

LADDM: A Flexible Tool for Log Anomaly Detection

Gianni D'Angelo(✉), Raffaele Esposito, Mariano Caiazzo, and Francesco Palmieri

University of Salerno, Fisciano, Italy
giadangelo@unisa.it

Abstract. Log files keep track of everything that happens when services are used, and looking through them is important for finding any threats or problems that might come up because of mistakes. This paper looks at different cutting-edge ways to find log anomalies and presents a new one called LADDM (Log Anomaly Detection as a Docker Microservice). LADDM was made with a focus on the importance of logs at the enterprise level. To see how well it works, a synthetic dataset simulating enterprise logs was made. The solution is set up in a Dockerized environment, which makes sure it runs on its own and doesn't depend on any other services.

This approach not only highlights the importance of efficient log analysis in safeguarding enterprise operations but also demonstrates the practicality and scalability of containerized solutions in addressing security challenges.

Keywords: Anomaly Detection · Log Analysis · Machine Learning · Deep Learning · Log Parser

1 Introduction

The Clusit report [1] - Report on Cybersecurity in Italy and in the world - shows that in 2024 incidents worldwide had increased by 36% compared to 2023. The global trend for the first half of 2025 shows a further growth, equal to +36% compared to the previous half of the year.

In the first half of 2025, not only did the frequency of incidents increase, but also their average severity. In 2024, incidents with a "Critical" or "High" impact accounted for 77% of the total. In the first half of 2025, the impact increased further compared to 2024 (82% of incidents with a critical or high severity). The sector with the highest number of cyber incidents in Italy is the Government/Military/Law Enforcement sector (38% of the total), with a 279% increase compared to the entire previous year.

Check Point Research, in January 2026, released the Global Threat Intelligence Report [2] which shows that Italy records 2,400 cyber attacks per week, significantly higher than the global average. Europe is highly vulnerable from a cyber perspective, paying a very high price. New research conducted by Howden [3], a global insurance broker, has estimated the losses suffered by the four major European economies—Italy, France, Germany, and Spain—due to cyber attacks between 2020 and 2025 at €307

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16759, pp. 592–606, 2027.
https://doi.org/10.1007/978-3-032-30518-3_37

billion. The resulting situation is alarming and we must do everything possible to prevent and block these attacks by limiting and reducing the risks and resulting damage.

In recent years, cybersecurity systems based on log anomaly detection have been proliferating. These systems identify logs that highlight unexpected and potentially dangerous activity, allowing timely intervention to prevent or reduce system damage and avoid negative cascading effects.

Logs are chronological and sequential records of operations performed by a user or computer system. Log files are stored in systems and remain available for monitoring and administration. They are essential for IT security as they allow the tracking and recording of activities performed by users or other systems, and are often used in the event of litigation.

Each type of log plays a specific role in managing and monitoring IT systems, enabling system administrators to identify and resolve problems, ensure data security, and monitor user and application activity.

Specifically, applications also generate logs when the system enters an unexpected or undesirable state, there are availability issues, or security incidents. We can use these logs, which highlight unexpected and potentially dangerous activities, to take timely action to prevent or mitigate system damage and avoid negative cascading effects.

A complex system typically produces a huge amount of logs. For example, Alibaba Inc.'s cloud computing systems produce approximately 30–50 gigabytes (about 120–200 million lines) of tracking logs per hour, typically in the form of unstructured text messages [4].

It is essential to adopt anomaly detection techniques capable of automatically learning patterns describing normal system behavior, so that any deviations from these patterns can be identified and flagged as potential malicious activity requiring operator intervention. Anomaly detection is a methodology used to identify atypical events or data within a data set.

2 Background

2.1 Anomaly Detection

Anomalies are patterns in data that do not conform to a well defined notion of normal behavior [5] (Chandola et al., 2009).

Anomaly detection is a process used to identify behaviors that significantly deviate from expectations [5] (Chandola et al., 2009). In other words, it involves finding exceptions or deviations from the norm within a dataset. This process is fundamental in many fields and applications, such as cybersecurity, fraud detection, predictive maintenance, and system health monitoring.

Identifying real anomalies is not easy because false positives or data noise are often found. A balance must be struck between identifying too many anomalies (false positives) and failing to identify real anomalies (false negatives). This activity is important for preventing system security attacks and serves to optimize decision-making processes by anticipating and mitigating potential risks, improving operational efficiency.

There are two main ways to find anomalies: outlier detection and novelty detection.

In the first instance, identifying extreme or atypical values that deviate from the normal distribution of data frequently signifies measurement inaccuracies, extraordinary variations, or potential security risks.

In the second case, novelty detection focuses on the discovery of patterns or instances that have not been previously observed in the training dataset and may indicate system evolution or the presence of new scenarios or threats.

2.2 Log Anomaly Detection Techniques

Log anomaly detection techniques can vary depending on the specific needs and characteristics of the system under consideration. However, they generally involve analyzing patterns in log data using machine learning algorithms, statistical analysis, or rule-based anomaly detection techniques. By applying advanced data analytics, artificial intelligence, and machine learning techniques, cybersecurity experts and system administrators can promptly identify significant deviations from normal system behavior and take appropriate action.

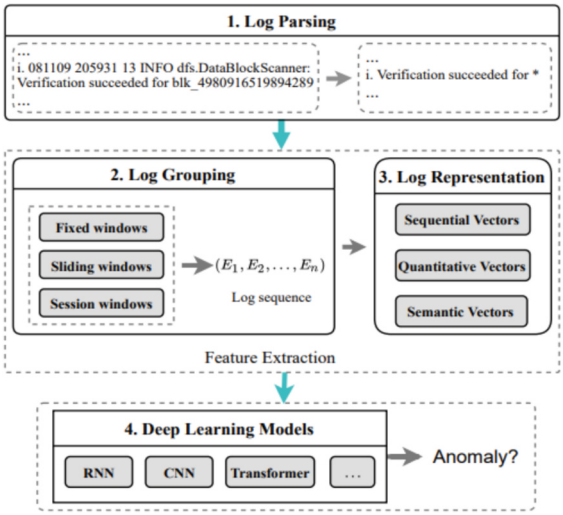

Fig. 1. Workflow Deep Learning models [4]

Figure 1 illustrates the typical workflow adopted by the most widely used deep learning approaches for log anomaly detection [4]. This process is structured into four stages:

1. **Log Parsing**. Log parsers process raw log entries at first to get log templates. This step makes the data uniform by turning each log message into a structured template. This lets you tell the difference between constant elements and variable parameters. The logs are now easier to read and better for further analysis.
2. **Log Grouping**. After being parsed, the logs are put into groups or sessions using methods like fixed windows, sliding windows, or session-based windows. This step

organizes the data so that it shows event sequences or operational sessions. This is important for finding temporal relationships between log entries.

3. **Log Representation**. The log sequences are turned into numbers after they are grouped. These representations can hold sequential, quantitative, or semantic information, depending on what you want to learn from the analysis. To make the data work with deep learning algorithms, you need to change logs into vector form.
4. **Deep Learning Models.** In the last step, the numerical vectors are used to teach deep learning models to tell the difference between normal and abnormal log sequences. These models learn about patterns and trends in the data through training. This lets them find problems by using their ability to model complex relationships.

3 Related Work

Several deep learning–based approaches have been proposed for detecting anomalies in system logs [4]. One of the most notable methods is DeepLog, which employs an LSTM model to learn the normal execution patterns of a system by analyzing sequences of log events. The model predicts the next log event based on previous ones, and an anomaly is detected when an incoming event does not match the predicted outcome. This approach has demonstrated high effectiveness on the HDFS dataset, achieving an F-measure of 0.96.

DeepLog [6] were the first to propose an approach based on deep learning that enables detection of anomalous event sequences in log data.

Another related approach is LogAnomaly [7], which combines log count vectors with an LSTM model. In this method, log templates are represented as semantic vectors generated through a technique based on synonym and antonym relationships. The model predicts the next log event and flags any event that deviates from the predicted result as anomalous. The experimental results are very good, with F-measures of 0.95 on the HDFS dataset and 0.96 on the BGL dataset.

PLELog [8] deals with the problem of having too little labeled data. It uses a probabilistic label estimation method to make up for missing annotations and an attention-based GRU neural network to find strange patterns in log sequences. The model sorts sequences into two groups: normal and strange. It has shown to work better than other semi-supervised methods on both the HDFS and BGL datasets.

LogRobust [9] is another method that focuses on making logs more stable and less prone to change. It uses TF-IDF weighting and a pre-trained Word2Vec model (FastText) to learn vector representations of log templates. An attention-based Bi-LSTM model then processes these vectors, which makes the system better at finding problems even when log events are different.

There are methods that use Convolutional Neural Networks (CNNs) [10]. In this case, logs are put into sessions and turned into a trainable matrix that the CNN model can use as input. The network learns to tell the difference between normal and strange log sequences. This method worked especially well on the HDFS dataset, getting an F-measure of 0.98.

The difficulties in enhancing the efficacy of deep learning models for log anomaly detection indicate potential avenues for future research and development.

The study of these typical models of the contemporary state of the art has elucidated the obstacles in enhancing their efficacy in detecting log anomalies:

- Training data selection: The way you choose your training data can have a big effect on how well your model works. Random selection can cause data leakage problems, where the training set may have data from the future, making the models less useful in real life.
- Log data clustering: the right window size for log clustering is critical. Wrong sizes can make it difficult to capture anomalies
- Unbalanced class distribution: An unbalanced class distribution, where anomalous log sequences make up a small part of the dataset, can make it much harder to find anomalies
- Data quality: Noise in the data, like false positives and false negatives that happen during the manual labeling process, can make the model work worse. Also, errors in log parsing can add wrong log events, which makes it harder to find anomalies.

Deepcase [11] is an innovative system designed to enhance cybersecurity monitoring. It uses deep learning techniques to analyze security events within their context, with the goal of reducing the workload of security operators by automating the correlation of security events.

This approach allows the system to filter a large portion of events, decreasing the need for manual analysis while improving the accuracy of threat detection. Additionally, it provides detailed explanations of why certain events are classified as malicious.

The deep learning model is made to find and study complicated relationships between sequences of events, both new and old. The model uses an advanced similarity function to group event sequences and show why different sequences are thought to be similar. This method lets security operators focus on a small number of representative sequences for each group. This makes it easier for them to decide what to do and cuts down on their work by a large amount.

LogAnomaly and Deeplog are just two of the many tools used to find strange things in log data. But the current methods have a big problem: they use predefined log patterns to train their models, and these patterns usually only work well in environments that are stable and predictable.

Isolation Forest [12], Random Forest [13], K-Nearest Neighbors [14], density-based clustering [6], and Template-to-Vec [7] are some of the most popular clustering-based methods. These methods frequently encounter difficulties associated with restricted accuracy, elevated false positive rates, and significant computational expenses.

Du et al. [4] present a technique that integrates a log parser with an LSTM network to autonomously detect patterns during standard system operations. This method is adaptable and can be used in a variety of situations. However, it has a big problem: it has trouble finding anomalies when log patterns change a lot from what it learned during training.

Guo et al. [15] introduced BERT, a model that learns sequence patterns using two new self-supervised learning methods. This model does a good job of finding differences between log sequences that are expected and those that are not. But it can't generalize well when it sees data that is very different from what it saw during training.

Liu et al. [16] use a self-attention neural network to find problems and help people make decisions. One of the best things about this method is that it uses data augmentation techniques to set decision thresholds using only the normal training data that is already available.

Nipa et al. [17] performed a comparative analysis of current machine learning and deep learning methodologies for anomaly detection by assessing their efficacy in identifying anomalies across various metrics and efficiency criteria.

The results show the best way to find problems in this system, which lets you do maintenance ahead of time and makes the system more reliable overall.

4 Our Solution

We came up with the name LADDM (Log Anomaly Detection as Docker Microservice) for our system. Because it is built on containers, it is a log anomaly detection system that is made for business settings.

Containers are standardized packages that hold an application's code and all the parts it needs to work. This makes it possible for the app to run quickly and reliably in any setting, no matter what operating system, libraries, or runtime versions are installed.

Virtualization focuses on the operating system instead of the hardware when using containers. Each container has only the application and the libraries it needs to work, so it doesn't need a full guest operating system. Because of this, containers are much lighter than regular virtual machines, which makes them faster and easier to move around.

We made a container-based architecture using Docker [18], which is an open-source platform. Docker lets applications run separately from the infrastructure they are built on, which makes it easy and quick to deploy software. Docker makes sure that development, testing, and production environments are all the same by putting an application and all of its dependencies into a standard container that can run on any system with Docker installed. The client-server architecture, in which the client interacts with the Docker daemon, which manages containers, images, and volumes, makes resource management more efficient and keeps processes separate.

In this case, one container acts like the business environment by making logs at regular intervals, while the other container uses and tests the anomaly detection models, separating and reporting logs that are out of the ordinary. LADDM's modular design gives users a lot of freedom. They can easily add or remove anomaly detection models by just adding the new model definition to the system.

Furthermore, LADDM supports integration with **DeepCase** and leverages models such as **IsolationForest** with semantic vector transformation and **RandomForest** with pattern recognition. To validate LADDM, a synthetic dataset designed to reflect an enterprise environment was used, where each record follows the format:

$$\text{"Action"}\ \textit{generated}\ \text{"}x\text{"}\ \textit{bytes from User}\ \text{"}Y\text{"}$$

In this context, "Action" indicates the operation performed by user "Y," resulting in the generation of "x" bytes. To identify anomalous logs, each action in the dataset produces a quantity of bytes that varies randomly within a predetermined range for that specific action. A log is considered anomalous when the volume of data generated by an

action is N times greater than the norm, thus providing a clear and quantifiable criterion for distinguishing anomalies within the dataset.

LADDM Structure. The operation of LADDM is described in Fig. 2.

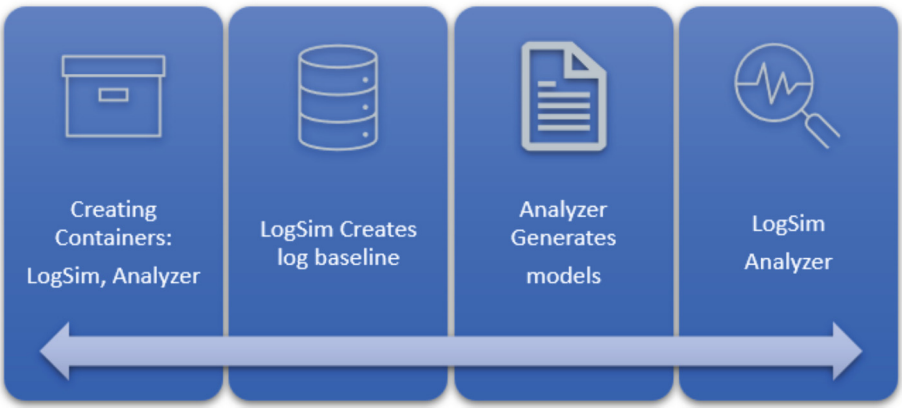

Fig. 2. LADDM Structure.

LADDM operates through a well-structured process that begins with the establishment of two Docker containers, named **LogSim** and **Analyzer**, each configured with a dedicated dockerfile detailing the dependencies required for their operation.

The **LogSim** container performs two crucial tasks: first, it generates an initial set of logs that will serve as a baseline for training anomaly detection models, and then it simulates the operation of an application by producing periodic logs.

LADDM takes on the role of model management: it acquires the log baseline, trains the anomaly detection models, and, if necessary, creates new ones. Next comes the monitoring phase: LADDM allows you to select specific log ranges to analyze and decide which models to use for the scan. If all available models are used, the analysis proceeds in parallel, optimizing the efficiency and effectiveness of the anomaly detection process.

5 Experiments

A Log Anomaly Detection Tool can become a key element for cybersecurity. Experimental evaluation is a crucial part of the development and validation of such systems and allows us to understand their effectiveness and validity, assessing their reliability, especially when dealing with complex and highly heterogeneous data. In this work, the evaluation is performed using two complementary types of datasets: a synthetic one, generated through log simulation, and a real one, HDFS_v1. This methodological choice combines the advantages of experimental testing with the benefits of analyzing realistic scenarios.

The synthetic dataset allows for a controlled environment where anomalies can be introduced explicitly and parameterizably. This approach allows for systematic variation in the type, frequency, and distribution of anomalies, thus offering the ability to analyze system behavior under specific conditions and validate design assumptions.

The HDFS_v1 dataset is a standard dataset in the literature for evaluating log anomaly detection techniques, and is extracted from Hadoop Distributed File System (HDFS) logs. It contains millions of log lines generated by a real cluster and organized into sequences associated with specific operations (e.g., read or write requests to data blocks). The dataset has an important feature: it has sequence-level labels that identify each session as normal or anomalous based on the outcome of the operation. Anomalies are typically caused by system failures or operational errors, so the dataset is highly representative of real-world scenarios. Furthermore, HDFS_v1 exhibits high structural variability, noise in log messages, and a highly skewed distribution between normal and anomalous samples, which increases its complexity and makes it a valuable benchmark for evaluating the robustness and generalization of models.

The use of synthetic and real datasets in the experimental evaluation phase allows for a more complete and reliable analysis, helping to highlight the strengths and limitations of the proposed log anomaly detection system.

The first step is to generate two containers: **LogSim**, which is oriented towards business operations, and **Analyzer**, which focuses on anomaly detection models and the ability to interact with **LogSim**. The docker file used is shown Fig. 3 .

```
1. FROM      python:3.10-slim
2. RUNapt-get update
3. && apt-get install -y --no-install-recommends \
4. build-essential \
5. libssl-dev \
6. libffi-dev \
7. python3-dev \
8. bash \
9. && rm -rf /var/lib/apt/lists/*
10. COPY      ./requirements.txt .
11. RUN       pip install --upgrade pip && \
12. pip install -r requirements.txt
13. RUN       rm ./requirements.txt
```

Fig. 3. Docker file.

To optimize the management and control of these containers, Docker Compose was adopted. This YAML configuration file allows you to orchestrate the startup and coordination of services with a single command, significantly simplifying the administration of the associated components, networks, and volumes.

After the containers have been created and launched, the next step is to develop a log baseline, which will serve as the training dataset for the anomaly detection models. A dedicated log generation function produces random logs following the structure outlined

in the introduction. Specifically, one thousand logs were generated incorporating four distinct action types

—"LOGIN," "LOGOUT," "CREATE," and "EDIT"—

performed by a set of twenty-six different users, ranging from "User_A" to "User_Z."

Each action type is associated with a defined range of generated bytes. Normal logs are marked with a "1," while anomalous logs, created with a 10% probability, are marked with a "-1." A log is considered anomalous when the number of bytes generated exceeds N times the typical range for that specific action.

The generated data is organized in a CSV file, structured into four main columns:

- "event" describing the log,
- "machine" indicating the log source (in this case, a single machine),
- "timestamp" recording the time the log was generated,
- "label" classifying the log as normal or abnormal.

Fig. 4 shows an example of what a generated file looks like.

```
1. event,machine,timestamp,label
2. Login generate 73 bytes performed by USER_C,1,1705949473,1
3. Edit generate 2000 bytes performed by USER_P,1,1707677473,1
4. Login generate 56 bytes performed by USER_V,1,1706122273,1
5. Edit generate 6528 bytes performed by USER_X,1,1707591073,1
6. Create generate 54290 bytes performed by USER_R,1,1707677473,1
7. Create generate 89030 bytes performed by USER_M,1,1706554273,1
8. Edit generate 2451 bytes performed by USER_T,1,1707331873,1
9. Logout generate 279 bytes performed by USER_V,1,1707159073,1
10. Edit generate 5159 bytes performed by USER_U,1,1706381473,1
11. Logout generate 819 bytes performed by USER_E,1,1706813473,1
12. Login generate 51 bytes performed by USER_B,1,1707331873,1
13. Logout generate 808 bytes performed by USER_V,1,1706035873,1
14. Logout generate 56700 bytes performed by USER_N,1,1705863073,-1
15. Login generate 21 bytes performed by USER_W,1,1705690273,1
16. Login generate 89 bytes performed by USER_W,1,1705603873,1
17. Login generate 4 bytes performed by USER_P,1,1706554273,1
```

Fig. 4. Example of baseline.

After generating the baseline file in CSV format, the next step is to develop the anomaly detection models. For each selected model, a dedicated directory is created to store the developed models. In this particular analysis, three separate directories were set up for the various models:

- "deepcase" for the DeepCase model,
- "if" for Isolation Forest, and
- "rf" for Random Forest.

Once the model generation is complete, you can activate the **Analyzer** scanning function.

All models available in their respective directories are run simultaneously, analyzing one log per second. If a model identifies an anomaly, it is logged in a file called "anomalies.csv" (Fig. 5), which documents the affected log and the name of the model that detected the anomaly.

```
1. event,machine,timestamp,label,models
2. Edit generate 959800 bytes performed by USER_G,1,1705525458,-1,IF
3. Create generate 13280 bytes performed by USER_Z,1,1705266259,1,IF
4. Login generate 5400 bytes performed by USER_F,1,1706907927,-1,RF
5. Edit generate 362100 bytes performed by USER_V,1,1706907935,-1,RF
6. Logout generate 75300 bytes performed by USER_J,1,1706130343,-1,RF
7. Edit generate 659400 bytes performed by USER_G,1,1705611976,-1,RF
8. Edit generate 384500 bytes performed by USER_V,1,1707599185,-1,RF
9. Login generate 3600 bytes performed by USER_H,1,1705957591,-1,RF
10. Logout generate 57500 bytes performed by USER_L,1,1706735195,-1,RF
11. Login generate 9600 bytes performed by USER_J,1,1706389598,-1,RF
12. Edit generate 603400 bytes performed by USER_Y,1,1705439249,-1,RF
```

Fig. 5. File anomalies.csv.

6 Results

For this synthetic dataset, the DeepCase model did not perform satisfactorily, demonstrating significant uncertainty in anomaly detection. Conversely, the Isolation Forest model, enriched with semantic transformation, performed better than DeepCase, although not without some ambiguities (Fig. 6). This latter finding is supported by the confusion matrix, which reveals a significant number of false negatives, indicating that there were several cases where the model did not correctly identify anomalies in the dataset.

The Random Forest model was the most effective, excelling both in the training evaluation phase and in real-time practical application, proving particularly suitable for analyzing this dataset. It demonstrated a high ability to accurately identify anomalous patterns in the logs, confirming its optimality as an anomaly detection tool in this specific context (Fig. 7).

6.1 HDFS_v1 Dataset

HDFS_v1 is a collection of logs produced in a private cloud environment using benchmark workloads and manually labeled using rules to identify anomalies. The dataset consists of 575,062 log sequences, of which 558,319 are normal (97.1%) and 16,743 are anomalous (2.9%). Pre-processing steps included grouping the log entries by block ID and filtering incomplete tracks. Other than the provided models, no further feature engineering technique has been applied.

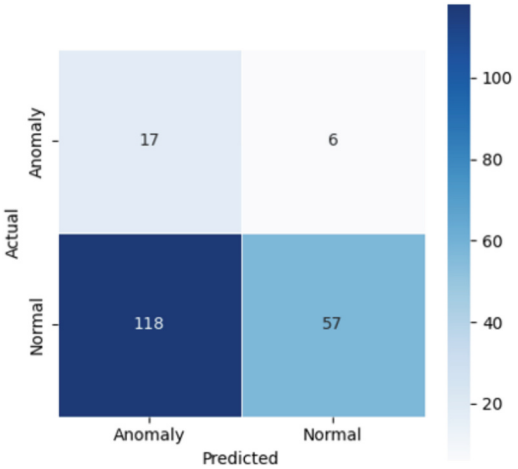

Fig. 6. Confusion matrix IF.

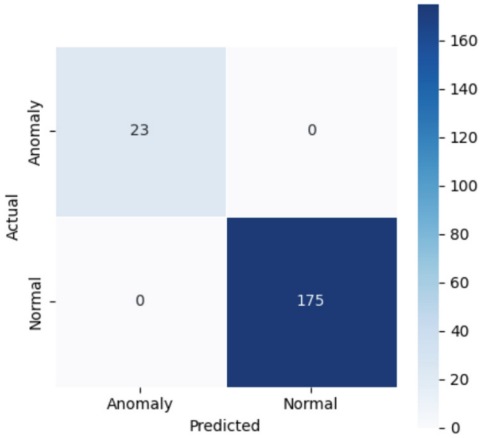

Fig. 7. Confusion matrix RF.

The logs are divided into traces based on block IDs. Each trace associated with a specific block ID is assigned a ground truth label: normal/anomaly.

For example

Record dataset HDFS_v1:

BlockId, Label, Type, Features, TimeInterval, Latency

blk_-3102267849859399193, Fail, 16, "[E5,E5,E22,E7,E11,E10,E14,E7]", "[0.0, 0.0, 60.0, 1.0, 0.0, 657.0, 0.0]", 718

where:

BlockId = "blk_-3102267849859399193"

Label = "Fail"

Type = "16"

Features = "[E5,E5,E22,E7,E11,E10,E14,E7]"
TimeInterval = "[0.0, 0.0, 60.0, 1.0, 0.0, 657.0, 0.0]"
Latency = 718

We used LADDM with the HDFS_v1 dataset. Deepcase's performance significantly outperforms the synthetic dataset (Fig. 8). This was expected, considering that HDFS is a frequently cited and recommended dataset in the Deepcase documentation, indicating potential optimization or specific adaptation of the model for this type of data.

	precision	recall	f1-score	support
0	0.6090	0.1832	0.2817	50178
1	0.8451	0.5529	0.6685	178358
2	0.5898	0.8079	0.6818	149575
3	0.8195	0.9997	0.9006	729051
4	0.6411	0.9057	0.7507	1124
5	0.9950	0.9873	0.9911	727926
6	0.9106	0.9366	0.9234	727926
7	0.8625	0.4791	0.6160	1100
8	0.9628	0.8474	0.9014	1101
9	0.0000	0.0000	0.0000	97
10	0.9990	0.9997	0.9993	593633
11	0.9994	0.3402	0.5077	242643
12	0.8905	0.9529	0.9207	592612
13	0.7429	0.3333	0.4602	1101
14	0.9382	0.9188	0.9284	730014
15	0.0000	0.0000	0.0000	2
16	0.0000	0.0000	0.0000	5
accuracy			0.9039	4726446
macro avg	0.6944	0.6026	0.6195	4726446
weighted avg	0.9110	0.9039	0.8946	4726446

Fig. 8. Classification report Deepcase - HDFS dataset

The confusion matrices for the Isolation Forest (Fig. 9) and Random Forest (Fig. 10) models show similar results, characterized by high levels of performance, reflecting the effectiveness of these machine learning techniques in correctly identifying anomalies in the dataset under study.

To improve readability and comparability, we report precision, recall, F1-score, and accuracy for Isolation Forest and Random Forest on both the synthetic dataset and HDFS_v1. These metrics are derived from the confusion matrices shown.

In Table 1 it is clear that the Random Forest algorithm outperforms Isolation Forest. In fact, Random Forest demonstrates a greater ability to detect anomalies, achieving significantly fewer false positives, which indicates improved accuracy in identifying irregularities.

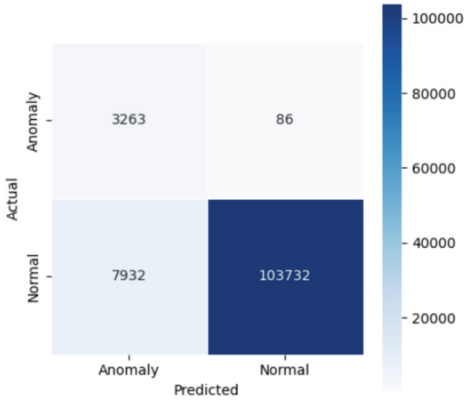

Fig. 9. Confusion matrix IF - HDFS dataset

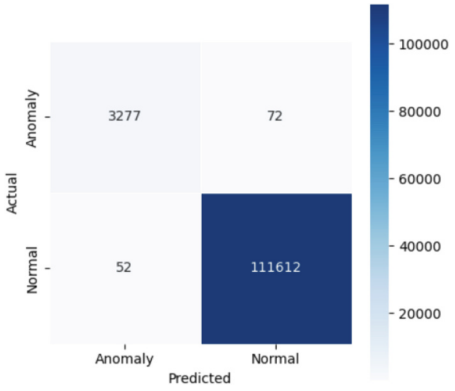

Fig. 10. Confusion matrix RF - HDFS dataset

Table 1. Results

Dataset	Model	Precision	Recall	F1-score	Accuracy
Synthetic	Isolation Forest	0,13	0,74	0,22	0,37
Synthetic	Random Forest	1	1	1	1
HDFS_v1	Isolation Forest	0,29	0,97	0,45	0,93
HDFS_v1	Random Forest	0,98	0,98	0,98	1

7 Conclusions

LADDM was conceived as a flexible tool that can be integrated into business processes to monitor services, leveraging a modular architecture of anomaly detection models. Analyzing the use of the "synthetic" dataset, the Random Forest model showed the best

performance, with similar results also obtained with the "hdfs" dataset. However, unlike the first "hdfs" dataset, it also showed good confidence with the deepcase model. It is therefore essential to recognize the possibility of selecting the most suitable algorithm or algorithms for each specific dataset.

LADDM's unique feature is its ability to interchange or replace anomaly detection models, allowing for tailored adaptation to specific enterprise datasets or the introduction of new, more advanced methodologies.

This flexibility opens the door to many future developments, such as the testing of new models or the handling of larger datasets that show the unusual behavior or real-world attacks that businesses deal with every day. As for future plans, the goal is to make LADDM a fully autonomous tool that runs in the cloud and is hosted in a distributed way. This would make it possible to handle a lot of data and create and host proprietary anomaly detection models, which would make sure that everything works well and is reliable.

To come up with solutions that are not only cutting-edge but also sustainable and scalable, LADDM development must be open and proactive, welcoming new challenges and adapting to change. These solutions must be able to change as the market and social needs change. In conclusion, LADDM could be both a new technology for finding anomalies and a way to make business processes more efficient and change things. LADDM could become a support for data security and analytics in the modern business world if it keeps looking into and using new technologies.

References

1. Rapporto Clusit: (2026). https://clusit.it/rapporto-clusit/. Accessed 15 Mar 2026
2. Check Point Research (CPR). https://www.checkpoint.com/
3. Howden's 2025 cyber insurance report https://www.howdengroupholdings.com/sites/default/files/2025-09/howden-2025-cyber-report-rebooting-growth.pdf
4. Le, V.-H., Zhang, H.: Log-based anomaly detection with deep learning: how far are we? In: Proceedings of the 44th International Conference on Software Engineering (ICSE '22), pp. 1356–1367. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3510003.3510155
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 1–58 (2009)
6. Du, M., Li, F., Zheng, G., Srikumar, V.: Deep log: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings ACM SIGSAC Conference Computer and Communication Security, pp. 1285–1298 (2017)
7. W. Meng, et al.: LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs
8. Lin, Y., et al.: Semi-supervised log-based anomaly detection via probabilistic label estimation. In: ICSE 2021, pp. 1448–1460. IEEE (2021)
9. Zhang, X., et al.: Robust log-based anomaly detection on unstable log data. In: ESEC/FSE 2019, pp. 807–817 (2019)
10. Lu, S., Wei, X., Li, Y., Wang, L.: Detecting anomaly in big data system logs using convolutional neural network. In: 2018 IEEE 16th Intl Conference on Dependable, Autonomic and Secure Computing, 16th Intl Conference on Pervasive Intelligence and Computing, 4th Intl Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), pp. 151–158. IEEE (2018)

11. Ede, T.V., et al.: DEEPCASE: semi-supervised contextual analysis of security events. In: 2022 IEEE Symposium on Security and Privacy (SP), pp. 522–539, San Francisco, CA, USA (2022)
12. Ding, Z., Fei, M.: An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proc. Vol. **46**(20), 12–17 (2013)
13. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
14. Wang, B., Ying, S., Yang, Z.: A log-based anomaly detection method with efficient neighbor searching and automatic k neighbor selection. Sci. Program. **2020**, 1–17 (2020)
15. Guo, H., Yuan, S., Wu, X.: LogBERT: log anomaly detection via BERT. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, Shenzhen, China (2021). https://doi.org/10.1109/IJCNN52387.2021.9534113
16. Liu, Z., Qin, T., Guan, X., Jiang, H., Wang, C.: An integrated method for anomaly detection from massive system logs. IEEE Access. **6**, 30602–30611 (2018)
17. Nipa, N.A., et al.: A comparative study of log-based anomaly detection methods in real-world system logs. In: International Conference on Internet of Things, Big Data and Security (2025)
18. DOCKER https://www.docker.com