine learning model is indicated by a high F1 score. combining precision and recall to increase model correctness. The accuracy statistic quantifies the frequency with which a model correctly predicts a dataset.

$$F1 = 2 \cdot \frac{(Recall \cdot \Pr e\,cision)}{(Recall + \Pr e\,cision)}$$

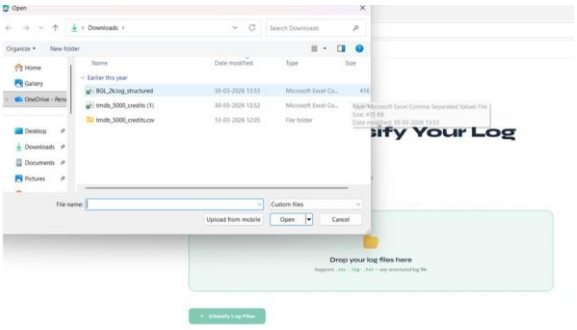

Fig2 upload input

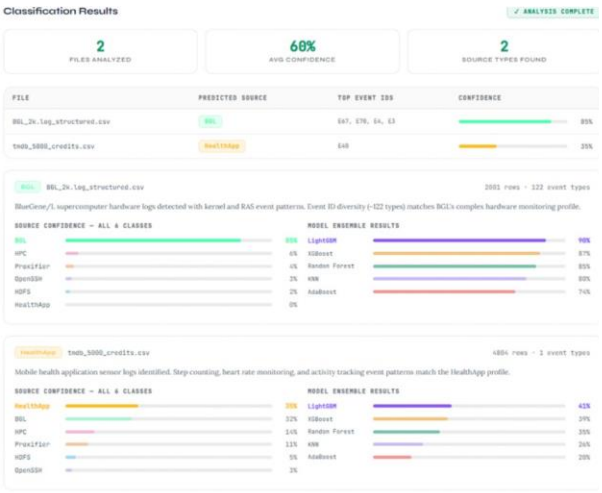

Fig3 results

5. CONCLUSION

This paper presents an efficient log classification approach using template mining by converting log messages into compact event IDs, significantly reducing data size and computational complexity. The integration of machine learning algorithms, particularly XGBoost and LightGBM, along with SMOTE for class balancing, achieves high classification accuracy (up to 99%) and improved processing efficiency.

6. FUTURE SCOPE

- Integration of deep learning models such as LSTM and Transformer for sequential log analysis and improved anomaly detection.
- Development of real-time log classification systems using streaming frameworks like Apache Kafka and Spark.
- Implementation of automated log parsing techniques to further enhance feature extraction and reduce preprocessing time.
- Deployment of the system in cloud-based and distributed environments for scalable and efficient monitoring.
- Incorporation of explainable AI techniques to improve interpretability and trust in classification results.

REFERENCES

[1] Fawzia Omer, A., Mohammed, H. A., Awadallah, M. A., Khan, Z., Abrar, S. U., & Shah, M. D. (2022). Big data mining using K-Means and DBSCAN clustering techniques. In Big Data Analytics and Computational Intelligence for Cybersecurity (pp. 231-246). Cham: Springer International Publishing.

[2] He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. ACM computing surveys (CSUR), 54(6), 1-37.

[3] Meena Siwach, D. S. M. (2022). Anomaly detection for web log data analysis: A review. Journal of Algebraic Statistics, 13(1), 129-148.

[4] Catillo, M., Pecchia, A., & Villano, U. (2022). AutoLog: Anomaly detection by

deep autoencoding of system logs. Expert Systems with Applications, 191, 116263.

[5] Plaisted, D., & Xie, M. (2022, April). DIP: a log parser based on" disagreement index token" conditions. In Proceedings of the 2022 ACM southeast conference (pp. 113-122).

[6] J. Svacina, J. Raffety, C. Woodahl, B. Stone, T. Cerny, M. Bures, D. Shin, K. Frajtak, and P. Tisnovsky, ‘‘On vulnerability and security log analysis: A systematic literature review on recent trends,’’ in Proc. Int. Conf. Res. Adapt. Convergent Syst., 2020, pp. 175–180.

[7] C.N.Kabat,D.L.Defoor, P. Myers, N. Kirby, K. Rasmussen, D. L. Saenz, P. Mavroidis, N. Papanikolaou, and S. Stathakis, ‘‘Evaluation of the elekta agility MLC performance using high-resolution log files,’’ Med. Phys., vol. 46, no. 3, pp. 1397–1407, Mar. 2019.

[8] J. Kimball, R. A. Lima, and C. Pu, ‘‘Finding performance patterns from logs with high confidence,’’ in Proc. 27th Int. Conf., Held Services Conf. Fed., SCF Web Services–ICWS, Honolulu, HI, USA. Springer, Sep. 2020, pp. 164–178.

[9] P. Jaccard, ‘‘The distribution of the flora in the alpine zone. 1,’’ New Phytologist, vol. 11, no. 2, pp. 37–50, Feb. 1912.

[10] C. D. Cantrell, Modern Mathematical Methods for Physicists and Engineers. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[11] Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, ‘‘Experience report: Deep learning-based system log analysis for anomaly detection,’’ 2021, arXiv:2107.05908.

[12] B. Waggener and W. N. Waggener, Pulse Code Modulation Techniques. Berlin, German: Springer, 1995.

[13] H.Guo, J.Yang, J.Liu, J.Bai, B.Wang, Z.Li, T.Zheng, B.Zhang, J.Peng, and Q. Tian, ‘‘LogFormer: A pre-train and tuning pipeline for log anomaly detection,’’ in Proc. AAAI Conf. Artif. Intell., Mar. 2024, vol. 38, no. 1, pp. 135–143.

[14] M.DuandF.Li,‘‘Spell: Streaming parsing of system event logs,’’ in Proc. IEEE 16th Int. Conf. Data Mining (ICDM), Dec. 2016, pp. 859–864.

[15] K. Shima, ‘‘Length matters: Clustering system log messages using length of words,’’ 2016, arXiv:1611.03213.

[16] Y. Freund and R. E. Schapire, ‘‘A decision-theoretic generalization of on line learning and an application to boosting,’’ J. Comput. Syst. Sci., vol. 55, no. 1, pp. 119–139, Aug. 1997.

[17] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, ‘‘Tools and benchmarks for automated log parsing,’’ in Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Softw. Eng. Pract. (ICSE-SEIP), 2019, pp. 121–130.

[18] M.Nagappan, ‘‘Analysis of execution log files,’’ in Proc. ACM/IEEE 32nd Int. Conf. Softw. Eng., vol. 2, May 2010, pp. 409–412.

[19] J. H. Andrews, ‘‘Testing using log file analysis: Tools, methods, and issues,’’ in Proc. 13th IEEE Int. Conf. Automated Softw. Eng., 1998, pp. 157–166.

[20] F. Ertam and M. Kaya, ‘‘Classification of firewall log files with multiclass support vector machine,’’ in Proc. 6th Int. Symp. Digit. Forensic Secur. (ISDFS), Mar. 2018, pp. 1–4. 96386

Author Profiles

Ms. K. Baby Ramya is working as an Assistant Professor in the Department of MCA at SRK Institute of Technology, Enikepadu, Vijayawada, NTR District. She completed her MCA from Krishna University. She has around 3 years of teaching experience at SRK Institute of Technology. Her areas of interest include Machine Learning, Data Science, and Computer Applications.

Mrs. K. Pavani is working as an Assistant and Head of Department of MCA, in SRK Institute of technology in Vijayawada. She completed her MCA and M.Tech in Computer Science. She has 10 years of teaching experience in SRK Institute of technology, Enikepadu, Vijayawada, NTR District. Her areas of interest include AI and ML, etc.

Ms.M.Sreevalli is an MCA Student in the Department of Computer Application at SRK Institute Of Technology, Enikepadu, Vijayawada, NTR District. She has Completed Degree in BCA from Krishna University. Her area of interests are DBMS and Machine Learning with Python.