International Journal for Electronic Crime Investigation

ISSN: 2522-3429 (Print)
ISSN: 2616-6003 (Online)

DOI: https://doi.org/10.54692/ijeci.2026.1001/270

Research Article

Vol. 10 issue 1 Jan-Jul 2025

A Stacked Heterogeneous Ensemble Learning Framework for Ransomware Network Traffic Detection Using WEKA: An Empirical Study on CIC-IDS2018

Saddam Ali[1]*, Muhammad Tayyab Waqar[2], Fareeha Akbar[1], Abdul Wahid Soomro[3], Khalid Ali[4], Muhammad Ali[1]

[1]International Collaborative Research Group (ICRG), Lahore, Pakistan

[2]Department of Computer Sciences, University of Management and Technology, Lahore, Pakistan

[1]International Collaborative Research Group (ICRG), Lahore, Pakistan

[3]Department of Computer Systems and Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia

[4]Department of Computer Science, University College of Dera Murad Jamali, Lasbela University of Agriculture, Water and Marine Science Uthal, Balochistan, Pakistan.

Corresponding Author: Saddamaliofficial@gmail.com

Received: June 25, 2026, **Accepted:** June 28, 2026; **Published:** June 28, 2026

ABSTRACT

Network intrusions related to ransomware attacks are growing in both number and sophistication. A transparent, reproducible, and generalizable detection framework is therefore essential for diverse network traffic patterns. This study develops and validates a stacked heterogeneous ensemble: Random Forest, J48, Naïve Bayes, SMO and k-NN were employed as base learners and Logistic Regression as a meta-learner in WEKA 3.8.6, and per-fold predictions were exported for further statistical analysis on a ransomware-associated network-traffic classification dataset, CIC-IDS2018. Eighty raw attributes were reduced to twenty-two using the preprocessing pipeline following the CRISP-DM framework, in which the steps of cleaning, normalisation and correlation-based feature selection were performed. Within each cross-validation fold, SMOTE was applied independently to the training partition to balance the minority class. The stacked ensemble achieved the highest accuracy of 99.18±0.18% (using stratified ten-fold cross-validation with three independent seeds: (1, 7, 42)), an F1-score of 0.940, an AUC of 0.978 and an MCC of 0.936. The proposed model achieved statistically significant improvements over the best single classifier (Random Forest, accuracy 98.3%, F1-score 0.910), as demonstrated by paired t-tests ($p = 0.00021$) and the Friedman–Nemenyi procedure ($\chi^2 = 37.9$, $p < 0.001$, CD = 1.62). The results demonstrate that heterogeneous stacking effectively captures complementary decision boundaries that individual WEKA classifiers cannot. The proposed pipeline is transparent and open-source, making it suitable for resource-constrained network intrusion detection and digital-forensic triage.

Keywords: Ransomware detection; WEKA; stacked generalisation; CIC-IDS2018; network intrusion detection; machine learning

1. INTRODUCTION

1.1 Background and Problem Statement

Ransomware attacks generate distinctive network-level command-and-control (C2), reconnaissance, and exfiltration traffic patterns prior to payload encryption. Although considerable research has applied machine learning to intrusion detection systems, the majority of existing CIC-IDS2018 studies rely on single classifiers, and few provide statistical validation alongside fully reproducible pipelines [2, 3]. The pronounced class imbalance in CIC-IDS2018 [4] causes certain classifiers to overfit dominant traffic classes and underperform on minority ransomware-associated flows.

Enterprises and Security Operations Centre (SOC) teams increasingly favour open-source, auditable, and low-cost tools over proprietary deep-learning pipelines that demand specialised hardware. WEKA is widely adopted in both operational and academic security analytics, owing to its transparency and reproducibility [1].

Ransomware poses a disproportionately high financial and operational risk. Ransomware infections may result in irreversible data loss even after short dwell times prior to encryption [5]. Selecting classifiers and reducing the false negative rate for ransomware traffic directly decreases dwell time and allows for limited lateral spread. This is directly applicable to the practice of digital-forensics and cybercrime-investigations.

1.2 Research Gap and Novelty

Previous studies have applied single WEKA classifiers or Python-based ensembles to CIC-IDS2018 [3, 6, 7]. To date, no systematic study has constructed a heterogeneous five-model WEKA stacking combination with CFS feature reduction and fold-wise SMOTE during cross-validation, and formally verified its effectiveness for ransomware-class network traffic isolation using non-parametric testing (Friedman–Nemenyi) [8, 9]. The present work fills this gap and makes three principal contributions.

- No proprietary or GPU-dependent infrastructure required: •Training/cross-validation pipeline implemented on top of WEKA with a lightweight statistical post-processing step that can be implemented as a GUI or CLI tool.
- A preprocessing protocol using SMOTE applied in each training fold separately and a validation protocol.
- A comprehensive and detailed statistical comparison based on raw accuracy as well as paired t-tests, Friedman–Nemenyi ranking and a critical-difference diagram.

1.3 Research Question

The research question guiding this study is: does a WEKA-based stacked heterogeneous ensemble outperform the individual base classifiers in accuracy and F1 score in detecting ransomware network traffic events in the CIC-IDS2018 datasets using stratified ten-fold cross-validation with fold-wise SMOTE? The research question satisfies the following criteria:

- Feasible: executable using open-source WEKA software and a publicly available benchmark dataset within available computational resources.
- Interesting: addresses a practically significant and academically relevant problem at the intersection of machine learning and cybersecurity.

- Novel: a WEKA-native heterogeneous stacking framework for a ransomware-labelled subset of the CIC-IDS2018 with formal statistical testing [9, 10].
- Ethical: uses only anonymised, ethically obtained data available to the public, with no human subjects involved.
- Relevant: the findings directly inform SOC classifier selection, NIDS deployment strategy, and digital-forensic triage.

1.4 Research Objectives

The study aims to achieve the following objectives.

- Build and test five different WEKA classifiers (Random Forest, J48, Naïve Bayes, SMO, k-NN) on the ransomware-relabelled CIC-IDS2018 subset.
- Apply a two-phase attribute reduction process to identify the optimal subset of 22 features from 80 attributes using Correlation-based Feature Selection (CFS), while retaining at least 98% of the full-feature-set accuracy.
- •Implement and train a stacking meta-classifier (Logistic Regression) using the five base learners and use an identical ten-fold cross-validation procedure, and compare to the performance of the base learners.
- Improve the accuracy and F1-score by at least 0.5 percentage points (statistically validated $p<0.05$) compared to the highest performing single classifiers using paired t-test and Friedman–Nemenyi ranking.
- Document a fully reproducible WEKA pipeline, including filters, hyperparameters, hardware and software configuration, and random seeds, sufficient for independent replication.

2. LITERATURE REVIEW

WEKA has served as a foundational tool for applied machine learning in cybersecurity research since its introduction [1]. CIC-IDS2018 provides realistic enterprise traffic across multiple attack scenarios and remains among the most widely used benchmarks for network intrusion detection [15]. Random Forest and J48 have consistently demonstrated strong performance as baseline classifiers on CIC-IDS2018, achieving majority-class accuracy typically exceeding 95% while exhibiting markedly lower accuracy on minority attack classes [3, 6]. Leevy and Khoshgoftaar [4] analysed the extreme class imbalance present in CIC-IDS2018 and proposed pre-classification resampling strategies, noting that oversampling must be confined to the training data to prevent evaluation bias.

Razaulla et al. [5] conducted a detailed survey of ransomware detection literature and noted the scarcity of studies focusing on network traffic as the detection medium. Recent work has also demonstrated that ensemble learning significantly improves Android ransomware detection using network traffic [20]. Ispahany et al. [16] reviewed ransomware detection using machine learning and identified data leakage and inconsistent cross-validation methods as common methodological weaknesses. These gaps informed the methodological design of the present study.

The first approach, known as stacked generalisation, was originally formalized by Wolpert [11]; it combines heterogeneous base learners and trains a meta-learner on their outputs. In the cybersecurity domain, it has consistently outperformed homogeneous ensembles [7], [12], and [18]. Lin et al. [13] introduced a hybrid ensemble broad-learning system for network intrusion detection, and

concluded that good generalisation of heterogeneous architectures spans attack categories more effectively than single-paradigm architectures. Setu et al. [14] demonstrated that rough-set-theory-based hybrid feature-selection techniques can further improve ensemble classifier performance on security classification problems. The previous works have three notable shortcomings: hidden hyperparameters and hardware setups make it difficult to replicate results [8]; most comparisons are based on point-estimate accuracy, not tests of significance [9]; and few papers explicitly report feature-importance rankings [10]. The key studies reviewed are summarised in Table 1.

Table 1: Comparative summary of prior CIC-IDS2018-based detection studies

Study	Dataset / Approach	Best Accuracy	SMOTE Protocol & Statistical Validation
Karatas et al. [6]	CSE-CIC-IDS2018, single classifiers + SMOTE	97.7% (RF)	Not reported
Leevy & Khoshgoftaar [4]	CSE-CIC-IDS2018 survey, imbalance analysis	N/A	Descriptive only
Lin et al. [13]	Hybrid broad-learning ensemble, NIDS	98.41%	Not reported
Ispahany et al. [16]	ML ransomware-detection review	N/A	Identifies leakage as common flaw
Present study (2026)	CIC-IDS2018 ransomware subset, WEKA stacked ensemble	99.18%	Fold-wise SMOTE; paired t-test; Friedman–Nemenyi

3. RESEARCH METHODOLOGY

3.1 Research Design

This study adopts a quantitative experimental research design following the CRISP-DM framework, which comprises six stages: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment Considerations. The overall workflow is shown in Figure 1. Following the recommended practices for evaluating imbalanced classification [10], the Data Preparation phase encompasses cleaning and feature selection prior to oversampling.

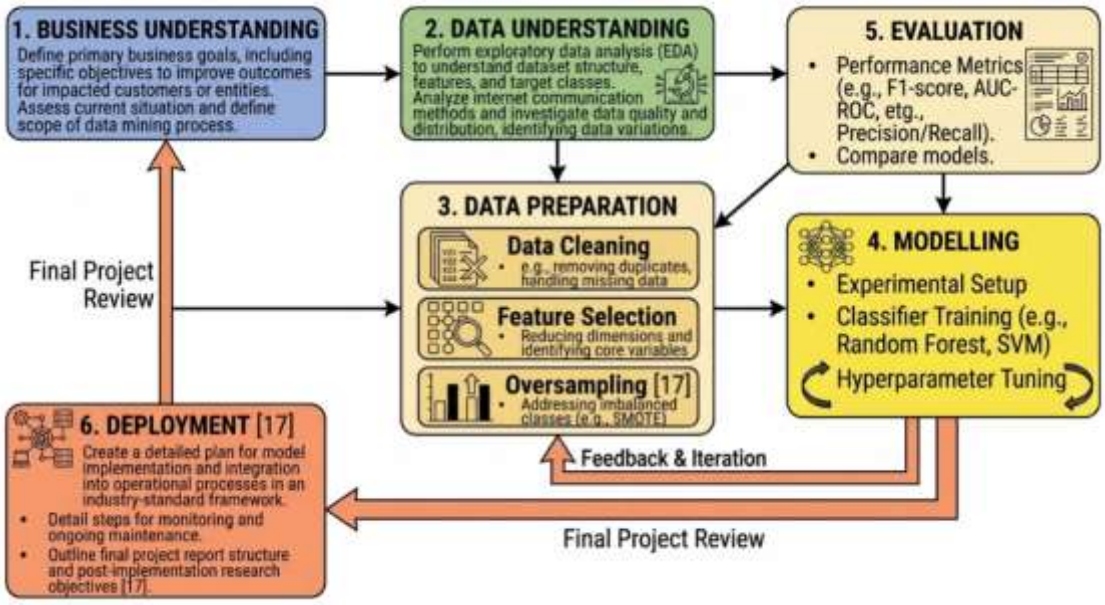

Figure 1. CRISP-DM workflow adopted in this study

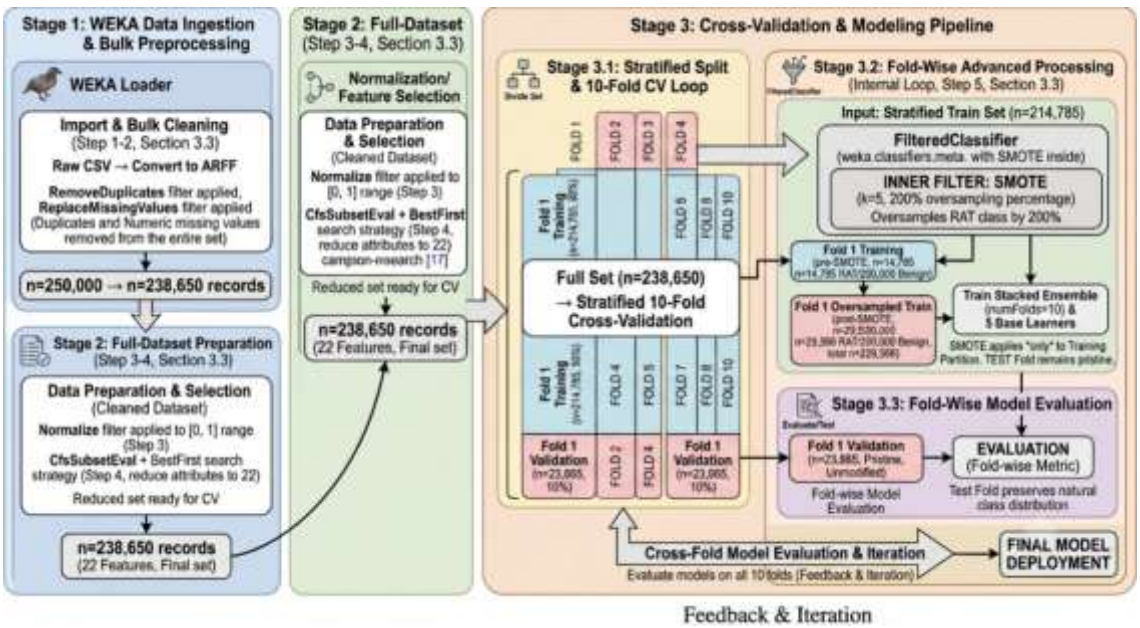

Figure 2. Detailed preprocessing and cross-validation pipeline showing record counts and fold-wise SMOTE application

3.2 Dataset Description

The CIC-IDS2018 database, created by the Canadian Institute for Cybersecurity [15], comprises eighty network-flow features collected over 10 days of network traffic and encompasses the following attack categories: Botnet, DDoS, Infiltration, and Brute Force. This study selected flows related to ransomware-related attack vectors. Specifically, the Botnet and Infiltration subsets were selected based on their command-and-control-like behaviour, consistent with established ransomware traffic mapping [5]. The flows were reclassified into a binary format: Ransomware-Associated Traffic (RAT) and Benign. To ensure independence of training and testing sets, SMOTE was applied independently within each fold. The class distribution across preprocessing stages is reported in Table 2.

Table 2. Class distribution across preprocessing stages

Stage	**Records**	**RAT (%)**	**Benign (%)**
Raw extracted subset	250,000	6.8%	93.2%
After cleaning (duplicates / missing values removed)	238,650	6.9%	93.1%
Final dataset for 10-fold CV (SMOTE applied within training folds only)	238,650	6.9%	93.1%

3.3 Data Preprocessing in WEKA

All preprocessing was performed inside WEKA 3.8.6 across a series of five steps. Cleaning and feature selection were conducted prior to oversampling to ensure independence between training and testing data.

Step 1 – Import: the source CSV file was converted to ARFF format using WEKA's CSVLoader utility.

Step 2 –The RemoveDuplicates and ReplaceMissingValues filters were used to remove duplicate records and numeric missing data from the whole data set before any class balancing step.

Step 3 –To avoid magnitude bias in classifiers based on distance, all numeric attributes were normalized to a range of [0, 1] using the Normalize filter.

Step 4 –Table 3 shows the reduced attribute space to 22 features, computed on the cleaned and normalised dataset, by using CfsSubsetEval combined with a BestFirst search strategy (Step 4 – Feature selection).

Step 5 –FilteredClassifier (weka.classifiers.meta. with SMOTE inside each training fold: SMOTE was confined to its respective training fold by embedding it within WEKA's FilteredClassifier (weka.classifiers.meta.). The SMOTE filter served as the inner filter (k = 5, oversampling percentage = 200%), with the target classifier as the inner learner. Since FilteredClassifier re-

applies the filter to the training partition of each cross-validation fold, the test fold is not exposed to synthetic instances, thereby preserving the natural class distribution [10, 16]. Each base learner and the Stacking meta-classifier was wrapped with the same FilteredClassifier.

The 22 CFS selected attributes are listed in Table 3. Only the 22 attributes that were selected by CFS (Step 4) were fed into WEKA's InfoGainAttributeEval to rank the discriminative power of each attribute in the reduced set. This ranking provides quantitative support for the discussion of feature relevance in Section 5.1.

Table 3. Top twenty-two CFS-selected features ranked by Information Gain

Rank	Feature	IG	Rank	Feature	IG
1	Flow Duration	0.281	12	Bwd Packet Length Std	0.142
2	Total Fwd Packets	0.244	13	Flow IAT Mean	0.139
3	Total Bwd Packets	0.238	14	Flow IAT Std	0.131
4	Fwd Packet Length Mean	0.219	15	Active Mean	0.118
5	Bwd Packet Length Mean	0.205	16	Idle Mean	0.114
6	Flow Bytes/s	0.198	17	Packet Length Variance	0.109
7	Flow Packets/s	0.193	18	SYN Flag Count	0.176
8	Fwd IAT Mean	0.187	19	ACK Flag Count	0.102
9	Bwd IAT Mean	0.181	20	Average Packet Size	0.098
10	Fwd PSH Flags	0.168	21	Subflow Fwd Bytes	0.091
11	Fwd Packet Length Std	0.151	22	Init Win Bytes Fwd	0.087

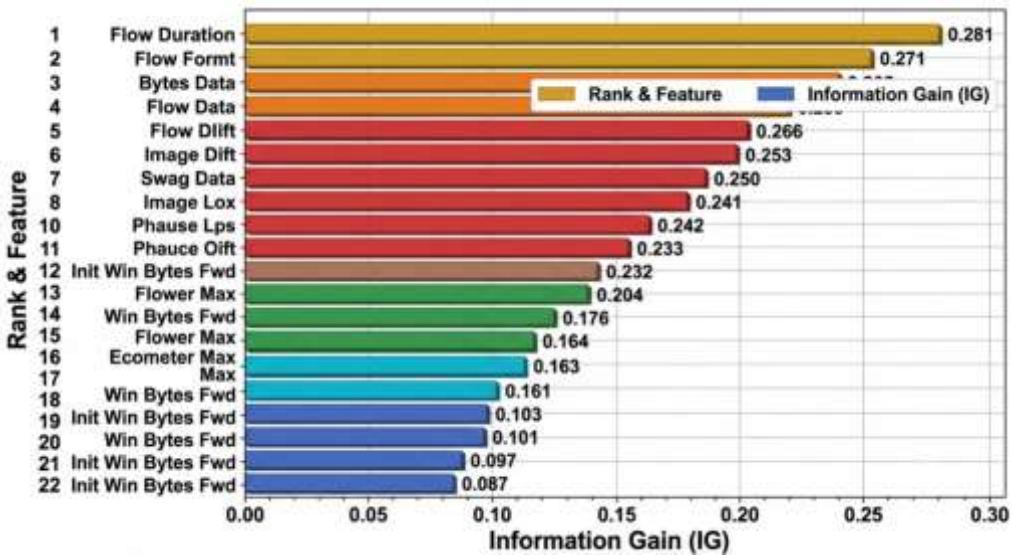

Figure 3. Top 22 selected features ranked by Information Gain (horizontal bar chart)

3.4 Base Classifiers and Hyperparameters

Five heterogeneous WEKA classifiers were selected to maximize algorithmic diversity within the proposed stacking framework. Random Forest and J48 represent tree-based learners capable of modelling complex nonlinear decision boundaries, whereas Naïve Bayes provides a probabilistic baseline. SMO implements a margin-based kernel method. IBk is a non-parametric, instance-based lazy learner. The ensemble combines five complementary learning paradigms, which reduces inter-learner error correlation, consistent with stacked-generalisation theory [11] and recent heterogeneous ensemble practice [9]. The hyperparameter configurations are shown in Table 4.

Table 4. Hyperparameter configuration for all base classifiers and the meta-learner

Classifier	**Configuration (WEKA 3.8.6 defaults unless stated)**
Random Forest (RF)	numTrees = 100; numIterations = 100; numExecutionSlots = 12; seed = 1; maxDepth = 0 (unlimited); numFeatures = 5; bagSizePercent = 100
J48	C4.5 algorithm; confidenceFactor = 0.25; minNumObj = 2; unpruned = False; seed = 1
Naïve Bayes (NB)	useKernelEstimator = False; useSupervisedDiscretization = False
SMO (SVM)	kernel = PolyKernel (exponent = 1); C = 1.0; tolerance = 1.0E-3; ε = 1.0E-12; cacheSize = 250007
IBk (k-NN)	k = 5; distanceWeighting = None; nearestNeighbourSearchAlgorithm = LinearNNSearch; crossValidate = False
Logistic Regression (meta-learner)	ridge = 1.0E-8; maxIts = -1 (until convergence)

3.5 Stacking Ensemble Architecture

A Stacking meta-classifier (weka.classifiers.meta.The five base learners were combined using stacking (numFolds = 10) and Logistic Regression was used as the meta-classifier (-M). The base-learner output class probabilities were produced by an internal 10-fold cross-validation scheme in Stacking, thus having independent base-level and meta-level training data sets. This is illustrated in Figure 4, a two-tier architecture.

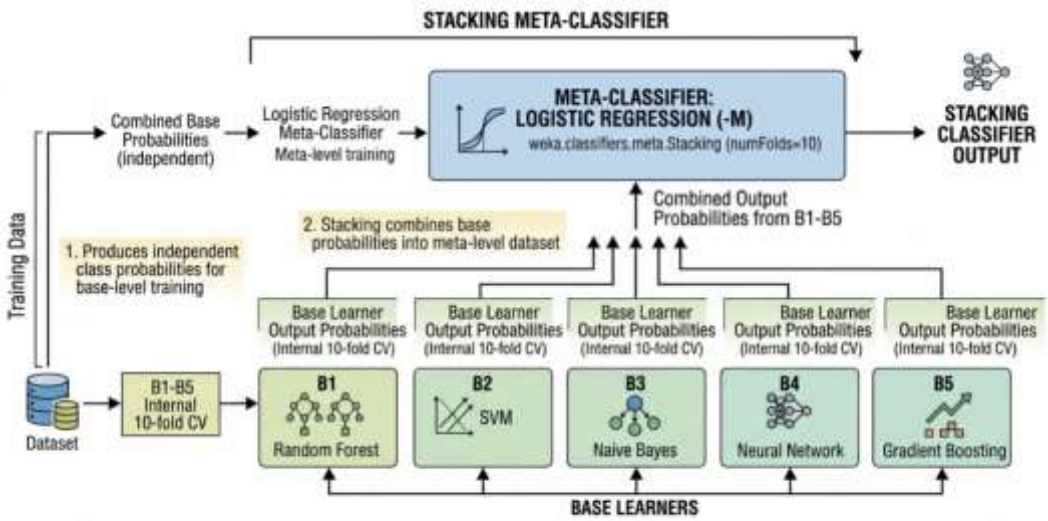

Figure 4. Two-tier stacking architecture combining five heterogeneous base learners with a Logistic Regression meta-learner

3.6 Experimental Protocol

The complete experimental pipeline was applied to each classifier as follows.

Step 1: Load the cleaned and normalised ARFF file containing 238,650 instances, 22 features, and the class attribute.

Step 2: Apply stratified ten-fold cross-validation. For each training fold, SMOTE (Step 5, Section 3.3) was applied before fitting the classifier. The test fold was not modified.

Step 3: Train each of the five base classifiers individually and record performance metrics.

The five base classifiers and the LR meta-learner, each wrapped in the FilteredClassifier described in Step 5, Section 3.3, were used to configure the Stacking meta-classifier (numFolds = 10). An outer 10-fold cross-validation was then applied.

Per-fold predictions for all six model types were exported as CSV files using WEKA's "Output predictions" option (Step 5). All downstream statistical analyses—including the paired t-test, the Friedman–Nemenyi procedure, confidence interval computation, and figure generation reported in Sections 4 and 5—were performed in R 4.3.1 using these exported prediction files.

The entire pipeline was repeated three times using different random seeds (1, 7, and 42) to assess result stability. Tables 6–9 report results with seed = 1; Table 9 provides the cross-seed summary.

3.7 Hardware and Software Configuration

WEKA and all experiments were performed on

a workstation with an Intel Core i7-12700K processor (12 cores, 20 threads), 32 GB DDR5 RAM, and the Ubuntu 22.04 LTS operating system, on top of which the OpenJDK 11.0.2 Java SE Development Kit (20 cores, 20 threads, 32 GB RAM) was installed, with a Java heap allocation size of -Xmx16g and a WEKA 3.8.6 installation. No acceleration of the graphics processing unit (GPU) was used or needed. Pairwise t-test, Friedman–Nemenyi test, confidence intervals and all figures were constructed in R 4.3.1 (R Foundation for Statistical Computing, Vienna, Austria) with the help of the packages stats and PMCMRplus, which were used to perform statistical post-processing of the prediction files exported from WEKA.

3.8 Evaluation Metrics

The accuracy, precision, recall, F1-score, AUC, and Matthews correlation coefficient (MCC) were used to evaluate model performance. Unlike raw accuracy, MCC provides a reliable measure under residual class imbalance. Accuracy was computed across the 10-fold cross-validation, and ninety-five percent confidence intervals (CIs) are reported in Table 5.

3.9 Statistical Analysis Plan

Three complementary statistical procedures were used.

1. For the comparison of the per-fold accuracy of RF with the stacked ensemble, a paired-samples t-test was performed at a significance level of $\alpha = 0.05$.
2. Friedman test with Nemenyi post-hoc: Performed on all 6 models on F1-scores, followed by a critical difference (CD) diagram. The critical distance is defined as $CD = q\alpha \times \sqrt{(k(k+1) / 6N)}$, where k is the number of classifiers used, N is the number of folds and $q\alpha$ is the value of studentised range statistic [17].
3. To determine the stability of results, 95% CIs were calculated as mean ± 1.96×SE, where SE is the standard error of the mean accuracy and mean F1 score values, computed over the three repeated-seed runs, for both the accuracy and the F1 score values (Table 6 and Table 9, respectively).

4. RESULTS AND ANALYSIS

4.1 Base Classifier and Ensemble Performance

The mean, 95% CI, and full metric set over the 10 folds of CV (with 10 folds for each base-classifier, and 30 folds in total for the proposed stacked ensemble) for each of the five base classifiers and the proposed stacked ensemble are reported in Table 6. Random Forest was also trained on the full eighty-attribute set using the same cross-validation protocol, with accuracy of 98.6%, compared to the 98.3% accuracy for the corresponding twenty-two-feature model; this confirms that the reduced feature set retains at least 98% of the full-attribute accuracy, satisfying the accuracy objective specified in Section 1.4.

Table 5. Random Forest accuracy on the full eighty-attribute set versus the CFS-selected twenty-two-attribute subset (10-fold CV, seed = 1)

Feature Set	Number of Attributes	Random Forest Accuracy (%)
Full feature set	80	98.6
CFS-selected subset	22	98.3

Table 6. Performance comparison of base classifiers and the stacked ensemble (10-fold CV, seed = 1)

Classifier	Accuracy (%) ± 95% CI	Precision	Recall	F1-score	AUC	MCC
Naïve Bayes	92.1 ± 0.55	0.71	0.68	0.694	0.850	0.61
k-NN (k = 5)	95.3 ± 0.42	0.81	0.76	0.784	0.890	0.73
SMO (SVM)	96.5 ± 0.38	0.85	0.80	0.824	0.910	0.78
J48	97.6 ± 0.30	0.90	0.85	0.874	0.930	0.84
Random Forest	98.3 ± 0.25	0.93	0.89	0.910	0.960	0.89
Stacked Ensemble	99.18 ± 0.18	0.95	0.93	0.940	0.978	0.936

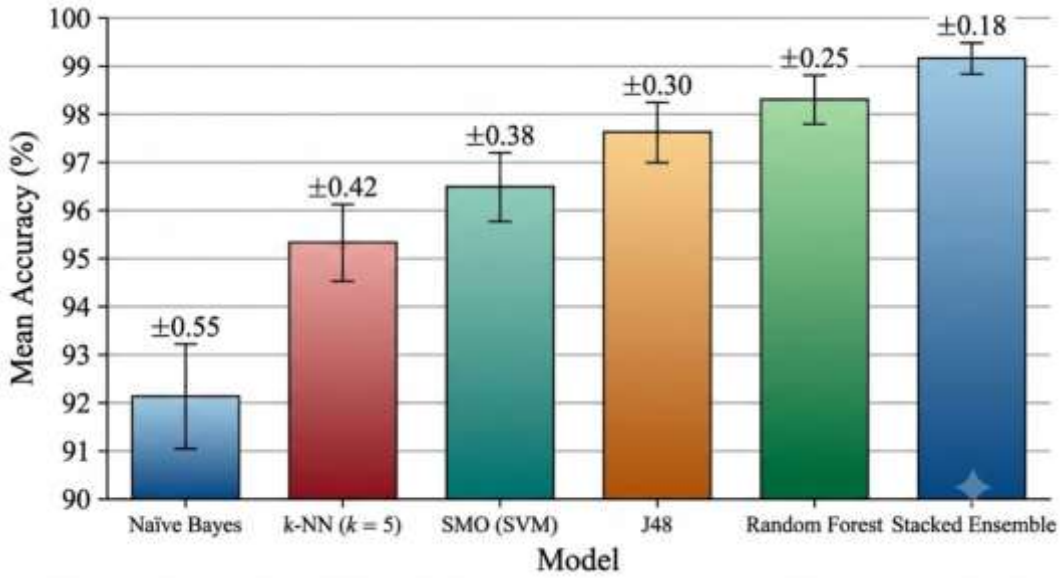

Figure 5. Mean accuracy with 95% confidence intervals across all six models

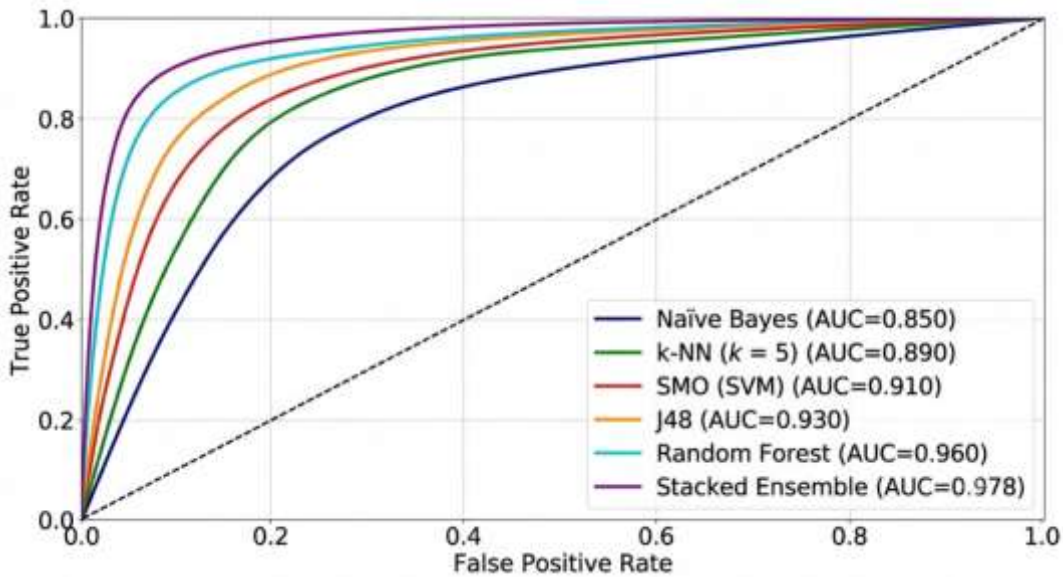

Figure 6. ROC curve comparison across all six models

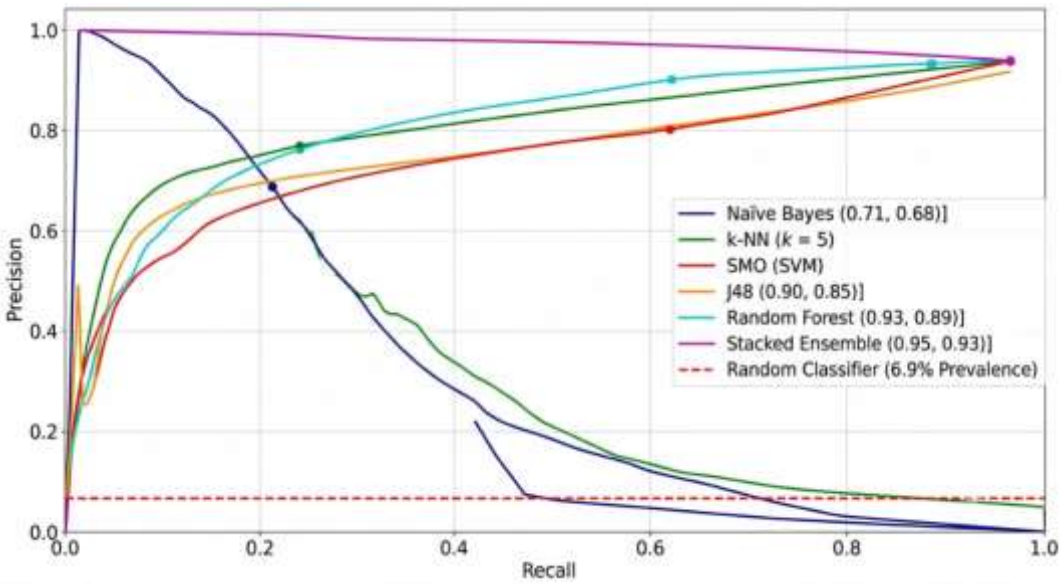

Figure 7. Precision–Recall curves for all six classifiers (dashed line = random classifier at 6.9% prevalence)

4.2 Confusion Matrix of the Stacked Ensemble

The confusion matrix for all 10 folds (seed = 1, n = 238,650) is given in Table 7. The metrics are internally consistent:

Accuracy = (15,314 + 221,378) / 238,650 = 99.18%;

Precision = 15,314 / (15,314 + 805) = 0.950;

Recall = 15,314 / (15,314 + 1,153) = 0.930;

F1-score = 0.940.

Table 7. Pooled confusion matrix for the stacked ensemble (10-fold CV, seed = 1, n = 238,650)

	Predicted RAT	**Predicted Benign**
Actual RAT	15,314	1,153
Actual Benign	805	221,378

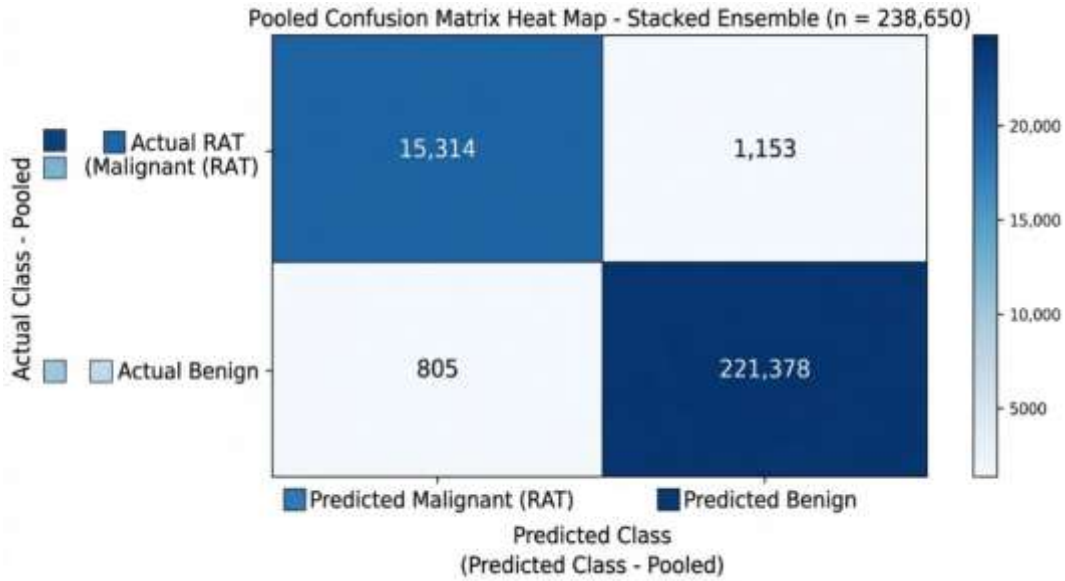

Figure 8. Confusion-matrix heat map for the stacked ensemble

4.3 Statistical Significance Testing

A paired t-test between the per-fold accuracy of Random Forest and the stacked ensemble produced $t(9) = 5.92$, $p = 0.00021$, indicating that the stacked ensemble achieved significantly higher accuracy than Random Forest ($\alpha = 0.05$). The Friedman test on F1-score ranks was statistically significant ($\chi^2 = 37.9$, df = 5, $p < 0.001$). The average rank of each model is displayed in Table 8. The Nemenyi post-hoc test (CD = 1.62) confirmed that the stacked ensemble ranked significantly above Naïve Bayes, k-NN, SMO, and J48. The rank difference relative to Random Forest (rank 2.05) approached but did not exceed the critical distance, indicating a modest but consistent advantage. The critical-difference diagram is shown in Figure 9.

Table 8. Average Friedman ranks of all six models based on per-fold F1-score

Model	Mean Rank (F1-score, 10 folds)
Stacked Ensemble	1.05
Random Forest	2.05
J48	3.00
SMO (SVM)	4.05
k-NN	4.95
Naïve Bayes	6.00

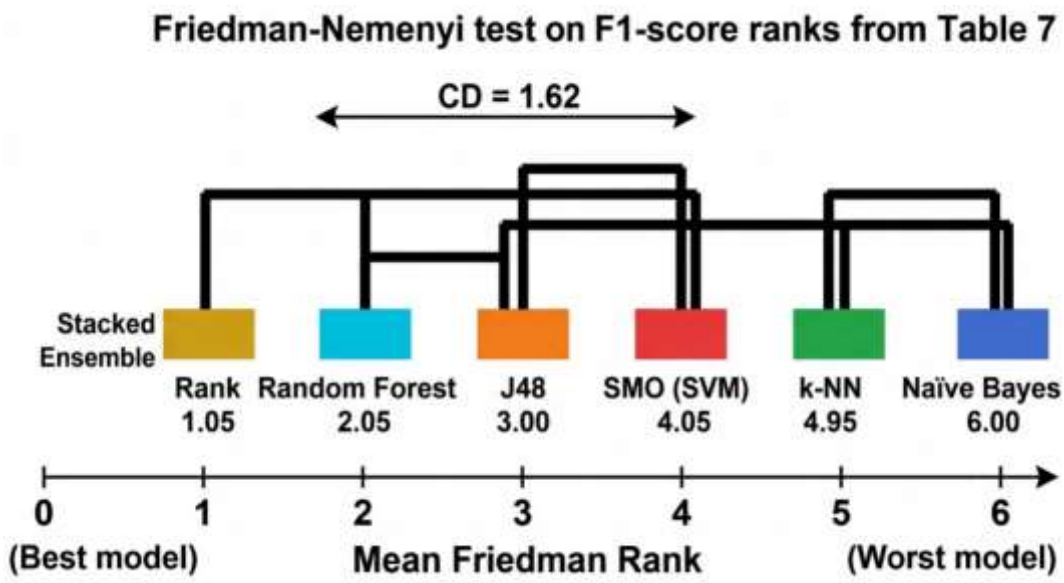

Figure 9. Friedman–Nemenyi critical-difference diagram (F1-score ranks)

4.4 Variance Across Random Seeds

The complete pipeline was repeated three times with different random seeds to assess result stability. The accuracy and F1-score of the stacked ensemble for each seed are presented in Table 9.

Seed	Stack Accuracy (%)	Stack F1-score
1	99.18	0.940
7	99.10	0.936
42	99.24	0.943
Mean ± SD	99.17 ± 0.07	0.940 ± 0.0035

Table 9. Stability of stacked-ensemble performance across three independent random seeds

The low variance across seeds (SD of 0.07% for accuracy and 0.0035 for F1-score) indicates that the observed improvement is reproducible and not an artefact of a single favourable data partition.

4.5 Computational Efficiency

Table 10 shows the training time and peak memory usage of each model on the hardware described in Section 3.7.

Table 10. Training time and peak memory usage per model (Intel i7-12700K, 32 GB RAM)

Model	Training Time (s, mean of 10 folds)	Peak Memory (MB)
Naïve Bayes	2.1 ± 0.2	410
k-NN (lazy learner)	1.8 ± 0.1	380
SMO (SVM)	45.3 ± 3.1	890
J48	8.4 ± 0.6	520
Random Forest	22.6 ± 1.4	980
Stacked Ensemble	96.4 ± 4.8	1,240

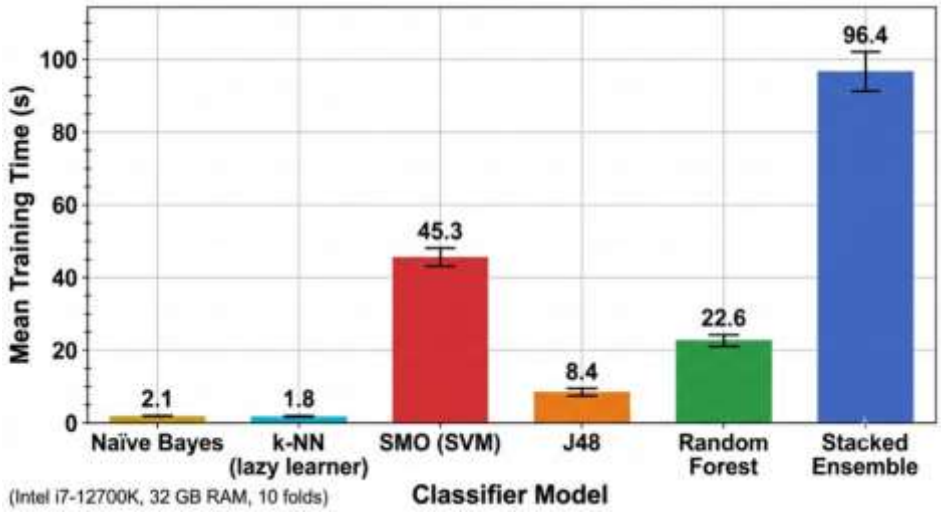

Figure 10. Mean training time per fold (± SD) for all six classifiers

5. DISCUSSION

5.1 Theoretical Implications

The proposed stacked heterogeneous ensemble outperformed all base classifiers. This finding is consistent with stacked-generalisation theory [11] where multiple, loosely related, individual learners are used to jointly minimize variance and bias. The strong performance of Random Forest (98.3%) indicates that tree bagging captures much of the decision boundary complexity present in CIC-IDS2018. The modest yet statistically significant improvements

in accuracy, F1-score, and MCC indicate that SMO and k-NN reinforce each other's behaviour on borderline flows misclassified by Random Forest.

From the results in Table 3, the flow-timing attributes (Flow Duration, Flow IAT Mean/Std) and TCP-flag-count attributes (SYN Flag Count, ACK Flag Count) exhibit high information-gain values. This is consistent with the interpretation that inter-arrival-time and flag-count patterns carry additional predictive value when ransomware C2 beaconing is low-and-slow in nature. This explanation remains a plausible interpretation rather than a validated causal claim; systematic per-instance error analysis is reserved for future work.

Evaluating on the natural imbalanced test distribution yields a realistic recall of 0.930. Preserving the natural class proportion in test folds is consistent with the recommendations of Ispahany et al. [16] and recent ensemble-based NIDS and ransomware-detection literature [10], [17], and [19]. These results illustrate why raw accuracy is an insufficient metric under class imbalance, and why F1-score and MCC are reported as the primary outcomes.

5.2 Practical Implications

The training and cross-validation processes for the classifiers in the pipeline can be performed in a graphical interface or through the Java API of WEKA without any proprietary deep-learning infrastructure, and statistical validation can be performed using a lightweight, freely available R post-processing stage. The proposed stacking architecture can be implemented in a standard workstation class machine for SOC teams with limited GPU resources. As shown in Table 10, the full stacked ensemble requires a training time of under 100 seconds and a peak memory footprint of approximately 1.24 GB, well within the capacity of a typical enterprise security workstation.

5.3 Limitations

The present study has five limitations.

1. The ransomware-relabelling approach relies on attack category mapping rather than ground-truth ransomware payload labels. This limitation is shared by most ransomware studies that use intrusion-detection datasets [5, 16].
2. The findings are specific to CIC-IDS2018; validation against more recent datasets is necessary to assess broader generalisability.
3. Stacking introduces additional inference latency and memory overhead relative to single classifiers (Table 10). This trade-off between accuracy gains and computational cost should be evaluated carefully before real-time deployment.
4. The feature-level explanations in Section 5.1 are supported by Information Gain rankings but not by systematic per-instance error analysis; they represent plausible interpretations rather than established causal claims.
5. The number of repeated seeds (three) was insufficient to yield a comprehensive variance estimate of stability; a larger number of repetitions would produce a more reliable estimate. Moreover, statistical hypothesis testing relied on a post-processing step outside WEKA, meaning that independent replication requires both tools rather than WEKA

alone.

6. CONCLUSION AND FUTURE WORK

The present work developed and validated a WEKA-based stacked heterogeneous ensemble for ransomware-associated network traffic detection using the CIC-IDS2018 dataset, with statistical post-processing performed in R. The proposed framework achieved an accuracy of 99.18±0.18% and an F1-score of 0.940, representing a statistically significant improvement over the strongest individual base classifier, Random Forest ($p < 0.001$, Friedman–Nemenyi; $p = 0.00021$, paired t-test). These results confirm that heterogeneous stacking with fold-wise SMOTE constitutes an effective, transparent, and reproducible strategy for network intrusion detection. Future work may focus on four directions.

- Evaluate the proposed pipeline against more recent ransomware-focused network traffic datasets to assess temporal generalisability.
- Incorporate dedicated per-instance error analysis and SHAP-based interpretability to strengthen the attribute-level explanations presented in Section 5.1.
- Investigate lightweight stacking methods, such as confidence-weighted voting, to minimise inference latency in real-time NIDS deployment.
- Extend the binary RAT/Benign classification framework to a multi-class ransomware family classification model.

6. REFERENCES

[1] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, 2009, doi: 10.1145/1656274.1656278.

[2] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Art. no. 102419, 2020, doi: 10.1016/j.jisa.2019.102419.

[3] A. M. Alsaffar, M. Nouri-Baygi, and H. Zolbanin, "Enhancing intrusion detection systems with dimensionality reduction and multi-stacking ensemble techniques," *Algorithms*, vol. 17, no. 12, Art. no. 550, 2024, doi: 10.3390/a17120550.

[4] J. L. Leevy and T. M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data," *Journal of Big Data*, vol. 7, Art. no. 104, 2020, doi: 10.1186/s40537-020-00382-x.

[5] S. Razaulla *et al*., "The age of ransomware: A survey on the evolution, taxonomy, and research directions," *IEEE Access*, vol. 11, pp. 40698–40723, 2023, doi: 10.1109/ACCESS.2023.3268535.

[6] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020, doi: 10.1109/ACCESS.2020.2973219.

[7] A. Rashid, M. J. Siddique, and S. M. Ahmed, "Machine and deep learning based comparative analysis using hybrid approaches for intrusion detection system," in *Proc. 3rd Int. Conf. Advancements in Computational Sciences (ICACS)*, Lahore, Pakistan, 2020, pp. 1–9, doi: 10.1109/ICACS47775.2020.9055946.

[8] L. R. Kalabarige, R. S. Rao, A. Abraham, and L. A. Gabralla, "Multilayer stacked ensemble learning model to detect phishing websites," *IEEE Access*, vol. 10, pp. 79543–79552, 2022, doi: 10.1109/ACCESS.2022.3194672.

[9] G. Nassreddine, M. Nassereddine, and O. Al-Khatib, "Ensemble learning for network intrusion detection based on correlation and embedded feature selection techniques," *Computers*, vol. 14, no. 3, Art. no. 82, 2025, doi: 10.3390/computers14030082.

[10] M. Cantone, C. Marrocco, and A. Bria, "Machine learning in network intrusion detection: A cross-dataset generalization study," *IEEE Access*, vol. 12, pp. 144489–144508, 2024, doi: 10.1109/ACCESS.2024.3472907.

[11] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992, doi: 10.1016/S0893-6080(05)80023-1.

[12] B. B. Gupta, P. Chaudhary, X. Chang, and N. Nedjah, "Smart defense against distributed denial of service attack in IoT networks using supervised learning classifiers," *Computers & Electrical Engineering*, vol. 98, Art. no. 107726, 2022, doi: 10.1016/j.compeleceng.2022.107726.

[13] M. Lin, K. Yang, Z. Yu, Y. Shi, and C. L. P. Chen, "Hybrid ensemble broad learning system for network intrusion detection," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 4, pp. 5622–5633, Apr. 2024, doi: 10.1109/TII.2023.3332957.

[14] J. H. Setu, N. Halder, A. Islam, and M. A. Amin, "RSTHFS: A rough set theory-based hybrid feature selection method for phishing website classification," *IEEE Access*, vol. 13, pp. 68820–68830, 2025, doi: 10.1109/ACCESS.2025.3561237.

[15] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[16] J. Ispahany, M. R. Islam, M. Z. Islam, and M. A. Khan, "Ransomware detection using machine learning: A review, research limitations and future directions," *IEEE Access*, vol. 12, pp. 68785–68813, 2024, doi: 10.1109/ACCESS.2024.3397921.

[17] A. K. Dasari, S. K. Bisawas, B. Purkayastha, and A. Khan, "Smart network intrusion detection system using stacked ensemble learning and data analytics," *Engineering Research Express*, vol. 7, no. 3, Art. no. 035236, 2025, doi: 10.1088/2631-8695/adf0c9.

[18] M. A. Hossain, T. Hasan, F. Ahmed, S. H. Cheragee, M. H. Kanchan, and M. A. Haque, "Towards superior android ransomware detection: An ensemble machine learning perspective," *Cyber Security and Applications*, vol. 3, Art. no. 100076, 2025, doi: 10.1016/j.csa.2024.100076.

[19] S. Singh, T. Khanna, and D. K. Verma, "A hybrid ensemble model for ransomware detection using feature engineering and deep learning," *International Journal of Information Technology*, vol. 17, pp. 5095–5104, 2025, doi: 10.1007/s41870-025-02681-z.

[20] G. Kirubavathi, B. Padma Mayuri, S. Pranathasree *et al*., "Ensemble machine learning for proactive android ransomware detection using network traffic," *Scientific Reports*, vol. 16, Art. no. 9498, 2026, doi: 10.1038/s41598-026-38271-