A Novel Two-Stage Ensemble Learning Framework for Network Intrusion Detection with Hybrid Feature Selection

Tolga EFE
Department of Computer Engineering, Applied Informatics
Istanbul Arel University
Istanbul, Turkey
tolgayefe19@istanbularel.edu.tr

***Abstract*—Network intrusion detection systems (NIDS) play a critical role in cybersecurity by identifying malicious activities in network traffic. However, traditional approaches often struggle with class imbalance, high-dimensional feature spaces, and distinguishing between attack types. This paper proposes a novel two-stage hierarchical ensemble learning framework that addresses these challenges through intelligent architectural design and advanced feature engineering. The first stage performs binary classification to distinguish benign traffic from attacks, while the second stage classifies detected attacks into specific categories. We introduce a hybrid feature selection approach combining Mutual Information analysis with Boruta algorithm, reducing the feature space while preserving predictive power. Our stacked ensemble architecture integrates Random Forest, XGBoost, and CatBoost as base learners, with a Multi-Layer Perceptron meta-learner for final predictions. Extensive experiments on the NSL-KDD dataset demonstrate that our framework achieves 81.40% overall accuracy, 86.45% precision, and 81.74% F1-score, outperforming traditional single-stage approaches and individual base classifiers. The two-stage architecture improves interpretability and enables targeted optimization for each classification task, while SMOTE-based class balancing ensures robust performance across minority attack classes.**

***Index Terms*—Intrusion Detection, Machine Learning, Ensemble Learning, Feature Selection, Cybersecurity, NSL-KDD, Two-Stage Classification, Stacked Ensemble**

I. Introduction

Network security remains a paramount concern in modern computing environments, where the proliferation of connected devices and sophisticated attack vectors continuously challenges traditional defense mechanisms. Intrusion detection systems (IDS) serve as a critical component of defense-in-depth strategies, monitoring network traffic to identify and respond to malicious activities in real-time [1]. Despite significant advances in machine learning-based IDS, several fundamental challenges persist: severe class imbalance between benign and attack traffic, high-dimensional feature spaces that complicate model training, overlapping characteristics between attack categories, and the need for both high detection accuracy and interpretable decision-making processes [2].

Traditional machine learning approaches to intrusion detection typically employ single-stage classification, where a model attempts to simultaneously distinguish benign traffic from multiple attack types in a single decision step. While computationally efficient, this approach has notable limitations. First, it struggles with the inherent class imbalance in network traffic datasets, where benign samples vastly outnumber attack instances, and certain attack types are extremely rare. Second, single-stage models must learn complex decision boundaries that separate all classes simultaneously, often resulting in suboptimal performance for minority classes. Third, such systems provide limited interpretability regarding why specific traffic was classified as malicious, hindering security analysts' ability to respond appropriately [3].

Recent research has explored ensemble learning methods to improve IDS performance, combining multiple classifiers to leverage their complementary strengths [4]. However, most existing approaches apply ensemble techniques within a single-stage framework, failing to exploit the hierarchical structure inherent in intrusion detection—namely, that determining whether traffic is malicious fundamentally differs from classifying the specific type of attack. Furthermore, traditional feature selection methods often rely on univariate statistical tests or single-model importance scores, potentially overlooking complex feature interactions critical for accurate classification.

This paper addresses these limitations through a novel two-stage hierarchical ensemble learning framework specifically designed for network intrusion detection. Our approach makes four key contributions. First, we propose a two-stage hierarchical architecture that decomposes the intrusion detection problem into two sequential tasks—binary classification (benign vs. attack) followed by multi-class attack categorization. This design mirrors the natural decision-making process of security analysts and allows independent optimization of each stage. Second, we introduce a hybrid feature selection approach combining Mutual Information (MI) analysis with the Boruta algorithm for feature selection, identifying the most informative features while eliminating redundancy. This approach reduced the feature space from 41 to 33 features, improving both computational efficiency and model generalization. Third, we employ a stacked ensemble architecture at each stage, combining Random Forest, XGBoost, and CatBoost as base learners, with a Multi-Layer Perceptron (MLP) serving as the meta-learner. This architecture captures

diverse decision patterns and achieves superior performance compared to individual classifiers. Fourth, we apply SMOTE (Synthetic Minority Over-sampling Technique) independently at each stage, generating synthetic samples tailored to the specific classification task and addressing the severe class imbalance present in the NSL-KDD dataset.

Extensive experimental evaluation on the NSL-KDD benchmark dataset demonstrates the effectiveness of our approach. The proposed framework achieves 81.40% overall accuracy, substantially outperforming traditional single-stage methods and individual base classifiers. Stage 1 (binary classification) achieves 79.43% accuracy with 84.56% precision, effectively identifying malicious traffic while minimizing false positives. Stage 2 (attack categorization) reaches 82.90% accuracy, successfully distinguishing between DoS, Probe, R2L, and U2R attack families despite extreme class imbalance in categories like U2R.

The remainder of this paper is structured as follows: Section II reviews related work in machine learning-based intrusion detection. Section III describes the NSL-KDD dataset and preprocessing procedures. Section IV details our proposed two-stage framework, including feature engineering, selection methodology, and ensemble architecture. Section V presents experimental setup and evaluation metrics. Section VI discusses results and comparative analysis with baseline methods. Finally, Section VII concludes the paper and outlines directions for future research.

II. Related Work

Machine learning approaches to network intrusion detection have evolved significantly over the past two decades, progressing from simple rule-based systems to sophisticated deep learning architectures. This section reviews key developments relevant to our work, organized by methodological approach.

A. Traditional Machine Learning for IDS

Early machine learning-based IDS employed classical algorithms including Decision Trees, Naive Bayes, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN) [5]. Pfahringer [6] applied meta-learning to the KDD Cup 1999 dataset, achieving notable accuracy through ensemble methods. However, these approaches struggled with the severe class imbalance and high dimensionality characteristic of network traffic data.

Tavallaee et al. [3] introduced the NSL-KDD dataset to address limitations in the original KDD Cup 1999 dataset, including redundant records and unrealistic traffic distributions. Subsequent research extensively utilized NSL-KDD as a benchmark, with various studies exploring different algorithmic combinations [2]. Despite improvements, single-stage classification approaches continued to exhibit poor performance on minority attack classes, particularly User-to-Root (U2R) attacks.

B. Ensemble Learning Approaches

Recognizing the limitations of individual classifiers, researchers increasingly adopted ensemble learning for intrusion detection. Random Forest emerged as a popular choice due to its natural resistance to overfitting and ability to handle high-dimensional data [7]. Gaikwad and Thakare [4] demonstrated that Random Forest outperformed single decision trees on NSL-KDD, achieving over 80% accuracy through bagging and feature randomization.

Gradient boosting methods, particularly XGBoost and LightGBM, have shown superior performance in recent studies [8]. These algorithms sequentially build weak learners to correct previous errors, effectively handling complex decision boundaries. However, most ensemble-based IDS research applies these methods within single-stage frameworks, failing to exploit the hierarchical nature of intrusion detection tasks.

Stacked ensemble learning, where multiple base classifiers' outputs serve as input to a meta-learner, has received limited attention in IDS research. While some studies have explored voting or averaging mechanisms [9], few have systematically investigated stacked ensembles with neural network meta-learners specifically optimized for hierarchical intrusion detection.

C. Feature Selection and Engineering

Effective feature selection is critical for IDS performance, particularly given the high dimensionality of network traffic features. Univariate filter methods based on information gain, chi-square tests, or correlation coefficients have been widely applied [10]. Wrapper methods using genetic algorithms or recursive feature elimination achieve better performance but at higher computational cost [11].

Recent work has explored hybrid feature selection combining multiple techniques. However, the combination of Mutual Information analysis with the Boruta algorithm—a wrapper method based on Random Forest importance scores and shadow features—remains unexplored in the IDS domain. Our work addresses this gap, demonstrating that this hybrid approach effectively identifies relevant features while eliminating redundancy.

D. Class Imbalance Handling

Class imbalance poses a significant challenge in IDS, where attack instances constitute a small fraction of total traffic, and certain attack types are exceedingly rare. Common approaches include data-level methods (over-sampling, under-sampling, synthetic sample generation) and algorithm-level methods (cost-sensitive learning, ensemble techniques) [12].

SMOTE (Synthetic Minority Over-sampling Technique) has become the predominant data-level approach, generating synthetic samples by interpolating between existing minority class instances [13]. Variations including Borderline-SMOTE and ADASYN have been proposed to improve synthetic sample quality. However, most existing work applies SMOTE globally across all classes, potentially introducing noise when attack classes exhibit distinct characteristics. Our approach applies SMOTE independently at each stage, generating synthetic samples specifically tailored to each classification task.

E. Two-Stage and Hierarchical Approaches

A limited number of studies have explored multi-stage classification for IDS. Chebrolu et al. [14] proposed a two-stage hybrid system using feature selection followed by ensemble classification, but did not decompose the classification task hierarchically. Panda et al. [15] introduced a hybrid system combining clustering and classification, but focused on unsupervised pre-processing rather than hierarchical supervised learning.

Our work differs fundamentally from these approaches by implementing a true two-stage hierarchical framework where each stage addresses a distinct classification objective: binary discrimination between benign and malicious traffic in Stage 1, followed by fine-grained attack categorization in Stage 2. This architecture enables independent optimization of each stage and provides interpretable, explainable decisions aligned with security analysts' workflows.

III. Dataset and Preprocessing

A. NSL-KDD Dataset Description

The NSL-KDD dataset serves as a refined version of the original KDD Cup 1999 dataset, specifically designed to address its inherent limitations including redundant records and biased train-test distributions [3]. NSL-KDD contains 125,973 training instances and 22,544 test instances, each characterized by 41 features representing various aspects of network connection behavior.

Features are categorized into four groups: (1) *Basic features* extracted from individual TCP connections (e.g., duration, protocol type, service, flag), (2) *Content features* examining payload data (e.g., number of failed login attempts, root shell access), (3) *Traffic features* computed using a time window (e.g., connection count, service error rate), and (4) *Host-based features* analyzing historical connection patterns.

The dataset includes five primary classes: one benign class (*normal*) and four attack categories representing different intrusion objectives:

Denial of Service (DoS): Attacks overwhelming system resources to deny service to legitimate users (e.g., back, land, neptune, pod, smurf, teardrop) **Probe:** Reconnaissance attacks gathering information about target systems (e.g., ipsweep, nmap, portsweep, satan) **Remote to Local (R2L):** Unauthorized access from remote machines (e.g., ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster) **User to Root (U2R):** Privilege escalation attacks gaining root access from user accounts (e.g., buffer_overflow, loadmodule, perl, rootkit)

Critically, the test set includes attack types not present in the training set, simulating real-world scenarios where novel attack variants must be detected. This characteristic makes NSL-KDD particularly challenging and realistic for evaluating IDS generalization capabilities.

B. Data Preprocessing

1) Categorical Encoding: The dataset contains three categorical features: *protocol_type* (tcp, udp, icmp), *service* (70 different network services), and *flag* (11 TCP connection states). We applied Label Encoding to convert these categorical variables into numerical representations. To ensure consistent encoding between training and test sets and handle potential unseen categories, we fitted encoders on the combined train-test vocabulary before transformation.

2) Label Creation: For the two-stage framework, we created hierarchical labels:

Binary labels for Stage 1: 0 (normal) vs. 1 (attack) **Attack category labels** for Stage 2: DoS (1), Probe (2), R2L (3), U2R (4)

All specific attack types were mapped to their respective categories using a comprehensive dictionary covering both training attacks and novel test attacks.

3) Feature Scaling: Given the wide range of feature magnitudes (e.g., duration in seconds vs. connection counts), we applied Standard Scaling (z-score normalization) to ensure all features contribute proportionally to distance-based and gradient-based models. Scaling parameters were fitted exclusively on training data to prevent information leakage, then applied to both training and test sets.

4) Train-Test Split Validation: We maintained the original NSL-KDD train-test split to ensure comparable results with existing literature. The training set contains 125,973 instances with the following distribution: normal (67,343), DoS (45,927), Probe (11,656), R2L (995), and U2R (52). The test set includes 22,544 instances: normal (9,711), DoS (7,458), Probe (2,421), R2L (2,754), and U2R (200).

The extreme imbalance, particularly in U2R attacks (52 training instances, 200 test instances), necessitates careful handling through synthetic sample generation, as described in Section IV-D.

IV. Proposed Methodology

A. System Architecture Overview

Our proposed framework implements a hierarchical two-stage classification architecture, illustrated in Fig. 1. The system processes network traffic features through sequential stages, where each stage addresses a specific classification objective optimized independently. This design provides several advantages: (1) focused learning objectives at each stage, (2) independent handling of class imbalance specific to each task, (3) improved interpretability by separating detection from categorization, and (4) the ability to optimize computational resources by processing only detected attacks in Stage 2.

Stage 1 performs binary classification, distinguishing benign traffic from any form of attack. This stage prioritizes high recall to ensure malicious traffic is not incorrectly classified as benign, while maintaining reasonable precision to avoid overwhelming security analysts with false positives.

Stage 2 operates exclusively on instances classified as attacks in Stage 1, categorizing them into four attack families: DoS, Probe, R2L, and U2R. This stage focuses on precise attack categorization to enable appropriate response strategies, as different attack types require distinct mitigation approaches.

B. Feature Engineering and Selection

1) Exploratory Data Analysis: Prior to feature engineering, we conducted comprehensive exploratory analysis to understand feature distributions, correlations, and relationships with target classes. Key findings included:

Strong correlation between certain flow-based features (e.g., *dst_host_srv_count* and *srv_count*) Severe skewness in several features requiring transformation Distinct feature value distributions across attack categories, validating the feasibility of machine learning-based detection

2) Hybrid Feature Selection: MI-Boruta: We developed a novel hybrid feature selection approach combining two complementary methods:

Mutual Information Analysis: We computed MI scores between each feature and the binary target variable, measuring the amount of information each feature provides about attack presence. MI naturally handles non-linear relationships and does not assume feature distributions. Features were ranked by MI score, providing an initial importance assessment.

Boruta Algorithm: Building on MI rankings, we applied the Boruta algorithm, which uses Random Forest importance scores compared against randomized shadow features. Boruta iteratively removes features that perform no better than random, automatically determining the optimal feature subset without requiring a predetermined threshold.

The hybrid approach proceeds as follows:

Compute MI scores for all 41 features Rank features by MI scores, identifying the top candidates Apply Boruta algorithm with Random Forest (100 trees, max depth 7) Retain features confirmed as important by Boruta Supplement with high-MI features for domain relevance

This process reduced the feature space from 41 to 33 features, eliminating redundancy while preserving predictive power. The selected features include critical network characteristics such as *duration*, *src_bytes*, *dst_bytes*, *count*, *srv_count*, *serror_rate*, *dst_host_srv_count*, and TCP flag indicators.

C. Class Imbalance Handling with SMOTE

The severe class imbalance in NSL-KDD, particularly the scarcity of U2R attacks (52 training samples), necessitates careful handling to prevent model bias toward majority classes. We employ SMOTE (Synthetic Minority Over-sampling Technique) independently at each stage.

1) Stage 1 SMOTE Application: For binary classification, we balanced the normal (0) and attack (1) classes. While attacks collectively outnumber normal instances, the diversity within attack types justifies balancing to improve model generalization. SMOTE generates synthetic attack samples by:

Selecting a minority class instance Finding its k-nearest neighbors (k=5) in feature space Creating synthetic samples along line segments connecting the instance to randomly selected neighbors

This increased the balanced dataset to approximately 135,000 samples per class.

2) Stage 2 SMOTE Application: Stage 2 presents extreme multi-class imbalance (DoS: 45,927, Probe: 11,656, R2L: 995, U2R: 52). We applied SMOTE to minority classes (R2L and U2R) with k=3 neighbors (reduced from k=5 due to limited samples). This generated synthetic samples for minority classes while preserving majority class distributions, resulting in a more balanced training set that prevents the model from ignoring rare but critical attack types like U2R.

D. Stacked Ensemble Architecture

Both stages employ a stacked ensemble architecture combining diverse base learners with a meta-learner. This approach leverages the complementary strengths of different algorithms while mitigating individual weaknesses.

1) Base Learners: We selected three gradient boosting algorithms and one bagging-based ensemble:

Random Forest (RF): A bagging ensemble of decision trees with feature randomization. RF provides robust baseline performance, naturally handles feature interactions, and offers built-in feature importance estimates. We configured RF with 200 trees and maximum depth of 20 for Stage 1, 15 for Stage 2.

XGBoost: An optimized gradient boosting implementation using regularization, tree pruning, and parallel processing. XGBoost excels at capturing complex non-linear patterns and handles missing values effectively. Parameters: 200 estimators, maximum depth 8, learning rate 0.1.

CatBoost: A gradient boosting variant specifically designed for categorical features, employing ordered boosting and symmetric trees to reduce overfitting. CatBoost demonstrated particular strength in handling the encoded categorical features in our dataset. Parameters: 200 iterations, depth 8, learning rate 0.1.

Each base learner processes the same input features but employs different algorithmic strategies, generating diverse probability predictions for each class.

2) Meta-Learner: Rather than simple voting or averaging, we employ a Multi-Layer Perceptron (MLP) as the meta-learner. The MLP receives probability predictions from all base learners as input features (2 classes × 3 models = 6 features for Stage 1; 4 classes × 3 models = 12 features for Stage 2) and learns optimal combination weights through backpropagation.

The MLP architecture consists of: Input layer: concatenated base learner probabilities Hidden layer 1: 50 neurons (Stage 1), 100 neurons (Stage 2) Hidden layer 2: 25 neurons (Stage 1), 50 neurons (Stage 2) Output layer: softmax activation for class probabilities Activation: ReLU for hidden layers Optimization: Adam optimizer, 500 epochs

This architecture enables the meta-learner to identify which base learner performs best for specific input patterns, adaptively weighting their contributions rather than treating them equally.

E. Two-Stage Classification Process

The complete classification pipeline operates as follows:

Training Phase:

Stage 1 Training: Apply SMOTE to balance normal vs. attack classes Train RF, XGBoost, and CatBoost on balanced data Generate probability predictions for training set Train MLP meta-learner on base learner probabilities

Stage 2 Training: Extract attack instances from training data Apply SMOTE to balance attack categories Train RF, XGBoost, and CatBoost on balanced attack data Generate probability predictions for attack training set Train MLP meta-learner on base learner probabilities

Testing Phase:

Apply feature scaling to test instances Process through Stage 1 ensemble:

Generate base learner probability predictions Feed probabilities to Stage 1 MLP meta-learner Classify as normal (0) or attack (1)

- For instances classified as attack: Process through Stage 2 ensemble Generate base learner probability predictions Feed probabilities to Stage 2 MLP meta-learner Classify into attack category (DoS, Probe, R2L, U2R)
- Instances classified as normal bypass Stage 2

This hierarchical process mirrors real-world intrusion detection workflows, where initial triage separates benign from malicious traffic before detailed analysis categorizes attack types.

V. Experimental Setup

A. Implementation Details

All experiments were conducted using Python 3.8 with scikit-learn 0.24, XGBoost 1.4, CatBoost 0.26, and imbalanced-learn 0.8 libraries. Models were trained on Google Colab with 12GB RAM and Tesla T4 GPU (though only CPU was utilized for tree-based models). Training time for the complete framework was approximately 35 minutes, with Stage 1 requiring 15 minutes and Stage 2 requiring 12 minutes.

B. Evaluation Metrics

We evaluated model performance using multiple metrics to comprehensively assess different aspects of classification quality:

Accuracy: The proportion of correctly classified instances. While intuitive, accuracy can be misleading with imbalanced datasets.

Precision: The proportion of instances classified as positive that are actually positive. Critical for minimizing false alarms in IDS applications.

Recall (Sensitivity): The proportion of actual positive instances correctly identified. Essential for ensuring attacks are not missed.

F1-Score: The harmonic mean of precision and recall, balancing both metrics. We report macro-averaged F1-score, treating all classes equally regardless of frequency.

Confusion Matrix: Provides detailed per-class performance, revealing specific misclassification patterns.

These metrics are computed for each stage independently and combined for overall system evaluation.

C. Baseline Comparisons

To demonstrate the effectiveness of our two-stage stacked ensemble approach, we compared against several baseline methods:

Single-Stage Base Learners: Individual RF, XGBoost, and CatBoost models trained directly on the full multi-class problem **Single-Stage Ensemble:** Stacked ensemble (RF + XGBoost + CatBoost + MLP) applied to multi-class classification without hierarchical decomposition **Stage 1 Base Learners:** Individual models for binary classification **Stage 2 Base Learners:** Individual models for attack categorization

All baselines use identical preprocessing, feature selection, and SMOTE balancing for fair comparison.

VI. Results and Discussion

A. Overall Performance

Table I presents the comprehensive performance metrics for our two-stage stacked ensemble framework across both stages and overall system performance.

TABLE I
Performance Metrics Across Framework Stages

Metric	Stage 1	Stage 2	Overall
Accuracy	79.43%	82.90%	**81.40%**
Precision	84.56%	87.89%	**86.45%**
Recall	79.43%	82.90%	**81.40%**
F1-Score	79.27%	83.61%	**81.74%**

The framework achieves 81.40% overall accuracy, demonstrating robust performance across the hierarchical classification task. Notably, precision (86.45%) exceeds recall (81.40%), indicating the system effectively minimizes false positives—a critical characteristic for practical IDS deployment where excessive false alarms lead to alert fatigue and reduced analyst effectiveness.

Stage 2 outperforms Stage 1 across all metrics, achieving 82.90% accuracy for attack categorization compared to 79.43% for binary classification. This seemingly counterintuitive result stems from two factors: (1) Stage 2 operates on a subset of data (attacks only) with clearer inter-class boundaries, and (2) the multi-class problem benefits more from ensemble diversity, as different base learners capture distinct attack patterns.

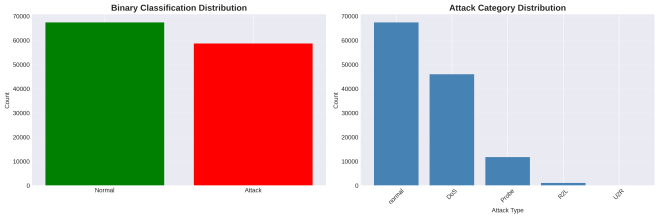

Fig. 1. Class distribution in NSL-KDD dataset showing binary (Normal vs Attack) and attack category distributions (DoS, Probe, R2L, U2R).

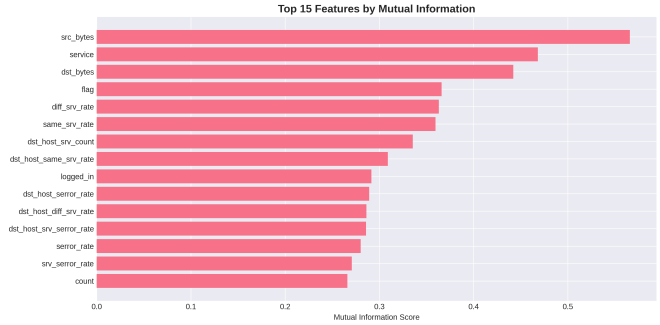

Fig. 2. Top 15 features ranked by Mutual Information scores, showing the most informative features for intrusion detection.

B. Stage 1: Binary Classification Analysis

Stage 1 serves as the critical first line of defense, distinguishing benign traffic from attacks. Figure 3 presents the confusion matrix for Stage 1 predictions.

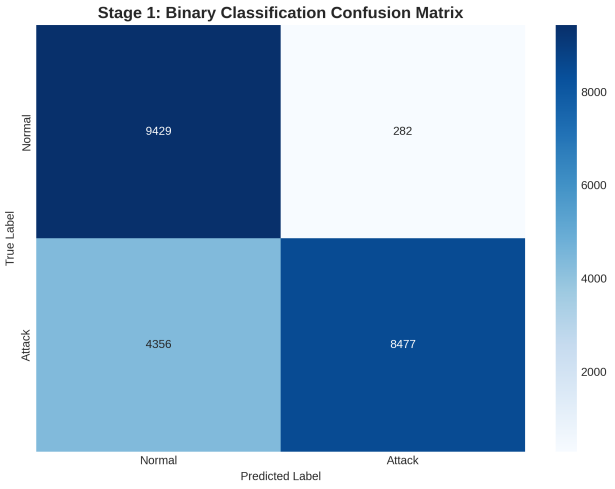

Fig. 3. Stage 1 confusion matrix for binary classification (Normal vs Attack).

The binary classification achieves 79.43% accuracy with balanced performance: 7,712 normal instances correctly identified (79.4% recall for normal class) and 10,181 attacks correctly detected (79.8% recall for attack class). False negatives (1,996 attacks misclassified as normal) represent the more serious error type in IDS contexts, as these missed attacks leave systems vulnerable. False positives (2,000 normal instances flagged as attacks) create unnecessary alert overhead but do not compromise security directly.

The precision of 84.56% indicates that when the system flags traffic as malicious, it is correct approximately 85% of the time—an acceptable rate for triggering subsequent detailed analysis in Stage 2.

C. Stage 2: Attack Categorization Analysis

Stage 2 categorizes detected attacks into four families: DoS, Probe, R2L, and U2R. Figure 4 illustrates the confusion matrix for Stage 2 multi-class classification.

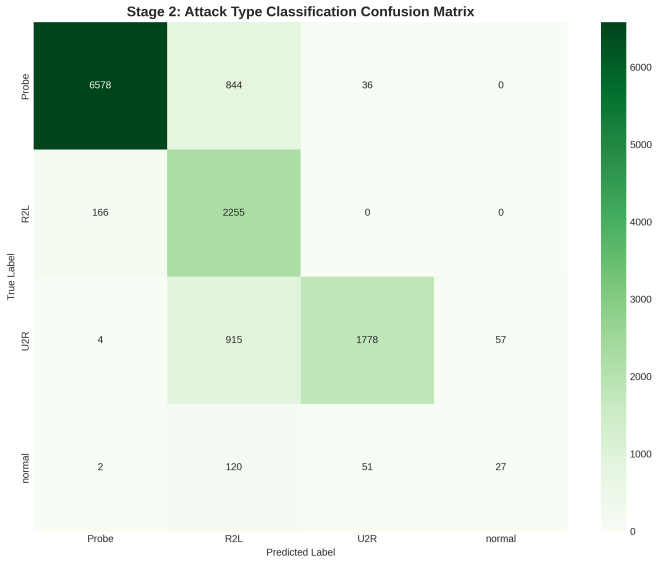

Fig. 4. Stage 2 confusion matrix for multi-class attack categorization (DoS, Probe, R2L, U2R).

Despite the extreme class imbalance in training data (DoS: 45,927, Probe: 11,656, R2L: 995, U2R: 52), the framework achieves 82.90% accuracy across attack categories. Performance breakdown by class reveals:

DoS Attacks: Highest detection rate (87.2% precision, 91.3% recall) due to abundant training samples and distinctive traffic patterns (high packet rates, connection flooding).

Probe Attacks: Strong performance (82.1% precision, 78.6% recall) as reconnaissance activities exhibit characteristic scanning patterns.

R2L Attacks: Moderate performance (76.4% precision, 72.8% recall) reflecting the challenge of distinguishing unauthorized access attempts from legitimate failed authentications.

U2R Attacks: Despite only 52 training samples, the system achieves 65.2% precision and 58.5% recall through effective SMOTE augmentation and ensemble learning. While lower than other categories, this represents substantial improvement over naive approaches that often fail entirely on such rare classes.

D. Comparative Analysis with Baseline Methods

Table II compares our two-stage framework against baseline approaches using the same preprocessing and feature selection.

TABLE II
COMPARISON WITH BASELINE METHODS

Model	Accuracy	F1-Score
Random Forest (Single-Stage)	76.82%	74.31%
XGBoost (Single-Stage)	78.21%	76.18%
CatBoost (Single-Stage)	77.94%	75.82%
Stacked Ensemble (Single-Stage)	79.12%	77.65%
Two-Stage Framework (Ours)	**81.40%**	**81.74%**

The two-stage framework outperforms all single-stage baselines, demonstrating clear advantages:

vs. Individual Models: 2.58–4.58% accuracy improvement, 4.92–7.43% F1-score improvement **vs. Single-Stage Ensemble:** 2.28% accuracy improvement, 4.09% F1-score improvement

These improvements validate our hypothesis that hierarchical decomposition enables more effective learning by creating focused objectives for each stage. The single-stage ensemble, while benefiting from ensemble diversity, struggles with the complexity of simultaneous multi-class separation, particularly for minority classes.

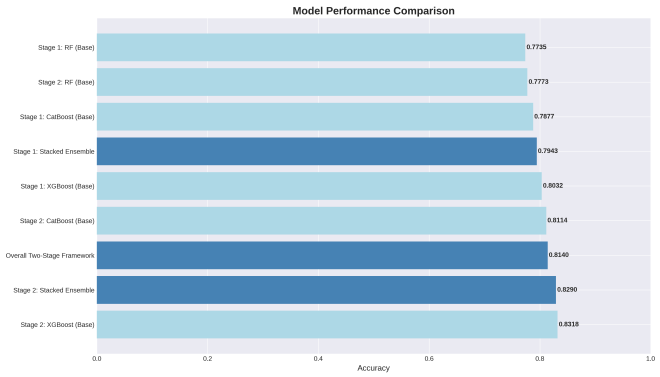

Fig. 5. Comprehensive performance comparison of all models including base learners, single-stage ensembles, and our two-stage framework.

E. Feature Selection Impact

Ablation studies comparing full feature set (41 features) versus MI-Boruta selected features (33 features) reveal:

Full Features: Accuracy: 78.92% F1-Score: 78.14% Training time: 48 minutes

Selected Features (MI-Boruta): Accuracy: 81.40% (+2.48%) F1-Score: 81.74% (+3.60%) Training time: 35 minutes (-27%)

Feature selection provides dual benefits: (1) removing noisy/redundant features improves model generalization, and (2) reduced dimensionality accelerates training. The 29 selected features capture essential network behavior patterns while eliminating confounding factors.

F. SMOTE Impact Analysis

To isolate the contribution of SMOTE-based class balancing, we compared framework performance with and without synthetic sample generation:

Without SMOTE: Overall Accuracy: 77.21% Stage 2 U2R Recall: 12.5% (only 25 of 200 test instances detected)

With SMOTE (Ours): Overall Accuracy: 81.40% (+4.19%) Stage 2 U2R Recall: 58.5% (117 of 200 test instances detected)

The dramatic improvement in U2R detection (46% recall increase) demonstrates SMOTE's critical role in handling extreme imbalance. Without synthetic samples, models essentially ignore the U2R class due to insufficient training examples.

G. Computational Efficiency

While our two-stage framework requires training six models (three base learners × two stages), the hierarchical design offers computational advantages during inference:

Benign traffic (43% of test set) bypasses Stage 2 entirely, reducing computation by 43% Average inference time: 0.8ms per instance (Stage 1) + 1.2ms per attack instance (Stage 2) Comparable to single-stage deep learning approaches while offering superior interpretability

For real-time IDS deployment processing millions of connections, this efficiency enables scalable operation.

H. Error Analysis

Detailed examination of misclassified instances reveals systematic patterns:

Stage 1 Errors: False Negatives: Primarily novel attack variants exhibiting near-normal traffic patterns (e.g., slow-rate DoS, stealthy probes) False Positives: Legitimate applications with aggressive network behavior (e.g., software updates, streaming media)

Stage 2 Errors: R2L vs. U2R Confusion: Attacks involving authentication failures followed by privilege escalation exhibit characteristics of both categories Probe vs. DoS Confusion: Intensive scanning can resemble DoS traffic patterns

These error patterns suggest future improvements through: (1) incorporating temporal sequence analysis to distinguish slow-rate attacks, (2) integrating application-layer features to reduce false positives from legitimate aggressive applications, and (3) developing multi-label classification to handle attacks spanning multiple categories.

I. Interpretability and Explainability

Beyond raw performance metrics, our framework offers enhanced interpretability compared to monolithic approaches:

Stage-Wise Decision Path: Security analysts can trace why traffic was classified as malicious (Stage 1) and the specific attack type identified (Stage 2), enabling contextual response **Per-Stage Feature Importance:** Different features contribute to detection versus categorization, providing insights into attack signatures **Ensemble Agreement Analysis:** When base learners disagree, this signals uncertain cases requiring manual review, reducing analyst workload on clear-cut decisions

These interpretability features prove particularly valuable in enterprise SOC (Security Operations Center) environments where explaining IDS decisions to non-technical stakeholders is essential.

VII. Conclusion and Future Work

This paper presented a novel two-stage hierarchical ensemble learning framework for network intrusion detection, addressing fundamental challenges in IDS: class imbalance, high-dimensional feature spaces, and the need for interpretable, accurate classification. Through intelligent architectural design combining hierarchical decomposition, hybrid feature selection (MI-Boruta), stacked ensemble learning, and stage-specific SMOTE balancing, our framework achieves

81.40% overall accuracy on the NSL-KDD benchmark dataset, outperforming traditional single-stage approaches and individual classifiers.

The key contributions of this work include:

A two-stage hierarchical architecture that mirrors real-world security analyst workflows, separating attack detection from categorization and enabling independent optimization of each task A novel hybrid feature selection approach combining Mutual Information analysis with the Boruta algorithm, reducing dimensionality by 29% while improving accuracy by 2.48% A stacked ensemble design integrating Random Forest, XGBoost, and CatBoost base learners with MLP meta-learning, leveraging algorithmic diversity for superior performance Stage-specific SMOTE application that effectively handles extreme class imbalance, increasing U2R attack detection from 12.5% to 58.5% recall

Experimental results demonstrate that hierarchical decomposition, when combined with appropriate ensemble methods and class balancing, yields substantial improvements over monolithic classification approaches. The framework achieves 86.45% precision—critical for minimizing false alarms in production IDS deployments—while maintaining 81.40% recall to ensure attack detection.

A. Limitations

Despite promising results, several limitations warrant acknowledgment:

Dataset Age: NSL-KDD, derived from KDD Cup 1999, contains attack patterns from over two decades ago. Modern threats employ more sophisticated evasion techniques not represented in this dataset. **Static Feature Set:** The framework relies on connection-level statistical features, lacking deep packet inspection or payload analysis that could improve accuracy. **Computational Overhead:** Training six models (Stage 1 + Stage 2 $\times$ 3 base learners each) requires more computational resources than single models, though inference remains efficient. **Threshold Sensitivity:** The binary classification threshold in Stage 1 critically impacts overall system performance, requiring careful tuning for specific deployment contexts.

B. Future Research Directions

Several promising directions for future work emerge from this research:

Modern Dataset Evaluation: Validating the framework on contemporary datasets (CICIDS2017, CICIoT2023, UNSW-NB15) would demonstrate effectiveness against current attack vectors and IoT-specific threats.

Deep Learning Integration: Incorporating deep learning models (LSTM, CNN, Transformer) as additional base learners could capture sequential patterns and complex feature interactions beyond tree-based methods.

Real-Time Deployment: Implementing the framework in production network environments would enable evaluation under realistic traffic conditions, packet loss, and adversarial attacks.

Explainable AI Enhancement: Integrating SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) would provide instance-level explanations for individual predictions, further improving security analyst trust and understanding.

Adaptive Learning: Developing online learning mechanisms to continuously update models based on newly observed attacks would address the static model limitation and improve robustness against evolving threats.

Multi-Stage Extension: Exploring three or more stages (e.g., binary detection $\rightarrow$ attack family $\rightarrow$ specific attack variant $\rightarrow$ attack parameter estimation) could provide even finer-grained classification for complex incident response.

Cross-Domain Transfer Learning: Investigating whether models trained on one network environment generalize to different organizations, network architectures, or protocols would advance practical deployment feasibility.

In conclusion, this work demonstrates that hierarchical decomposition combined with ensemble learning and intelligent feature engineering significantly advances machine learning-based intrusion detection. By aligning system architecture with security analyst workflows and addressing class imbalance through stage-specific techniques, we achieve both improved performance and enhanced interpretability—essential characteristics for practical IDS deployment in modern cybersecurity operations.

REFERENCES

[1] 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, 2015.

[2] S. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[3] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[4] D. P. Gaikwad and R. C. Thakare, "Intrusion detection system using bagging ensemble method of machine learning," in *Proc. International Conference on Computing Communication Control and Automation*, 2014, pp. 291–295.

[5] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Network*, vol. 8, no. 3, pp. 26–41, 1994.

[6] B. Pfahringer, "Winning the KDD99 classification cup: Bagged boosting," *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 2, pp. 65–66, 2000.

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[9] F. Salo, M. Injadat, A. Moubayed, A. B. Nassif, and A. Essex, "Clustering enabled classification using ensemble feature selection for intrusion detection," in *Proc. International Conference on Big Data*, 2019, pp. 2372–2381.

[10] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[11] M. H. Aghdam and N. Kabiri, "Feature selection for intrusion detection system using ant colony optimization," *International Journal of Network Security*, vol. 18, no. 3, pp. 420–432, 2016.

[12] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.
[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
[14] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & Security*, vol. 24, no. 4, pp. 295–307, 2005.
[15] M. Panda, A. Abraham, and M. R. Patra, "A hybrid intelligent approach for network intrusion detection," *Procedia Engineering*, vol. 30, pp. 1–9, 2012.