Detection of Anomalies in Computer Networks using Machine Learning

Alexandre Aparecido da Silva
alexandreaparecido@alunos.utfpr.edu.br
Federal University of Technology – Paraná (UTFPR)
Apucarana, Paraná, Brazil

Rafael Gomes Mantovani
rafaelmantovani@utfpr.edu.br
Federal University of Technology – Paraná (UTFPR)
Apucarana, Paraná, Brazil

Luiz Fernando Carvalho
luizfcarvalho@utfpr.edu.br
Federal University of Technology – Paraná (UTFPR)
Apucarana, Paraná, Brazil

ABSTRACT

Given the significant growth of cyberattacks, it becomes urgent to develop effective methods for identifying anomalies in computer networks. Traditional security mechanisms, which predominantly rely on static signatures, often struggle to detect novel or sophisticated threats such as zero-day attacks and polymorphic malware. Consequently, the research community is increasingly turning towards data-driven approaches capable of learning dynamic traffic patterns to distinguish between benign behavior and malicious intent with high precision. This work benchmarks four supervised machine learning algorithms: SVM, Random Forest, LSTM, and Logistic Regression. These were applied to the recognition of malicious traffic using the CICIDS2017 dataset, which is known for its diverse set of attacks. The methodology adopted includes preprocessing, normalization, feature selection, and hyperparameter tuning for each algorithm to ensure greater reliability in the results. The F1-score metric was used for model evaluation due to the original data imbalance on the CICIDS2017 data. This methodological choice aims to minimize common distortions in real-world environments, where attacks typically constitute a minority of traffic compared to legitimate traffic. The experimental results indicated that all the algorithms obtained high F1-scores in the $\{0.937; 0.999\}$ interval, with the best results obtained by the RF-induced model. These findings reinforce the potential of combining machine learning techniques with updated datasets to create robust intrusion detection systems, contributing to the security of modern networks.

KEYWORDS

Cybersecurity, Anomaly Detection, Machine Learning, CICIDS2017.

1 INTRODUCTION

The proliferation of connected devices and the exponential growth of data traffic have transformed computer networks into critical infrastructure for modern society. However, this connectivity also exposes organizations to increasingly sophisticated cyber threats [1]. As the volume and complexity of cyberattacks – such as Denial of Service (DoS), Brute Force, and web attacks—continue to increase, traditional signature-based intrusion detection systems (IDS) often struggle to identify new or encrypted malicious activity[2]. Consequently, there is an urgent need to develop more effective, adaptive methods for identifying anomalies in network traffic.

Machine learning (ML) has emerged as a promising approach to address these challenges, offering the ability to learn patterns from historical data and detect deviations indicative of attacks [3]. The adoption of ML in cybersecurity is driven by the limitations of traditional signature-based detection methods, which are often ineffective against novel or polymorphic threats (zero-day attacks) that lack predefined signatures. Unlike static rule sets, ML algorithms excel at analyzing high-dimensional network data to establish baselines of "normal" behavior, enabling the automated detection of subtle anomalies and significantly improving the scalability of defense mechanisms in real-time environments [4]. This study evaluates the performance of four distinct supervised machine learning algorithms: Support Vector Machines (SVM), Random Forest (RF), Long Short-Term Memory (LSTM), and Logistic Regression (LR). Our focus is on detecting malicious traffic using the CICIDS2017 dataset [5], a benchmark known for its diverse representation of modern attack scenarios.

The primary contribution of this work is a rigorous comparative analysis of these algorithms, employing a comprehensive pipeline that includes preprocessing, normalization, and hyperparameter tuning. Special attention is given to the F1-Score metric, which is chosen over accuracy to better account for the class imbalance inherent in network traffic data. This paper is organized as follows: Section 2 presents theoretical concepts necessary to understand this study and some similar proposals. The experimental methodology is presented in Section 3, while the results are discussed in Section 4. Finally, Section 5 presents the final considerations of the study.

2 BACKGROUND AND RELATED WORK

This section defines the fundamental concepts of network anomaly detection and discusses the machine learning paradigms applied in this study. Furthermore, it reviews recent works that have utilized similar datasets and methodologies.

2.1 Anomalies and Network Traffic

Network traffic encompasses the continuous flow of data packets transmitted across a network infrastructure. Within the domain of cybersecurity, this traffic is fundamentally categorized as either *benign*—representing authorized and legitimate user operations—or *malicious*, which denotes activities designed to compromise network integrity, confidentiality, or availability [6].

An anomaly is defined as a pattern of traffic that deviates significantly from the expected behavior of the network. These deviations often signal the presence of cyberattacks. The dataset explored in this study is the CICIDS2017, which includes several specific categories of attacks such as: Brute Force, DoS/DDoS, and Web Attacks [7]. Sharafaldin et al. [7] introduced this dataset to address the shortcomings of older benchmarks, such as KDD99 [8], which lacked the diversity of modern attack scenarios. They generated

the data within a simulated network environment over five days, capturing realistic background traffic alongside malicious flows.

To effectively detect malicious activity, it is essential to understand the behavioral characteristics that distinguish attacks from benign traffic. The attacks analyzed in this study exhibit distinct patterns:

- **Brute Force:** This attack involves an automated system attempting numerous password combinations to gain unauthorized access. In network traffic, this manifests as a high frequency of connection requests to authentication protocols (such as SSH or FTP) from a single source IP within a short time window, often resulting in multiple "failure" response codes [2].
- **DoS and DDoS:** Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks aim to exhaust network resources, rendering services unavailable to legitimate users. These attacks are characterized by an abnormal spike in traffic volume (volumetric attacks) or a flood of specific packet types (e.g., TCP SYN floods) designed to saturate bandwidth or deplete server processing power [1].
- **Web Attacks:** These target vulnerabilities in web applications, such as SQL Injection (SQLi) or Cross-Site Scripting (XSS). Unlike volumetric attacks, web attacks may appear as normal HTTP traffic in terms of volume but contain malicious payloads within the packet data, requiring deep packet inspection (DPI) for detection [6].

Prior to the widespread adoption of machine learning, network intrusion detection relied primarily on two conventional methodologies: signature-based and heuristic-based detection.

Signature-based Detection is the traditional industry standard. It operates by comparing incoming network packets against a database of known malicious signatures (e.g., specific byte sequences or malware hashes). While highly effective against known threats with low false-positive rates, this method fails to detect *zero-day* attacks or polymorphic malware where the signature has been altered [2].

Heuristic and Statistical Detection attempts to overcome these limitations by defining a baseline of "normal" network behavior (e.g., average bandwidth usage or login frequency). Any deviation exceeding a predefined threshold is flagged as an anomaly. However, defining these thresholds is manually intensive; strict thresholds lead to high false-alarm rates (blocking legitimate high-usage traffic), while loose thresholds allow attacks to pass undetected. Machine learning automates this thresholding process, learning complex, non-linear boundaries between benign and malicious behavior without explicit programming.

The classification process typically follows a pipeline: preprocessing, model induction, and evaluation [9]. Given that attack traffic is often a minority class compared to benign traffic, relying on accuracy alone can lead to the "accuracy paradox," where a model appears effective simply by predicting the majority class. Therefore, metrics such as the F1-Score (the harmonic mean of precision and recall) are essential for a reliable evaluation [10].

2.2 Related Works

ML is a branch of artificial intelligence that focuses on the design and development of algorithms capable of learning from and making predictions on data. Rather than following strictly static program instructions, ML algorithms build a mathematical model based on sample data, known as "training data," to make decisions or predictions without being explicitly programmed for the specific task [3].

Generally, machine learning tasks are categorized into three primary paradigms based on the nature of the signal or feedback available to the learning system:

- **Supervised Learning:** The algorithm learns from a labeled dataset where inputs are paired with the correct outputs.
- **Unsupervised Learning:** The algorithm explores unlabeled data to discover hidden structures or patterns (e.g., clustering).
- **Reinforcement Learning:** An agent learns to make decisions by performing actions in an environment and receiving rewards or penalties.

This study specifically employs *Supervised Learning*. Formally, let X denote the input space (feature vectors) and Y denote the output space (labels). We are given a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where each $x_i \in X$ and $y_i \in Y$. The objective of supervised learning is to approximate a mapping function $f : X \rightarrow Y$ such that $f(x)$ accurately predicts the label y for new, unseen input instances x [11].

The classification process typically follows a pipeline: preprocessing, model induction, and evaluation. Given that attack traffic is often a minority class compared to benign traffic, relying on accuracy alone can lead to the "accuracy paradox," where a model appears effective simply by predicting the majority class. Therefore, metrics such as the F1-Score (the harmonic mean of precision and recall) are essential for a reliable evaluation [10].

Several studies have explored the application of ML to the CICIDS2017 dataset. Rodriguez et al. [12] conducted a comparative evaluation of machine learning algorithms specifically for flow-based intrusion detection on CICIDS2017. They reported that tree-based models, such as Random Forest (RF) and J48[1], consistently achieved F1-Scores above 0.99, outperforming other statistical methods in terms of both detection accuracy and computational efficiency.

Building on this, more recent work by Senthilkumar and Kumaresan [13] validated the resilience of Random Forest on the CICIDS2017 dataset. Their 2024 analysis highlighted that, beyond standard accuracy metrics, RF maintained superior performance in noisy data environments compared to other ensemble methods, reinforcing its suitability for practical deployment. Conversely, Udurume et al. [14] evaluated a complex hybrid Deep Learning architecture combining CNN and Bi-LSTM layers. While their model demonstrated robust detection capabilities, they explicitly concluded that traditional ensemble models like Random Forest often provide a superior balance between accuracy and computational resource usage, particularly in resource-constrained environments.

While Deep Learning (DL) [15] models like LSTM [16] are often praised for their ability to capture complex temporal dependencies, they do not always outperform ensemble methods in tabular

[1] J48 is an open-source Java implementation of the C4.5 algorithm provided by the WEKA data mining toolkit. It constructs decision trees using the concept of Information Gain to select the most discriminative attributes and employs pruning techniques to reduce overfitting.

flow data. Vinayakumar et al. [10] highlighted that while DL architectures can be highly effective, they often require significantly more computational resources and training time compared to shallow learning algorithms like RF, which can be a critical factor in real-time IDS deployment.

Furthermore, the quality of the dataset itself is a subject of ongoing research. Engelen et al. [17] performed a critical analysis of CICIDS2017, identifying certain artifacts in the traffic generation process. Their work emphasizes that high performance in ML models must be carefully scrutinized to ensure the model is learning actual attack patterns rather than dataset-specific irregularities, reinforcing the necessity of the rigorous data cleaning and preprocessing steps adopted in our methodology.

3 EXPERIMENTAL METHODOLOGY

This section details the methodology employed for the development and evaluation of the anomaly detection models. It covers the description of the dataset used, the pre-processing steps, the selected classification algorithms, and the experimental evaluation design.

3.1 Dataset

The dataset utilized in this study is the Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CICIDS2017). This is a publicly available benchmark dataset designed for the evaluation of Intrusion Detection Systems (IDS). It was generated within a simulated yet realistic network environment over a period of five days, capturing both benign network traffic and a variety of modern cyberattacks. The benign traffic was generated using a profiling system to simulate realistic human behaviors.

As presented in Table 1, the included attacks cover several categories, such as Brute Force, Denial of Service (DoS), Distributed Denial of Service (DDoS), Web Attacks, and Infiltration. The dataset was obtained from the official source [2] and organized into directories separated by capture days. For the purpose of this study, data from various days were unified, and the problem was treated as a binary classification task.

Table 1: General datasets labels

Labels	Number of Examples
BENIGN	2273097
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack - Brute Force	1507
Web Attack - XSS	652
Infiltration	36
Web Attack - Sql Injection	21
Heartbleed	11

3.2 Pre-processing

The CICIDS2017 dataset consists of labeled network flows. Each record represents a bidirectional flow defined by the 5-tuple (Source IP, Destination IP, Source Port, Destination Port, Protocol) and contains over 80 statistical features. These features include temporal attributes (e.g., `Flow Duration`, `Flow IAT`), volumetric attributes (e.g., `Total Fwd Packets`, `Total Length of Fwd Packets`), and content-based statistical measures (e.g., `Packet Length Mean`, `Packet Length Std`).

Data pre-processing is a critical step to ensure the quality of the model inputs. An automated pre-processing pipeline was applied considering the following steps:

- **Handling Infinite Values:** The dataset contains values labeled as `Infinity` or `-Infinity`. These artifacts typically arise in rate-based features, such as `Flow Packets/s` or `Flow Bytes/s`, when the denominator (flow duration) is zero or extremely close to zero during the feature extraction process. These values were treated as nulls to prevent numerical instability in the classifiers.
- **Removal of Null and Erroneous Data:** A consistency check identified a negligible percentage of rows containing `NaN` (Not a Number) values resulting from the conversion of infinite values or extraction errors. Given the large volume of the dataset (millions of flows), removing these few incomplete instances does not statistically impact the class distribution or the model's ability to learn patterns. Therefore, all rows containing null values were dropped.
- **Label Encoding:** The target column, `Label`, which contains categorical string values (e.g., *BENIGN*, *DoS Hulk*, *PortScan*), was encoded into a binary numeric format (0 for Benign, 1 for Attack) to suit the binary classification objective.
- **Feature Normalization:** The features in the dataset exhibit varying scales (e.g., `Flow Duration` is in microseconds, while `Packet Count` is an integer). To prevent features with larger magnitudes from dominating the objective function, the *MinMaxScaler* was applied. This scaler transforms all features to the range $[0, 1]$. Crucially, the scaler was fitted **only** on the training set and then applied to the test set to avoid data leakage.
- **Reshaping (LSTM Only):** Specifically for the Long Short-Term Memory (LSTM) model, the 2D input vectors were reshaped into a 3D format [(samples, timesteps, features)] to represent each network flow as a sequence with a single timestep, meeting the architectural requirements of Recurrent Neural Networks.

3.3 Algorithms

Four supervised learning algorithms were examined in this study, encompassing both classical statistical methods and deep learning architectures.

[2] https://www.unb.ca/cic/datasets/ids-2017.html

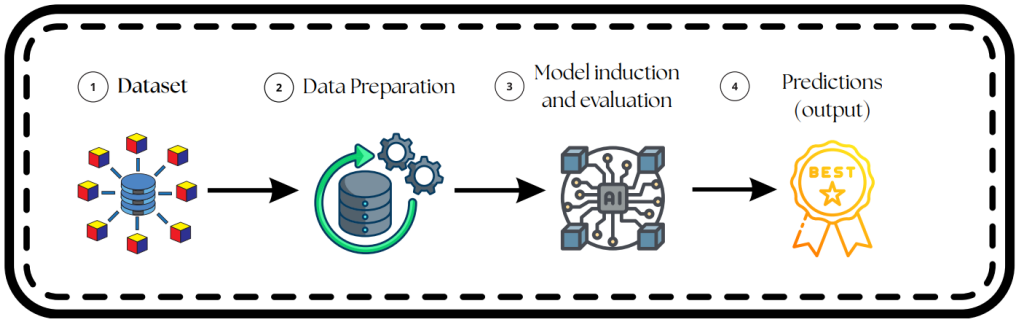

Figure 1: The complete pipeline adopted in the experiments

Logistic Regression was employed as a fundamental baseline due to its simplicity, interpretability, and effectiveness in linearly separable classification tasks, following the foundational principles established by Cox [18].

Random Forest was included for its ability to model non-linear relationships and to handle high-dimensional data with robustness against overfitting, as introduced by Breiman [19].

The **S**upport Vector Machine classifier was also evaluated, given its capacity to maximize the decision margin between classes and its flexibility through kernel functions—including the linear and Radial Basis Function (RBF) kernels—which enables the modeling of complex, non-linear boundaries in the feature space, as defined in the seminal work of Cortes and Vapnik [20].

Finally, a Long Short-Term Memory neural network was adopted as the deep learning component of the study. Although the CIC-IDS2017 dataset does not inherently contain temporal sequences, the LSTM architecture was incorporated to investigate its ability to capture higher-order feature interactions and to evaluate whether recurrent structures can enhance anomaly detection performance beyond traditional machine learning models, an approach supported by the findings of Roopak et al. [21], who demonstrated effective intrusion detection using LSTMs on this dataset.

3.4 Experimental Evaluation

The experimental evaluation followed a systematic and controlled pipeline designed to ensure fairness, reproducibility, and computational efficiency across all tested models. After the pre-processing stage, the dataset was partitioned into training and testing subsets using a 70/30 stratified split to maintain the original class distribution. To ensure the reproducibility of the results and optimize model performance, we employed a Grid Search strategy for hyperparameter tuning. Table 2 details the specific search space defined for each algorithm. For the Random Forest and Logistic Regression models, we explored variations in tree structure and regularization techniques, respectively. The Support Vector Machine tuning focused on kernel types and the regularization parameter (C). Finally, for the Long Short-Term Memory network, we adjusted architecture-specific parameters such as the number of units, dropout rates, and optimizer learning rates, while fixing the maximum epochs at 50 with an Early Stopping mechanism to prevent overfitting.To reduce the computational burden associated with hyperparameter exploration, tuning was executed in parallel through the *joblib* library, leveraging multi-core processing to accelerate experimentation.

Table 2: Hyperparameter Search Space for Grid Search

Algorithm	Hyperparameter	Search Space
Logistic Regression	Regularization (C)	{0.1, 1.0, 10.0}
	Solver	{liblinear, saga}
Random Forest	Number of Estimators	{100, 200}
	Max Depth	{10, 20, None}
	Min Samples Leaf	{1, 2}
SVM	Regularization (C)	{0.1, 1, 5}
	Kernel	{Linear, RBF}
LSTM	Units	{32, 64}
	Dropout Rate	{0.2, 0.4}
	Learning Rate	{0.001, 0.01}
	Batch Size	{64, 128}

In contrast, the LSTM model, implemented using TensorFlow's Keras API, was trained sequentially due to the memory constraints imposed by GPU execution and the model's inherently higher computational cost. For this model, early stopping was employed to prevent overfitting, monitoring the validation loss with a patience of five (5) consecutive epochs, while model checkpointing ensured that the weights corresponding to the best validation performance were preserved.

All models were assessed using the F1-Score evaluation metric due to the dataset's class imbalance. The choice of the F1-Score is due to the fact that it represents the harmonic mean between precision and recall, making it more suitable than accuracy in class imbalance scenarios, which are common in intrusion detection systems. his choice is supported by Ruffo [22], who emphasizes that traditional metrics, such as isolated accuracy, can generate distorted interpretations in situations with unequal class distributions. Throughout the evaluation process, the same train/test split and scaling parameters were consistently reused for all algorithms, thereby ensuring comparability across different experimental configurations.

Given a confusion matrices for a binary classification problem contains:

- **TP (True Positives):** Instances correctly classified as attacks.
- **TN (True Negatives):** Instances correctly classified as normal traffic.
- **FP (False Positives):** Normal instances incorrectly classified as attacks.
- **FN (False Negatives):** Attack instances incorrectly classified as normal.

so, the metrics used in this study can be defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

3.5 Reproduction of Experiments

To ensure strict reproducibility of the study, all procedures involving stochasticity such as dataset splitting, model initialization, and random sampling were executed using a fixed random seed (*random_state=42*). The entire experiment was coded with Python with the libraries: Scikit-learn [23], TensorFlow [24], NumPy, and Pandas. The normalization model generated during pre-processing, specifically the MinMaxScaler instance fitted on the training set, was serialized and stored using to guarantee consistent application across future inferences or replications. Each trained model was also saved to disk, enabling full restoration of model states and hyperparameters. Additionally, all classification reports, confusion matrices, and log files generated during experimentation were archived in both text, CSV and joblib formats, allowing for external verification and complete traceability of the decision-making process. These measures collectively ensure that every stage of the methodology can be reproduced exactly, thereby reinforcing the scientific rigor and transparency of the results presented in this work. The complete experimental setup is presented in Table 3.

Table 3: Experimental configuration for reproducibility.

Item	Description
Dataset	CICIDS2017 (Benign vs Attack)
Split	70/30 stratified
Pre-processing	NaN removal, label encoding; MinMaxScaler (train-fit only); LSTM 3D reshape
Algorithms	LR, RF, SVM (Linear/RBF), LSTM
Tuning	GridSearchCV (ML); early stopping, checkpointing (LSTM)
Metrics	F1-score
Reproducibility	random_state=42, saved scaler & models
Environment	Python 3.12.x, Sklearn, TensorFlow, NumPy, Pandas

4 RESULTS AND DISCUSSION

Table 4 summarizes the test set performance of the optimal configuration identified for each algorithm after the hyperparameter tuning phase.

Table 4: Comparative performance of the best model configurations sorted by F1-Score.

Model	F1-Score	Precision	Recall
Random Forest	**0.9991**	0.9991	0.9991
LSTM	0.9964	0.9964	0.9964
SVM (Linear)	0.9788	0.9789	0.9788
Logistic Regression	0.9368	0.9365	0.9373

The obtained results suggest that ensemble and deep learning methods tend to outperform linear baseline models within the evaluated experimental setup. The Random Forest (RF) model achieved the highest overall performance, reaching a near-perfect Weighted F1-Score of 0.9991. In the context of Intrusion Detection Systems (IDS), this metric reflects a critical balance. The high Precision indicates that when the system flags an anomaly, it is highly likely to be a genuine threat, thereby reducing administrative fatigue caused by false alarms. Simultaneously, the high Recall ensures that the system minimizes 'missing' attacks (False Negatives), which is the primary security objective.

Although the Random Forest model achieved the highest numerical performance, the difference compared to the LSTM network (0.9964) resides in the third decimal place. While this accuracy gain is marginal, the decisive advantage of the RF model lies in its computational efficiency. Unlike the LSTM, which was implemented with computationally intensive layers and is sensitive to hyperparameter complexity (e.g., units, dropout), the Random Forest model proved to be significantly faster to train and less resource-demanding. Given that both models perform essentially at the same level of accuracy, the Random Forest is the preferable choice for this specific deployment scenario due to its lower complexity.

The similarity in performance between the sequence-based model LSTM and the ensemble method RF warrants a critical analysis of the data nature. The dataset consists of flow-based statistical features rather than raw packet sequences. Effectively, these features condense temporal behaviors into a tabular format. Furthermore, the LSTM input was reshaped to, treating each flow as an isolated event rather than a time-series window. Consequently, the problem remains fundamentally tabular. The Random Forest model, which excels at partitioning tabular feature spaces, was able to capture the boundaries between benign and malicious traffic just as effectively as the recurrent architecture, without the overhead of managing statefulness.

In contrast, the linear baseline (Logistic Regression) achieved an F1-Score of 0.9368. While this represents a drop in efficacy compared to the non-linear models, this result is remarkably robust for such a simple algorithm. A score above 0.93 indicates that the vast majority of network traffic is linearly separable in the feature space. The Logistic Regression model primarily struggles with 'edge cases'—sophisticated attacks whose statistical signatures overlap non-linearly with benign traffic. However, considering the trade-off, it remains a strong candidate for resource-constrained environments where a 93% detection rate is acceptable in exchange for extreme speed and model transparency.

4.1 Best Prediction Algorithm

To quantify the actual impact of hyperparameter tuning, we compared the optimal model against the default *scikit-learn* configuration (*n_estimators: 100, min_samples_leaf: 1, max_depth: None*). The default setup yielded a Weighted F1-Score of 0.9989, whereas the optimized configuration (*n_estimators: 200, min_samples_leaf: 2*) achieved 0.9991. Consequently, the hyperparameter tuning resulted in a net performance gain of approximately 0.02%.

Although this numerical improvement is marginal, confirming the inherent robustness of the Random Forest algorithm for this dataset, the preference for the optimized configuration is justified by structural stability. The adjustment of *min_samples_leaf* from 1 to 2 introduces a critical regularization constraint. By preventing the creation of leaf nodes containing single samples, the optimized model reduces variance and mitigates the risk of overfitting to noise, a common issue in high-dimensional network traffic data. Thus, the tuned model offers superior generalization reliability for production deployment, ensuring that the slight metric gain translates into more robust anomaly detection in practice.

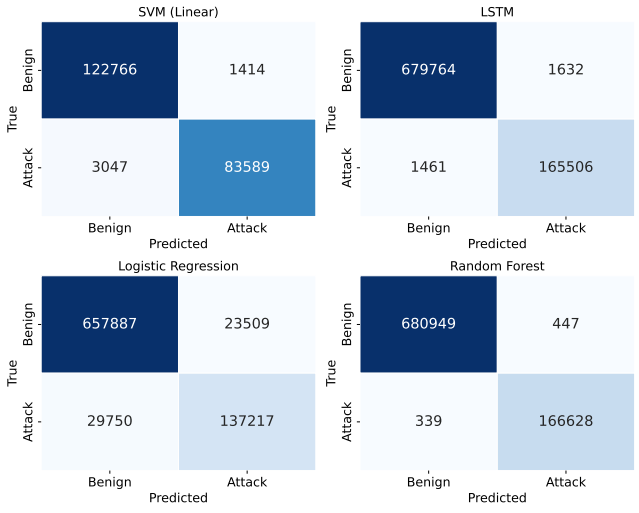

Figure 2: Confusion matrices of the best setups evaluated in the experiments.

The confusion matrices for the Random Forest (RF) model and the baseline classifiers (SVM, LSTM, and Logistic Regression) are collectively presented in Figure 2. The model exhibited a negligible False Positive Rate (FPR), effectively minimizing false alarms—a critical requirement for production-grade Intrusion Detection Systems (IDS). Similarly, the low False Negative Rate indicates that the model successfully detected the vast majority of malicious flows.

Analysis of the secondary diagonal of the confusion matrix reveals that, although the RF model is highly robust, it has not reached perfection. The 447 False Positives (FP) indicate benign flows that mimic malicious behavior, possibly due to traffic bursts (*traffic bursts*) or legitimate services operating on unconventional ports.

On the other hand, the 339 False Negatives (FN) represent the area of greatest criticality. In the CICIDS2017 dataset, these errors tend to focus on "low intensity" attacks or minority classes (such as Infiltration or *Heartbleed*), whose statistical signatures are insufficient for the algorithm to generate clear partitions without incurring *overfitting*. As attacks represent only a fraction of total traffic, the lack of numerical representation of certain sub-attacks causes the model to prioritize the majority class decision boundary, resulting in a subtle overlap where "stealth" attacks end up classified as normal traffic.

When comparing the Random Forest predictions against the other baselines, the distinction in performance is evident. As observed in the Logistic Regression quadrant, the linear model struggles significantly with minority attack classes, resulting in a substantially higher rate of missed detections (False Negatives) compared to the Random Forest. While the LSTM model served as a strong competitive baseline, the Random Forest offered a clear edge in performance, reducing errors while requiring significantly lower computational resources for training and inference.

The false alarms represent benign traffic flows that the model incorrectly classified as attacks. In the context of the massive volume of benign traffic processed, these errors likely stem from benign network behaviors that exhibit statistical outliers resembling attack signatures, such as sudden bursts in packet size or unusual port usage, which fall into the leaf nodes predominantly associated with malicious classes. While these "false alarms" can increase the workload for security analysts, this error type is generally preferred over missing an active threat.

Of greater concern in an Intrusion Detection scenario are the False Negatives, where actual attack traffic is misclassified as benign. Although the Random Forest missed fewer attacks than the LSTM or Logistic Regression models, the persistence of these errors suggests the presence of sophisticated or "stealthy" attack vectors. These specific attacks likely operate within the decision boundary overlap, mimicking the feature distribution of normal traffic and lacking the distinct high-variance features that the algorithm typically relies on for easy discrimination.

The analysis suggests that the system is optimized to prioritize sensitivity. While minimizing False Negatives is the primary goal for an IDS to ensure system integrity, the results show that the model achieves a high degree of optimization by accepting the trade-off of a marginally higher rate of False Positives. This configuration ensures that while a small amount of legitimate traffic may be flagged for review, the vast majority of threats are successfully intercepted.

4.2 Best Model - Importance of Features

Given that the Random Forest (RF) classifier as the superior algorithm in our comparative evaluation, we further investigated the model to verify the patterns driving its predictions. A key advantage of the Random Forest architecture is interpretability factor. Due to the algorithm's operation, which prioritizes features that maximize information gain at each split, we can quantitatively estimate the relative "importance" of specific input characteristics in the decision-making process. Figure 3 illustrates the top features that contributed most significantly to the model's predictive capability.

The analysis highlights that *Destination Port*, *Packet Length Variance*, and *Bwd Packet Length Std* are the dominant discriminators.

- **Destination Port:** Ranked as the most critical feature, this indicates that specific attack vectors in the CICIDS2017 dataset (such as Brute Force or Web Attacks) consistently target distinct services (e.g., SSH on port 22, HTTP on port 80). This allows the decision trees to efficiently isolate attack classes based on the targeted service endpoint.
- **Packet Length Statistics (Variance & Std):** The high importance of *Packet Length Variance* and *Bwd Packet Length Std* suggests that the model successfully learned to distinguish payload characteristics. Automated attacks, such as DoS floods, often exhibit uniform packet sizes (low variance), whereas legitimate user traffic displays high variability due to the dynamic nature of human interaction and diverse media content.
- **Average Packet Size:** Similar to variance, the average size of packets serves as a key indicator of the traffic type, helping to identify anomalies where packet payloads deviate significantly from the expected baseline of normal flows.

Beyond these primary indicators, the model places significant weight on *Backward* traffic statistics, specifically *Avg Bwd Segment Size*, *Bwd Packet Length Mean* and *Total Length of Bwd Packets*. These features effectively capture the network server's response dynamics, allowing the algorithm to detect traffic asymmetry typical of malicious behavior. For instance, scanning activities often trigger numerous small error responses (low backward mean), whereas data exfiltration events are marked by disproportionately large backward payloads relative to the forward requests. Furthermore, boundary indicators like *Max Packet Length* and *Bwd Packet Length Max* (0.0403) are critical for distinguishing between the standardized payloads seen in flooding attacks and the inherent variability found in legitimate interactive sessions. By synthesizing these flow-based statistical markers, the model achieves high discrimination without requiring deep packet inspection.

5 CONCLUSIONS

This study presented a comprehensive evaluation of supervised machine learning and deep learning algorithms for network anomaly detection. Through an experimental pipeline using the CICIDS2017 dataset, we demonstrated that supervised methods, specifically Random Forest, successfully identified both benign traffic and various attack vectors, showing particular efficacy in distinguishing malicious patterns. The Random Forest model achieved an F1-Score of 0.9991, validating its capability in distinguishing between benign traffic and various attack vectors with minimal false positives.

However, despite these promising results, it is essential to acknowledge the limitations inherent in the experimental design. A critical area for improvement is the experimental rigor; Future evaluations should repeat experiments across different random seeds to report the mean and standard deviation of the algorithms' performance. Additionally, reporting the training times is necessary to assess computational efficiency, and employing more advanced hyperparameter tuning methods would further optimize model robustness. Finally, regarding the dataset, while CICIDS2017 is a high-quality benchmark, it represents a static snapshot of network traffic captured in a simulated environment, whereas real-world operational scenarios involve highly dynamic user behaviors and continuously emerging attack patterns. Consequently, future work will move beyond batch learning approaches and focus on *Data Stream Mining*. This paradigm allows for the continuous updating of models as new data arrives, enabling the system to adapt to evolving threats in real-time without the need for complete retraining. In this context, the present research serves as a critical background study. By identifying the most discriminative features (such as Destination Port and Packet Length statistics) and establishing a high-performance baseline with Random Forest, this work provides the foundational knowledge necessary to develop and evaluate adaptive, stream-based intrusion detection systems in future iterations.

REFERENCES

[1] Cisco Systems. Cisco annual internet report (2018–2023) white paper. Technical report, Cisco Systems, 2020. Accessed: 2025-12-10.
[2] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.
[3] Stephen Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015.
[4] Richard Kimanzi, Peter Kimanga, Dedan Cherori, and Patrick K Gikunda. Deep learning algorithms used in intrusion detection systems: A review. *arXiv preprint arXiv:2402.17020*, 2024.
[5] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. CICIDS2017: Intrusion detection evaluation dataset. Canadian Institute for Cybersecurity, 2017. Available at: https://www.unb.ca/cic/datasets/ids-2017.html.
[6] William Stallings. *Network Security Essentials: Applications and Standards*. Pearson, 6th edition, 2017.
[7] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116, 2018. URL https://www.scitepress.org/papers/2018/66398/66398.pdf.
[8] The UCI KDD Archive. KDD Cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999. Information and Computer Science, University of California, Irvine.
[9] Sotiris B Kotsiantis. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.
[10] R Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and S Venkatraman. Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study. *IEEE Access*, 7: 122524–122538, 2019. URL https://ieeexplore.ieee.org/document/8822452.
[11] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
[12] Manuel Rodríguez, Álvaro Alesanco, Luis Mehavilla, and Joaquín García. Evaluation of machine learning techniques for traffic flow-based intrusion detection. *Sensors*, 22(23):9326, 2022. URL https://www.mdpi.com/1424-8220/22/23/9326.
[13] P. Senthilkumar and D. Kumaresan. Comparative performance analysis of machine learning algorithms for intrusion detection systems using the CIS IDS 2017 dataset. *Turkish Journal of Engineering*, 9(3):535–543, 2024.
[14] Miracle Udurume, Vladimir Shakhov, and Insoo Koo. Comparative analysis of deep convolutional neural network—bidirectional long short-term memory and machine learning methods in intrusion detection systems. *Applied Sciences*, 14 (16):6967, 2024. URL https://www.mdpi.com/2076-3417/14/16/6967.
[15] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, Cham, 2018. ISBN 978-3-319-94463-0. doi: 10.1007/978-

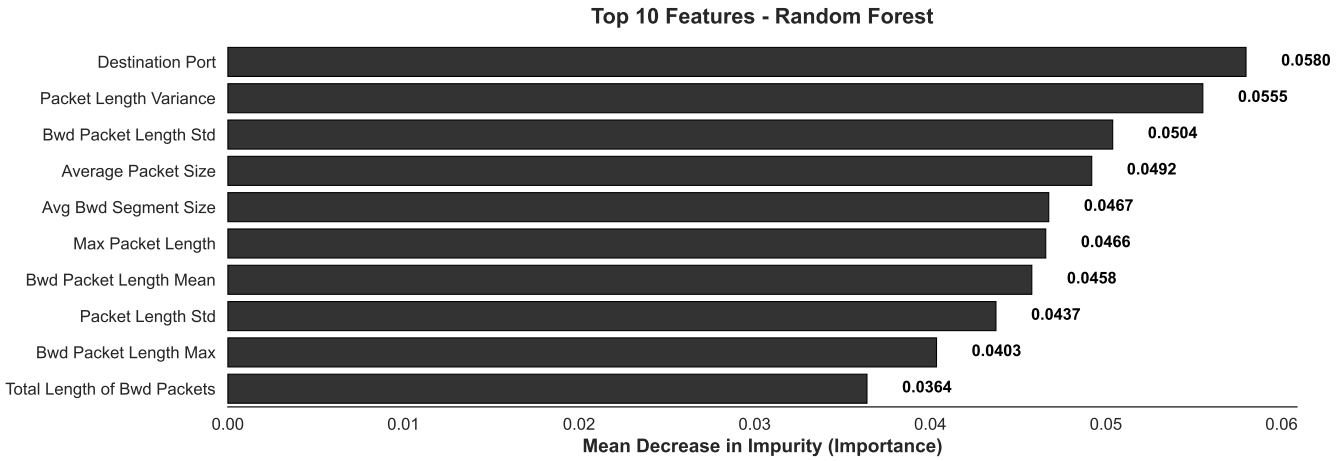

Figure 3: Importance of Features Random Forest

3-319-94463-0.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[17] G Engelen, V Rimmer, and W Joosen. Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12. IEEE, 2021. URL https://ieeexplore.ieee.org/document/9557993.

[18] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[19] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[21] Monowar H Roopak, Gui Yun Tian, and Jonathon Chambers. Deep learning models for cyber security in IoT networks. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 452–457. IEEE, 2019. doi: 10.1109/CCWC.2019.8666588.

[22] Vitor Gabriel da Silva Ruffo, Daniel Matheus Brandão Lent, Mateus Komarchesqui, Vinícius Ferreira Schiavon, Marcos Vinicius Oliveira de Assis, Luiz Fernando Carvalho, and Mario Lemes Proença. Anomaly and intrusion detection using deep learning for software-defined networks: A survey. *Expert Systems with Applications*, 256:124982, 2024. ISSN 0957-4174. doi: 10.1016/j.eswa.2024.124982.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.