Comparative Analysis of Supervised and Unsupervised Learning for Intrusion Detection in Network Logs

Paulo Castro *, Fernando Santos, Pedro Lopes

Posted Date: 20 March 2026

doi: 10.20944/preprints202603.1632.v1

Keywords: cybersecurity; anomaly detection; machine learning; XGBoost; deep learning; recurrent networks

 doi:10.20944/preprints202603.1632.v1

Article

Comparative Analysis of Supervised and Unsupervised Learning for Intrusion Detection in Network Logs

Paulo Castro [1,*], Fernando Santos [2] and Pedro Lopes [2]

[1] Escola Superior de Tecnologia e Gestão de Lamego, Instituto Politécnico de Viseu, Portugal

[2] CISeD – Research Centre in Digital Services, Instituto Politécnico de Viseu, Portugal

* Correspondence: estgl4262@estgl.ipv.pt

Abstract

The growing complexity of network infrastructures and the sophistication of cyber threats require increasingly robust and automated Intrusion Detection Systems (IDS). This article presents a comparative investigation of the effectiveness of various Machine Learning and Deep Learning architectures in identifying anomalies in network logs. The methodology ranged from classic supervised and ensemble algorithms, such as Random Forest and XGBoost, to sequential Deep Learning approaches (LSTM, GRU) and unsupervised models based on latent reconstruction (VAE, DeepLog). The results demonstrate that supervised approaches significantly outperformed unsupervised methods in the analyzed context. The optimized XGBoost model established benchmark performance, achieving a Recall of 0.96 and a Precision of 0.85, offering the ideal balance between detecting rare threats and minimizing false alarms. In contrast, unsupervised models revealed critical limitations, suggesting that statistical mimicry between normal and anomalous traffic hinders detection based solely on reconstruction error. Additionally, the study documents the technical interoperability challenges when attempting to integrate state-of-the-art language models, such as BERT. In conclusion, this work validates the effectiveness of *Boosting* algorithms and recurrent networks as viable and scalable solutions for critical network security, providing guidelines for model selection in real monitoring environments.

Keywords: cybersecurity; anomaly detection; machine learning; XGBoost; deep learning; recurrent networks

1. Introduction

The exponential growth of network infrastructures and the sophistication of cyber threats have exposed systems to unprecedented vulnerability, rendering traditional defense strategies insufficient [1,2]. Signature-based and static rule-based systems, while fundamental, demonstrate critical limitations in identifying zero-day attacks and generate high volumes of false positives, compromising the response capabilities of security teams.

In this context, intelligent analysis of network logs using Artificial Intelligence (AI) emerges as a promising solution, although it faces complex challenges such as data heterogeneity and the extreme imbalance between legitimate activity and intrusions. The motivation for this work lies in the need to optimize the reliability of detection models, prioritizing the reduction of false negatives to ensure operational continuity in critical infrastructures.

This article proposes a comparative methodology between supervised and unsupervised models applied to real data from an institutional environment. The main contributions of this study are:

- The design of a robust pipeline for processing network logs, ensuring data integrity for predictive analysis;

 doi:10.20944/preprints202603.1632.v1

- The implementation of optimized feature engineering techniques to ensure interoperability between Machine Learning and Deep Learning algorithms;
- A rigorous evaluation of model performance under class imbalance conditions, with a focus on maximizing recall.

The organization of this paper is as follows: Section 2 reviews related work on intrusion detection using machine learning and deep learning approaches; Section 3 describes the computing environment and hardware specifications used; Section 4 details the experimental methodology, including preprocessing and feature engineering; Section 5 discusses the model evaluation strategy and analyzes the results for supervised and unsupervised architectures; and Section 6 concludes the paper with final remarks.

2. Related Works

Intrusion detection based on log and network traffic analysis has been extensively investigated using machine learning and deep learning approaches. The evolution and sophistication of cyber threats have driven the transition from traditional signature-based systems to intelligent models capable of identifying anomalous patterns and zero-day attacks with greater effectiveness than classic misuse detection methods [3–5]. However, the practical implementation of these models faces critical challenges, namely extreme class imbalance, where anomalies represent a minimal fraction of the records, and the heterogeneous and sequential nature of the data, which requires advanced preprocessing strategies for the preservation of temporal information [6].

In the field of Ensemble Learning, algorithms such as XGBoost and Random Forest stand out for their robustness and generalization capacity [7–9]. Recent studies point to XGBoost as one of the most effective models overall, often reporting accuracy values above 98% in the identification of complex attacks [9,10]. Nevertheless, the literature recognizes that this high performance in global metrics does not always translate into operational effectiveness, since an excessive focus on accuracy can neglect Recall, a vital metric in security contexts where a false negative can compromise the entire infrastructure [8].

At the same time, deep neural networks have emerged as key solutions due to their ability to model long-term temporal dependencies [11–13]. The gate mechanism present in architectures such as LSTM and GRU allows for selective information retention, resulting in effective learning of network event sequences. Recent work indicates that the combination of convolutional networks with recurrent units (CNN+GRU) offers an optimized balance between accuracy and computational efficiency, making it particularly suitable for real-time applications [12,13].

Despite these advances, significant gaps remain in current research. Most studies favor global metrics over minimizing false negatives and lack validation in simulated scenarios that test the generalization of models in the face of artificially reproduced intrusions. This study seeks to fill these gaps by presenting a comparative analysis focused on reducing false negatives and evaluating the robustness of models in conditions close to real environments.

3. Computing Environment

The practical implementation and comparative evaluation of the algorithms were conducted on a workstation equipped with an Intel® Core™ i7-10870H CPU @ 2.20GHZ (8 cores, 16 threads), configured to optimize the parallel processing required by decision tree-based models. The system has 16 GB of DDR4 RAM for efficient handling of large volumes of data in memory and an NVIDIA GeForce RTX 2060 (6 GB GDDR6) graphics processing unit (GPU), essential for hardware acceleration during the training of Deep Learning architectures. Storage and input/output (I/O) operations were ensured by an NVMe SSD drive, minimizing latency in the data pipeline.

 doi:10.20944/preprints202603.1632.v1

4. Methodology and Results

The systematic evaluation of the proposed models is based on the use of real security data provided by the Polytechnic Institute of Viseu. This dataset, characterized by its high dimensionality and complexity, allows the empirical results to be compared with the theoretical assumptions in the literature, ensuring the relevance of the results in a practical application scenario.

Given the critical nature of network security, the methodology favors Recall as the central performance metric. This choice is justified by the imperative need to minimize false negatives, since the failure to detect a real intrusion has significantly more serious consequences than the occurrence of false positives.

4.1. Preprocessing and Feature Engineering

The original *dataset*, consisting of multiple CSV files, presented significant challenges of high dimensionality, data type heterogeneity, and a high incidence of Not a Numbers (NaNs). The data preparation process was structured to ensure data integrity and computational efficiency of the models.

Cleaning and Consolidation: After concatenating the records, rigorous filtering was applied based on the informational relevance of the variables. Columns with an incidence of null values greater than 70% were discarded, and the remaining (NaNs) were filled in using the mode (categorical variables) and the median (numeric variables). In addition, the metadata was normalized, removing coding inconsistencies and invisible characters in column names to ensure referential consistency in the pipeline.

Feature Engineering and Temporal Transformation: Knowledge extraction from temporal attributes was crucial to capture the dynamic behavior of the network. Timestamp and duration variables were decomposed into discrete components (year, month, day of the week, hour, and second) and converted to Epoch format. This process allowed the transformation of raw date/time data into numerical features interpretable by learning algorithms.

Dimensionality Control and Encoding: To mitigate memory explosion and redundancy issues, a multilevel dimensionality reduction strategy was implemented:

- **Identifier Filtering:** Removal of high-cardinality variables with no predictive value (e.g., registration IDs).
- **Hashing Encoding:** Applied to variables such as Source and Destination to map thousands of categories into a fixed and manageable vector space [14].
- **Rare Category Clustering:** Classes with less than 0.5% representation were consolidated into a generic category, reducing noise in subsequent encoding.
- **One-Hot Encoding and Variance:** After binarizing the categorical variables, low-variance features (where more than 99.9% of the values were identical) were eliminated, optimizing the dataset for the training phase.

Target Variable Definition and Class Imbalance: The target variable was defined based on the "Severity" column rather than the "Action" column. This methodological decision is based on the premise that severity reflects the intrinsic risk of the event, regardless of the system's reactive response (blocking or acceptance). By categorizing events of 'High' and 'Critical' severity as anomalies (1) and the rest as normal traffic (0), the model can identify threats that may have bypassed existing security rules.

The final *dataset* is extremely unbalanced, with the anomalous class representing only 0.02% of the total records. This characteristic requires rigorous evaluation metrics focused on minimizing false negatives.

 doi:10.20944/preprints202603.1632.v1

5. Model Evaluation and Selection Strategy

Given the high volume of *the dataset* and the diversity of architectures explored, a phased optimization methodology was adopted to ensure the computational feasibility of the study. The experimental procedure was structured in two main stages:

- **Baseline Evaluation:** All models were initially trained and evaluated using their default settings. This phase allowed a direct comparison of the intrinsic performance of each algorithm in relation to the specific nature of the network data, functioning as a feasibility filter.
- **Selective Optimization:** Only the algorithm that demonstrated the most promising and relevant performance in the context of intrusion detection (balance between *Recall* and latency) was selected for a subsequent phase of hyperparameter fine-tuning. This approach allows computational resources to be concentrated on the architecture with the greatest potential for practical application.

5.1. Supervised Machine Learning Classification

This section presents the experimental procedures performed with the supervised machine learning models. Each model is trained, evaluated, and compared based on the same data set and metrics to ensure a consistent analysis of its performance.

5.1.1. Random Forest

The implementation of the Random Forest algorithm was structured to reconcile computational feasibility with statistical robustness, given the high volume of the dataset (2,250,300 records). The experimental protocol began with an exploratory phase using a stratified sample of 10%, allowing the sensitivity of the extracted network metrics to be validated before expanding to full training.

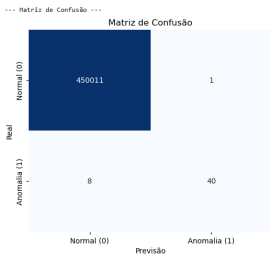
When processing the complete dataset, an 80/20 split was applied to the training and test sets, respectively. The preservation of the minority class proportion in both subsets was ensured through stratified sampling ('stratify=y'). Prior to training, the numerical variables underwent a 'StandardScaler' process, ensuring that discrepancies in magnitude between traffic characteristics did not introduce biases in the construction of decision trees.

The strategic choice of the 'class_weight=balanced' parameter proved to be decisive for the effectiveness of the model. By assigning a higher penalty to classification errors in the minority class, this technique mitigated extreme imbalance without resorting to synthetic oversampling methods such as SMOTE. This choice preserves the integrity of the original data and avoids introducing artificial variance, a critical factor in institutional network security analyses where synthetic noise can mask real intrusion patterns.

After training, the model obtained the following results in Figure 1:

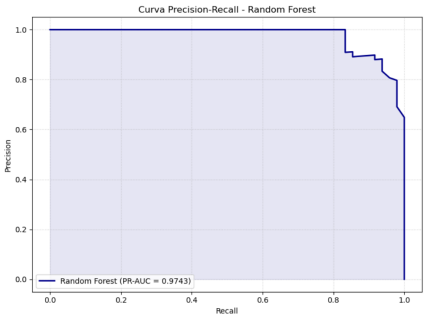
```
--- Relatório de Classificação ---
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    450012
           1       0.98      0.83      0.90        48

    accuracy                           1.00    450060
   macro avg       0.99      0.92      0.95    450060
weighted avg       1.00      1.00      1.00    450060
```

1 Classification Report



2 Confusion Matrix

 doi:10.20944/preprints202603.1632.v1



3 Precision-Recall Curve

Figure 1. Evaluation metrics for the Random Forest model on (1) Classification Report, (2) Confusion Matrix, (3) Precision-Recall Curve.

The results obtained revealed high effectiveness in filtering false positives, with a Precision of 0.98. The F1-score of 0.90 and the PR-AUC of 0.97 confirm the robustness of the model in distinguishing between legitimate and anomalous traffic. However, the analysis focused on Recall (0.83) identified a limitation, namely the failure to detect approximately 17% of actual intrusions (false negatives). This sensitivity deficit, although acceptable in generic contexts, is considered high for critical infrastructure cybersecurity systems, motivating the exploration of *gradient boosting* algorithms with greater adjustment capacity.

5.1.2. XGBoost

The XGBoost algorithm was selected as representative of *Gradient Boosting* architectures, being widely recognized for its efficiency in high-dimensional tabular data. The technical implementation followed a binary classification protocol using the 'binary:logistic' cost function. To control complexity and prevent overfitting, the learning rate was set at 0.1, coordinated with the use of the logloss metric to monitor convergence during the 100 training iterations.

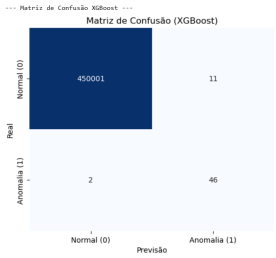
Given the unbalanced nature of *the dataset*, the model configuration focused on *the cost-sensitive* learning technique. To this end, the 'scale_pos_weight' parameter was used, whose value was calculated using the ratio between the normal class and the anomalous class. This parameterization allows the algorithm to adjust its decision boundary, giving greater importance to the correct classification of the minority class, without the need for external data manipulation, such as oversampling techniques.

After training, the performance metrics obtained for the XGBoost algorithm, presented in Figure 2, were as follows:

--- Relatório de Classificação XGBoost ---

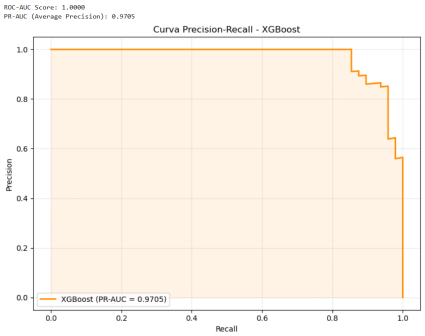
	precision	recall	f1-score	support
0	1.00	1.00	1.00	450012
1	0.81	0.96	0.88	48
accuracy			1.00	450060
macro avg	0.90	0.98	0.94	450060
weighted avg	1.00	1.00	1.00	450060

1 Classification Report



2 Confusion Matrix

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 March 2026 doi:10.20944/preprints202603.1632.v1



3 Precision-Recall Curve

Figure 2. Evaluation metrics for the XGBoost model on (1) Classification Report, (2) Confusion Matrix, (3) Precision-Recall Curve.

5.1.3. Stochastic Gradient Descent

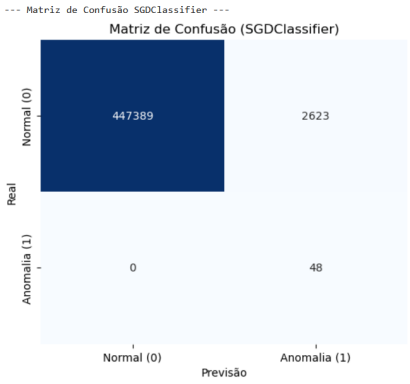
The exploration of linear decision boundaries using Support Vector Machines (SVM) proved computationally unfeasible for the volume of data under study (approx. 1.8 million samples). Traditional SVM algorithms, such as Sequential Minimal Optimization (SMO), have a complexity that typically scales between $\left(0(n^2)\right)$ a $\left(0(n^3)\right)$, which require unaffordable RAM resources and processing times.

To circumvent this limitation, we opted to use Stochastic Gradient Descent (SGD), configured with a *Hinge* loss function to behave as a linear SVM classifier. Unlike batch methods, SGD updates the model parameters iteratively using individual samples, reducing the complexity to $\left(O(n)\right)$. This approach ensures the scalability needed to handle the high dimensionality of the dataset without compromising memory efficiency.

As a result, the model after training achieved the following results, presented in Figure 3:

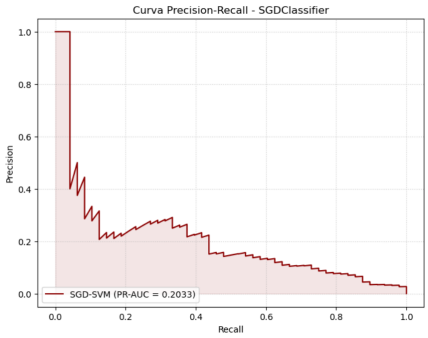
```
--- Relatório de Classificação SGDClassifier ---
              precision    recall  f1-score   support

           0       1.00      0.99      1.00    450012
           1       0.02      1.00      0.04        48

    accuracy                           0.99    450060
   macro avg       0.51      1.00      0.52    450060
weighted avg       1.00      0.99      1.00    450060
```

1 Classification Report



2 Confusion Matrix

 doi:10.20944/preprints202603.1632.v1



3 Precision-Recall Curve

Figure 3. Evaluation metrics for the SGD model on (1) Classification Report, (2) Confusion Matrix, (3) Precision-Recall Curve.

The results obtained by the SGD classifier present a statistical phenomenon that is relevant in the context of intrusion detection. The model achieved a Recall of 1.00, successfully identifying all 48 anomalies in the test set. However, this performance was accompanied by an extremely low Precision of 0.02, due to the generation of 2623 false positives.

This discrepancy highlights the risk of using ROC-AUC as the sole evaluation metric in cybersecurity. Although the ROC-AUC of 0.993 suggests an almost perfect theoretical separation, the metric is insensitive to the impact of false positives when the minority class is tiny. The low F1-score (0.04) and PR-AUC (0.20) confirm that, despite its overall sensitivity, the model lacks specificity for practical implementation in a real monitoring system.

5.1.4. Supervised Machine Learning – Analytical Comparison

The comparative analysis of supervised models, summarized in Table 1, shows a clear dichotomy between theoretical sensitivity and practical applicability. The SGD classifier, despite reaching the maximum *Recall* limit (1.00), proved to be operationally unfeasible. The massive generation of false alerts (2623 occurrences) would result in "alert fatigue" for cybersecurity teams, neutralizing the practical usefulness of the detection system. The F1-score and PR-AUC metrics corroborate this inadequacy, demonstrating that total sensitivity does not compensate for the critical loss of accuracy.

Table 1. Comparison of the results of the Supervised Machine Learning models.

Metrics	Random Forest	XGBoost	SGD
Precision	0.98	0.81	0.02
Recall	0.83	0.96	1.00
F1-score	0.90	0.88	0.04
False Positives	1	11	2623
False Negatives	8	2	0
ROC-AUC	1.00	1.00	0.9993
PR-AUC	0.9743	0.9705	0.2033

Regarding the balance between tree-based models, there is a distinct technical *trade*-off between Random Forest and XGBoost:

- Random Forest favored specificity, presenting the lowest false alarm rate but failing to identify 8 critical anomalies (false negatives).

 doi:10.20944/preprints202603.1632.v1

- XGBoost demonstrated superior sensitivity, capturing 46 of the 48 real threats. Although this performance resulted in a residual increase in false positives (11 cases), the operational cost of this inaccuracy is largely offset by risk mitigation.

In conclusion, XGBoost's ability to detect 12.5% more anomalies than Random Forest positions it as the most resilient solution for the domain under study. In critical infrastructure contexts, the top priority is visibility into threats. Therefore, XGBoost's robustness in reducing false negatives gives it a decisive strategic advantage for integration into proactive monitoring systems.

5.1.4. XGBoost Optimization

After identifying XGBoost as the most promising model, a systematic optimization phase was carried out to maximize the balance between Precision and Recall. To ensure computational feasibility given the vast dimensionality, the Randomized Search Cross-Validation ('RandomizedSearchCV') technique was used, exploring 50 different combinations of hyperparameters. The process was supported by stratified cross-validation (Stratified K-Fold, K = 3), ensuring that the representativeness of the minority class was preserved in all partitions. The target metric for optimization was the F1-score of the minority class, as it represents the most robust compromise between detection sensitivity and operational reliability.

As shown in Table 2, the comparison between the *baseline* model and the optimized version reveals a maintenance of high detection capacity, with Recall set at 0.96 (identification of 46 out of 48 anomalies). This result confirms that the refinement of the parameters did not compromise the fundamental sensitivity of the algorithm.

Table 2. Comparative performance between the base and optimized XGBoost models.

Metrics	Base XGboost	Optimized XGBoost
Precision	0.81	0.85
Recall	0.96	0.96
F1-score	0.88	0.90
False Positives	11	8
False Negatives	2	2
ROC-AUC	1.00	0.9998
PR-AUC	0.9705	0.9703

The most significant gain was in Precision, which rose from 0.81 to 0.85. This increase translates into a reduction of approximately 27% in the volume of false positives (from 11 to 8 occurrences). In a production environment, this refinement is crucial, as it reduces the cognitive load on security analysts and reinforces confidence in the alerts generated by the system. The F1-score followed this evolution, reaching 0.90. The ROC-AUC and PR-AUC metrics remained at excellent levels, validating the stability and generalization of the final optimized model.

5.2. Unsupervised Machine Learning

This section presents the experimental procedures performed with the Unsupervised *Machine Learning* models. These algorithms were applied under the same methodological conditions defined for the previous models, allowing us to evaluate their ability to identify anomalous patterns without resorting to labeled data.

5.2.1. Isolation Forest

The unsupervised approach began with the Isolation Forest algorithm, with the aim of identifying divergent patterns without the guidance of the target variable. Unlike supervised

 doi:10.20944/preprints202603.1632.v1

machine learning models, this architecture is based on the premise that anomalies are rare observations and are more susceptible to isolation in decision tree structures.

The model initialization included 100 estimators and a sensitivity analysis focused on the contamination parameter, which defines the expected proportion of outliers. Initially, the actual anomaly ratio (0.000106%) was used, resulting in a conservative convergence that classified all records as legitimate traffic. To diagnose the model's response to different decision thresholds, the contamination rate was raised to 0.01 (1%), forcing a more comprehensive decision boundary.

The results obtained demonstrate a critical flaw in the applicability of this architecture to the institutional dataset. The model recorded Precision, Recall, and F1-score values of 0.00, failing to identify any of the 48 anomalies present in the test set. At the same time, the generation of 50 false positives introduced unjustified noise into the system. The PR-AUC value (0.0004) confirms the practical inability of the algorithm to establish an effective anomaly score.

After completing the training, the algorithm was evaluated on the test set, with the performance metrics detailed in Table 3:

Table 3. Comparative results of the performance of the Isolation Forest model with different levels of contamination.

Metrics	Isolation Forest (0,000106)	Isolation Forest (0,01)
Precision	0,00	0.00
Recall	0.00	0.00
F1-score	0.00	0.00
False Positives	50	4358
False Negatives	48	48
ROC-AUC	0.7996	0.7996
PR-AUC	0.0004	0.0004

The analysis also revealed a discrepancy between the configured contamination parameter (1%) and the volume of instances isolated (50 out of 450,060). This behavior suggests that the anomalies present in the network logs do not manifest themselves as obvious global outliers in terms of distance or isolation, but rather as subtle patterns integrated into the normal traffic density. In summary, Isolation Forest's inability to detect any anomalies, coupled with the generation of false positives, makes it unfeasible for the research objective.

5.2.2. K-Means

The application of the K-Means algorithm was based on spatial density analysis, starting from the premise that instances significantly distant from the centroids of the dominant clusters represent anomalous behavior. Given the sensitivity of the technique to the definition of the number of groups (k), an exploratory strategy was implemented, varying the granularity of the grouping to assess the ability to isolate threats in the feature hyperspace.

Initially, $k = 8$ was used to consolidate legitimate traffic into macro behavioral patterns. Given the low sensitivity observed, the hyperparameter was increased to $k = 50$, allowing for more detailed segmentation of the data space and testing the hypothesis that anomalies would be associated with small residual clusters. Anomaly classification was determined by calculating the Euclidean distance between each observation and its respective centroid, using decision thresholds based on the 99.99% and 99.9% distance percentiles.

The performance metrics resulting from the two configurations are presented in Table 4.

 doi:10.20944/preprints202603.1632.v1

Table 4. Comparative performance results of the K-means model with different numbers of clusters.

Metrics	K-means (8)	K-means (50)
Precision	0.00	0.00
Recall	0.00	0.00
F1-score	0.00	0.00
False Positives	901	451
False Negatives	48	48
ROC-AUC	0.5631	0.6152
PR-AUC	0.0001	0.0001

The experimental results demonstrated the ineffectiveness of this approach for the cybersecurity scenario under study. Regardless of the granularity applied, the model recorded zero Precision, Recall, and F1-score values, failing to identify any of the 48 real threats present in the test set. The increase in the number of clusters only resulted in a substantial increase in false positives, without any gain in the detection rate.

Visualization using Principal Component Analysis (PCA), illustrated in Figure 4, corroborated the failure of the algorithm, demonstrating that anomalies (real attack vectors) do not manifest as isolated points, but are scattered non-linearly among legitimate traffic data. The low performance of PR-AUC (0.0001) and a ROC-AUC close to a random classifier suggest that the centroid distance premise is insufficient to capture the complexity of institutional intrusions, where malicious patterns can mimic the statistical structure of normal traffic.

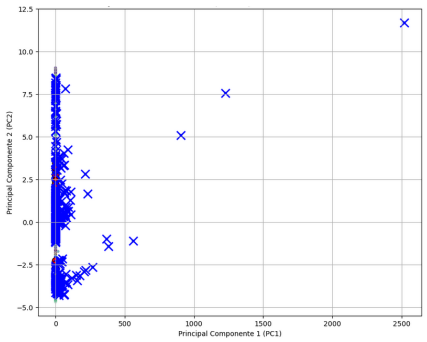


Figure 4. Visualization of K-means clustering with PCA.

5.2.3. DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was explored for its theoretical ability to identify clusters of arbitrary geometry and isolate low-density points, categorizing them as noise (potential anomalies). The effectiveness of this approach critically depends on the definition of the neighborhood radius ('eps') and the minimum number of points for forming dense clusters ('min_samples').

Given the high dimensionality of the dataset (297 features), a two-step evaluation was performed to determine the feasibility of the model:

- **Restrictive Configuration:** 'min_samples=594' was defined, based on the heuristic of $2 * \text{número de features}$, with eps=0.5. This parameterization proved to be overly conservative, resulting in total fragmentation of the data where almost all observations were labeled as noise.
- **Flexible Configuration:** To promote the convergence of legitimate clusters, 'min_samples' was reduced to 30. The objective was to assess whether a less demanding density definition would allow normal traffic to be isolated, reducing the volume of false positives.

The results obtained, summarized in Table 5, were as follows:

 doi:10.20944/preprints202603.1632.v1

Table 5. Comparative performance results of the DBSCAN model in 'min_samples' configurations.

Metrics	DBSCAN (594)	DBSCAN (30)
Precision	0	0.00
Recall	1.00	1.00
F1-score	0.00	0.0
False Positives	1,783,931	1,662,820
False Negatives	0	0
ROC-AUC	0.7301	0.8425
PR-AUC	0.0002	0.0003

The results show that DBSCAN, in both configurations, was unable to establish a useful distinction between normal traffic and anomalies. Although the model recorded a Recall of 1.00, this value is of no practical relevance, since zero Precision indicates that the algorithm classified approximately 1.78 million normal instances as anomalies.

This systematic failure suggests that the structure of the network logs under study exhibits high dispersion in the feature hyperspace. The inability to form stable clusters, even with the reduction of the 'min_samples' parameter, indicates that the data does not exhibit the local density necessary for the effectiveness of algorithms based on spatial connectivity. As observed in the SGD model, the unbalanced performance of DBSCAN makes its operational application unfeasible, reinforcing the premise that detection in these environments requires models capable of learning more complex decision boundaries or compressed latent representations.

5.3. Supervised Deep Learning

This section addresses the practical implementation of Supervised Deep Learning models. The training, validation, and evaluation processes of the different architectures are described, ensuring methodological uniformity regarding Machine Learning approaches.

5.3.1. LSTM

The Long Short-Term Memory (LSTM) architecture was selected for its intrinsic ability to model sequential dependencies in data flows, being widely recognized in the literature for its effectiveness in network log analysis. For implementation, the data was transformed from a two-dimensional structure to a three-dimensional format (samples, time windows, characteristics), an essential requirement for processing in recurrent layers.

To optimize computational efficiency and minimize processing latency, a unitary time window ('n_timesteps=1') was defined. This configuration allows the model to process events atomically, maintaining compatibility with the LSTM memory structure. The sequential architecture consisted of:

- **Recurrent Layer:** 64 LSTM units for latent pattern extraction;
- **Regularization:** A Dropout layer (0.2) to prevent overfitting through stochastic omission of neurons;
- **Output Layer:** Densely connected neuron with Sigmoid activation function for binary classification.

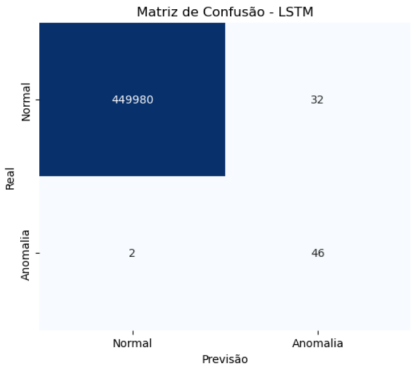
Training was optimized by applying class weights, compensating for the extreme imbalance in the dataset. The Adam optimizer and the Binary Cross-Entropy loss function were used, with GPU acceleration support. To ensure generalization, an early stopping mechanism was incorporated with a patience of 3 epochs, monitoring the validation loss.

Thus, after training the model, the following results obtained, illustrated in Figure 5, were as follows:

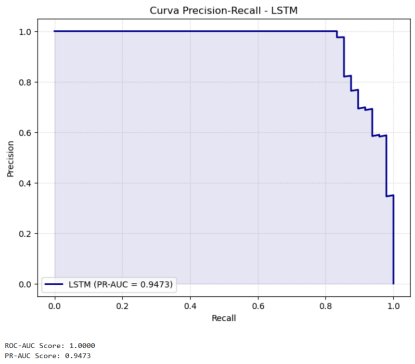
 doi:10.20944/preprints202603.1632.v1

	precision	recall	f1-score	support
0	1.00	1.00	1.00	450012
1	0.59	0.96	0.73	48
accuracy			1.00	450060
macro avg	0.79	0.98	0.87	450060
weighted avg	1.00	1.00	1.00	450060

1 Classification Report



2 Confusion Matrix



3 Precision-Recall Curve

Figure 5. Evaluation metrics for the LSTM model on (1) Classification Report, (2) Confusion Matrix, (3) Precision-Recall Curve.

The LSTM model demonstrated high robustness, achieving a Recall of 0.96 (identification of 46 out of 48 anomalies). This performance matches the sensitivity of the XGBoost model, confirming the effectiveness of neural networks in detecting rare threats.

However, an operational compromise was observed in Precision (0.59), which resulted in a higher volume of false positives compared to Ensemble models. This trade-off is characteristic of deep models driven by class weights, which tend to be more aggressive in signaling anomalies. Despite lower accuracy, the ROC-AUC (1.00) and PR-AUC (0.9473) values validate LSTM as a powerful tool for security systems that prioritize total visibility over infrastructure, even at the cost of a greater need for alert screening.

5.3.2. BERT

The integration of the Bidirectional Encoder Representations from Transformers (BERT) model was planned with the aim of establishing a performance benchmark based on large-scale pre-trained models. BERT is widely recognized for its ability to capture bidirectional semantic context, offering a theoretical advantage over traditional sequential architectures in the classification of complex patterns in network logs.

However, the practical implementation of this architecture revealed critical technical constraints related to interoperability between Deep Learning frameworks. The execution of the 'TFBertForSequenceClassification' class in the TensorFlow environment depends on the dynamic conversion of pre-trained weights originating from PyTorch. During this process, structural incompatibility was found between the Transformers library and recent versions of Keras (v3.x), resulting from the depreciation of low-level functions in the library's backend.

 doi:10.20944/preprints202603.1632.v1

Mitigation attempts, including reverting to legacy versions of TensorFlow and using the 'tf-Keras' compatibility package, proved unfeasible due to dependency conflicts in the operating system and hardware infrastructure used.

The definitive resolution of these anomalies would require the complete migration of the experimental pipeline to PyTorch. However, such restructuring was ruled out for methodological reasons. The framework transition would introduce external variables that could skew the direct comparison between the model, preventing a reliable assessment of whether performance variations stemmed from the model architecture or the intrinsic optimizations of the execution platform. In short, the exclusion of BERT is based on preserving the methodological integrity of the study and documenting the real portability challenges encountered when integrating cutting-edge models into rapidly evolving development ecosystems.

5.3.3. GRU

The Gated Recurrent Unit (GRU) model was implemented as an optimized alternative to the LSTM architecture, aiming to assess the impact of a faster recurrent structure on intrusion detection. The motivation for choosing GRU lies in its computational efficiency, by merging the control mechanisms in the Update and Reset Gates vectors. The algorithm reduces the volume of trainable parameters, accelerating convergence without compromising the ability to model long-term temporal dependencies.

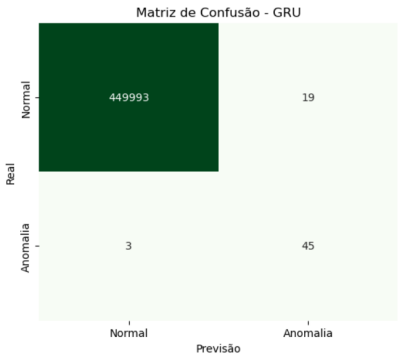
To ensure experimental comparability, the GRU adopted a sequential structure equivalent to that of the LSTM model:

- **Processing Layer:** 64 GRU units for sequential feature extraction;
- **Regularization:** Dropout layer at 0.2 for overfitting mitigation;
- **Classifier:** Dense layer with Sigmoid activation for generating binary probabilities.

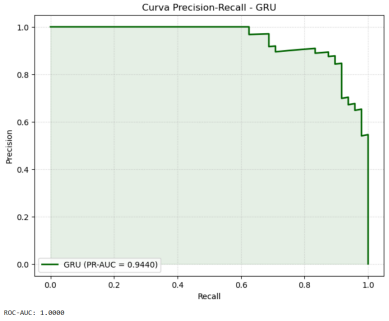
Based on this process, the results obtained are presented in Figure 6:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	450012
1	0.70	0.94	0.80	48
accuracy			1.00	450060
macro avg	0.85	0.97	0.90	450060
weighted avg	1.00	1.00	1.00	450060

1 Classification Report



2 Confusion Matrix



3 Precision-Recall Curve

 doi:10.20944/preprints202603.1632.v1

Figure 6. Evaluation metrics for the GRU model on (1) Classification Report, (2) Confusion Matrix, (3) Precision-Recall Curve.

5.3.4. Supervised Deep Learning – Analytical Comparison

As shown in Table 6, a comparative analysis between Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures reveals distinct approaches to managing the trade-off between sensitivity and specificity. The LSTM model proved to be the most balanced architecture, achieving an F1-score of 0.89. This robustness translates into effective mitigation of both false negatives and false positives, positioning LSTM as a consistent classifier for environments where alert reliability is critical. The PR-AUC value (0.9470) corroborates its effectiveness in maintaining selectivity even under the pressure of an extremely unbalanced dataset.

Table 6. Comparison of the results of Supervised Deep Learning models.

Metrics	LSTM	BERT	GRU
Precision	0.59	-	0.70
Recall	0.96	-	0.94
F1-score	0.73	-	0.80
False Positives	32	-	19
False Negatives	2	-	3
ROC-AUC	1.00	-	1.00
PR-AUC	0.9473	-	0.9440

On the other hand, the GRU stood out for its superiority in Recall, ensuring the identification of 46 of the 48 real threats (96% capture effectiveness). However, this high operational sensitivity comes at a cost in terms of Precision, resulting in an increase in false alarms that can increase the cognitive load on security analysts. The GRU's high PR-AUC confirms that the architecture is extremely powerful in extracting minority class features, prioritizing total threat visibility over classification accuracy.

5.3.5. GRU Optimization

The refinement of the GRU model aimed to harmonize the trade-off between Precision and Recall, mitigating the discrepancy observed in the baseline configuration. The optimization strategy evolved from a simple architecture to a stacked layer configuration (Stacked GRU), designed to enhance the extraction of hierarchical features in complex time series.

The new architecture was structured in two successive layers:

- **Primary Layer:** 64 GRU units with 'return_sequences=True', allowing the propagation of the temporal dimension to the next level;
- **Secondary Layer:** 32 GRU units, focused on distilling learned temporal dependencies and reducing latent dimensionality.

To prevent overfitting in this deeper structure, Dropout layers were integrated between the recurrent blocks. The training protocol remained consistent, using the Adam optimizer, the Binary Cross-Entropy loss function, and the application of class weights to compensate for severe data imbalance. The process was monitored via Early Stopping to ensure optimal convergence and computational resource efficiency.

The results, detailed in Table 7, show a significant reconfiguration in the balance of metrics. Precision showed remarkable improvement, rising from 0.59 to 0.91. In practical terms, this improvement translates into a drastic reduction in false alarms (from 32 to only 4 occurrences), which minimizes security analyst fatigue. The F1-score followed this trend, reaching 0.88.

 doi:10.20944/preprints202603.1632.v1

Table 7. Comparative performance between the base and optimized GRU models.

Metrics	Base GRU	Optimized GRU
Precision	0.7	0.91
Recall	0.94	0.85
F1-score	0.80	0.88
False Positives	19	4
False Negatives	3	7
ROC-AUC	1.00	1.0
PR-AUC	0.9440	0.9253

Although there was a marginal reduction in Recall (from 0.96 to 0.85), resulting in 7 false negatives compared to the original 2, the optimized model established a higher level of reliability. The stability of the ROC-AUC and PR-AUC metrics (0.9253) confirms that the stacked architecture is more resilient and suitable for production environments, where alert accuracy is critical to the effectiveness of incident response plans.

5.4. Unsuprvised Deep Learning

In this section, experiments corresponding to unsupervised Deep Learning models are conducted. The configuration, training, and analysis of results steps are presented to evaluate their practical effectiveness in the context of the dataset under study.

5.4.1. VAE

The exploration of unsupervised methods based on deep learning focused on the Variational Autoencoder (VAE). The objective was to learn a compact and probabilistic representation of the "normality" of network data, using reconstruction error as a discriminatory metric for intrusion detection.

The model was trained exclusively with instances of the normal class, if this subsample defines the nominal behavior of the system. The architecture implemented followed the classic encoder-decoder structure:

- **Encoder:** Maps input features to Gaussian distribution parameters in a latent space of dimensions;
- **Latent Space:** Acts as an information bottleneck, forcing the compression of essential attributes;
- **Decoder:** Reconstructs the original data from samples in the latent space.

Training was conducted with the *Adam* optimizer for 100 epochs, using dense layers of 128 neurons to capture subtle variations in the data distribution. Detection was based on the definition of *thresholds* applied to the reconstruction error, calculated from the error percentiles of the training set.

After training, the model was evaluated on the test set, obtaining the following results as shown in Figure 7:

5.4.2. LogBERT

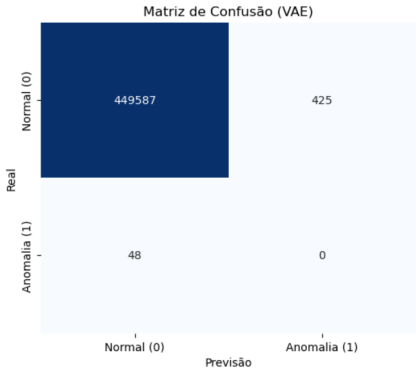
LogBERT architecture was considered a natural extension to the study, representing the state of the art in the specialization of Transformer-based models for the log domain. Its conceptual advantage lies in the use of Self-Attention mechanisms and the Masked Language Modeling task, which allow capturing long-range contextual dependencies and semantic relationships between network events that escape traditional sequential models.

Despite its theoretical potential, the practical implementation of LogBERT proved unfeasible in this study, inheriting the interoperability limitations documented in the previous section on the BERT model. Given its direct dependence on the Transformers library, the integration of LogBERT would require an exclusive migration to the PyTorch ecosystem.

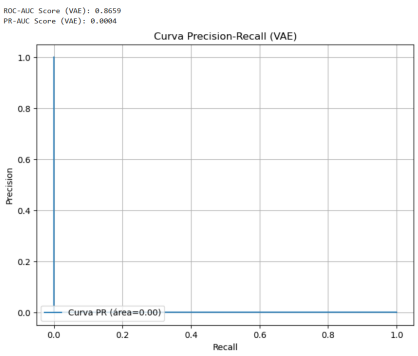
 doi:10.20944/preprints202603.1632.v1


```
--- Relatório de Classificação (VAE) ---
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    450012
           1       0.00      0.00      0.00        48

    accuracy                           1.00    450060
   macro avg       0.50      0.50      0.50    450060
weighted avg       1.00      1.00      1.00    450060
```

1 Classification Report



2 Confusion Matrix



3 Precision-Recall Curve

Figure 7. Evaluation metrics for the VAE model on (1) Classification Report, (2) Confusion Matrix, (3) Precision-Recall Curve.

The decision not to proceed with this implementation was based on two critical pillars:

- **Ecosystem Consistency:** Introducing an exclusive dependency on PyTorch would create a **methodological asymmetry**, since all other Deep Learning architectures in this study were developed and optimized on the TensorFlow/Keras backend.
- **Comparative Validity:** To ensure the reliability of the analysis, it is imperative that performance variations be attributed to the nature of the algorithms and not to intrinsic optimizations or processing differences between frameworks.

Thus, the exclusion of LogBERT does not stem from conceptual limitations of the model, but rather from a strategic choice to ensure a uniform and controlled experimental environment. This analysis reinforces the persistent challenge of ecosystem fragmentation in AI research, where the portability of cutting-edge models across development platforms remains an obstacle to the integration of heterogeneous tools.

5.4.3. DeepLog

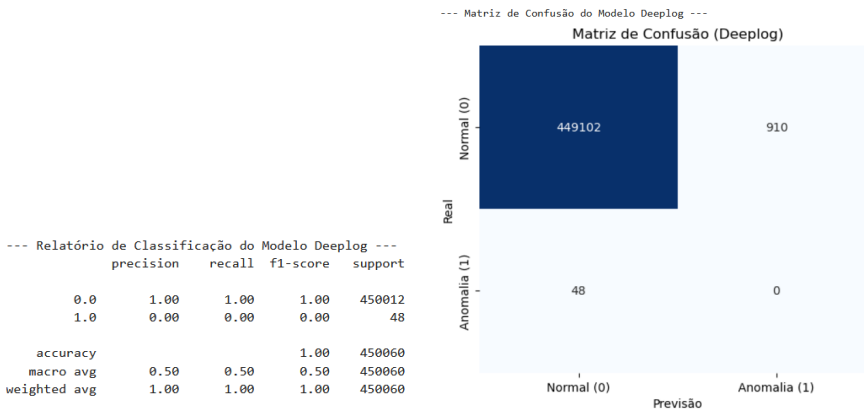
The DeepLog architecture was implemented with the aim of evaluating the effectiveness of Recurrent Autoencoders in modeling nominal patterns in network logs. The fundamental premise of this approach lies in the hypothesis that anomalous instances, by diverging from the learned sequentiality, will present a significantly higher reconstruction error (MSE) than legitimate traffic.

 doi:10.20944/preprints202603.1632.v1

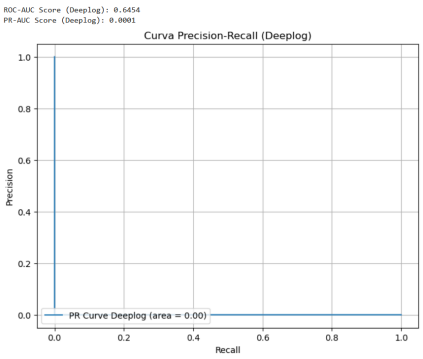
The architecture was developed on a sequential structure composed of stacked GRU layers (128 and 64 units, respectively), integrating a 'TimeDistributed' layer in the decoding block to allow element-by-element reconstruction of the original sequences.

Given the extreme disproportion of the dataset, a balanced subsampling strategy was applied for training, consolidating a set of samples. The model was optimized using the Adam algorithm, using Mean Squared Error (MSE) as the loss function and an Early Stopping mechanism to ensure convergence without overfitting.

The following results, detailed in Figure 8, demonstrate the model's performance:



1 Classification Report **2** Confusion Matrix



3 Precision-Recall Curve

Figure 8. Evaluation metrics for the DeepLog model on (1) Classification Report, (2) Confusion Matrix, (3) Precision-Recall Curve.

The experimental results revealed that this approach was not feasible for the scenario under study. The model recorded zero values for Recall, Precision, and F1-score, failing to identify all 48 threats present in the test set. The PR-AUC value (0.0001) confirms that the classifier operated at a level of statistical indistinguishability.

This limitation suggests a structural incompatibility between the nature of DeepLog and the specific characteristics of the institutional logs analyzed. Contrary to what architecture assumes, anomalies in this context do not manifest themselves through deviations in temporal sequentiality that generate high reconstruction residues. This residual convergence phenomenon indicates that the model was able to reconstruct malicious events with the same accuracy as legitimate events, making it impossible to define a detection threshold.

 doi:10.20944/preprints202603.1632.v1

6. Discussion

In contrast to the success of Boosting algorithms, unsupervised architectures revealed critical limitations in the segregation of anomalous events. This performance disparity stems fundamentally from the intrinsic nature of the dataset and the high data compression during preprocessing, which may have mitigated behavioral variables that are decisive for the identification of structural deviations. In the absence of marked temporal dependencies or distinctive informational variability in the features, models based on reconstruction error converged on a representation where legitimate and malicious traffic become latently indistinguishable.

Thus, methodological evidence suggests that the success of supervised approaches in this domain stems from the ability to map complex nonlinear relationships through explicit feedback from the target variable. In contrast, the effectiveness of unsupervised methods remains strictly dependent on contextual richness that allows the model to isolate anomalies without prior supervision. For the institutional scenario analyzed, explicit labeling and severity-focused feature engineering proved to be the pillars of a robust detection system, overcoming the hypothesis of detection by mere statistical divergence.

7. Conclusions

This paper presented a comprehensive comparative analysis of several Machine Learning and Deep Learning architectures applied to intrusion detection in institutional network logs. Through a rigorous methodology that covered supervised and unsupervised models, it was possible to identify the most effective strategies for dealing with the unbalanced nature and high dimensionality of cybersecurity data.

The results unequivocally demonstrate the superiority of supervised approaches, with particular emphasis on the XGBoost algorithm. After the hyperparameter optimization process, this model established the performance benchmark for the study, achieving a Recall of 0.96 and a Precision of 0.85. This balance is fundamental in operational scenarios, as it ensures the visibility of almost all real threats, while minimizing the cognitive load on security analysts by drastically reducing false alarms.

Recurrent neural networks, specifically Stacked GRU, proved to be robust alternatives, offering remarkable generalization capabilities through deep latent representations. In contrast, unsupervised models based on reconstruction error, such as VAE and DeepLog, revealed critical limitations. Evidence suggests that anomalies in this specific context mimic the statistical structure of legitimate traffic, making reconstruction error an insufficient discriminator for class separation.

The research also documented the practical challenges of interoperability between state-of-the-art frameworks. The inability to integrate Transformer-based architectures, such as BERT and LogBERT, due to backend incompatibilities between TensorFlow and PyTorch, underscores the need for more agnostic and portable development ecosystems in the field of Artificial Intelligence.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Networks
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
I/O	Input/Output
IDS	Intrusion Detection System
LSTM	Long Short-Term Memory
MSE	Mean Squared Error

 doi:10.20944/preprints202603.1632.v1

NaN	Not a Number
PCA	Principal Component Analysis
PR-AUC	Precision-Recall Area Under the Curve
ROC-AUC	Receiver Operating Characteristic Area Under the Curve
SGD	Stochastic Gradient Descent
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
VAE	Variational Autoencoder

References

1. Castro, P.; Santos, F.; Lopes, P. Artificial Intelligence Models for Log Event Analysis. *Millenium - Journal of Education, Technologies, and Health* **2025**, *2025*, e41569–e41569, doi:10.29352/MILL0220E.41569.
2. Pinto, J.C.O. Sistema de Detecção de Intrusão Em Redes Informáticas, Instituto Superior de Engenharia do Porto, 2009.
3. Gutierrez-Garcia, J.L.; Sanchez-DelaCruz, E.; Pozos-Parra, M. del P. A Review of Intrusion Detection Systems Using Machine Learning: Attacks, Algorithms and Challenges. *Lecture Notes in Networks and Systems* **2023**, *652 LNNS*, 59–78, doi:10.1007/978-3-031-28073-3_5.
4. Evolution and Advancements in Intrusion Detection Systems: From Traditional Methods to Deep Learning and Federated Learning Approaches. *ACCENTS Transactions on Information Security* **2024**, *9*, doi:10.19101/TIS.2024.935002.
5. Ali, A.H.; Charfeddine, M.; Ammar, B.; Hamed, B. Ben; Albalwy, F.; Alqarafi, A.; Hussain, A. Unveiling Machine Learning Strategies and Considerations in Intrusion Detection Systems: A Comprehensive Survey. *Front. Comput. Sci.* **2024**, *6*, 1387354, doi:10.3389/FCOMP.2024.1387354/FULL.
6. Landauer, M.; Onder, S.; Skopik, F.; Wurzenberger, M. Deep Learning for Anomaly Detection in Log Data: A Survey. *Machine Learning with Applications* **2023**, *12*, 100470, doi:10.1016/J.MLWA.2023.100470.
7. Sharma, V.; Kumar, M. Comparative Analysis of Machine Learning Models for Intrusion Detection Systems. *Panamerican Mathematical Journal* **2025**, *35*, 273–285, doi:10.52783/PMJ.V35.I3S.3891.
8. Fan, Z.; You, Z. Research on Network Intrusion Detection Based on XGBoost Algorithm and Multiple Machine Learning Algorithms. **2024**, doi:10.54254/2753-8818/31/20241171.
9. Vikrant Sharma Comparative Analysis of Machine Learning Models for Intrusion Detection Systems. *Panamerican Mathematical Journal* **2025**, *35*, 273–285, doi:10.52783/PMJ.V35.I3S.3891.
10. Dhaliwal, S.S.; Nahid, A. Al; Abbas, R. Effective Intrusion Detection System Using XGBoost. *Information 2018, Vol. 9, Page 149* **2018**, *9*, 149, doi:10.3390/INFO9070149.
11. Lindemann, B.; Maschler, B.; Sahlab, N.; Weyrich, M. A Survey on Anomaly Detection for Technical Systems Using LSTM Networks. *Comput. Ind.* **2021**, *131*, 103498, doi:10.1016/J.COMPIND.2021.103498.
12. What Is LSTM - Long Short Term Memory? - GeeksforGeeks Available online: https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory/ (accessed on 10 November 2025).
13. Immastephy, A.J.A.; Punitha, K. A Systematic Review on Network Intrusion Detection System Based on Machine Learning and Deep Learning Approach. *E3S Web of Conferences* **2024**, *540*, 14006, doi:10.1051/E3SCONF/202454014006.
14. Alaref, A. Hash Encoding(Or Feature Hashing) Available online: https://www.kaggle.com/code/adnanalaref/hash-encoding-or-feature-hashing (accessed on 20 October 2025).