

Article

Comparative Analysis of Machine Learning Algorithms for Malicious Network-Traffic Classification

Byron Wladimir Oviedo-Bayas * , Stefany Michelle Perachimba Panezo , Jorge Humberto Guanin-Fajardo 
and Stalin Daniel Carreño Sandoya 

Facultad de Ciencias de la Computación, Carrera de Telemática, Universidad Técnica Estatal de Quevedo (UTEQ), Quevedo 120302, Los Ríos, Ecuador; sperachimba@uteq.edu.ec (S.M.P.P.); jguaninf@uteq.edu.ec (J.H.G.-F.); sdcarreno@uteq.edu.ec (S.D.C.S.)

* Correspondence: boviedo@uteq.edu.ec; Tel.: +593-5-275-6850

Abstract

The classification of malicious network-traffic is critical to cybersecurity. However, to the best of our knowledge, no previous studies have performed a comparative analysis of supervised algorithms for classifying malicious traffic, specifically within the network environment of UTEQ, an academic setting with distinctive traffic patterns and security policies. This study compared the performance of four supervised machine learning algorithms (K-Nearest Neighbors, Decision Tree, SVM-RBF, and SVM-Polynomial) using the CRISP-DM methodology. The dataset consisted of 1182 records with 30 variables from Hillstone Networks firewall logs at UTEQ, representing three categories: Normal (74.3%), Botnet_Activity (16.4%), and Other_Malware (9.3%). Preprocessing techniques included SMOTE balancing and Relief-based feature selection (reducing the variables to eight). The area under the curve (AUC) was used as a primary discriminant metric under two complementary one-vs-rest aggregation schemes. Using a support-weighted AUC, K-Nearest Neighbors ($k = 7$) obtained the highest value (AUC = 0.6147), followed by SVM-Polynomial (0.5846), Decision Tree (0.5724), and SVM-RBF (0.5784), with SVM-RBF obtaining the highest accuracy on the unified eight-feature test set (73.8%). Using a macro-averaged AUC, SVM-Polynomial obtained the highest value (0.6166), closely followed by KNN (0.6133). All AUC values fell within a narrow range (0.57–0.62). A class-wise analysis showed that no single model provides strong discrimination for all traffic classes, and that algorithm selection for operational deployment should be guided by the specific class of interest rather than by a single aggregate metric.

Keywords: K-Nearest Neighbors; Support Vector Machines; SMOTE; class imbalance; firewall logs; malicious traffic classification; network intrusion detection



Academic Editor: Carolina Del-Valle-Soto

Received: 21 May 2026

Revised: 28 June 2026

Accepted: 7 July 2026

Published: 8 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Network traffic in modern organizations generates millions of events daily that must be analyzed to identify potential threats. In academic and corporate environments, security systems continuously record information about connections, data transfers, and communication patterns, creating massive volumes of logs that exceed the capacity for manual analysis [1]. The timely detection of anomalous patterns is critical for information security; threats, such as botnets, DNS malware, and Trojans, exhibit characteristic behaviors identifiable by analyzing network metrics, temporal patterns, and communication characteristics [2].

The practical urgency of this problem is substantial. According to the IBM Cost of a Data Breach Report, the global average cost of a data breach reached USD 4.88 million in 2024, the highest figure on record, with network-borne intrusions accounting for a significant share of incidents [3]. Botnet activity has surged: the Spamhaus Botnet Threat Report documents millions of active command-and-control (C&C) servers operating globally each year, with universities disproportionately targeted due to their high-bandwidth connectivity and heterogeneous device populations [4]. DNS-based malware exploits port 53 to exfiltrate data or receive commands, blending with legitimate recursive DNS queries [2]. These observations motivate the design of adaptive, learning-based classifiers that can distinguish malicious from benign traffic in real operational settings.

The proportion of malicious traffic relative to legitimate traffic is typically very low, making identification with static rules difficult [5]. Supervised machine learning has demonstrated superior capabilities for recognizing complex patterns in large datasets, learning to accurately distinguish between traffic classes based on features extracted from historical examples [6]. Traffic classification is a foundational task in network management, underpinning Quality of Service (QoS) and Quality of Experience (QoE) in telecommunication systems [7]. The ITU provides reference guidelines, including technical guidelines for detecting malicious activity by mobile applications [8] and recommendations characterizing the security risks posed by impermissible traffic [9].

Several recent studies have addressed malicious traffic classification. Wang et al. [10] proposed an edge-intelligence approach for classifying malicious traffic in IIoT environments. Helen et al. [11] compared five algorithms and found that Random Forest outperformed others on their dataset. These studies illustrate that algorithm rankings depend strongly on dataset characteristics, feature sets, and class imbalance, which motivates dataset- and context-specific studies, such as the one presented here.

Prior comparative studies have largely focused on benchmark datasets (e.g., NSL-KDD) or industrial/IoT settings [5,10–12]. The present study differs in three respects: (i) it uses operational firewall log data from a real academic network; (ii) it jointly examines accuracy, Precision, Recall, F1-score, and AUC under two complementary aggregation schemes; and (iii) it explicitly documents the preprocessing pipeline, providing a transparent, reproducible reference pipeline for similarly resourced academic network environments.

Accordingly, the distinctive contributions of this study are: (i) demonstrating ML for identifying malicious-traffic patterns in a resource-constrained academic environment without costly commercial solutions; (ii) analyzing the impact of SMOTE class balancing and Relief feature selection; and (iii) identifying the most discriminating variables while reporting the sensitivity of model rankings to the choice of evaluation metric.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 describes the methodology; Section 4 presents the results; Section 5 discusses the findings; and Section 6 presents the conclusions.

2. Related Work

2.1. Machine Learning for Network-Traffic Classification

The application of supervised machine learning to network-traffic classification has been studied extensively. Early benchmark evaluations on the NSL-KDD dataset showed that even shallow models achieve high accuracy on majority attack categories while systematically failing on minority classes [13]. Subsequent work confirmed that this majority-class bias persists even under ensemble methods unless explicit rebalancing is applied [14]. Wang et al. [10] noted that port-based and machine-learning methods are more practical than deep packet inspection for encrypted traffic. Dawood et al. [5] reported that feature quality dominates algorithmic choice in traffic classification performance. A systematic

review by Subhashini et al. [6] covering 78 studies (2015–2024) found that SVMs and KNN remain competitive baselines when labeled training data are available.

2.2. Class Imbalance in Network Security Datasets

SMOTE, introduced by Chawla et al. [15], generates synthetic minority-class instances through linear interpolation between real observations and their nearest neighbors, and remains one of the most widely used rebalancing methods [14]. Ullah et al. [16] found that SMOTE applied exclusively to the training set improved minority-class F1-scores by 12–18 percentage points without inflating the false-positive rate. With fewer than 80 minority instances, k values of 3 or lower are recommended to prevent extrapolation beyond the empirical support of the minority class [15].

2.3. Feature Selection for Traffic Classification

The Relief algorithm, introduced by Kira and Rendell [17], estimates feature relevance by examining how well feature values distinguish instances of different classes within a nearest-neighbor context. Its primary advantages over chi-squared and mutual information filters are its sensitivity to feature interactions and its applicability to numerical features without requiring discretization [17]. In network-traffic classification, Relief consistently ranks port-based and volumetric features highly [5].

2.4. Evaluation Metrics Under Class Imbalance

Sokolova and Lapalme [18] demonstrated that no single scalar metric suffices under imbalance. Fawcett [19] established the ROC curve and AUC as threshold-independent measures of discriminative ability. For multiclass problems, Hand and Till [20] proposed weighted and macro one-vs-rest aggregation schemes, noting that the two schemes can produce different model rankings when minority-class performance differs. Patrick [21] identified false-positive “alert fatigue” in intrusion detection systems, demonstrating that models selected on accuracy alone frequently exhibit unacceptably high false-positive rates for minority-class threats.

3. Materials and Methods

3.1. CRISP-DM Methodology

This study adopted the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology [22,23]. It builds on the KDD framework of Fayyad et al. [24] and has maintained approximately 43% adoption in industry surveys [25]. The 2021 systematic literature review confirmed it as the predominant methodology in data science and cybersecurity analytics [26]. CRISP-DM structures the project into six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment [27]. Figure 1 details the project roadmap, adapted from Guanin-Fajardo et al. [28].

The data come from UTEQ’s Hillstone Networks firewall, comprising 1182 event records covering 11 original traffic categories: APT, Botnet, CnC_Server, DNS_Botnet, DNS_CnC, DNS_Miner, DNS_Trojan, Malware, Normal, Sinkhole, and Trojan. Each record contains network connection characteristics, traffic metrics (bytes sent, bytes received, bandwidth), port information (source, destination, NAT translation), and temporal metadata.

3.2. Dataset Description

The final dataset of 1182 records exhibits significant class imbalance: Normal 74.3% (878 instances), Botnet_Activity 16.4% (194 instances), and Other_Malware 9.3% (110 instances). The dataset contains 30 original variables. An initial cleaning step reduced this to 17 candidate predictors by removing raw IP addresses and redundant administra-

tive fields. Of these, 15 were retained for modeling: ports (dest_port, source_port, nat_translated_port), traffic metrics (bytes_sent, bytes_received, packets_sent, flow_bandwidth_current, flow_bandwidth_max), temporal information (month, day, hour, minute, second, day_of_week_num), and one categorical variable (botnet_tag). Two candidates (policy_id, rule_id) were excluded as deployment-specific identifiers.

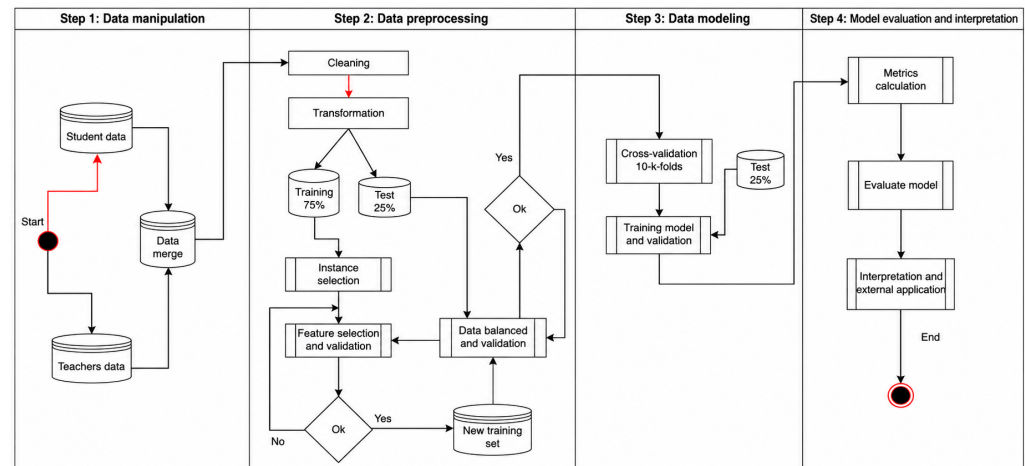


Figure 1. Diagram of the carried-out activities. The processes are described in four stages. Source: adapted from Guanin-Fajardo et al. [28].

3.3. Evaluation Metrics

Multiple complementary metrics were computed on the unmodified test set, following the standard definitions of Sokolova and Lapalme [18] and Fawcett [19]. For a given class treated as “positive” under a one-vs-rest scheme: TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives. Sensitivity (Recall) equals the True Positive rate $P(TP)$; Specificity equals the True Negative rate $P(TN)$; Precision equals the positive predictive value.

Accuracy [18]:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

Precision [18]:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

Recall (Sensitivity) [18]:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

Specificity [18]:

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (4)$$

F1-Score [18]:

$$F_1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

AUC [19]: the ROC curve plots Recall (TPR) against $FPR = 1 - \text{Specificity}$ as the decision threshold varies. The AUC is estimated using the trapezoidal rule:

$$\text{AUC} \approx \sum_{i=1}^{n-1} \frac{\text{TPR}_i + \text{TPR}_{i+1}}{2} \times (\text{FPR}_{i+1} - \text{FPR}_i) \quad (6)$$

where TPR_i and FPR_i denote the true and false positive rates at the i -th of n decision thresholds ordered by increasing FPR [19]. For the multiclass case, AUC was computed

under the one-vs-rest scheme and aggregated under two strategies: a support-weighted average (Table 1) and a macro-average (Figure 3). The macro-average is the primary metric for evaluating minority-class detection [20].

3.4. Data Preprocessing

The preprocessing phase was carried out through an iterative sequence of transformations with validation at each stage, following the decision flowchart in Figure 2.

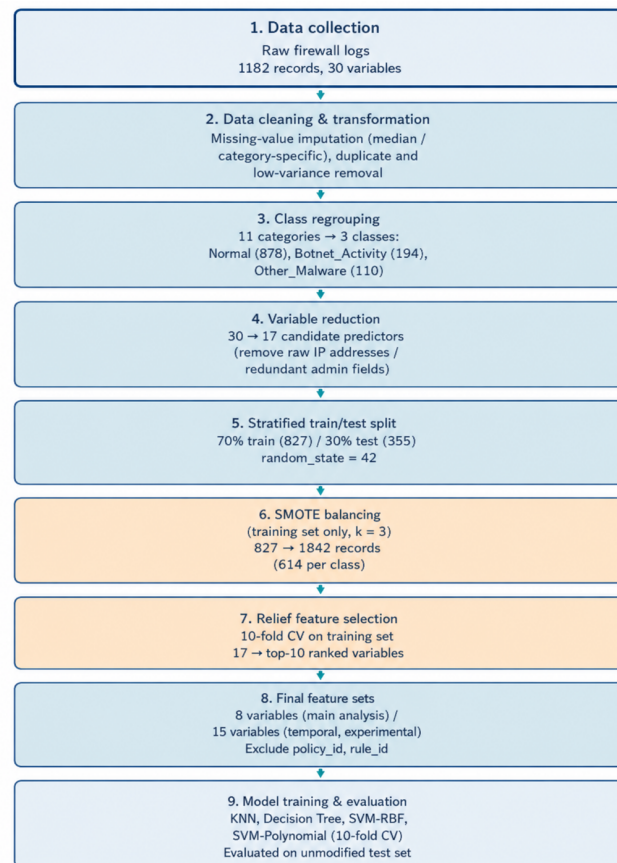


Figure 2. Decision flowchart for the data-preprocessing pipeline (Section 3.4), from raw firewall logs to the final feature sets used for model training and evaluation.

3.4.1. Cleansing and Transformation

Of the 30 original variables, 8 contained missing values at rates between 2% and 17%. For numerical variables, median imputation was applied because the traffic-volume features exhibit pronounced right skew due to large burst events, making the median a more robust location estimator than the mean [18]. For categorical variables, missing values were analyzed individually: the botnet_tag field represented structured absence (no recognized botnet signature) and was retained as a valid category. Records with more than 30% missing values (seven records) and variables with zero variance (two fields) were removed.

The 11 original categories were grouped into 3 main classes: Botnet_Activity (Botnet, CnC_Server, Sinkhole); Normal (unchanged); and Other_Malware (APT, DNS_Botnet, DNS_CnC, DNS_Miner, DNS_Trojan, Malware, Trojan). This grouping was necessary because individual categories contained as few as 3–7 instances each [15]. The dataset was divided into training (70%, 827 records) and test (30%, 355 records) sets using a stratified split with random_state = 42. A 70/30 ratio was preferred over 80/20 to retain a test set

with 33 Other_Malware instances, the minimum considered adequate for stable per-class Recall estimation [14].

3.4.2. Feature Selection and Validation

Feature selection was performed exclusively on the training set to prevent information leakage. Starting from 17 candidate predictors, the Relief filter algorithm [17] was used to rank variables by discriminative weight via 10-fold cross-validation. Relief was selected over chi-squared and mutual information filters because: (1) it is distribution-free and requires no discretization; (2) it is sensitive to feature interactions; and (3) benchmark comparisons show that Relief consistently produces competitive feature subsets on network-traffic data [5]. The top eight ranked variables after excluding deployment-specific identifiers (policy_id, rule_id) were: dest_port, source_port, nat_translated_port, bytes_sent, bytes_received, packets_sent, flow_bandwidth_current, flow_bandwidth_max.

3.4.3. Data Reconciliation (SMOTE Balancing)

SMOTE [15] was applied exclusively to the training data using $k = 3$ nearest neighbors. The parameter $k = 3$ was selected because it is the maximum feasible value given the size of the smallest minority class (77 Other_Malware instances in the training set): using $k > 3$ risks generating synthetic samples with unrealistic feature combinations [15]. SMOTE generated 1015 additional synthetic samples, increasing the training set from 827 to 1842 records and balancing the three classes to 614 each. The test set was kept completely unmodified (264 Normal, 58 Botnet_Activity, 33 Other_Malware) to ensure realistic operational evaluation [16].

3.5. Classification Algorithms

3.5.1. K-Nearest Neighbors (KNN)

KNN classifies new instances by voting on the k nearest neighbors in feature space. Values $k \in \{1, 3, 5, 7, 9\}$ were evaluated using the Manhattan distance (L1) and inverse-distance weighting. Selection of $k = 7$ was based on maximum accuracy (73.24%) following 10-fold cross-validation. StandardScaler normalized each feature to zero mean and unit standard deviation, essential for distance-based metrics [6].

3.5.2. Decision Tree

The Decision Tree uses recursive partitions maximizing class homogeneity (entropy criterion), with maximum depth of 7, minimum of 10 samples per node split, and a minimum of 5 samples per leaf. class_weight = 'balanced' automatically adjusts class weights to compensate for residual imbalance.

3.5.3. Support Vector Machine with RBF Kernel

SVM-RBF captures nonlinear relationships via the RBF kernel. Hyperparameters were optimized by Grid Search with 3-fold cross-validation over $C \in \{1, 10, 50, 100\}$ and $\gamma \in \{\text{scale}, 0.1, 0.01\}$. The optimal configuration was $C = 100$, $\gamma = \text{scale}$, with class_weight = 'balanced' and probability = True for AUC computation.

3.5.4. Support Vector Machine with Polynomial Kernel

SVM-Polynomial uses second-degree polynomial functions to capture quadratic feature interactions, configured with $C = 10.0$, $\gamma = 0.01$, coef0 = 1, and class_weight = 'balanced'. Training time was significantly longer (2–3 minutes vs. seconds), a characteristic typical of SVMs with complex kernels.

3.5.5. Cross-Validation

All models used 10-fold cross-validation during training to obtain robust performance estimates. The Python source code and raw output files (classification metrics and confusion-matrix/ROC-curve generation scripts) used to produce the results reported in Sections 4 and 4.2 are provided in File S1 (Supplementary Materials).

4. Results

4.1. Overall Performance of the Classification Models

Table 1 presents evaluation results on the unbalanced test set ($N = 355$). SVM-RBF achieved the best overall accuracy (73.8%) with Precision 0.7079, Recall 0.7380, and F1-score 0.7012. KNN ($k = 7$) achieved the highest weighted AUC (0.6147), followed by SVM-Polynomial (0.5846), Decision Tree (0.5724), and SVM-RBF (0.5784).

Table 1. Performance of classification algorithms on the test set ($N = 355$).

Model	Accuracy	Precision	Recall	F1-Score	AUC
KNN ($k = 7$)	0.7324	0.6783	0.7324	0.6936	0.6147
Decision Tree	0.7324	0.6828	0.7324	0.6959	0.5724
SVM-RBF	0.7380	0.7079	0.7380	0.7012	0.5784
SVM-Polynomial	0.7352	0.7101	0.7352	0.7020	0.5846

Models trained on SMOTE-balanced data ($k = 3$) and evaluated on the unmodified test set: 264 Normal (74.4%), 58 Botnet_Activity (16.3%), 33 Other_Malware (9.3%). Precision, Recall, and F1-Score are support-weighted averages; AUC is the support-weighted, one-vs-rest AUC (Equation (6)). A complementary macro-averaged AUC is reported in Figure 3.

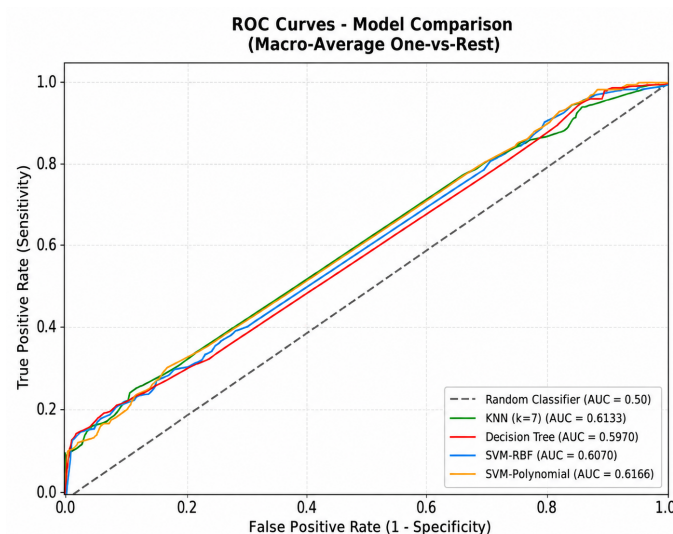


Figure 3. Comparative macro-averaged one-vs-rest ROC curves for the four models.

4.2. ROC Curve Analysis

Figure 3 shows the macro-averaged, one-vs-rest ROC curves. Under this macro-averaging scheme, SVM-Polynomial obtained the highest AUC (0.6166), followed by KNN (0.6133), SVM-RBF (0.6070), and Decision Tree (0.5970); all four values lie within 0.02 AUC points. All models outperform the random classifier ($AUC = 0.50$). The macro-averaged ranking differs from the support-weighted AUC ranking in Table 1 because the macro-average gives the minority classes the same weight as the majority Normal class [20]. In both cases, the AUC values fall within a narrow band (0.57–0.62), reflecting the intrinsic difficulty of this classification problem.

4.3. Feature Importance

The Permutation Importance analysis for KNN ($k = 7$) revealed that `dest_port` is the most discriminating variable (accuracy decrease of ≈ 0.05 when permuted), followed by `source_port`, `packets_sent`, `bytes_sent`, `nat_translated_port`, and `bytes_received` (Figure 4). Bandwidth variables (`flow_bandwidth_current`, `flow_bandwidth_max`) were of lesser importance, suggesting that static port and volume patterns are more informative than dynamic transfer rates. This is consistent with Wang et al. [10], who report that destination port remains a stable discriminating feature across IIoT traffic datasets.

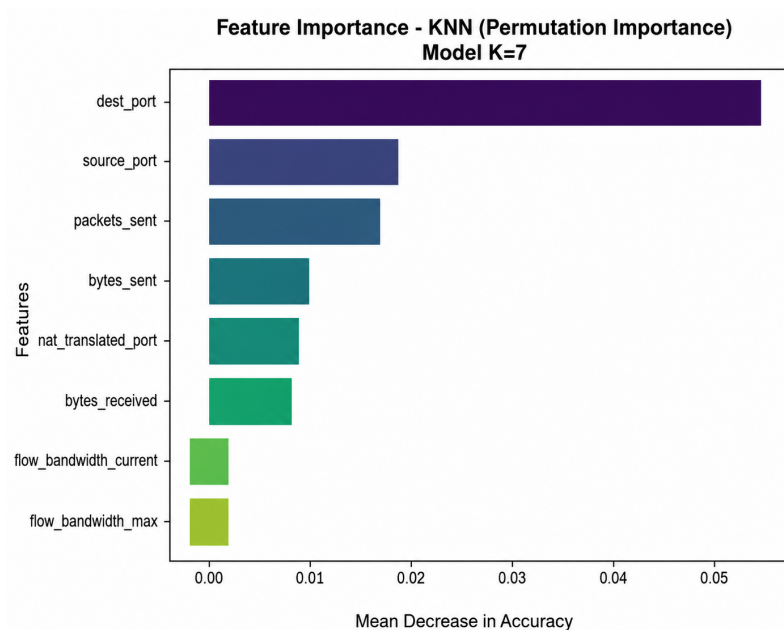


Figure 4. Assessment of the importance of features (Permutation Importance, KNN $k = 7$).

4.4. Confusion Matrices

Figures 5–8 show the confusion matrices for all four models on the unmodified test set.

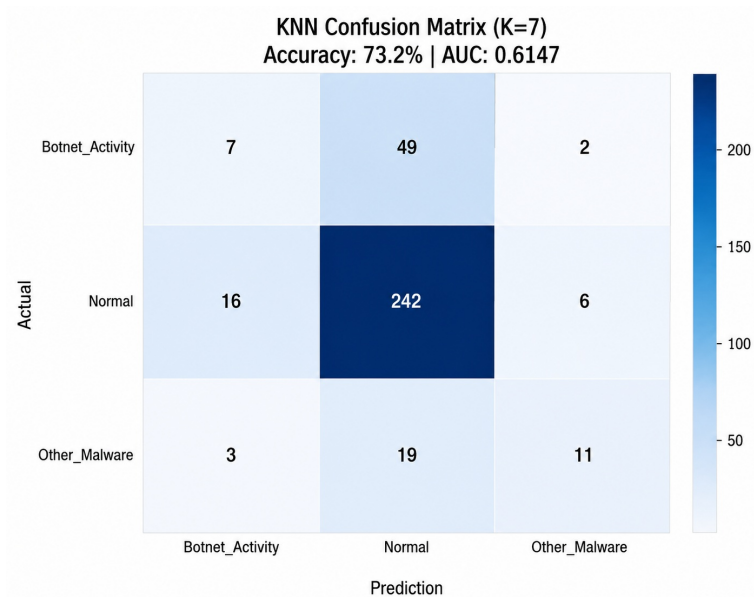


Figure 5. KNN confusion matrix ($k = 7$). Accuracy: 73.2%. AUC: 0.6147.

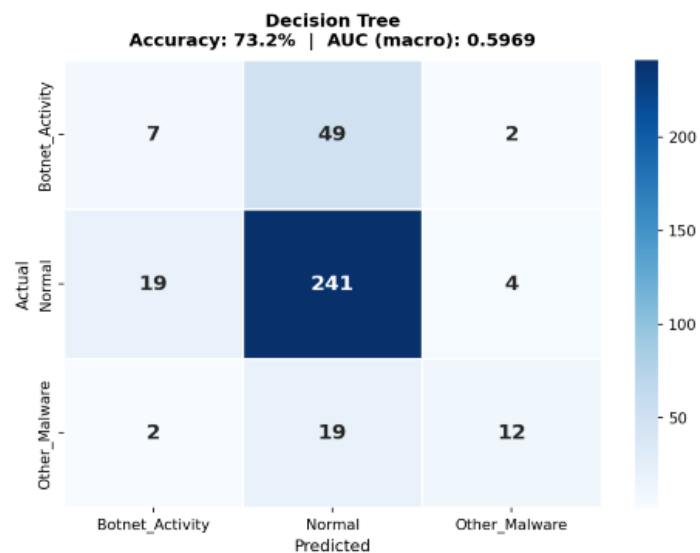


Figure 6. Confusion matrix for the Decision Tree model ($N = 355$). Accuracy: 73.2%. AUC (macro): 0.5969.

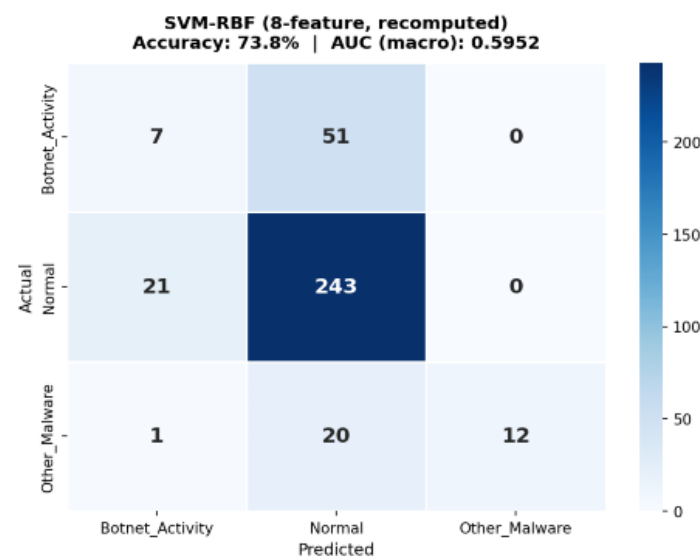


Figure 7. Confusion matrix for the SVM-RBF model ($N = 355$), re-evaluated on the 8-feature set. Accuracy: 73.8%. AUC (macro): 0.5952.

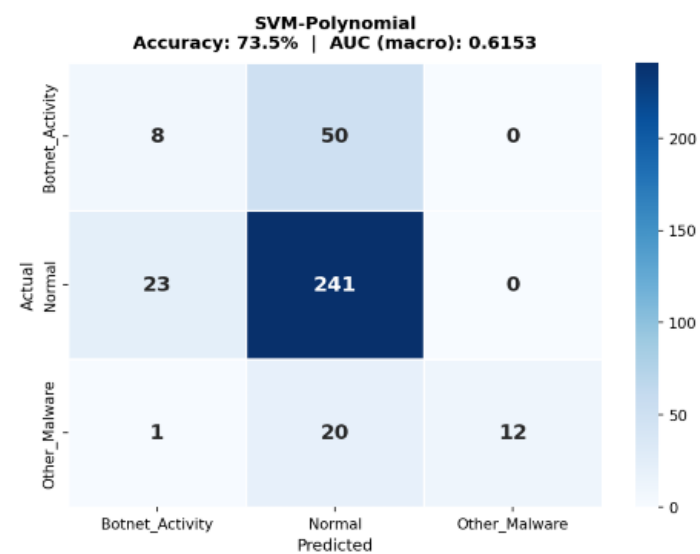


Figure 8. Confusion matrix for the SVM-Polynomial model ($N = 355$). Accuracy: 73.5%. AUC (macro): 0.6153.

4.5. Class-Wise Performance of All Four Models

Because overall accuracy and AUC can mask class-specific weaknesses under severe imbalance, Table 2 reports per-class Precision, Recall, and F1-score for all four classifiers. The weighted averages reproduce the values in Table 1, confirming internal consistency.

Table 2. Class-wise performance of all four classifiers on the unmodified test set ($N = 355$). SVM-RBF * was re-evaluated on the same 8-feature set as the other models.

Model	Class	Precision	Recall	F1-Score	Support
KNN ($k = 7$)	Botnet_Activity	0.269	0.121	0.167	58
	Normal	0.781	0.917	0.843	264
	Other_Malware	0.579	0.333	0.423	33
	Macro avg.	0.543	0.457	0.478	355
	Weighted avg.	0.678	0.732	0.694	355
Decision Tree	Botnet_Activity	0.250	0.121	0.163	58
	Normal	0.780	0.913	0.841	264
	Other_Malware	0.667	0.364	0.471	33
	Macro avg.	0.566	0.466	0.492	355
	Weighted avg.	0.683	0.732	0.696	355
SVM-RBF *	Botnet_Activity	0.241	0.121	0.161	58
	Normal	0.774	0.920	0.841	264
	Other_Malware	1.000	0.364	0.533	33
	Macro avg.	0.672	0.468	0.512	355
	Weighted avg.	0.708	0.738	0.701	355
SVM-Polynomial	Botnet_Activity	0.250	0.138	0.178	58
	Normal	0.775	0.913	0.838	264
	Other_Malware	1.000	0.364	0.533	33
	Macro avg.	0.675	0.472	0.517	355
	Weighted avg.	0.710	0.735	0.702	355

The class-wise results reveal a consistent pattern: performance on the majority Normal class is strong ($F1 \geq 0.838$ for all models), whereas performance on the minority classes is substantially weaker. For KNN, Botnet_Activity $F1 = 0.167$ and Other_Malware $F1 = 0.423$, meaning approximately two-thirds of Other_Malware instances are misclassified as Normal traffic. The two SVM variants yield perfect Precision for Other_Malware (1.000) but the same low Recall (0.364), resulting in $F1 = 0.533$ —the highest Other_Malware $F1$ across all models.

5. Discussion

The central empirical finding is that the four evaluated algorithms achieve broadly similar and modest discriminative performance (AUC 0.57–0.62), with the identity of the “best” model depending on the evaluation criterion. Under the support-weighted AUC, KNN obtained the highest value; under the macro-averaged AUC, SVM-Polynomial obtained the highest value [20]. When compared to the literature, partial convergence is observed with Moreno Ruiz and Robles [29], who reported limited Decision Tree performance (64% accuracy) on NSL-KDD. Our Decision Tree result (73.24%) suggests that rigorous preprocessing and careful hyperparameter specification can substantially improve performance, while the Decision Tree’s AUC (0.5724) remained among the lowest, reinforcing that accuracy alone is insufficient under class imbalance [18].

Three contributing factors explain the weak minority-class performance. First, although SMOTE increased minority training examples, the number of real anchor points remained small (77 Other_Malware and 136 Botnet_Activity instances), limiting the di-

versity of the synthetic subspace [15]. Second, the Botnet_Activity port distribution overlaps substantially with Normal traffic. Third, the 33 Other_Malware and 58 Botnet_Activity test instances may be insufficient for stable per-class Recall estimates [18]. Future work should address minority-class performance through: (a) cost-sensitive learning; (b) anomaly-detection approaches; (c) collection of additional operational data; and (d) threshold optimization.

Claims regarding the operational value of any single model for reducing false-positive “alert fatigue” [21] should be made cautiously until a threshold-based operational analysis is completed. A possible explanation for KNN’s relatively favorable weighted AUC is that Normal-class traffic in this environment forms dense local clusters in feature space (particularly around `dest_port`), which KNN captures effectively through proximity voting.

Limitations

This study has five main limitations: (1) The dataset is small (1182 records) from a single firewall at a single university, limiting generalizability; no comparison against public benchmarks (NSL-KDD, CICIDS2017, UNSW-NB15) was performed [13]. (2) No exhaustive SVM hyperparameter search beyond Grid Search was performed. (3) The study is retrospective and does not evaluate performance under concept drift. (4) Root-cause analysis of minority-class misclassification remains qualitative. (5) All reported AUC values (0.57–0.62) indicate only modest discriminative ability.

6. Conclusions

This study compared four supervised machine learning algorithms for classifying malicious network traffic on the UTEQ campus network. All four models achieved modest and broadly comparable discriminative performance (AUC 0.57–0.62), and the identity of the “best” model depended on the evaluation criterion: KNN obtained the highest support-weighted AUC, SVM-RBF obtained the highest overall accuracy (73.8%), and SVM-Polynomial obtained the highest macro-averaged AUC. The main practical implication is that algorithm selection should be guided by the specific operational objective together with class-wise diagnostics, not by a single aggregate score. The `dest_port` variable emerged as the most informative predictor, suggesting that port-aware feature engineering is valuable regardless of the algorithm selected. Before any operational deployment, a threshold-based analysis of false-positive and false-negative rates would be required.

Future work should: (a) extend the analysis to longitudinal and multi-university datasets including public benchmarks; (b) conduct a more exhaustive SVM hyperparameter search; (c) investigate cost-sensitive learning and anomaly-detection approaches to improve minority-class Recall; (d) perform a threshold-based operational analysis; and (e) explore classifier robustness under adversarial attacks obfuscating key variables such as the destination port.

Supplementary Materials: The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/a19070560/s1>; File S1: Python source code (KNN, Decision Tree, and SVM-Polynomial classifiers, and the multiclass ROC-curve generation script) and raw model-output files (classification-report metrics and confusion-matrix figures) underlying the results reported in the Results section.

Author Contributions: Conceptualization, B.W.O.B.; methodology, B.W.O.B. and S.M.P.P.; formal analysis, B.W.O.B.; investigation, S.M.P.P. and S.D.C.S.; resources, J.H.G.F.; data curation, S.D.C.S.; writing—original draft, B.W.O.B.; writing—review and editing, all authors; supervision, B.W.O.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data will be made available upon reasonable request to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Raschka, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv* **2020**, arXiv:1811.12808.
2. Hassan, S.E.H.; Duong-Trung, N. Machine Learning in Cybersecurity: Advanced Detection and Classification Techniques for Network Traffic Environments. *EAI Endorsed Trans. Ind. Netw. Intell. Syst.* **2024**, *11*, 5237. <https://doi.org/10.4108/EETINIS.V11I3.5237>.
3. IBM Security. *Cost of a Data Breach Report 2024*; IBM: Armonk, NY, USA, 2024.
4. Spamhaus Technology. *The Spamhaus Botnet Threat Update Q4 2023*; Spamhaus: Geneva, Switzerland, 2024.
5. Dawood, M.; Xiao, C.; Tu, S.; Alotaibi, F.A.; Alnfaia, M.M.; Farhan, M. Intelligent model for the detection and classification of encrypted network traffic in cloud infrastructure. *PeerJ Comput. Sci.* **2024**, *10*, e2027. <https://doi.org/10.7717/PEERJ-CS.2027>.
6. Subhashini, L.D.C.S.; Li, Y.; Tao, X.; Yong, J. Machine Learning for Privacy Threat Classification: A Systematic Review. *Preprint* **2025**. <https://doi.org/10.21203/RS.3.RS-6934585/V1>.
7. Poryazov, S.; Andonov, V.; Saranova, E.; Atanasov, I. Overall Model Normalization towards Adequate Prediction and Presentation of QoE. In Proceedings of the TELSIS 2019, Nis, Serbia, 23–25 October 2019; pp. 360–363. <https://doi.org/10.1109/TELSIS46999.2019.9002295>.
8. International Telecommunication Union. *Recommendation ITU-T X.1130*; ITU: Geneva, Switzerland, 2020.
9. International Telecommunication Union. *Recommendation ITU-T E.371*; ITU: Geneva, Switzerland, 2026.
10. Wang, M.; Zhang, B.; Zang, X.; Wang, K.; Ma, X. Malicious Traffic Classification via Edge Intelligence in IIoT. *Mathematics* **2023**, *11*, 3951. <https://doi.org/10.3390/math11183951>.
11. Helen, S.; Lakshmi, S.; Gokila, S.; Selvarasu, S.; Nayak, S.R. Recent Trends in AI and Data Sciences. *Lect. Notes Electr. Eng.* **2025**, *1447*, 169. https://doi.org/10.1007/978-981-96-9203-3_14.
12. Martínez, W.V.O.; Toasa, M.R.; Urdaneta, M. Evaluación del Desempeño de Algoritmos de Machine Learning para la Búsqueda de Patrones de Ciberataques. Master's Thesis, Universidad Israel, Quito, Ecuador, 2024.
13. Ullah, I.; Mahmoud, Q.H. A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. In *Proceedings of the Canadian Conference on AI*; Springer: Cham, Switzerland, 2020; pp. 508–520. https://doi.org/10.1007/978-3-030-47358-7_52.
14. Hatamlou, A.; Jafari, S. Comparative Analysis of Machine Learning Approaches for Network Intrusion Detection with Class Imbalance. *Algorithms* **2025**, *18*, 749. <https://doi.org/10.3390/algorithms18120749>.
15. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. <https://doi.org/10.1613/JAIR.953>.
16. Ullah, U.; Hassan, A.; Asif, M.; Farooq, M.S. Intelligent Intrusion Detection System for Apache Web Server Empowered with Machine Learning. *Int. J. Comput. Innov. Sci.* **2022**, *1*, 21–27.
17. Kira, K.; Rendell, L.A. The feature selection problem: Traditional methods and a new algorithm. In Proceedings of the AAAI-92, San Jose, CA, USA, 12–16 July 1992; pp. 129–134.
18. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>.
19. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. <https://doi.org/10.1016/j.patrec.2005.10.010>.
20. Hand, D.J.; Till, R.J. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Mach. Learn.* **2001**, *45*, 171–186. <https://doi.org/10.1023/A:1010920819831>.
21. Patrick, B. Reducing False Positives in Intrusion Detection Systems with Adaptive Machine Learning Algorithms. *ResearchGate* **2025**.
22. NCR; Clinton, J. *CRISP-DM 1.0: Step-by-Step Data Mining Guide*; SPSS Inc.: Chicago, IL, USA, 1999.
23. Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; Wirth, R. CRISP-DM 1.0: Towards a Standard Process Model for Data Mining. In Proceedings of the 4th International Conference on the Practical Application of KDD, Manchester, UK, 11–13 April 2000.
24. Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In *Proceedings of the KDD-1996*; AAAI Press: Menlo Park, CA, USA, 1996; pp. 82–88.
25. Saltz, J.S.; Hotz, N.; Wild, D.; Stirling, K. Exploring Project Management Methodologies Used Within Data Science Teams. In Proceedings of the AMCIS 2018, New Orleans, LA, USA, 16–18 August 2018.
26. Schröer, C.; Kruse, F.; Gómez, J.M. A systematic literature review on applying CRISP-DM process model. *Procedia Comput. Sci.* **2021**, *181*, 526–534. <https://doi.org/10.1016/J.PROCS.2021.01.199>.

27. Saltz, J.S.; Shamshurin, I.; Crowston, K. Comparing data science project management methodologies via a controlled experiment. In Proceedings of the HICSS 2017, Hilton Waikoloa Village, HI, USA, 4–7 January 2017; pp. 1013–1022.
28. Guanin-Fajardo, J.H.; Guaña-Moya, J.; Casillas, J. Predicting Academic Success of College Students Using Machine Learning Techniques. *Data* **2024**, *9*, 60. <https://doi.org/10.3390/DATA9040060>.
29. Moreno Ruiz, J.; Robles Gómez, A. Aplicación del Machine Learning en la Detección de Intrusiones para Ciberseguridad. UNED. 2024. Available online: <https://hdl.handle.net/20.500.14468/26237> (accessed on 7 July 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.