
FEATURE SELECTION USING RANDOM FOREST FEATURE IMPORTANCE FOR EFFICIENT NETWORK INTRUSION DETECTION ON THE NSL-KDD DATASET

Shafiu Shuaibu¹, Basira Yahaya¹, A.d. Adekale¹, Aminu Yakubu^{2,*}

¹Ahmadu Bello University Zaria

²Federal University of Education Zaria

Email: shafson6@gmail.com, basiraee@gmail.com, adekale@abu.edu.ng, aminuyakubu@fuez.edu.ng*

*Corresponding author: Aminu Yakubu

DOI: <https://doi.org/10.67203/sp.ICESID.2026>

ABSTRACT

Network Intrusion Detection Systems (NIDS) require informative features to distinguish benign from malicious traffic. This study investigates Random Forest feature importance for reducing the dimensionality of NSL-KDD. Using KDDTrain+, an 80:20 stratified split was applied after preprocessing, producing 121 predictive features. The top 30 features selected by Random Forest reduced dimensionality by 75.21%. Evaluated with an optimized Random Forest, the reduced representation achieved 99.83% accuracy, 99.93% precision, 99.70% recall, 99.82% F1-score, and 0.0594% false-positive rate. The results demonstrate that Random Forest can identify a compact and informative feature subset under the stated experimental conditions. However, computational efficiency was not directly measured, and a full 121-feature baseline must be rerun before confirming performance preservation or improvement.

Keywords: Network Intrusion Detection; Feature Selection; Random Forest; Feature Importance; NSL-KDD; Machine Learning; Dimensionality Reduction

1. INTRODUCTION

The continued growth of computer networks and Internet services has increased the need for automated detection of malicious activities. Network Intrusion Detection Systems (NIDS) monitor network traffic and classify connections as normal or suspicious. Machine-learning techniques are widely used because they can learn attack patterns and support automated detection (Buczak & Guven, 2016; Khraisat et al., 2019).

The NSL-KDD dataset remains a widely used benchmark for evaluating machine-learning-based intrusion detection methods. It was developed to address limitations of the original KDD Cup 1999 dataset and provides a structured basis for IDS research (Tavallaei et al., 2009). However, categorical encoding can produce a high-dimensional feature space containing variables with limited contribution to classification.

Feature selection reduces this problem by identifying relevant variables while preserving their original meaning. Common approaches include filter, wrapper, and embedded methods (Chandrashekar & Sahin, 2014). Random Forest provides an embedded feature-importance mechanism that can rank variables according to their contribution to model performance (Breiman, 2001).

Recent studies have highlighted the importance of feature selection in IDS research, including the use of Random Forest on NSL-KDD (Umar et al., 2025). However, differences in preprocessing, data partitioning, and evaluation procedures make direct comparison difficult.

Therefore, this study investigates Random Forest feature importance for ranking the 121 encoded predictive features of NSL-KDD and selecting the top 30 features for intrusion detection classification. The study (i) applies a reproducible Random Forest-based feature-ranking procedure, (ii) reduces the feature space from 121 to 30 features, and (iii) evaluates the selected features using an optimized Random Forest classifier on a held-out test set.

2. RELATED WORK

Machine learning is widely used for intrusion detection, with studies emphasizing dataset characteristics, algorithm selection, and computational considerations (Buczak & Guven, 2016).

Feature selection reduces redundant and irrelevant variables and can improve IDS efficiency. Methods are commonly classified as filter, wrapper, and embedded approaches, with Random Forest providing embedded feature-importance measures during training.

Recent NSL-KDD studies demonstrate the value of feature selection. RF-based and other selection methods have achieved strong detection performance while reducing feature dimensionality.

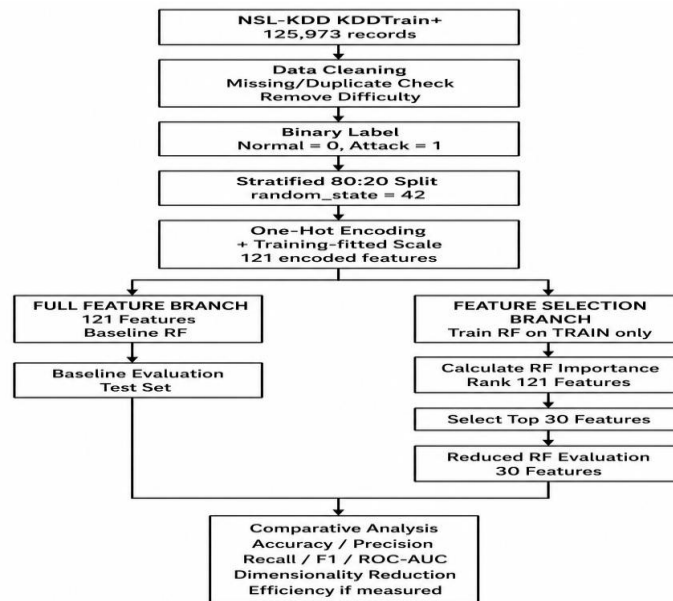
This study applies Random Forest impurity-based feature importance to select the top 30 features from 121 one-hot-encoded NSL-KDD features and evaluates them using an optimized Random Forest under an 80:20 stratified KDDTrain+ split.

3. METHODOLOGY

3.1 Research Design

An experimental machine-learning design was adopted. The complete workflow consisted of data preparation, train-test partitioning, binary label transformation, categorical encoding, feature scaling, Random Forest feature-importance estimation, feature ranking, top-30 selection, classification and comparative analysis.

The test partition was not used to calculate feature-importance scores or select features.



3.2 Dataset

The experiment used the KDDTrain+ file from the NSL-KDD dataset. The loaded file contained 125,973 records and 43 columns, comprising 41 predictive attributes, the attack label and the auxiliary difficulty attribute.

The three categorical attributes were:

- protocol_type
- service
- flag

The original attack labels were retained during initial loading and subsequently transformed into a binary classification target.

KDDTest+ was not used in the reported experiment. Instead, KDDTrain+ was partitioned into training and testing subsets.

3.3 Data Preprocessing

Data integrity checks were performed before modelling. The notebook recorded no missing values and no duplicate rows in the loaded KDDTrain+ data.

The auxiliary difficulty attribute was removed before the final experimental pipeline. Consequently, the model input contained 41 predictive attributes before categorical encoding.

The preprocessing procedure therefore consisted of:

1. loading KDDTrain+;
2. checking missing and duplicate records;
3. removing difficulty;
4. separating predictors and target;
5. splitting the data into training and testing partitions;
6. converting the target to binary labels;
7. one-hot encoding categorical predictors; and
8. standardizing the resulting numerical representation.

3.4 Binary Label Transformation

The original attack categories were transformed into a binary target:

$$y = \begin{cases} 0, & \text{normal} \\ 1, & \text{attack} \end{cases}$$

After the 80:20 stratified partition, the training set contained 100,778 records, comprising 53,874 normal and 46,904 attack records. The test set contained 25,195 records, comprising 13,469 normal and 11,726 attack records.

3.5 Categorical Encoding

The categorical attributes protocol_type, service and flag were converted into binary indicator variables using one-hot encoding.

After removal of the difficulty attribute, the resulting representation contained 121 encoded predictive features.

The encoding was performed separately on the training and test partitions, after which the test representation was aligned to the training feature columns. This ensured that both partitions had the same model-input structure.

3.6 Feature Scaling

Standardization was applied using StandardScaler.

For a feature x , the standardized value is:

$$z = \frac{x - \mu}{\sigma}$$

where μ and σ represent the mean and standard deviation estimated from the training data.

Importantly, the scaler was fitted only on the training partition:

$$X_{train}^{scaled} = \text{Scaler}.fit(X_{train})$$

and subsequently applied to the held-out test partition:

$$X_{test}^{scaled} = \text{Scaler}.transform(X_{test})$$

Thus, test-set statistics were not used when fitting the scaler.

Although Random Forest is relatively insensitive to feature scaling, scaling was retained as part of the implemented preprocessing pipeline.

3.7 Train/Test Strategy

The 125,973 KDDTrain+ observations were divided using an 80:20 stratified train-test split:

- training: 100,778 records;
- testing: 25,195 records;
- test_size=0.20;
- random_state=42;
- shuffle=True;

- stratification based on the binary target.

The stratified procedure preserved the relative distribution of normal and attack instances between the two partitions.

No KDDTest+ records were used in the reported experiment. Consequently, the results should be interpreted as performance on a random held-out portion of KDDTrain+ rather than as results obtained using the standard KDDTrain+/KDDTest+ benchmark separation.

3.8 Random Forest Feature Importance

Random Forest was used as the feature-importance estimator. The optimized Random Forest configuration obtained through five-fold randomized cross-validation was:

Parameter	Value
Number of trees	100
Maximum depth	30
Minimum samples split	10
Minimum samples leaf	1
Maximum features	sqrt
Criterion	Gini (default)
Bootstrap	True (default)
Class weight	None (default)
Random state	42
Number of parallel jobs	-1

The hyperparameter search considered 20 randomly selected parameter combinations using five-fold cross-validation and accuracy as the scoring measure. The best cross-validation accuracy recorded in the notebook was 0.9995634.

Feature importance was obtained from the Random Forest `feature_importances_` attribute. This corresponds to impurity-based, or Mean Decrease in Impurity (MDI), importance. For a feature j , its overall importance can be represented as the sum of the impurity reductions attributable to that feature across the trees, weighted by the number of samples reaching the corresponding nodes.

The feature-importance estimator was fitted using the training partition only. The test labels were not used in feature ranking.

3.9 Feature Ranking and Top-30 Selection

The 121 encoded training features were ranked in descending order according to their Random Forest importance scores. The 30 highest-ranked features were retained.

The reduction was:

$$121 - 30 = 91$$

and the percentage reduction was:

$$\frac{121 - 30}{121} \times 100 = 75.21\%$$

Thus, the selected representation contains:

$$\frac{30}{121} \times 100 = 24.79\%$$

of the original encoded feature space.

Table 1: Feature Representation Progression

Representation	Features/Attributes	Description
Raw predictive representation	41	Original predictive attributes after removing label and difficulty
Encoded representation	121	Predictive features after one-hot encoding
Selected representation	30	Top-ranked features selected using Random Forest importance
Features removed	91	Features excluded from the reduced representation
Dimensionality reduction	75.21%	Percentage of encoded features removed

Table 2: Top-Ranked Features from Random Forest Feature Importance

FEATURE SELECTION USING RANDOM FOREST FEATURE IMPORTANCE FOR EFFICIENT
NETWORK INTRUSION DETECTION ON THE NSL-KDD DATASET

Rank	Feature	Importance
1	dst_bytes	0.148207
2	src_bytes	0.127277
3	flag_SF	0.065321
4	dst_host_srv_count	0.054315
5	diff_srv_rate	0.053405
6	same_srv_rate	0.050875
7	dst_host_same_srv_rate	0.036270
8	dst_host_diff_srv_rate	0.033897
9	count	0.030726
10	srv_serror_rate	0.030007
11	flag_S0	0.029867
12	service_http	0.023957
13	dst_host_same_src_port_rate	0.023873
14	logged_in	0.023026
15	dst_host_srv_diff_host_rate	0.022148
16	dst_host_count	0.021498
17	service_private	0.021402
18	protocol_type_icmp	0.021364
19	dst_host_serror_rate	0.020812
20	serror_rate	0.015389
21	protocol_type_tcp	0.013379
22	service_eco_i	0.011072
23	srv_count	0.011048
24	service_ecr_i	0.010700
25	dst_host_srv_serror_rate	0.010415
26	hot	0.010077
27	service_ftp_data	0.009848
28	dst_host_rerror_rate	0.009014
29	dst_host_srv_rerror_rate	0.007972
30	srv_rerror_rate	0.007518

3.10 Final Classifier and Configuration

The classifier used to generate the reported 99.83% performance was the optimized Random Forest classifier operating on the 30 selected features.

This distinction is important. Random Forest was first used to estimate feature importance and rank the 121 encoded training features. The resulting 30-feature subset was then used to retrain the Random Forest classifier for the final evaluation.

The final classifier used:

- 100 trees;
- maximum depth of 30;
- minimum samples split of 10;
- minimum samples leaf of 1;
- square-root feature selection at each split;
- Gini impurity criterion;
- random_state=42;
- parallel execution using n_jobs=-1.

The classifier was fitted using the training records and evaluated using the held-out test records.

3.11 Baseline and Reduced-Feature Experiments

A valid comparison requires evaluation of the same classifier and experimental procedure using both the complete 121-feature representation and the reduced 30-feature representation.

The supplied notebook contains earlier baseline experiments, but those experiments were conducted before removal of difficulty and therefore used a different 122-feature representation. Those values should not be presented as the 121-feature baseline required for this comparison.

Table 3: Full-Feature Versus Reduced-Feature Random Forest Performance

Feature Representation	Accuracy	Precision	Recall	F1-score
Full 121 features	99.96%	99.97%	99.94%	99.96%
Selected 30 features	99.8293%	99.9316%	99.7015%	99.8164%

The full-feature baseline must be rerun using exactly the corrected 121-feature pipeline before the paper can state whether the selected representation preserves, improves or reduces predictive performance relative to all available encoded features.

3.12 Computational Efficiency Evaluation

The experiment did not record training time, prediction time or memory consumption. Therefore, computational efficiency cannot be claimed as an experimentally demonstrated outcome.

The demonstrated result is feature-space reduction from 121 to 30 features, corresponding to 75.21% dimensionality reduction.

Measure	Full 121 Features	Selected 30 Features
Training time	Not measured	Not measured
Inference time	Not measured	Not measured
Memory usage	Not measured	Not measured
Number of input features	121	30
Feature reduction	—	75.21%

4. RESULTS AND DISCUSSION

4.1 Feature Selection Results

Random Forest feature importance produced a ranked representation of the 121 encoded features. The most important features were `dst_bytes`, `src_bytes`, `flag_SF`, `dst_host_srv_count`, `diff_srv_rate` and `same_srv_rate`.

The importance scores indicate that byte-volume characteristics and connection-level or host-level traffic statistics contributed substantially to the fitted Random Forest model. For example, `src_bytes` and `dst_bytes` quantify traffic volume associated with a connection, while variables such as `count`, `srv_count` and destination-host statistics describe patterns of repeated or related connections. State and service indicators, including `flag_SF`, `flag_S0` and service-specific encoded variables, also appeared among the highly ranked features.

These observations should not be interpreted as proving that any individual feature causes an intrusion. Rather, they indicate that the features provided useful discriminatory information to the trained Random Forest under the experimental conditions.

4.2 Reduced-Feature Classification Performance

Using the 30 selected features, the optimized Random Forest achieved an accuracy of 99.8293%. Precision was 99.9316%, while recall was 99.7015%. The resulting F1-score was 99.8164%.

The confusion matrix was:

	Predicted Normal	Predicted Attack
Actual Normal	13,461	8
Actual Attack	35	11,691

Only eight normal instances were incorrectly classified as attacks, giving a false-positive rate of 0.0594%. Thirty-five attack instances were incorrectly classified as normal.

4.3 Full-Feature Baseline Comparison

The supplied notebook does not contain a valid post-correction 121-feature baseline using the same final preprocessing configuration. Consequently, no scientific conclusion can yet be made about whether the 30-feature representation improves or preserves performance relative to the complete 121-feature representation.

The earlier baseline Random Forest result in the notebook should not be substituted for this comparison because it was generated before difficulty was removed and therefore corresponds to a different feature representation.

The required baseline experiment is:

1. use the corrected KDDTrain+ data after removal of difficulty;
2. retain all 121 encoded features;
3. use the same 80:20 stratified split with `random_state=42`;
4. fit preprocessing only on training data;
5. train the same optimized Random Forest configuration;
6. evaluate on the same held-out test partition; and
7. report accuracy, precision, recall and F1-score.

Only after this experiment is completed can the paper determine whether feature selection preserved, improved or reduced predictive performance.

4.4 Leakage and Evaluation Integrity

The corrected experimental pipeline follows an appropriate order for avoiding direct test-set leakage during preprocessing and feature selection.

The sequence was:

KDDTrain+ → Train/Test Split

followed by:

Train → Encoding/Scaling → RF Feature Importance

and:

RF Ranking → Top 30

The selected feature names were then applied to both the training and held-out test representations. The test labels were not used to calculate feature importance.

The scaling procedure was also training-dependent: the StandardScaler was fitted on training data and subsequently applied to test data.

Nevertheless, the very high performance warrants additional verification. NSL-KDD is a benchmark dataset with a structured distribution, and unusually high results can arise from dataset characteristics, feature distributions, preprocessing choices or differences in evaluation protocols. Recent research has similarly emphasized that NSL-KDD results can vary substantially according to preprocessing, feature-selection strategy, normalization and classifier configuration (Umar et al., 2025).

The present study therefore avoids interpreting the 99.83% result as evidence of equivalent performance on contemporary operational network traffic.

4.5 Comparison with Previous Studies

Comparison with previous NSL-KDD studies should be interpreted cautiously because differences in preprocessing, data partitioning, feature selection, class definitions, and classifiers can substantially affect performance.

Umar et al. (2025) reported 99.86% accuracy using Random Forest, while Anagha et al. (2025) reported F1-scores of 98.7–98.9% using SHAP- and correlation-based feature selection. These results provide useful context but are not directly comparable due to methodological differences.

The 99.83% accuracy obtained in this study is therefore reported specifically for the stated KDDTrain+ 80:20 experimental protocol and should not be interpreted as a universal NSL-KDD performance level.

4.6 Dimensionality Reduction and Potential Efficiency

The principal demonstrated reduction was from 121 encoded features to 30 selected features. This removes 91 model inputs and reduces the feature representation by 75.21%.

The reduction may potentially lower the amount of data processed by subsequent models. However, because training time, inference time and memory consumption were not measured, the study does not claim that computational efficiency was experimentally improved.

A future efficiency experiment should measure both training and prediction time under identical hardware and software conditions for the 121-feature and 30-feature representations.

5. LIMITATIONS

Several limitations should be considered. First, the study uses the NSL-KDD benchmark rather than contemporary network traffic, limiting generalisation to modern environments.

Second, an 80:20 stratified split of KDDTrain+ was used instead of the standard KDDTrain+/KDDTest+ separation, limiting direct comparison with studies using the official test set.

Third, a complete 121-feature baseline under the same preprocessing configuration is not yet available; therefore, claims about performance preservation after feature selection remain preliminary.

Fourth, computational efficiency was not directly measured, so the results demonstrate dimensionality reduction rather than confirmed reductions in training time, inference time, or memory.

Finally, feature-selection stability was not evaluated across repeated partitions or independent datasets, meaning the selected features may partly depend on the specific KDDTrain+ split.

6. CONCLUSION

This study investigated Random Forest feature importance for feature selection in network intrusion detection using the NSL-KDD KDDTrain+ dataset. After removing the difficulty attribute and one-hot encoding, 41 predictive attributes produced 121 features. A stratified 80:20 split was used, with the top 30 features selected from the training data, reducing dimensionality by 75.21%.

The optimized Random Forest achieved 99.83% accuracy, 99.93% precision, 99.70% recall, 99.82% F1-score, and 0.0594% false-positive rate. These results demonstrate that Random Forest

feature importance can identify a compact and informative feature subset under the stated experimental conditions.

However, computational gains were not measured, and the full 121-feature baseline requires rerunning under the same preprocessing configuration. Future work should therefore compare full and reduced representations, measure computational efficiency, perform repeated validation, and evaluate contemporary network-traffic datasets.

5. REFERENCES

- A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.
- A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 20, 2019, doi: 10.1186/s42400-019-0038-7.
- G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014, doi: 10.1016/j.compeleceng.2013.11.024.
- L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, Ottawa, Canada, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.
- M. A. Umar, Z. Chen, K. Shuaib, and Y. Liu, "Effects of feature selection and normalization on network intrusion detection," *Data Science and Management*, vol. 8, no. 1, pp. 23–39, 2025, doi: 10.1016/j.dsm.2024.08.001.
- A. S. Anagha, C. Thomas, and N. Balakrishnan, "Optimized intrusion predictions through feature selection methods," *Computers & Security*, vol. 157, Art. no. 104541, 2025, doi: 10.1016/j.cose.2025.104541.
- E. M. Oluwabenga, T. Kolajo, and J. A. Babatunde, "Development and evaluation of a hybrid machine learning-based intrusion detection system using NSL-KDD dataset," *UMYU Scientifica*, vol. 3, no. 3, pp. 277–283, 2024, doi: 10.56919/2433.030.