Proceedings of CONF-MLA 2026 Symposium: Learning and Decision Making in Multi Agent Software Systems
DOI: 10.54254/2755-2721/2026.BA36241

Comparison of Network Attack Detection Algorithms Based on NSL-KDD Dataset

Qi Wu

College of Computer Science and Engineering, Sanjiang University, Nanjing, China
Lubyonly@outlook.com

Abstract. Intrusion detection systems built on hand-written rules are losing ground as attackers find new ways to bypass fixed signatures. Working with the NSL-KDD benchmark, the paper put three classical machine learning methods—Decision Tree (DT), Random Forest (RF), and K-Nearest Neighbors (KNN)—through the same set of binary and five-class detection tests. Random forest handled the diverse five-class label set best, hitting 76.3% accuracy with an F1 of 0.720. In the simpler binary task, however, the decision tree pulled ahead (81.2%, F1 0.813). The reason was straightforward: the ensemble model grew too cautious when samples were scarce, suppressing false alarms at the cost of missing real attacks. DoS floods were caught almost without fail (F1 0.884). R2L and U2R, on the other hand, evaded detection almost entirely—their training presence was vanishingly small, at 0.83% and 0.04% of the training pool. Feature importance scores from the random forest pointed overwhelmingly to traffic volume and rate-based statistics as the strongest signals, while class skew stood out as the single largest obstacle. Rather than chasing marginal accuracy gains, the results argue for treating rare-class detection as the main priority in future work.

Keywords: Network intrusion detection, machine learning, NSL-KDD, random forest, feature importance

1. Introduction

The FBI logged more than 880,000 cybercrime complaints in 2023, with reported losses north of $12.5 billion [1]. Data breaches, crippled services, and attacks on critical infrastructure are no longer headlines—they are routine. In this environment, catching malicious traffic early and accurately has become a basic operational requirement, not a nice-to-have. The trouble is that most deployed intrusion detection systems still lean on static rule sets and signature databases. Those tools handle known threats well enough; they crumble when the attack is novel, quick to morph, or designed to mimic normal behavior. Machine learning offers an alternative that does not depend on analysts writing new rules every week. Algorithms trained on network traffic can pick up statistical patterns that are invisible to hand-crafted heuristics, and they can adapt as the threat landscape shifts.

Several research groups have already put machine learning to the test on public intrusion datasets. Ravipati and Abualkibash ran a head-to-head comparison across multiple classifiers on KDD99 and NSL-KDD; random forest and decision tree came out at 82.1% and 80.5%, handily beating naive

DOI: 10.54254/2755-2721/2026.BA36241

Bayes, which managed just 74.3% [2]. Ngueajio and colleagues reviewed SVM-based intrusion detection and highlighted the method's strong generalization on high-dimensional network features [3]. More recently, deep architectures have raised the ceiling: a CNN-BiLSTM fusion tested by Udurume et al. reached 85.7% five-class accuracy on NSL-KDD [4]. Deep neural nets clearly squeeze out extra percentage points, but they demand large labeled corpora and significant compute, and their decisions are famously hard to audit. Classical models, for all their simplicity, offer something deep networks do not: a paper trail. A security analyst can look at a decision tree's split points or a random forest's feature ranking and understand why a particular connection was flagged. Most of the existing comparison studies, though, stop at reporting a single accuracy. They rarely ask how the same algorithms behave when the classification task changes from binary to multi-class, or what happens to the rarely-seen attack categories that matter most in practice.

Dataset characteristics also shape outcomes. NSL-KDD, a cleaned-up successor to KDD99 that removed redundant records, became a standard benchmark precisely because it avoids the classifier bias problem that plagued the original [5]. Hakke et al. showed that a model's ranking can flip depending on whether it is tested on NSL-KDD, UNSW-NB15, or CICIDS2017 [6]. Arcos-Argudo et al. reported that applying SMOTE oversampling lifted recall for rare attack classes from under 10% to about 35% [7]. Zamri and Juremi stressed that reporting accuracy alone is misleading when class distributions are skewed [8]. These observations suggest that both dataset choice and imbalance handling deserve as much attention as model selection, yet most published comparisons still foreground a single accuracy.

The literature, for all its breadth, leaves a few questions hanging. Classical algorithms have rarely been compared at two different classification granularities—binary versus multi-class—inside the same experimental pipeline, so it do not have a clear picture of how each model's strengths shift when the label set expands or contracts. At the same time, class imbalance is widely acknowledged as a problem, but few papers trace exactly which attack categories pay the price and why. There is also a disconnect between feature importance rankings—something tree-based models produce almost for free—and the concrete failure patterns visible in a confusion matrix. This study tries to close those gaps. Decision Tree, Random Forest, and KNN are run through identical preprocessing and evaluated on accuracy, precision, recall, and F1-score across binary and five-class setups. Feature importance scores are then connected back to the misclassification patterns, with the goal of producing a comparison that is useful for both model selection and diagnosis.

2. Methods

2.1. Data source and description

The NSL-KDD dataset, originally built by Tavallaee et al. as an improved version of KDD99 and now distributed by the Canadian Institute for Cybersecurity, eliminates the massive redundancy that made the earlier dataset unsuitable for fair benchmarking [5, 9]. It contains 125,973 training records, a 20% subset of 25,192 records, and 22,544 test records. Each record carries 41 features and a label that falls into one of five categories: Normal, DoS, Probe, R2L, or U2R. It used the 20% subset for training—this keeps training time manageable while preserving the natural class proportions—and reserved the full test set for evaluation. The test set includes 17 attack variants absent from the training data, which makes it a genuine test of generalization rather than a simple memorization check.

DOI: 10.54254/2755-2721/2026.BA36241

2.2. Feature selection and description

The 41 features divide into four groups (Table 1): TCP connection basics (9 dimensions), content features (13 dimensions), and two categories of traffic statistics computed over the same host (9 dimensions) and the same destination host (10 dimensions), respectively. It kept all 41 features rather than pre-selecting a subset, partly to establish a full-feature baseline and partly because the random forest's built-in importance ranking would later reveal which ones mattered most. Categorical variables—protocol type (3 values), service (70 values), and flag (11 values)—were label-encoded instead of one-hot-encoded; with 70 service categories, one-hot encoding would have blown up the feature space to over 120 dimensions, hurting distance-based KNN. All features were then standardized to zero mean and unit variance.

Table 1. Overview of the 41 features in the NSL-KDD dataset

Feature Category	Dim	Representative Features
TCP Connection Basics	9	duration, protocol type, service, flag, source bytes, destination bytes, land, wrong fragment, urgent
Content Features	13	hot, number of failed logins, logged in, number of compromised, root shell, su attempted, number of root, number of file creations, number of shells, number of access files, number of outbound commands, is host login, is guest login
Traffic Statistics (Same Host)	9	count, service count, SYN error rate, service SYN error rate, REJ error rate, service REJ error rate, same service rate, different service rate, service different host rate
Traffic Stats (Same Dest. Host)	10	destination host count, destination host service count, destination host same service rate, destination host different service rate, destination host same source port rate, destination host service different host rate, destination host SYN error rate, destination host service SYN error rate, destination host REJ error rate, destination host service REJ error rate

2.3. Method description

Three algorithms, chosen for their contrasting design philosophies, were compared under identical conditions.

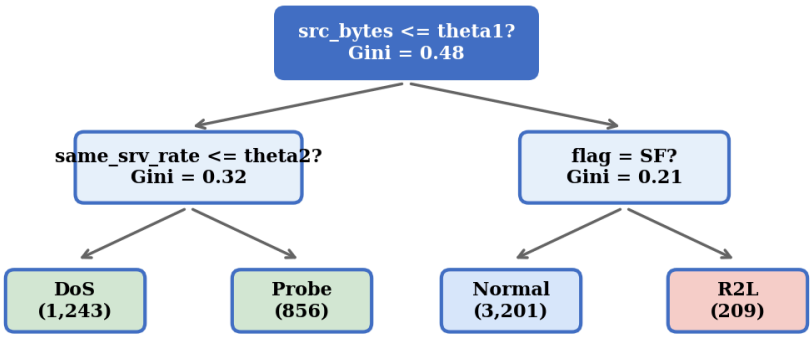

Figure 1. Schematic diagram of the Decision Tree model (photo/picture credit: original)

DOI: 10.54254/2755-2721/2026.BA36241

Decision Tree (DT) partitions the feature space through recursive binary splits, producing a tree whose leaf nodes carry class predictions. Its main draw is transparency: every decision can be traced back to a specific feature threshold, and no normalization is needed beforehand (Figure 1) [1].

Random Forest (RF) builds an ensemble of trees, each trained on a bootstrap sample of the data and constrained to split on a random subset of features at each node. The final prediction comes from majority voting across all trees. The injection of randomness at both the sample and feature levels curbs overfitting and yields a built-in estimate of feature importance (Figure 2) [1].

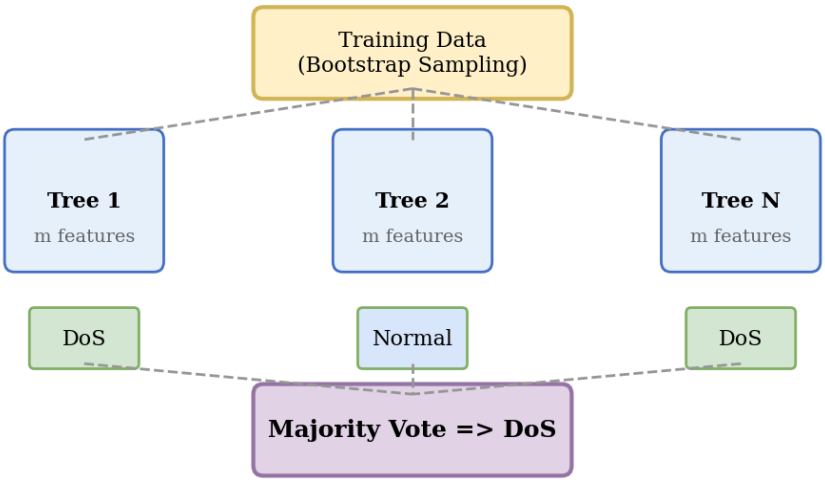

Figure 2. Schematic diagram of the Random Forest model (photo/picture credit: original)

K-Nearest Neighbors (KNN) takes a different approach entirely: it stores the training data and, for each new sample, finds the K closest points by Euclidean distance. The label is decided by a vote among those neighbors. It sets K = 5 throughout. KNN's appeal is its simplicity—no training phase, no hyperparameter grid—and its non-parametric nature can be advantageous when the decision boundary is irregular (Figure 3) [2].

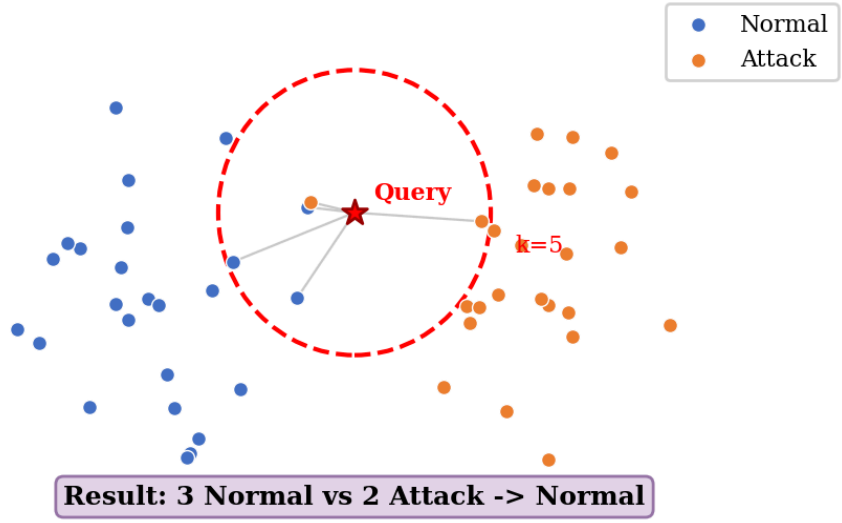

Figure 3. Schematic diagram of the KNN model (k = 5) (photo/picture credit: original)

DOI: 10.54254/2755-2721/2026.BA36241

3. Results and discussion

3.1. Overall performance comparison

Table 2 summarizes performance across both tasks. In five-class classification, random forest came out on top (76.3% accuracy, F1-score 0.720), though the margin over decision tree and KNN was narrow—within two percentage points. The binary classification results were more striking: decision tree reached 81.2% accuracy with an F1-score of 0.813, while random forest and KNN each posted precision above 0.96 but recall of only 0.626 and 0.616. What happened is that the ensemble and the neighbor-based classifier both grew very cautious on the minority class; they would rather stay silent than risk a false alarm. This pattern fits the observation by Zamri and Juremi that majority voting under class skew pulls ensemble predictions toward the dominant class [8]. A single tree, unencumbered by committee voting, found a better balance between catching attacks and avoiding false positives. For an intrusion detection context where missing an attack is usually costlier than raising a false alarm, that balance matters.

Table 2. Experimental results on five-class and binary classification tasks

Task	Algorithm	Accuracy	Precision	Recall	F1-score
5-Class	Decision Tree	0.750	0.800	0.750	0.715
5-Class	Random Forest	0.763	0.825	0.763	0.720
5-Class	KNN (k=5)	0.753	0.807	0.753	0.708
Binary	Decision Tree	0.812	0.937	0.718	0.813
Binary	Random Forest	0.775	0.967	0.626	0.760
Binary	KNN (k=5)	0.772	0.973	0.616	0.754

3.2. Per-class detection performance analysis

Breaking down the best five-class model—random forest—by category exposes sharp disparities (Table 3). DoS attacks were detected with near-production quality: F1-score 0.884, precision 0.958, recall 0.820. These attacks produce unmistakable traffic signatures—spikes in connection count, high SYN error rates—that the statistical features capture well. Normal traffic was rarely missed (recall 0.974), yet the model's precision on Normal was only 0.662, meaning about one in three samples labeled Normal was actually an attack—most of them R2L. Probe attacks sat in the middle ground (F1 = 0.723).

Table 3. Per-class performance of Random Forest in five-class classification

Class	Precision	Recall	F1-score	Test Samples
DoS	0.958	0.820	0.884	7,458
Normal	0.662	0.974	0.788	9,711
Probe	0.858	0.625	0.723	2,421
R2L	1.000	0.041	0.078	2,887
U2R	1.000	0.030	0.058	67

The real failure is with R2L and U2R. Their recall rates were 4.1% and 3.0%, respectively, and both F1-scores fell below 0.08. The normalized confusion matrix (Figure 4) makes the problem

DOI: 10.54254/2755-2721/2026.BA36241

visual: almost every R2L and U2R sample landed in the Normal column. Two factors converge here. First, the numbers are tiny—only 209 R2L and 11 U2R examples in the training set, or 0.83% and 0.04% of the data. At that scale, no learning algorithm can build a reliable decision boundary. Second, these attacks do not announce themselves through volume or connection frequency the way DoS attacks do. R2L techniques such as password guessing look, at the connection level, much like legitimate remote access. U2R exploits such as buffer overflows happen after an attacker has already gained a foothold, so the traffic-level statistics appear benign. Connection-level features—source bytes, connection count—simply do not carry enough signal for these categories. The same conclusion was reached by Arcos-Argudo et al. and Zamri and Juremi, both of whom pointed to class imbalance and feature limitations as the dual bottleneck for minority attack detection [7, 8].

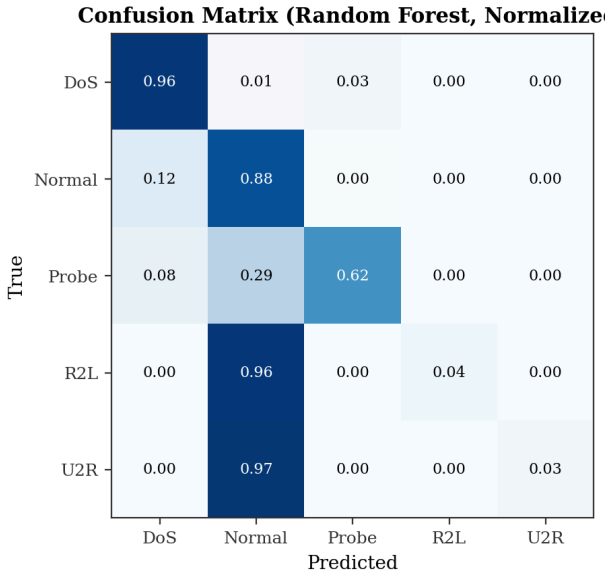

Figure 4. Normalized confusion matrix of Random Forest in five-class classification (photo/picture credit: original)

3.3. Feature importance analysis

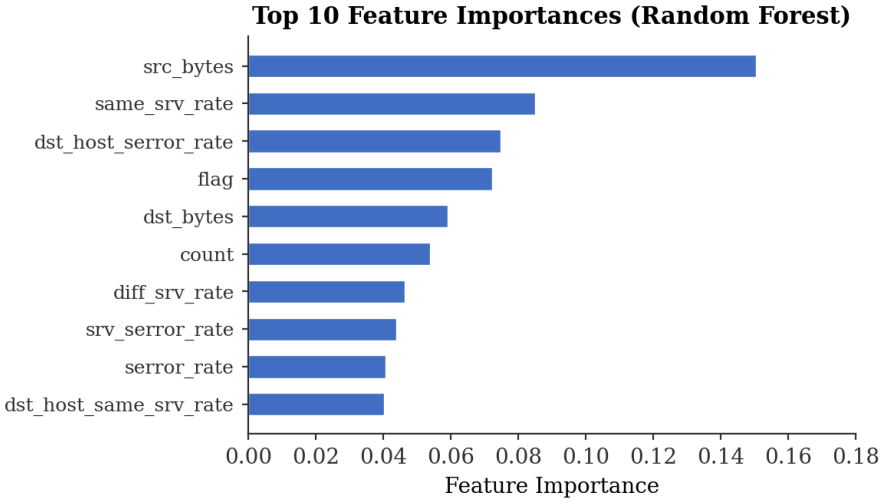

Figure 5. Top 10 feature importance scores from Random Forest (photo/picture credit: original)

DOI: 10.54254/2755-2721/2026.BA36241

One practical benefit of using random forest is the feature importance ranking it provides through mean decrease in impurity. The top five contributors (Figure 5) were source bytes (0.150), same service rate (0.085), destination host SYN error rate (0.075), flag (0.072), and destination bytes (0.059). Volume-related features naturally separate DoS flows from normal ones. Traffic statistics dominated the top ten, claiming six spots, while basic connection metadata such as protocol type ranked much lower. Only one content feature—hot—made the list, at ninth place. That a single content indicator survived suggests that the content dimension might be more useful than the ranking implies, but with so few R2L and U2R samples, the model never had a chance to exploit it. Das et al. and Kasongo and Sun reached similar conclusions about the value of feature selection and dimensionality reduction for intrusion detection [10, 11].

4. Conclusion

This study put three classical machine learning algorithms—Decision Tree, Random Forest, and K-Nearest Neighbors—through a controlled comparison on the NSL-KDD dataset, examining both binary and five-class detection scenarios.

On the algorithm front, random forest earned the top spot in five-class classification (accuracy 76.3%, F1-score 0.720), confirming the value of ensembling when the label space is diverse. Decision tree, however, was the better binary classifier (accuracy 81.2%, F1-score 0.813). Its strength was not raw accuracy alone but a more even precision-recall profile, which makes it the safer choice when analysts need to trace a model's reasoning.

At the data level, the imbalance problem proved decisive. DoS attacks were handled reliably, but R2L and U2R—the very categories that represent the most dangerous real-world threats—went almost entirely undetected because their training presence was negligible (0.83% and 0.04%). Addressing this disparity through oversampling or cost-sensitive objectives is not an optional refinement; it is a prerequisite for any system that aspires to operational relevance.

On the feature side, traffic statistics consistently outperformed connection metadata. The importance ranking also revealed a mismatch: content features that could plausibly help with R2L and U2R never got a fair trial because those classes were too small to learn from.

Several limitations should be acknowledged. The experiments were confined to a single, aging dataset; no hyperparameter search was performed; and the class imbalance problem was diagnosed but not treated with remedies such as SMOTE. Future work should validate these findings on newer datasets, incorporate systematic hyperparameter tuning, and test whether oversampling combined with lightweight deep architectures can close the detection gap for minority attack classes.

References

[1] Federal Bureau of Investigation. (2024). Internet crime report 2023. https://www.ic3.gov/Media/PDF/AnnualReport/2023_IC3Report.pdf

[2] Ravipati, R. D., & Abualkibash, M. (2019). Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets – a review paper. *International Journal of Computer Science and Information Technology, 11*, 1–14.

[3] Ngueajio, M. K., Washington, G., Rawat, D. B., et al. (2022). Intrusion detection systems using support vector machines on the KDDCUP'99 and NSL-KDD datasets: A comprehensive survey. *arXiv*. https://arxiv.org/abs/2209.05579

[4] Udurume, M., Shakhov, V., & Koo, I. (2024). Comparative analysis of deep convolutional neural network–bidirectional long short-term memory and machine learning methods in intrusion detection systems. *Applied Sciences, 14*(8), Article 3289.

DOI: 10.54254/2755-2721/2026.BA36241

[5] Tavallaee, M., Bagheri, E., Lu, W., et al. (2009). A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA)* (pp. 1–6).
[6] Hakke, D. G., Dixit, A. Y., Thorat, S., et al. (2025). Performance evaluation of machine learning-based intrusion detection using NSL-KDD, UNSW-NB15 and CICIDS2017 datasets. *International Journal of Applied Mathematics, 38*(2), 145–162.
[7] Arcos-Argudo, M., Bojorque, R., & Torres, A. (2025). A deterministic comparison of classical machine learning and hybrid deep representation models for intrusion detection on NSL-KDD and CICIDS2017. *Algorithms, 18*(3), 112–130.
[8] Zamri, I. T. B., & Juremi, J. (2023). Review of network intrusion systems evaluated on NSL-KDD and CIC-IDS2017 datasets. *Journal of Applied Technology and Innovation, 7*(1), 33–48.
[9] Canadian Institute for Cybersecurity. (2009). NSL-KDD dataset [Data set]. *University of New Brunswick*. https://www.unb.ca/cic/datasets/nsl.html
[10] Kasongo, S. M., & Sun, Y. (2023). A deep learning method with wrapper-based feature selection for wireless intrusion detection system. *Computers & Security, 124*, Article 102984.
[11] Das, S., Mahfouz, A. M., Venugopal, D., & Shiva, S. (2023). Dimensionality reduction for network intrusion detection: A comparative study of feature selection techniques. *IEEE Access, 11*, 89321–89337.