INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 5, May 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Machine Learning Framework for Intelligent Log Anomaly Detection and Root Cause Analysis in IT Infrastructure

1st. Nancy E
Department of Advanced Computing and Analytics
Vels Institute of Science, Technology & Advanced Studies
Chennai, India
nancyenancye@gmail.com

2nd. B. Kamatchy,
Assistant professor,
Department of Advanced Computing and Analytics
Vels Institute of Science, Technology & Advanced Studies
Chennai, India
kamatchi6282@gmail.com

***Abstract*- Modern IT infrastructures generate large volumes of log data.This log data is generated by various components of the system like servers, applications,and network devices. Log data produced contains crucial information regarding the system's behavior and failures.Analysis of the huge volume of log data is a time-consuming and inefficient process. This paper proposes a system based on the machine learning approach to detect log anomalies and perform root cause analysis of the system failures.The system processes the log data and converts the log data into numerical values using the Term Frequency-Inverse Document Frequency algorithm.Finally, the system uses the Isolation Forest algorithm to analyze the numerical values and detect anomalies in the system.After detecting the anomalies in the system, the system uses the Random Forest classifier to determine the potential root cause of system failure like memory overflow, disk failures, network failures, and security issues.Additionally, the system provides a facility to monitor the system's logs using the visual representation of the system's logs in the form of charts and graphs.The result of the experiment proves the system's ability to detect log anomalies and help system administrators to detect system failures.**

Keywords
Machine Learning, Log Analysis, Anomaly Detection, Isolation Forest, Random Forest, Root Cause Analysis, System Monitoring

1. INTRODUCTION

In the field of modern computing, we require complex infrastructures like servers, distributed architecture, cloud computing, and applications. These complex infrastructures are generating huge amounts of log information on various activities, failures, warnings, and events. Log information plays a vital role in monitoring various performances, failures, and security. With the increase in the level of complexity in these infrastructures, there is an explosion in the log information generated. It is almost impossible for the IT administrators to go through all the log information. Log information monitoring techniques are either rule-based or keyword-based techniques. These techniques are able to monitor failures in the infrastructures but are not able to monitor unknown failures. Machine learning is the best technique to solve this problem in log information monitoring. Machine learning techniques are able to identify both normal and abnormal log information without any rules.

This will help the IT administrators to identify failures in the system in advance and take appropriate actions to prevent failures in the future. In the study, the researchers are able to propose a machine learning model that can automatically scan the log information in the system

2. LITERATURE SURVEY

Log Analysis has recently gained recognition as an invaluable research area because understanding how a computer system operates; keeping it secure, and monitoring its performance are becoming increasingly important in today's world. Given that log analysis provides substantial insight into the operational characteristics of quite large computer systems, particularly with respect to cloud-based service providers and networks, it is clear that log data in its unstructured form can pose challenges for deriving value through log analysis.

Historical approaches to conducting log analysis primarily centered on either utilizing rules or toolkits to ascertain general patterns within the log files themselves. For example, recurring log entries in console logs may indicate multiple types of issues(this point has Furthermore, research studies show this applicability). However, this approach can only be beneficial when conducting retrospective analysis of log files given the relatively small amount of "raw" data utilized and the fact that the identification of patterns requires interpretation. Xu et al. have completed comparative studies demonstrating that clustering and pattern mining techniques may be applied to uncover hidden structures within log data. While these techniques enhance the accuracy of detection, they will require a significant amount of preprocessing and feature engineering before performing log analysisData mining algorithms are especially apparent in this domain.

Outlier detection is one of the most popular techniques for detecting abnormal data points that differ from the usual patterns of occurrence within a data set. The primary application of outlier detection techniques, according to Aggarwal, is to detect unusual and critical system events. The development of the Isolation Forest algorithm, as discussed by Liu, et al., has also become a popular

unsupervised method for detecting anomalous values within a very large data set by separating (or isolating) them from other values via random partitions of the data . This is much different than traditional distance-based methods, which rely on the estimation of density to identify outliers, and the ability to quickly and easily scale to very large log files makes it, as noted in numerous papers, the most well-liked choice for determining system log anomalies with a high level of precision and less processing cost . As with anomaly detection, classification is often another technique for determining the source of a system anomaly; yet according to Breiman, Random Forest is considered one of the most efficient algorithms available for ensemble learning .

By building a prediction model using multiple decision trees, this method provides increased prediction accuracy and reduced overfitting versus other machine learning approaches (i.e., classifiers) for the analysis of log files to identify broken systems.The benefits of log parsing techniques for detecting anomalies in unstructured log data have been widely established. As reported in a collective study by He, Chang, and Chen, various techniques exist for parsing log files, such as TF-IDF (Term Frequency – Inverse Document Frequency).

3. METHODOLOGY METHOD OF PROPOSED

The methodology of proposed includes actions to review logs from the system in order to determine if they contain any anomalies.

A. System architecture

Figure 1: Architecture of the proposed machine learning-based log anomaly detection system, illustrating the sequential processing pipeline from log collection to anomaly detection and alert generation.

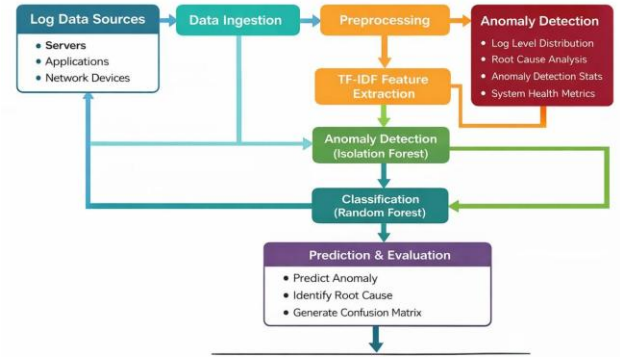


Figure 1

B. Working Methodology.

1. Log Data Collection.

log data is collected from the Github and kaggle.Most of the log data comprises log times,log level server IDs,and log messages.

.

2. Data Preprocessing

Raw log messages have many undesirable contents which include numbers, special characters, redundant text and many more. Machine learning algorithms need to be applied to the data that has to be cleaned by preprocessing.

The follows preprocessing steps are

- Changing the text to lower case.
- Eliminating characters and figures.
- Removal of redundant areas.
- Standardizing log messages.

All these are carried out to make sure that the text data is standard and more readily analyzed

.

3. Feature Extraction

The log messages are to be translated into numerical forms because machine learning algorithms does not care of textual data directly. The log messages are converted into feature vectors by waying of TF-IDF vectorization. TF-IDF is used to determine how significant words in a document are when compared to the whole dataset. Frequently used words in particular pieces of log messages are assigned greater levels of importance, and machine learning models have the ability to learn meaningful patterns

4. Isolation Forest to Detection Anomalies.

The Isolation Forest algorithm is used to detect any anomalies in the log dataset after the extraction of features. Isolation Forest is used by randomly choosing features and dividing the data until each individual observation is isolated. Abnormalities are normally clustered into fewer partitions as they are widely differentiated to the normal observations.

Root Cause analysis after the anomalies are identified, the Random Forest classifier attempts to establish the potential underlying cause of the anomaly. Random Forest tries to construct a number of decision trees and then make decision by mixing the outcomes of the decision trees in order to make a final classification decision.the classification tool classifies anomalies into a few possible causes as Root Cause Analysis After the anomalies are identified, the Random Forest classifier attempts to establish the potential underlying cause of the anomaly. Random Forest tries to construct a number of decision trees and then make decision by mixing the outcomes of the decision trees in order to make a final classification decision.

The classification tool classifies anomalies into a few possible causes as:

- Memory Overflow
- Disk Failure
- Network Issue
- Security Attack
- Application Error
- Visualization Dashboard

5. Visualization Dashboard

To assist the administrators in interpreting the outcomes, the system comes with a monitoring dashboard, which is developed with the Streamlit tool and has HTML and CSS styling. Some of the visualizations found in the dashboard include:
1. Log level distribution
2. .Root cause analysis charts
3. Statistic of anomaly detection.
4. Timeline plots of the anomaly tendencies.
5. Server error heatmaps

Through this interface, administrators can gain a quick insight into the behavior of the system and the problems with it.

A. Architecture Description
The architecture components and their functions are summarized in Table I.

TABLE I SYSTEM ARCHITECTURE DESCRIPTION

Step	Component	Function
1	Log Collection	Gather logs from multiple sources
2	Preprocessing	Clean and standardize text data
3	Feature Engineering	Transform logs using TF-IDF
4	Anomaly Detection	Identify abnormal patterns
5	Classification	Determine root cause of anomalies
6	Visualization	Present insights via dashboard

PSEUDOCODE AND IMPLEMENTATION

Source Data Log Files

Source data log files are derived from a variety of IT infrastructure components, including servers, applications, networks, etc. They are raw data with minimal organization and are typically composed of different data types, including time stamps, log severity levels (such as info, warning, error), and log sources or messages. They are raw data that needs to be cleaned and normalized prior to any processing. They can be obtained either directly from log files or through a real-time log system. The log data can be either in the format of "in.log" or "csv." Each log data represents a data point that needs to be used to train the machine learning model.

A. Pseudocode

Step 1: Load log dataset
Step 2: Preprocess logs
- Convert text to lowercase
- Remove special characters
- Remove stop words
- Remove duplicate entries
Step 3: Using TF-IDF to extract feature.

- Convert log messages to vectors containing numbers.

Step 4: Train Isolation Forest model.
- The Fit model runs on TF-IDF elements.
- Decode anomaly scores for every log.

Step 5: Identify anomalies.
IF anomaly score < threshold THEN Label as Anomaly
ELSE
Label as Normal END IF
Step 6: Train Random Forest classifier.

- Use labeled anomaly data
- Learn classification patterns

Step 7: Predict root cause.
- Memory Error
- Disk Failure
- Network Issue
- Security Attack
- Application Error
-
Step 8: Visualize results.

- Display charts and graphs on dashboard
End

B.Implementation

The log analysis system has been built with the assistance of the Python programming language and many different machine learning approaches. The following describes how this system performs its functions: The system obtains log data from servers, applications and network devices; since log data is not structured, preprocessing occurs independently for each log set; preprocessing involves lowercase conversion of log data use to remove special characters, stopwords and duplicate entries. Then, during feature extraction, log data is converted into numeric data through the TF-IDF method.

Anomalous log entries are detected using the Isolation Forest algorithm, which generates an anomaly score for every log entry and uses a threshold value to identify log entries as either "normal" or "anormalous."

Once classified as an anomaly, the log entries generated during the identification process can then undergo root cause analysis by applying the Random Forest algorithm for classification of anomalous logs as memory failures, disk failures, network issues, security breaches, or application errors. The Random Forest Classifier is an ensemble

classifier that builds multiple classifiers to increase its accuracy and reliability for classification.

The dashboard is a combination of log level charts and anomaly trend charts and will allow administrators to visually see the information of their logs. It will allow for root cause analysis of the logs based on previously known anomalies.

RESULT

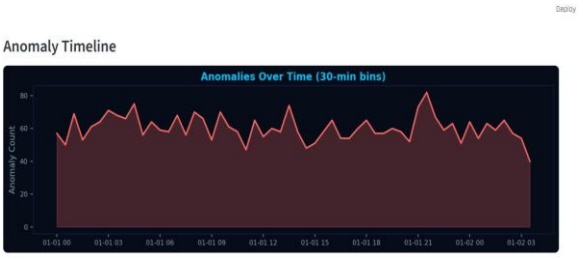
The final outputs of the system will allow for users to identify specific anomalous log entries, as well as the root cause(s) for those anomalous log entries.
The classification of logs, when either a log is Normal or Anomalous, is also done on an individual log-by-log basis based on the Anomaly Detection process described previously.

Once the logs are identified as being Anomalous by the Anomaly Detection process, the system performs an additional form of Root Cause Analysis to determine the root cause(s) of each log entry.

While the classifications of logs will be the same, the nature of the logs classified as Anomalous will be different and may include any of the following types of errors: memory errors, disk errors, network errors, security incidents, and application errors. This information will allow administrators to determine the reason for the abnormal behaviour of the system.

As well as a classification of logs, the system will also provide an output in the form of graphics or visualisation of logs.Each user will have the ability to view logs historically along with the trend(s) of the types of logs historically and anomaly types to demonstrate how logs are correlated with the causes of logs that are potentially deemed as being Anomalous.

That is to say, as mentioned earlier, this output will allow administrators to identify their systems anomalies and to rectify them, increasing the reliability of those systems.



I. RESULTS AND DISCUSSION

Summary of Log Data
There is a lot of log data, both normal and abnormal. So, there's enough data to be analyzed.

The temperature anomaly time series data shows several trends over time:

- - High-frequency y time periods.
- - High-frequency y periods.

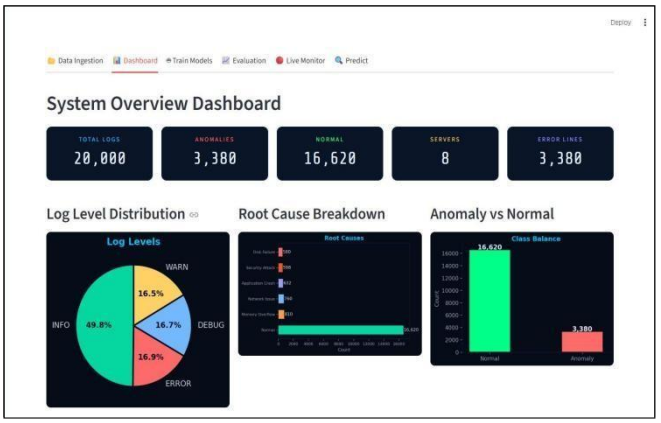
Another way of analyzing the data is by considering all the servers or the data at the server level, using a heat map.

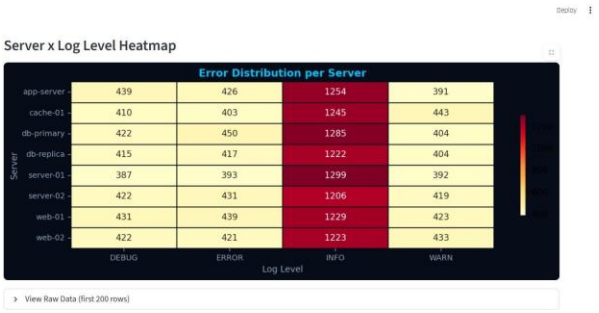
The heat map can be used to illustrate the participation of the components of the server.

A. Dashboard Insights

The dashboard provides:

- Total log anamolies
- Log level distribution
- Root cause breakdown





The combination of both supervised and unsupervised learning culminates into a very efficient process. The first characteristic of the Isolation Forest algorithm is its capability to detect unusual patterns. The effectiveness of this algorithm can be attributed to several important factors, including log data and samples.

4. CONCLUSION

The paper introduces a machine learning method for the detection of log anomalies and the reasons for the detection of the anomalies. The method is efficient for the detection of log anomalies by using TF-IDF for feature selection, the Isolation Forest method for the detection of anomalies, and the Random Forest method for classification.

The experiment results show that the method is efficient for the detection of log anomalies,
which helps in saving time and improving the overall performance of the monitoring process. Therefore, the method introduces a promising approach to the more efficient monitoring of distributed systems, which is a major challenge in the distributed computing environment of today's world. The method can be improved in the following ways in the near future.

5. REFERENCES

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, 2009.

[2] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation Forest," in *Proc. IEEE ICDM*, 2008, pp. 413–422.

[4] X. Xu, L. Chen, A. Fox, and D. Patterson, "Large-Scale System Log Analysis," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[5] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs using Deep Learning," in *Proc. ACM CCS*, 2017, pp. 1285–1298.

[6] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting Large-Scale System Problems by Mining Console Logs," in *Proc. ACM SOSP*, 2009, pp. 117–132.

[7] J. Lin, Q. Fu, H. Wei, J. Lou, and T. Zhang, "Log Clustering Based Problem Identification for Online Service Systems," in *Proc. IEEE ICSE*, 2016, pp. 102–111.

[8] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience Report: System Log Analysis for Anomaly Detection," in *Proc. IEEE ISSRE*, 2016,pp. 207–218.

[9] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An Evaluation Study on Log Parsing and Its Use in Log Mining," in *Proc. IEEE DSN*, 2016, pp. 654–661.

[10]Q. Fu, J. Lou, Y. Wang, and J. Li, "Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis," in *Proc. IEEE ICDM*, 2009, pp. 149–158.

[11]H. Dai, H. Li, C. Chen, and W. Wang, "Log-based Anomaly Detection Using Machine Learning Techniques," *IEEE Access*, vol. 8, pp. 146719–146731, 2020.

[12]Y. Zhang, X. Chen, and D. Zhang, "Robust Log-Based Anomaly Detection on Unstable Log Data," *IEEE Transactions on Reliability*, vol. 69, no. 3, pp. 987–1000, 2020.

[13]G. Lou, X. Zheng, and J. Li, "Log-Based Anomaly Detection with Feature Extraction and Clustering," *IEEE Access*, vol. 7, pp. 107940–107950, 2019.

[14]S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory,"
Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[15]D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. ICLR*, 2015.

[16]I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[17]A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proc. NeurIPS*, 2019, pp. 8024–8035.

[18]F. Pedregosa et al., "Scikit-learn: Machine Learning in Python,"
Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[19]J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proc. OSDI*, 2004, pp. 137–150.

[20]C. C. Aggarwal, *Outlier Analysis*, Springer, 2017.