DEPARTMENT OF STATISTICS
AHMADU BELLO UNIVERSITY, ZARIA
2nd International Conference on Statistical Sciences and Data Analytics (ICSSDA 2026) Proceedings Paper

Enhancing IoT Intrusion Detection using a Machine Learning Framework

Matthew Tunde Ogedengbe[1*], Caleb Faveren Shangbum[1], Aminat Bolatito Yusuf[1], and Sikirat Kehinde Aina[1]

[1]Department of Computer Science, Joseph Sarwuan University, Makurdi, Nigeria

[2]Department of Computer Science, Ahmadu Bello University, Zaria, Nigeria

[2]Department of Computer Science, Federal University, Gashua, Nigeria

*Correspondence: matt.oged@uam.edu.ng, faveren.shangbum@uam.edu.ng, abyusuf@abu.edu.ng, kennystain@fugashua.edu.ng

Abstract

The increasing adoption of the Internet of Things (IoT) in smart workplace environments has significantly expanded the cyberattack surface, exposing interconnected devices to increasingly sophisticated security threats. Conventional intrusion detection systems often face challenges associated with class imbalance, high-dimensional network traffic, feature representation, and evolving attack patterns. Addressing these challenges requires intrusion detection frameworks that combine effective classification with robust data preprocessing strategies to improve the quality and reliability of network traffic analysis. This study presents a machine learning-based IoT intrusion detection framework that integrates a data preprocessing pipeline with a Random Forest classifier for detecting malicious traffic in smart IoT environments. A simulation-based IoT cybersecurity dataset was developed using Python to emulate representative smart workplace network traffic under normal and malicious operating conditions. The preprocessing workflow incorporates data cleaning, categorical data encoding, training-only class balancing, feature scaling, and dimensionality reduction before model training, with the evaluation procedure designed to prevent information leakage between training and testing data. Experimental evaluation on the simulated dataset achieved an accuracy of 97.45%, precision of 99.11%, and recall of 92.33%, demonstrating strong capability in distinguishing normal and malicious IoT traffic. Additional ablation and robustness analyses were conducted to examine the contribution of the preprocessing stages and the stability of the framework under controlled evaluation conditions. The proposed framework provides a reproducible and effective approach for machine-learning-based intrusion detection in smart IoT environments.

Keywords: Internet of Things, Intrusion Detection System, Machine Learning, Random Forest.

1. Introduction

The Internet of Things (IoT) has transformed workplace environments by enabling seamless connectivity, automation, and real-time decision-making through interconnected smart devices

(Atzori *et al.,* 2010). Its widespread adoption has improved operational efficiency through intelligent resource management, monitoring, and communication among connected devices (Al-Garadi *et al.,* 2018; Singh *et al.,* 2021). However, the rapid growth of IoT devices has also expanded the cyberattack surface, exposing smart workplaces to threats such as Distributed Denial-of-Service (DDoS), Man-in-the-Middle (MITM), spoofing, and botnet attacks (Mosenia & Jha, 2017; Singh *et al.,* 2021). The heterogeneous nature of IoT devices, together with constrained computational resources, diverse communication protocols, and weak security configurations, further complicates the protection of IoT environments.

Intrusion Detection Systems (IDS) have therefore become an important component of IoT security for monitoring network activities and identifying malicious behaviour. Conventional signature-based IDS are effective against previously known attacks but have limited capability in detecting novel or zero-day threats, whereas anomaly-based approaches can identify previously unseen behaviour but may generate high false-alarm rates (Garcia-Teodoro *et al.,* 2009; Shone *et al.,* 2018). These limitations have encouraged the adoption of machine learning (ML) and deep learning (DL) techniques, which can learn patterns from network traffic and automatically distinguish legitimate from malicious activities (Javaid *et al.,* 2016; Al-Garadi *et al.,* 2018).

ML-based IDS have received considerable attention because algorithms such as Decision Trees, Support Vector Machines, Logistic Regression, K-Nearest Neighbours, Naïve Bayes, and Random Forest (RF) can provide effective classification with comparatively lower computational requirements than complex deep learning architectures (Mustafa *et al.,* 2024; Adeyemi, 2024; Kikissagbe & Adda, 2024). Recent studies have demonstrated the applicability of these approaches in different IoT security scenarios. Alfardus and Rawat (2024), for example, applied Binary Logistic Regression to intrusion detection in Controller Area Networks, demonstrating its suitability for lightweight binary classification. Devine et al. (2025) investigated federated Support Vector Machines for DDoS detection in IoT networks, while Kalaria *et al.* (2024) developed IoTPredictor for identifying malicious IoT device behaviours. Random Forest has also demonstrated competitive performance in IoT malware and botnet detection, with Souza and Arima (2024) and Das *et al.* (2025) reporting strong results using conventional machine-learning approaches. Nevertheless, ML-based IDS continue to face challenges associated with class imbalance, feature redundancy, data representation, and evolving attack behaviours (Kikissagbe & Adda, 2024).

Deep learning approaches have similarly demonstrated strong capability in learning complex representations from IoT network traffic. Architectures including Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks have been investigated for detecting sophisticated cyber threats. Awajan (2023) developed a deep learning-based IDS using a fully connected neural network and reported an accuracy of 93.74%. Yaras and Dener (2024) subsequently combined CNN and LSTM architectures to exploit both spatial and temporal characteristics of IoT traffic, while Jablaoui and Liouane (2025) investigated combined CNN-RNN architectures for IoT intrusion detection. More recent work has incorporated attention mechanisms with CNN-BiLSTM architectures to improve the detection of attacks in resource-constrained IoT networks (Shakir *et al.,* 2026). Although these approaches can achieve strong detection performance, their computational requirements, dependence on large labelled datasets, and increased model complexity can limit practical deployment in constrained IoT environments (Khaleel *et al.,* 2024).

Beyond classifier selection, the quality and representation of input data remain important determinants of IDS performance. IoT traffic datasets commonly contain categorical attributes, redundant or high-dimensional features, and imbalanced attack classes, which can adversely affect model learning and generalisation (Ferrag *et al.,* 2020; Hajiheidari *et al.,* 2019; Akintade *et al.,* 2025). Consequently, recent studies have incorporated preprocessing and feature-engineering techniques to improve the quality of network traffic representations. Souza and Arima (2024) demonstrated the importance of data preprocessing in IoT malware detection, while Ma et al. (2025) employed feature selection and synthetic data generation to reduce redundant features and address class imbalance.

Das *et al.* (2025) combined Principal Component Analysis (PCA), Synthetic Minority Over-sampling Technique **(**SMOTE), Recursive Feature Elimination, and normalisation for IoT botnet detection, while Hamidou and Mehdi (2025) incorporated SMOTE and model optimisation into a comparative IDS framework. These studies demonstrate the relevance of preprocessing for IoT intrusion detection; however, the individual contribution of preprocessing components and their interaction within a leakage-controlled end-to-end workflow require further examination.

Dataset selection is also an important consideration in IoT intrusion detection. Public benchmark datasets provide valuable resources for comparative evaluation; however, their traffic characteristics and attack distributions may not necessarily reflect the communication behaviour of a particular deployment scenario (Koroniotis *et al.,* 2019). Dataset quality can also affect model generalisation and the reliability of reported performance (Sudyana *et al.,* 2024). Controlled simulation-based datasets can therefore complement public benchmarks by allowing the IoT environment, device population, traffic-generation process, attack scenarios, and experimental parameters to be explicitly defined. Such control is particularly important when reproducibility and systematic evaluation of an intrusion detection pipeline are central objectives.

Despite the progress reported in ML- and DL-based IoT intrusion detection, two important gaps remain. First, existing studies frequently emphasise overall classifier performance while providing limited examination of how preprocessing operations such as categorical encoding, class balancing, scaling, and dimensionality reduction affect the resulting detection pipeline. Second, simulation-based studies do not always provide sufficient information about dataset-generation parameters or establish safeguards against information leakage during preprocessing and model evaluation. These limitations motivate a controlled and reproducible experimental framework in which both the data-generation process and the subsequent machine-learning pipeline are explicitly defined and evaluated.

Motivated by these limitations, this study proposes a machine learning-based IoT intrusion detection framework that integrates a controlled simulation-based IoT cybersecurity dataset, a structured preprocessing pipeline, and a Random Forest classifier for cyberattack detection in smart workplace environments. The simulation environment is developed in Python to generate representative IoT network traffic under normal and malicious operating conditions. The preprocessing workflow incorporates data cleaning, categorical feature encoding, training-only class balancing using SMOTE, feature scaling, and dimensionality reduction using PCA. Importantly, the dataset is divided into training and testing subsets before the training-dependent preprocessing operations are fitted, ensuring that information from the test partition is not used during model preparation. The framework is subsequently evaluated using controlled performance, preprocessing ablation, encoding-sensitivity, and source-file-grouped robustness analyses. The major contributions of this study are:

1. Development of a controlled simulation-based IoT cybersecurity dataset for intrusion-detection experimentation in a defined smart workplace environment, with explicit reporting of the dataset-generation parameters required for reproducibility.
2. Design of a leakage-controlled preprocessing and Random Forest framework in which data splitting precedes training-dependent operations, with SMOTE, scaling, and PCA fitted exclusively on the training data.
3. Systematic evaluation of the proposed framework through preprocessing ablation, label-encoding sensitivity, and source-file-grouped robustness analysis to examine the contribution and stability of the experimental pipeline.

2. Materials and Methods

2.1. Proposed IoT Intrusion Detection Framework

This study follows the proposed IoT intrusion detection framework, which integrates simulation-based dataset generation, data preprocessing, leakage-controlled model development, and performance evaluation. The framework provides a structured workflow for transforming simulated IoT network traffic into model-ready data and evaluating intrusion detection performance under controlled experimental conditions. The major stages of the framework are illustrated in Figure 1, while the following subsections describe the dataset generation, feature preparation, preprocessing, model development, and evaluation procedures in detail.

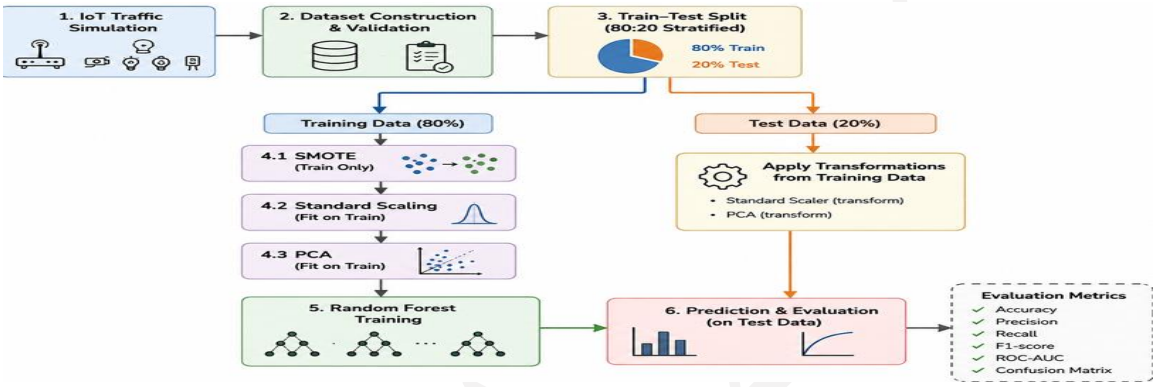

Figure 1. Proposed IoT Intrusion Detection Framework and Experimental Pipeline.

2.2. Simulation-Based IoT Dataset Generation

A controlled simulation-based IoT cybersecurity dataset was developed in Python to provide a reproducible environment for evaluating the proposed intrusion detection framework. The simulation was designed to represent heterogeneous devices operating within a smart workplace environment and generating both normal and malicious network traffic. Rather than relying on an uncontrolled collection of observations, the simulation parameters were explicitly defined before data generation, allowing the experimental conditions to be reproduced and the characteristics of the resulting dataset to be documented. The principal simulation and dataset-generation parameters used in the final experiment are presented in Table 1.

Table 1. Simulation and dataset-generation parameters

Parameter	Experimental specification
Simulation environment	Python-based simulation
Simulation duration	3,600 s (1 h)
Random seed	2026
Number of IoT/network devices	50
Target flows	10,000
Traffic generation mode	Flow-based
Final dataset observations	10,000
Benign observations	7,000
Malicious observations	3,000
Packet-size range	40–1,500 bytes
Packet-size representation	Integer bytes
Device categories	7

Attack-generation families	Brute Force, DDoS, DoS, Mirai, Recon, Spoofing, Web-based
Reference traffic schema	CICIoT2023-compatible
Reference feature count	46
Common modelling features	37

The device population was distributed across seven device categories to represent heterogeneous communication behaviour rather than treating all devices as identical traffic sources. The final population with the category counts summing to the 50 devices specified for the simulation is shown in Table 2.

Table 2. Simulated IoT device population

Device type	Number of devices
Temperature sensor	10
Humidity sensor	8
Smart camera	8
Smart light	8
Smart plug	7
Gateway	3
Workstation	6

Each device category used a predefined normal-traffic profile covering packet count, packet size, inter-arrival time, protocol, application, and destination port. Traffic was generated at the flow level to preserve differences in device communication behaviour. The resulting dataset followed a 46-feature CICIoT2023-compatible schema, with 37 common features retained as the final modelling space. Detailed attack-generation parameters and dataset validation procedures are presented in the following subsections.

2.3. Attack Traffic Generation

Malicious traffic was generated using parameterised mechanisms representing seven attack families: Brute Force, DDoS, DoS, Mirai, Recon, Spoofing, and Web-based attacks. The mechanisms were designed to reproduce traffic-level characteristics associated with representative malicious behaviours under controlled simulation conditions. Since the dataset is simulation-based, the generated attacks represent controlled behavioural approximations rather than exact replicas of real-world attack campaigns.

For DDoS traffic, the generation mechanism modified the corresponding normal-traffic profile using predefined intensity and temporal parameters. The principal parameters used in the final experiment are presented in Table 3.

Table 3. DDoS attack-generation parameters

Parameter	Experimental specification
Number of attackers	5
Attack-flow probability	0.80
Packet-rate multiplier	8.0×
Packet-rate multiplier	0.125×
Flow-duration multiplier	0.35×
TCP / UDP / ICMP distribution	0.40 / 0.40 / 0.20
Forward-packet probability	0.85

Target device type	Gateway
Random seed	2026

Here, the × symbol denotes a multiplicative scaling factor applied to the corresponding traffic-profile parameter. Thus, a packet-rate multiplier of 8.0× increases the corresponding packet-rate value eightfold, while an inter-arrival-time scale of 0.125× and a flow-duration multiplier of 0.35× reduce the corresponding values to 12.5% and 35%, respectively. If X_normal represents the corresponding parameter from the normal traffic profile and m is its attack-specific multiplier, the modified value can be expressed as:

$$X\_attack = m \times X\_normal \quad (1)$$

The protocol distribution and forward-packet probability were independently used to control the communication characteristics of the generated DDoS flows. These parameterised mechanisms provide explicit information for reproducing the simulated attack behaviour while maintaining the distinction between controlled simulation and attacks observed in operational networks.

2.4. Dataset Validation and Freezing

Before model development, the generated dataset was subjected to structural, numerical, and feature-consistency validation. The simulation produced traffic according to a 46-feature CICIoT2023 - compatible schema. Following schema alignment and exclusion of non-predictive fields, 37 common traffic features were retained as the final modelling feature space. The label was excluded from the predictors, and timestamp information was not used as a modelling feature.

The final dataset contained exactly 10,000 observations, comprising 7,000 benign and 3,000 malicious observations. Validation confirmed that all 37 modelling features were numeric, with no missing values or infinite values. Packet-level and feature-consistency checks were also performed during dataset construction to confirm that generated flows conformed to the intended traffic representation.

After validation, the dataset was frozen as the fixed input for all subsequent model experiments. Its integrity was verified using a SHA-256 checksum. The frozen dataset and associated experimental configuration records were retained unchanged throughout the subsequent preprocessing, training, and evaluation stages.

2.5 Leakage-Controlled Preprocessing Pipeline

To prevent information leakage, the frozen dataset was first partitioned into training and testing subsets before any data-dependent preprocessing operation was fitted. The test partition was then locked and excluded from preprocessing fitting, class balancing, and model-development decisions. The preprocessing stages followed the workflow presented in Figure 1, with the training and testing branches handled separately after the initial split. The statistical techniques employed in this study were as follows:

2.5.1 Train and Test Partitioning

A stratified 80:20 train/test split was applied to the frozen dataset, preserving the relative proportions of benign and malicious observations in both partitions. The dataset was therefore divided as:

$$D = D_{train} \cup D_{test} \quad (2)$$

$$D = D_{train} \cap D_{test} = \emptyset \quad (3)$$

Approximately 80% of the observations were used for model development, and 20% were retained as the locked test set. No information from the test partition was used to fit subsequent preprocessing transformations or make model-development decisions.

2.5.2 Label Encoding

Label encoding was selected because the experiment performs binary classification and the Random Forest classifier can directly use numerical class labels. The encoded label was treated solely as the response variable and was excluded from the predictor matrix.

A sensitivity comparison between label encoding and one-hot representation was also conducted. The outcome of this comparison is presented in Section 3.

2.5.3 SMOTE-Based Class Balancing

SMOTE was applied only to the training partition. Synthetic minority observations were generated from existing minority-class training observations using interpolation between a minority observation and one of its nearest neighbours:

$$\boldsymbol{x_{new} = x_i + \lambda(x_{nn} - x_i),\ 0 \leq \lambda \leq 1} \quad (4)$$

Where $\boldsymbol{x_i}$ is a minority-class observation, $\boldsymbol{x_{nn}}$ is a selected minority-class nearest neighbour, and $\boldsymbol{\lambda}$ is a randomly selected interpolation factor. In the source-file grouped robustness evaluation, the training data contained 5,119 benign and 2,787 DDoS observations before SMOTE. After oversampling, both classes contained 5,119 observations, producing 10,238 training observations. The locked test partition was not subjected to SMOTE.

2.5.5 Principal Component Analysis

PCA was fitted after standardisation using the training data only. Starting from the 37 modelling features, PCA retained 11 principal components, accounting for 95.63% of the variance in the grouped robustness evaluation. The PCA transformation can be represented as:

$$\boldsymbol{Z = XP} \quad (5)$$

where X represents the standardised training feature matrix, and P contains the selected principal-component directions. The same fitted PCA transformation was subsequently applied to the locked test data without refitting. Thus, the test observations did not contribute to the estimation of either the scaling parameters or the principal-component directions.

2.6 Random Forest Model Development

The processed training data were classified using an RF classifier. RF was selected because it can model nonlinear relationships among network-traffic features and is suitable for classification using multiple input variables. The classifier was trained using the transformed training data generated by the leakage-controlled preprocessing pipeline described in Section 2.5.

For the primary experiment, the RF classifier was configured with a fixed random seed of 2026 and 100 decision trees (estimators). The model was trained on the PCA-transformed training representation and evaluated on the locked test partition after application of the same fitted scaling and PCA transformations. No hyperparameter tuning or test-set model selection was performed for the primary evaluation.

After training, the fitted Random Forest was applied to the transformed test observations to generate the final predictions. Performance was assessed using accuracy, precision, recall, F1-score, and ROC-AUC. Additional RF configurations, including evaluation without PCA and hyperparameter tuning, were investigated separately as controlled analyses and are reported in the Results and Discussion section.

3. Results and Discussions

The experimental results were obtained using the frozen simulated IoT cybersecurity dataset and the leakage-controlled workflow described in Section 2. The primary evaluation used the complete preprocessing pipeline, including training-only class balancing, feature standardisation, PCA, and Random Forest classification. The primary performance is first presented, followed by controlled ablation and sensitivity analyses examining the contribution of preprocessing stages, dimensionality

reduction, and categorical encoding. Finally, the framework is evaluated using the established CIC-IoT2023 dataset to provide an additional external validation of its intrusion-detection capability. Unless otherwise stated, reported test results were obtained from data partitions that were not used to fit preprocessing transformations or train the final classifier.

3.1 Primary Full-Pipeline Performance

The primary experiment evaluated the complete proposed pipeline on the frozen simulated dataset. Following the stratified 80:20 partition, SMOTE was applied only to the training data, while the scaler and PCA transformation were fitted exclusively on the training partition. The locked test partition was subsequently transformed using the fitted parameters and used for final Random Forest evaluation. The performance of the classifier is shown in Table 4. These results demonstrate strong discrimination between benign and malicious IoT traffic under the leakage-controlled evaluation procedure. The result of the shown in Table 4.

Table 4. Primary full-pipeline performance

Metric	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Performance	97.45%	99.11%	92.33%	95.60%	99.10%

Also, the corresponding confusion matrix of Table 4 is shown in Figure 2. Figure 2 shows that the model correctly classified most observations in both classes, with only five benign observations incorrectly classified as malicious. The 46 false-negative predictions represent the principal source of classification error for malicious traffic and are reflected in the lower recall relative to precision.

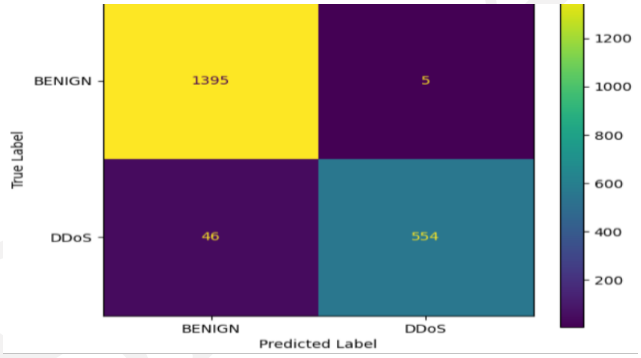

Figure 2. The Confusion matrix of the proposed Random Forest classifier

3.2 Preprocessing Pipeline Ablation Analysis

The examination of the contribution of the major preprocessing stages to the proposed framework, controlled ablation experiments were conducted using the frozen simulated dataset and the same Random Forest evaluation procedure. The analysis considered the effects of PCA and SMOTE on classification performance. The PCA analysis compares Random Forest performance with and without dimensionality reduction, while the SMOTE analysis examines whether training-set class balancing produces a measurable improvement. These complementary experiments provide evidence for interpreting the role of each preprocessing stage in the complete pipeline.

3.2.1 Principal Component Analysis and Random Forest Ablation Analysis

An ablation experiment was conducted to determine whether PCA contributed to the performance of the Random Forest classifier of section 3.1. The Random Forest with PCA used the frozen dataset, a stratified 80:20 train-test partition, training-only SMOTE, and the evaluation procedure as the primary experiment. The Random Forest without PCA used the frozen dataset, a stratified 80:20 train-test partition, training-only SMOTE, and the evaluation procedure only on Random Forest. The principal difference between the two configurations was the feature representation supplied to the Random Forest. The RF without PCA configuration used the original 37 modelling features, whereas the RF with PCA configuration used the PCA-transformed representation after training-only standardisation.

PCA was fitted exclusively on the training partition. The 37 modelling features were reduced to 11 principal components, retaining 95.56% of the variance in the training representation. The same fitted transformation was subsequently applied to the locked test partition without refitting.

The performance obtained from the two configurations is presented in Table 5. The table compares the models using accuracy, precision, recall, F1-score, and ROC-AUC, thereby allowing the effect of PCA on the Random Forest classifier to be assessed consistently across the principal evaluation measures.

Table 5. Random Forest performance with and without PCA

Configuration	Accuracy	Precision	Recall	F1-score	ROC-AUC
Random Forest without PCA	99.30%	100.00%	97.67%	98.82%	99.84%
Random Forest with PCA	97.45%	99.11%	92.33%	95.60%	99.10%

As shown in Table 5, these results show that PCA did not improve Random Forest performance for the simulated dataset. Although PCA reduced the feature representation from 37 variables to 11 components while retaining 95.56% of the training variance, the original feature representation provided more useful information for the Random Forest classifier. This is logical because PCA requires scaling when features have very different numerical ranges, but Random Forest itself generally does not require feature scaling. The findings therefore do not support a general claim that PCA improves tree-based intrusion detection. Instead, the effect of dimensionality reduction should be determined empirically for the dataset and classifier under investigation.

3.2.2 Synthetic Minority Over-sampling Technique and Random Forest Ablation Analysis

The contribution of the Synthetic Minority Over-sampling Technique (SMOTE) was examined by comparing Random Forest performance before and after applying class balancing to the training data. In the first configuration, Random Forest was trained directly using the raw modelling features without SMOTE. In the second configuration, SMOTE was applied to the training data before Random Forest classification. The remaining evaluation procedure was kept unchanged, allowing the effect of class balancing to be assessed independently.

The sequential ablation results, including the subsequent addition of standardisation and PCA, are presented in Table 6. This comparison provides a progressive view of how the preprocessing stages affected the Random Forest classifier.

Table 6. Sequential preprocessing ablation results

Configuration	Accuracy	Precision	Recall	F1-score	ROC-AUC
A - Raw features with RF	99.30%	100.00%	97.67%	98.82%	99.84%
B - Raw features, SMOTE and Random Forest	99.30%	100.00%	97.67%	98.82%	99.84%
C - Raw features, SMOTE, Standard Scaler and RF	99.30%	100.00%	97.67%	98.82%	99.10%
D - Raw features, SMOTE, Standard Scaler, PCA and RF	97.45%	99.11%	92.33%	95.60%	99.10%

As shown in Table 6, introducing SMOTE in Condition B did not change accuracy, precision, recall, or F1-score compared with the raw-feature Random Forest in Condition A. All four measures remained at 99.30%, 100.00%, 97.67%, and 98.84%, respectively. ROC-AUC increased slightly from 99.80% to 99.84% after SMOTE. The subsequent addition of standardisation in Condition C produced essentially the same classification results, with ROC-AUC remaining approximately 99.84%.

These findings indicate that SMOTE did not produce a measurable improvement in the principal classification metrics under the tested conditions. The absence of a change in accuracy, precision, recall, and F1-score suggests that the Random Forest classifier was already able to achieve strong

performance using the original feature representation. Nevertheless, SMOTE was retained as part of the proposed preprocessing pipeline because it provides an explicit mechanism for addressing class imbalance during training.

The final configuration, Condition D, introduced PCA after SMOTE and standardisation and produced a noticeable reduction in performance. Accuracy decreased to 97.45%, recall to 92.33%, F1-score to 95.60%, and ROC-AUC to 99.10%. This further supports the focused PCA comparison presented in Section 3.2.1, where the effect of dimensionality reduction on Random Forest performance is examined separately.

3.3 Label Encoding Sensitivity Analysis

A sensitivity analysis was conducted to determine whether the choice of categorical encoding affected Random Forest performance. Two encoding strategies, label encoding and one-hot encoding, were evaluated using the same dataset partitioning, class-balancing procedure, model configuration, and evaluation procedure. This allowed the effect of encoding to be examined independently of other experimental factors.

The results of the two encoding configurations are presented in Table 6. The comparison includes accuracy, precision, recall, F1-score, and ROC-AUC to determine whether either encoding strategy produced a measurable performance difference.

Table 6. Sensitivity analysis of categorical encoding

Configuration	Accuracy	Precision	Recall	F1-score	ROC-AUC
Label encoding	99.40%	100.00%	98.00%	98.99%	99.91%
One-hot encoding	97.40%	100.00%	98.00%	98.99%	99.91%

As shown in Table 6, both encoding strategies produced identical performance across all reported metrics. The absence of a measurable difference indicates that the encoding strategy did not materially affect Random Forest performance under the experimental conditions. Label encoding was therefore retained in the main preprocessing workflow because it provides a more compact representation than one-hot encoding while producing equivalent classification performance in this experiment. This result also indicates that the observed performance of the framework was not dependent on the use of a particular categorical encoding strategy.

3.4 Comparative Evaluation and External Dataset Validation

The final evaluation extended beyond the simulated dataset to provide both contextual comparison with previous IoT intrusion-detection studies and external validation using CICIoT2023. The proposed framework was first evaluated on the frozen simulated dataset using the leakage-controlled pipeline described in Section 2.

For contextual comparison, selected published results are presented in Table 7. Because the studies use different datasets, attack distributions, feature representations, classifiers, and evaluation protocols, the comparison is intended for contextual positioning rather than direct benchmarking.

Table 7. Comparative accuracy of selected IoT intrusion-detection studies and the proposed framework

Study	Dataset	Method	Accuracy
Awajan(2023)	Simulation-based IoT	Deep neural network	93.74%
Akif *et al.*(2025)	Simulation-based IoT	Random Forest	96.20%
Proposed framework	Simulation-based IoT	Proposed Pipeline with Random Forest	97.45%
CICIoT2023 Random split	CICIoT2023	Same modelling framework	99.84%
CICIoT2023 grouped split	CICIoT2023	Source file grouped evaluation	98.51%

As shown in Table 7, the proposed framework achieved higher accuracy on the simulated dataset than the selected studies, exceeding the results reported by Awajan (2023) and Akif et al. (2025). However, these differences should be interpreted cautiously because the studies used different datasets, attack distributions, feature representations, and evaluation protocols.

The framework was subsequently evaluated on CICIoT2023 using the 37 common modelling features identified in Section 2. The conventional 80:20 evaluation achieved 99.84% accuracy, while the stricter source-file-grouped evaluation achieved 98.51%, with zero source-file overlap between the training and testing groups. The relatively small reduction under the grouped condition indicates that the high performance was largely maintained when stricter source separation was imposed.

Overall, these experiments provide complementary evidence for the framework. The simulated-dataset evaluation demonstrates its performance under controlled and reproducible workplace IoT conditions, while CICIoT2023 provides external validation using an independent public dataset. The results also highlight the importance of dataset characteristics and partitioning strategies when interpreting IoT intrusion-detection performance.

4. Conclusion

This study presented a reproducible machine learning-based IoT intrusion detection framework for smart workplace environments, integrating simulation-based dataset generation, leakage-controlled preprocessing, and Random Forest classification. The Python-based simulation represented heterogeneous IoT devices and defined the generation parameters explicitly for reproducibility. The preprocessing workflow incorporated training-only SMOTE, feature standardisation, PCA, and Random Forest classification while preventing the held-out test data from influencing preprocessing or model development.

The complete pipeline achieved 97.45% accuracy, 99.11% precision, 92.33% recall, 95.60% F1-score, and 99.10% ROC-AUC on the simulated dataset. Ablation analysis showed that SMOTE and standardisation produced little change in the main classification measures, whereas PCA reduced performance. This demonstrates that dimensionality reduction should be evaluated empirically rather than assumed to benefit Random Forest-based IDS. Evaluation on CICIoT2023 also produced high performance, with 99.84% accuracy under the conventional evaluation and approximately 98.51% under source-file grouping, highlighting the influence of dataset and evaluation conditions on IDS performance.

The main limitation is the simulation-based nature of the primary dataset, which cannot fully capture the complexity of real-world IoT traffic. Future work should therefore evaluate the framework on larger and more diverse real-world datasets, additional attack scenarios, and cross-environment settings. Further investigation of feature-selection and dimensionality-reduction techniques may also identify conditions under which preprocessing can improve tree-based IoT intrusion detection.

References

Adeyemi, D. S. (2024). Effectiveness of machine learning models in intrusion detection systems: A systematic review. *Communication in Physical Sciences, 11*(4), 1060–1088.

Akif, M., Ahmed, M., Abdullah, A., & Alshammari, A. (2025). Intrusion detection system for IoT based on modified random forest algorithm. *International Journal of Computer Science and Mathematics, 6*(2).

Akintade, S., Roy, K., & Kim, S. (2025). Hybrid deep machine learning feature selection for high-dimensional cybersecurity data. *IEEE Access*.

Alfardus, A., & Rawat, D. B. (2024). Binary logistic regression-based intrusion detection system for CAN bus security. In *Proceedings of the 2024 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)* (pp. 141–147). IEEE.

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for Internet of Things (IoT) security. *IEEE Communications Surveys & Tutorials, 22*(3), 1646–1685.

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks, 54*(15), 2787–2805.

Awajan, A. (2023). A novel deep learning-based intrusion detection system for IoT networks. *Computers, 12*(2), 34.

Das, R., Deka, V., & Taye, G. (2025). IoT botnet detection: A comparative performance analysis of various ML models on IoT-23 dataset. *Indian Journal of Science and Technology, 18*(41), 3300–3318.

Devine, M., Ardakani, S. P., Al-Khafajiy, M., & James, Y. (2025). Federated machine learning to enable intrusion detection systems in IoT networks. *Electronics, 14*(6), 1176.

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications, 50*, 102419.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security, 28*(1–2), 18–28.

Hajiheidari, S., Wakil, K., Badri, M., & Navimipour, N. J. (2019). Intrusion detection systems in the Internet of Things: A comprehensive investigation. *Computer Networks, 160*, 165–191.

Hamidou, S. T., & Mehdi, A. (2025). Enhancing IDS performance through a comparative analysis of random forest, XGBoost, and deep neural networks. *Machine Learning with Applications*, 100738.

Jablaoui, R., & Liouane, N. (2025). Network security-based combined CNN-RNN models for IoT intrusion detection system. *Peer-to-Peer Networking and Applications, 18*(3), 129.

Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*.

Kalaria, R., Kayes, A. S. M., Rahayu, W., Pardede, E., & Salehi, A. (2024). IoTPredictor: A security framework for predicting IoT device behaviours and detecting malicious devices against cyber attacks. *Computers & Security, 146*, 104037.

Khaleel, Y. L., Habeeb, M. A., Albahri, A. S., Al-Quraishi, T., Albahri, O. S., & Alamoodi, A. H. (2024). Network and cybersecurity applications of defence in adversarial attacks: A state-of-the-art using machine learning and deep learning methods. *Journal of Intelligent Systems, 33*(1), 20240153.

Kikissagbe, B. R., & Adda, M. (2024). Machine learning-based intrusion detection methods in IoT systems: A comprehensive review. *Electronics, 13*(18), 3601.

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of a realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems, 100*, 779–796.

Ma, H., Zhang, W., Zhang, D., & Chen, B. (2025). An IoT intrusion detection framework based on feature selection and large language model fine-tuning. *Scientific Reports, 15*(1), 21158.

Mosenia, A., & Jha, N. K. (2017). A comprehensive study of security of Internet of Things. *IEEE Transactions on Emerging Topics in Computing, 5*(4), 586–602.

Mustafa, Z., Amin, R., Aldabbas, H., & Ahmed, N. (2024). Intrusion detection systems for software-defined networks: A comprehensive study on machine learning-based techniques. *Cluster Computing, 27*(7), 9635–9661.

Shakir, D. A. Q., Nafea, A. A., Jassim, S. A. G., & Kareem, A. K. (2026). Attention-based CNN-BiLSTM framework for intrusion detection in RPL-based IoT networks. In *Proceedings of the 2026 International Conference on Smart Multidomain Integrated Learning Environments (ICSMILE)* (pp. 1–6). IEEE.

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence, 2*(1), 41–50.

Singh, J., Pasquier, T., Bacon, J., Ko, H., & Eyers, D. (2021). Twenty security considerations for cloud-supported Internet of Things. *IEEE Internet of Things Journal, 8*(4), 2157–2181.

Souza, C. H., & Arima, C. H. (2024). Avaliação de algoritmos de machine learning para detecção de malware IoT no dataset IoT-23. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)* (pp. 767–772). SBC.

Sudyana, D., Verkerken, M., D'Hooge, L., Lin, Y.-D., Hwang, R.-H., Lai, Y.-C., … & De Turck, F. (2024). Quality analysis in IDS dataset: Impact on model generalisation. In *Proceedings of the 2024 IEEE Conference on Communications and Network Security (CNS)* (pp. 1–6). IEEE.

Yaras, S., & Dener, M. (2024). IoT-based intrusion detection system using new hybrid deep learning algorithm. *Electronics, 13*(6), 1053.