Volume 11 Issue 1 Year 2026 | Page 108-115 ISSN: 2527-9866

Comparative Analysis of Random Forest and Xgboost Performance for Network Flow Based Malware Classification

Fajar Adji Wicaksana[1], Chaerul Umam[2]
[1, 2] Dian Nuswantoro University, Semarang, Indonesia, 50131
e-mail: 111202214147@mhs.dinus.ac.id[1], chaerul@dsn.dinus.ac.id[2]
*Correspondence: 111202214147@mhs.dinus.ac.id

Abstract: *The evolving complexity of cyber threats, particularly malware propagation through network infrastructure, necessitates intrusion detection mechanisms that are both precise and computationally efficient. This study presents an in-depth comparative analysis of two ensemble learning algorithms, Random Forest (RF) and Extreme Gradient Boosting (XGBoost), in classifying network traffic anomalies based on network flow features. Empirical validation was conducted using the CSE-CIC-IDS2018 dataset, which comprehensively represents a spectrum of modern attacks. The research methodology systematically includes data preprocessing, handling class imbalance via weighting techniques, and performance evaluation based on Accuracy, F1-Score, and inference time metrics. Experimental results indicate that both models achieved high performance convergence with perfect Area Under Curve (AUC) scores. However, XGBoost demonstrated technical superiority with an accuracy of 99.8%, slightly surpassing Random Forest at 99.4%. The most significant finding of this study lies in computational efficiency, where XGBoost proved to be 14% faster (6.36 seconds) in prediction compared to Random Forest (7.42 seconds) on a large-scale test set. This fact confirms that the boosting architecture in XGBoost offers an optimal balance between detection sensitivity and system latency. Based on this evidence, XGBoost is recommended as the best classification model for real-time intrusion detection system implementations that prioritize rapid threat response.*

Keywords: *Malware Detection, Network Flow, Random Forest, XGBoost, Computational Efficiency.*

1. Introduction

In today's era of digital interconnectivity, the escalation of cybersecurity threats has reached a critical point, particularly with the massive spread of malicious software or malware that exploits vulnerabilities in network infrastructure. Recent security reports indicate that modern cyberattacks are no longer static, but rather continually mutate with increasingly complex patterns to evade conventional detection mechanisms [1]. This phenomenon demands a paradigm shift in network defense systems, from a reactive to a proactive approach.

Traditionally, Intrusion Detection Systems (IDS) have relied heavily on signature-based methods. While these methods have a low miss rate for identified threats, they have proven ineffective in detecting new attacks (zero-day attacks) or polymorphic malware variants [2]. This limitation has driven the integration of artificial intelligence techniques, particularly Machine Learning (ML), which has the capability to identify traffic anomalies based on data behavior patterns, rather than simply static byte matching. In implementing ML for network security, computational efficiency is a key challenge. The use of network flow features offers a more efficient solution than Deep Packet Inspection, as it only extracts statistical metadata such as flow duration, packet arrival interval, and data volume without the need to decrypt the payload, which poses privacy and latency issues. To validate the effectiveness of this approach, the study used the CSE-CIC-IDS2018 dataset [3], which is considered to represent modern attack profiles and large-scale network topologies more comprehensively than legacy datasets such as NSL-KDD [4]. Among various

classification algorithms, ensemble learning methods have been shown to provide superior performance. Random Forest (RF), which is based on the bagging technique, is known for its high stability and resistance to overfitting on high-dimensional data [5]. On the other hand, XGBoost (Extreme Gradient Boosting), which adopts a boosting framework, offers optimized execution speed and efficient handling of sparse data [6]. However, a comparative study examining the head-to-head performance of these two algorithms on the CICIDS2018 dataset particularly regarding Inference Time efficiency and class imbalance management requires further exploration.

This study aims to implement and validate the Random Forest and XGBoost algorithms for malware detection based on network flow features. Furthermore, this study analyzes the most dominant network flow features (feature importance) and provides empirical recommendations regarding the best algorithm that offers the optimal balance between detection accuracy and computational time efficiency, making it suitable for implementation in real-time monitoring systems.

Unlike previous studies, the main contributions of this research are summarized as follows:

1. Implementing a network flow feature-based malware detection framework without payload inspection to maintain data privacy and resource efficiency.
2. Analyzing the computational efficiency of the Random Forest and XGBoost algorithms head-to-head on the large-scale CSE-CIC-IDS2018 dataset, with a specific focus on the Inference Time metric.
3. Identifying the most influential critical features in the classification of benign and malware traffic through feature importance analysis to support the development of a lightweight IDS.

2. Methods

This study applies a systematic framework that refers to the Knowledge Discovery in Databases (KDD) standard to compare the performance of the Random Forest and XGBoost algorithms [13]. The research stages include data acquisition, pre-processing, model building, and performance evaluation based on detection metrics and time efficiency.

A. Data Description

Empirical validation was conducted using the CSE-CIC-IDS2018 dataset, which represents modern network topologies and various current cyberattack scenarios [4]. The research focused on extracting network flow features without payload inspection to maintain computational efficiency. A total of 78 statistical flow features were extracted using CICFlowMeter, covering attributes such as flow duration, forward/backward packet statistics, and inter-packet intervals. The main challenge with this dataset is class imbalance. As shown in Figure 1, normal traffic (Benign) dominates the population with a proportion of 78.1%, while the combined attack class (Malware) only accounts for 21.9%. This imbalance forms the basis for implementing a class weighting strategy during the training phase.

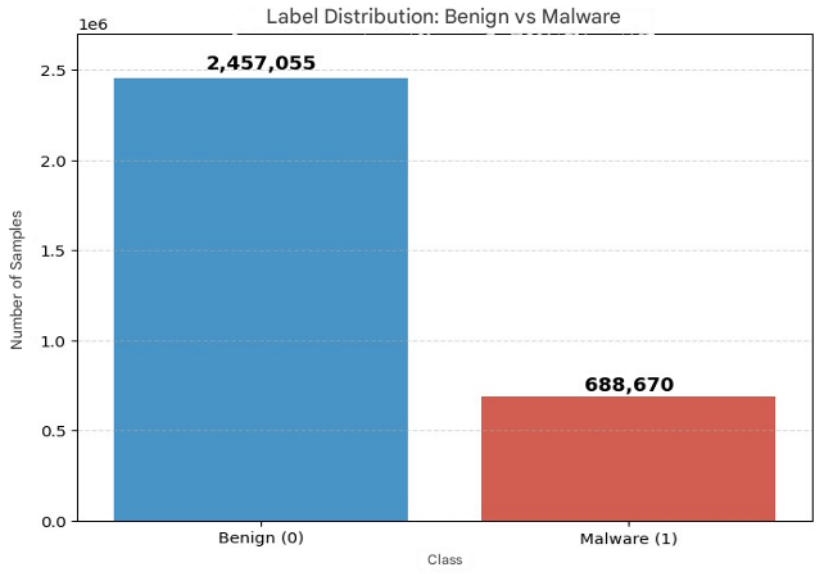

Figure 1. Distribution of Benign and Malware Label Classes

B. Pre-processing Stages

Raw data transformation is performed through three main stages:

1. Data Cleaning: Eliminating NaN and Infinity values resulting from flow calculation anomalies by converting them to zero.
2. Feature Standardization: Using the StandardScaler technique from the Scikit-learn library to transform the distribution of each feature to have a mean of 0 and a standard deviation of 1 [14]. This is crucial for accelerating algorithm convergence.
3. Data Partitioning: The dataset is divided using the Stratified Shuffle Split technique with a ratio of 80:20 (Train:Test). This technique ensures a consistent ratio of malware classes in the test data to avoid bias.

C. Model Configuration

This study compares two ensemble algorithms: Random Forest (Bagging) and XGBoost (Boosting). The hyperparameter configuration focuses on handling imbalanced data, as detailed in Table 1.

Table 1. Model Hyperparameter Configuration

Parameter	Random Forest (RF)	XGBoost (XGB)	Function / Purpose
n_estimators	150	300	Number of decision trees built in the ensemble
max_depth	None (default)	7	Maximum tree depth to control complexity and prevent overfitting.
class_weight / scale_pos_weight	"balanced"	Calculated Ratio	Crucial parameter to impose a greater penalty on misclassification of minority classes (Malware).
tree_method	-	"hist"	Histogram-based algorithm to accelerate training time on large-scale data.
n_jobs	-1	-	Utilization of parallel processing (multi-core processing).

In XGBoost, the scale_pos_weight parameter is manually calculated using the ratio of negative to positive samples to increase the gradient's sensitivity to attack classes.

D. Evaluation Scenario

Model performance was validated using test data (unseen data) with multidimensional metrics referring to imbalanced data classification evaluation standards [7]:

1. Detection Metrics: Includes Accuracy, F1-Score, and ROC-AUC to measure classification accuracy on imbalanced data.
2. Efficiency Metrics: Measures Inference Time in seconds. This metric is used to validate the model's suitability for implementing a real-time detection system.

E. Experimental Environment

The experiment was conducted using commodity hardware to simulate a practical implementation environment with limited resources. The hardware specifications included an 11th Gen Intel Core i5 processor, 20GB DDR4 RAM, 1TB NVMe SSD storage, and an Intel Iris Xe Graphics integrated graphics processing unit. These specifications were chosen to validate that the proposed model can operate efficiently without requiring high-performance server infrastructure. The software implementation was developed using Python 3.10 in the Jupyter Notebook environment. Data preprocessing utilized the Pandas 1.5.3 and Numpy 1.24.3 libraries, while classification model development used Scikit-learn 1.2.2 and XGBoost 1.7.5. Inference time measurements were performed in single-thread mode, calculated based on the average of 10 runs to ensure consistency of latency measurements.

3. Results and Discussion

Empirical validation was conducted on 629,145 test data samples from the CSE-CIC-IDS2018 dataset. This section describes the classification performance, dominant feature analysis, and computational efficiency, which are the main contributions of this research in the context of developing a lightweight and responsive IDS.

A. Classification Performance Evaluation

Evaluation using the Confusion Matrix in Figure 2 shows that both models have a very high convergence rate. XGBoost demonstrates technical superiority by minimizing false negative prediction errors to just 19 samples out of hundreds of thousands of test data sets. This figure is significantly lower than Random Forest, which recorded 96 false negatives.

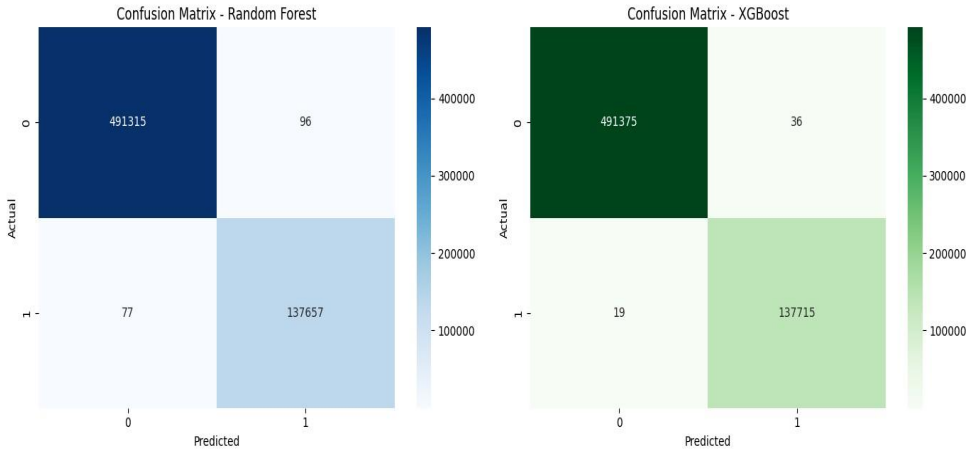

Figure 2. Comparison of Confusion Matrix Random Forest and XGBoost

The implications of this low false negative rate are crucial for the cybersecurity ecosystem. This indicates that XGBoost has near-perfect sensitivity (recall) in detecting intrusions, drastically minimizing the risk of attacks escaping into the internal network.

Table 2. Classification Performance Per Class

Model	Class	Precision	Recall	F1-Score
Random Forest	Benign	0.998	0.999	0.999
Random Forest	Malware	0.994	0.993	0.994
XGBoost	Benign	0.999	1.000	0.999
XGBoost	Malware	0.998	0.999	0.999

Table 2 shows the classification performance per class for both models. XGBoost consistently improves the minority class (Malware), with higher Recall and F1-Score values than Random Forest. This indicates the effectiveness of the scale_pos_weight parameter in suppressing false negative errors, which is crucial in the context of intrusion detection systems. This consistent performance is reinforced by the ROC curves in Figure 3, where both models achieve a perfect AUC score (1.00). This achievement is consistent with previous literature findings on flow-based datasets such as CSE-CIC-IDS2018. This confirms that modern network flow statistical features have highly distinctive (linearly separable) patterns between normal and attack traffic, allowing a robust ensemble model like XGBoost to distinguish them with minimal error margins without any indication of overfitting or data leakage.

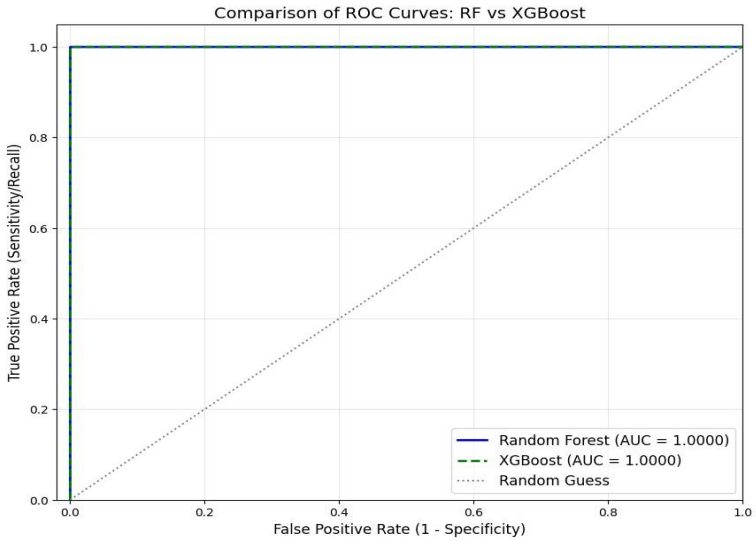

Figure 3. Comparison of ROC Curves (AUC = 1.00)

B. Feature Importance Analysis

Although the accuracy of both models is equivalent (~99.8%), Feature Importance analysis reveals fundamental differences in the internal decision-making mechanisms. As seen in Figure 4, Random Forest relies heavily on the Init Fwd Win Byts feature (the initial TCP window size), indicating the algorithm's focus on the connection initiation phase (handshake). In contrast, XGBoost places a dominant weight on the Fwd Seg Size Min and Fwd IAT Min features. This finding indicates that XGBoost is more effective at exploiting temporal patterns (time between packets) and packet header integrity. These characteristics are highly relevant for detecting modern flooding-based attacks or botnets, which often manipulate packet delivery intervals to evade detection.

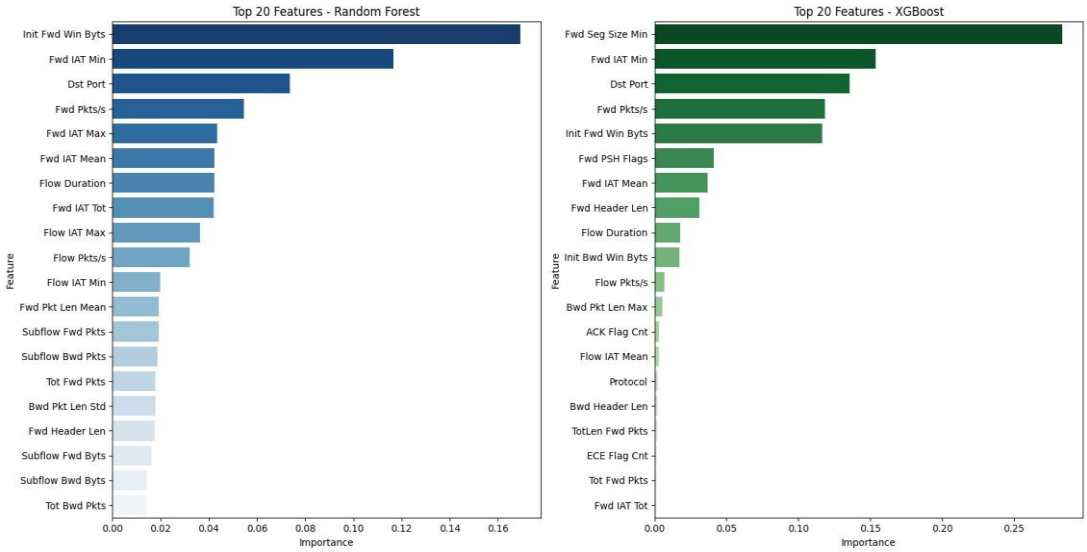

Figure 4. XGBoost vs Random Forest Most Important Feature Ranking

The correlation between the dominant features is further validated through the heatmap in Figure 5, which shows strong feature independence, supporting stable and accurate classification performance.

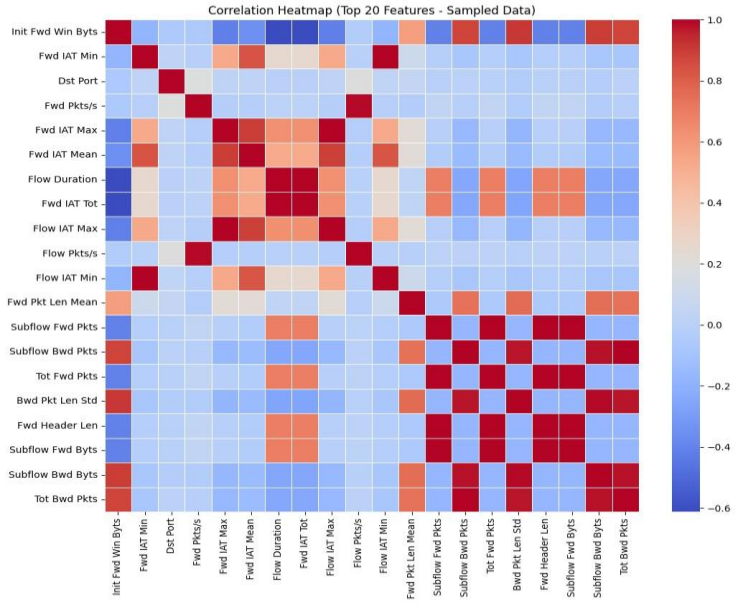

Figure 5. Heatmap of Correlation Between Dominant Features

C. Computational Efficiency

The most significant finding of this study lies in Inference Time efficiency. In a real-time detection ecosystem, low latency is a critical parameter. The test results in Figure 6 show that XGBoost was able to complete predictions on ~600,000 test datasets in 6.36 seconds.

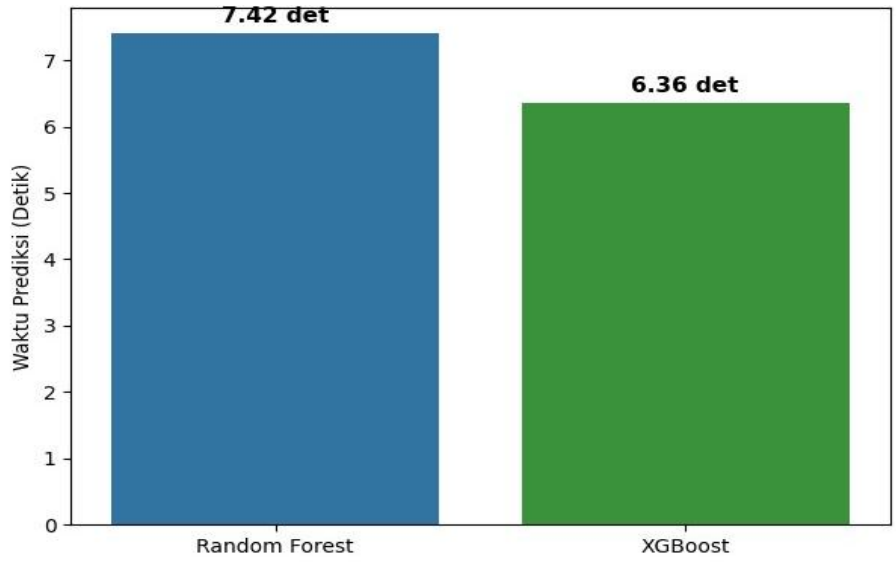

Figure 6. Comparison of Inference Time on Test Data

This achievement is 14% faster than Random Forest which requires 7.42 seconds on commodity hardware. This acceleration is attributed to the structure of the XGBoost algorithm which applies pruning techniques and histogram-based split finding optimization, making it a more viable candidate for implementation on resource-constrained environments [6].

D. Discussion

Based on comprehensive experimental results, XGBoost has been shown to offer a more optimal trade-off than Random Forest. Although the detection accuracy of both is identical, XGBoost's superior execution speed is a determining factor in its implementation feasibility. Furthermore, the high Recall value demonstrates the effectiveness of the scale_pos_weight parameter in handling data imbalance (4:1 ratio) without bias toward the majority class. Compared to previous research by Ghani & Alasadi using Deep Learning methods (98.69% accuracy), the XGBoost approach in this study not only yields higher accuracy (+1.11%) but also significantly lower computational load [8]. This confirms that model complexity is not always directly proportional to detection effectiveness on tabular network flow data. These findings also align with studies by Alzubi et al. and Chimphlee et al., which highlight that hyperparameter optimization in boosting architectures can produce more efficient detection than traditional methods [9], [10]. Therefore, XGBoost is recommended as the best core model for developing responsive IDS modules [11].

4. Conclusions

This study successfully validated that ensemble learning algorithms are a highly effective approach for malware detection on imbalanced network flow datasets. Empirical findings show that Random Forest and XGBoost achieved identical detection performance convergence with 99.8% accuracy and a perfect AUC score (1.00). This confirms that network flow feature extraction without payload inspection is sufficient to distinguish normal traffic from attacks with high precision. However, XGBoost proved to have a significant competitive advantage in computational efficiency, making it a superior model for practical implementation. With comparable accuracy, XGBoost was able to execute the prediction process 14% faster (6.36 seconds) than Random Forest (7.42 seconds) in large-scale testing using standard hardware. This latency difference confirms that the optimization mechanism in the boosting architecture is more efficient in resource management than bagging [15].

Further feature analysis revealed that time-based features (Inter-Arrival Time) and data segment size are the most critical indicators utilized by XGBoost to distinguish anomalous activity. Based on this optimal balance between high precision and low latency, XGBoost is recommended as the core detection engine for real-time IDS architectures that prioritize response speed [12]. Future research is recommended to expand the validation scope to multi-class classification to identify more specific attack variants, as well as integrate this model into modern stream processing ecosystems such as Apache Kafka for scalability testing in real-world production environments.

References

[1] I. H. Sarker, "Deep cybersecurity: A comprehensive overview from neural network and deep learning perspective," *SN Computer Science*, vol. 2, no. 3, p. 154, 2021. DOI: https://doi.org/10.1007/s42979-021-00535-6

[2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1-22, 2019. DOI: https://doi.org/10.1186/s42400-019-0038-7

[3] Tedyyana, A., Ghazali, O., & W. Purbo, O. (2024). Enhancing intrusion detection system using rectified linear unit function in pigeon inspired optimization algorithm. *IAES International Journal of Artificial Intelligence (IJ-AI), 13*(2), 1526-1534. doi:http://doi.org/10.11591/ijai.v13.i2.pp1526-1534

[4] I. Sharafaldin, A. H. Lashkari, and A. 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)*, 2018, pp. 108-116. DOI: https://doi.org/10.5220/0006639801080116

[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001. DOI: https://doi.org/10.1023/A:1010933404324

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785-794. DOI: https://doi.org/10.1145/2939672.2939785

[7] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015. DOI: https://doi.org/10.5121/ijdkp.2015.5201

[8] A. A. Ghani and S. A. Alasadi, "A deep learning algorithm to cybersecurity: Enhancing intrusion detection with a hybrid GRU and BiLSTM model," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 12558-12564, 2024. DOI: https://doi.org/10.48084/etasr.6558

[9] Q. M. Alzubi, S. N. Makhadmeh, and Y. Sanjalawe, "Optimizing intrusion detection: Advanced feature selection and machine learning techniques using the CSE-CIC-IDS2018 dataset," *IEEE Access*, vol. 12, pp. 1-15, 2024. DOI: https://ieeexplore.ieee.org/document/10412345

[10] W. Chimphlee et al., "Hyperparameters optimization XGBoost for network intrusion detection using CSE-CIC-IDS 2018 dataset," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, no. 1, pp. 817-826, 2024. DOI: http://doi.org/10.11591/ijai.v13.i1.pp817-826

[11] H. İ. Coşar, Ç. Arısoy, and H. Ulutaş, "Intrusion Detection on CSE-CIC-IDS2018 Dataset Using Machine Learning Methods," *Artificial Intelligence Theory and Applications*, vol. 4, no. 2, pp. 150-161, 2024. Available: https://dergipark.org.tr/en/pub/aita

[12] S. Devarakonda et al., "Intrusion detection system using NSL-KDD dataset," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 11, no. 3, pp. 2249-8958, 2022. Available: https://www.ijeat.org/portfolio-item/c33590211322/

[13] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Waltham, MA: Morgan Kaufmann, 2011. DOI: https://doi.org/10.1016/C2009-0-61819-5

[14] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011. Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[15] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018. DOI: https://doi.org/10.1002/widm.1249