NOTATION TRENDPLUS PUBLISHING
WHERE IDEAS MEET IMPACT

Volume-1, Issue 1, June 2026, Page No- 155-173
Journal of Machine Learning Innovations and Artificial Intelligence Horizons (JMLIAIH)

An Ensemble Voting Classifier Framework for High-Accuracy Intrusion Detection in Dynamic Network Environments

Amitava Biswas

Head of the Department, Department of Computer Science, Behala College

DOI: https://doi.org/10.70903/jmliaih.2026.1109

Article Received: 25-03-2026

Revised Version: 01-06-2026

Accepted: 17-06-2026

*Corresponding Author

Amitava Biswas

E-Mail Id: abiswas.seminar@gmail.com

Copyright: © The Author(s), 2026. Published by Notation Trendplus Publishing. This is an Open Access article, distributed under the terms of the Creative Commons Attribution 4.0 License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Abstract: We propose an ensemble machine learning framework for intrusion detection in dynamic network environments is presented in this paper. The rapidly evolving landscape of cyber threats demands detection systems that can adapt to novel attack patterns while maintaining high accuracy. We propose a multi-layered processing pipeline that begins with the extraction of ten high-dimensional network traffic features, including source and destination IP addresses, packet size, flow duration, and protocol type. Categorical attributes undergo label encoding, while continuous numerical features are normalized to a uniform range of zero to one via Min-Max scaling, thereby ensuring computational stability during model training. The core of our detection system is an Ensemble Voting Classifier that aggregates predictions from five distinct base learners: Decision Tree, Random Forest, Gradient Boosting Machine, and Extreme Gradient Boosting. Furthermore, a dedicated ensemble wrapper integrates these models through a hard voting strategy, where the final class label is selected based on the majority vote among the individual classifiers. This approach effectively combines the variance-reducing capability of bagging from Random Forest with the sequential error-correction strength of boosting methods. Binary Cross-Entropy serves as the loss function to guide parameter optimization during training, minimizing the discrepancy between predicted probabilities and true labels. The principal contribution of this work lies in demonstrating that a carefully constructed ensemble of heterogeneous tree-based models, operating on normalized and encoded features, yields superior generalization performance compared to individual classifiers. Moreover, the framework requires no complex architectural modifications, making it readily deployable in real-world network environments. We anticipate that this method will significantly enhance the accuracy and robustness of intrusion detection systems against both known and emerging cyber threats.

Keywords: Intrusion Detection System, Ensemble Voting Classifier, Network Security, Cybersecurity, Machine Learning

\

Introduction

The proliferation of sophisticated cyber-attacks targeting enterprise networks has rendered traditional signature-based intrusion detection systems increasingly inadequate [1]. Systems such as Snort and Suricata, which depend on predefined attack signatures, fail to generalize to novel or polymorphic threats [2]. This limitation has driven the adoption of anomaly-based detection methods supported by machine learning algorithms [3]. However, early classifiers such as Naive Bayes, K-Nearest Neighbors, and Support Vector Machines often suffer from high variance or overfitting when processing high-dimensional network traffic data [4]Consequently, recent research efforts have turned toward ensemble learning techniques that combine multiple weak learners to produce a robust and stable predictive model.

The transition from single-model to ensemble-based architectures represents a significant methodological advancement in cybersecurity [4]. Random Forest, a bagging-based ensemble of decision trees, reduces variance by averaging predictions across many decorrelated trees. On the other hand, boosting algorithms such as Gradient Boosting Machines and XGBoost iteratively correct errors made by predecessor models, thereby improving bias reduction. Despite these individual strengths, a single ensemble type often fails to handle the full diversity of attack vectors, including Denial of Service, brute force, and botnet communications. Hence, a hybrid strategy that integrates the complementary advantages of bagging and boosting through a voting mechanism has emerged as a promising direction.

Nevertheless, many existing ensemble frameworks for intrusion detection focus on homogeneous base learners or require extensive hyperparameter tuning to achieve acceptable performance [5]. Furthermore, the preprocessing of raw network traffic—namely, encoding categorical features and normalizing numerical attributes—is often underemphasized in these approaches despite its critical role in ensuring model convergence and stability. The Security Operations Center environment demands real-time detection capabilities combined with low false-positive rates, a requirement that many complex deep learning architectures fail to meet due to their computational overhead. Therefore, a lightweight yet accurate ensemble classifier that operates on carefully engineered features becomes essential for practical deployment in dynamic network environments [14]

The present paper introduces a novel ensemble voting classifier framework specifically designed for high-accuracy intrusion detection in such environments. Our core innovation lies in the integration of five heterogeneous base learners—Decision Tree, Random Forest, Gradient Boosting Machine, XGBoost, and a custom ensemble wrapper—combined through a hard voting mechanism that synthesizes individual model predictions into a single consensus output. This approach directly addresses the bias-variance tradeoff by simultaneously benefiting from the variance reduction of bagging and the bias reduction of boosting. In contrast to previous ensemble works that often rely on a single family of base models, our architecture explicitly diversifies the learning paradigms to capture a broader spectrum of traffic patterns. attack patterns. Moreover, the framework requires no architectural

modifications or deep neural network layers, making it computationally efficient and suitable for real-time threat detection in Security Operations Center workflows.

The principal contributions of this work are threefold. First, we propose a dedicated multi-step feature engineering pipeline that encodes categorical attributes via label encoding and normalizes continuous features via Min-Max scaling, ensuring that all base classifiers operate on a consistent numerical space. Second, we design an ensemble voting classifier that aggregates predictions from tree-based bagging and boosting models, demonstrating that the voting ensemble consistently outperforms each individual base learner across multiple evaluation metrics. Third, we validate the framework on a widely used benchmark dataset—specifically, the NSL-KDD dataset—and report its detection accuracy, precision, recall, and F1-score, thereby providing a reproducible baseline for future research. The remainder of this paper is organized as follows. Section 2 provides a review of related work on ensemble learning for intrusion detection. Section 3 details the feature engineering and traffic representation methodology. Section 4 describes the ensemble voting architecture for multi-vector threat detection. Section 5 outlines the experimental setup and evaluation metrics. Section 6 presents and analyzes the results. Section 7 discusses the implications and limitations of the proposed approach and outlines future research directions. Finally, Section 8 concludes the paper with a summary of findings.

Related Work

The field of intrusion detection has increasingly adopted machine learning approaches to overcome the limitations of traditional signature-based systems. Early data-driven methods applied individual classifiers such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) to network traffic features, but these models often exhibited high variance and sensitivity to hyperparameter settings. To address these weaknesses, researchers turned to ensemble learning, which combines multiple weak learners to produce a more robust and accurate predictive model.

Bagging and Random Forest for Intrusion Detection

Bagging, or Bootstrap Aggregating, is a foundational ensemble technique that reduces variance by training multiple base learners on bootstrapped subsets of the training data and averaging their predictions. The Random Forest (RF) algorithm extends this concept by building a collection of decorrelated collection of decision trees, which has proven particularly effective for high-dimensional network traffic data. For instance, the work by Zhang et al. demonstrated that a Random Forest-based intrusion detection system achieved high accuracy on the NSL-KDD dataset, effectively handling categorical features such as protocol type and service [6]. This variance-reduction property makes Random Forest a natural candidate for environments where false positive rates must be minimized. However, bagging-based ensembles can struggle with bias reduction, as each individual tree in the forest may have limited ability to learn complex, non-linear attack patterns.

Boosting and Gradient-Based Methods for Sequential Error Correction

In contrast to bagging, boosting algorithms such as Gradient Boosting Machines (GBM) and Extreme Gradient Boosting (XGBoost) focus on reducing bias by sequentially training models to correct the errors of their predecessors. These methods have shown strong performance in cybersecurity competitions and applied tasks, including cybersecurity threat detection. For example, Chen et al. applied XGBoost to network intrusion detection and found that it outperformed both SVM and Random Forest in terms of precision and recall when detecting minority attack classes [7]. The sequential nature of boosting allows these models to focus on hard-to-classify samples, thereby improving the detection rate for rare attack types such as User-to-Root (U2R) and Remote-to-Local (R2L) attacks. Nevertheless, boosting methods are prone to overfitting on noisy data, particularly when the dataset contains a significant number of anomalous but benign traffic flows.

Hybrid Ensemble Architectures for Security Operations Centers

Recognizing the complementary strengths of bagging and boosting, several studies have proposed hybrid ensemble architectures that integrate both families of models. These approaches typically employ either a stacking meta-learner or a voting mechanism to combine predictions. For example, Ahmad et al. introduced a stacked ensemble that trained a logistic regression meta-classifier on the outputs of Random Forest and XGBoost, achieving a higher F1-score than either base model alone [8]. Similarly, Kumar and Sharma proposed a soft voting ensemble of three tree-based classifiers for detecting botnet traffic, reporting improved generalization on unseen attack patterns [9]. These hybrid strategies leverage the bias reduction of boosting and the variance reduction of bagging within a single framework, making them well-suited for Security Operations Center (SOC) environments where detection accuracy and robustness are equally critical [10].

Feature Engineering and Preprocessing in Intrusion Detection

Despite the advances in model architecture, the importance of feature engineering and preprocessing is frequently underappreciated. Many existing studies apply standard normalization or one-hot encoding without considering the specific requirements of tree-based ensembles. However, as argued by Sarker et al., the choice of feature encoding can significantly impact the convergence and stability of boosting algorithms [11]. For example, label encoding preserves the ordinal relationship that tree-based models can exploit, whereas one-hot encoding can lead to an explosion in dimensionality. Min-Max normalization, which scales continuous features to a fixed range, is particularly beneficial for gradient-based methods such as GBM and XGBoost, as it prevents features with larger numerical ranges from dominating the gradient computation.

Comparison with the Proposed Framework

The ensemble voting classifier proposed in this paper differs from existing works in several key aspects. Unlike stacked ensembles that require an additional meta-classifier, our hard voting mechanism is computationally lightweight and avoids the risk of meta-overfitting. Furthermore, we explicitly incorporate both a dedicated ensemble wrapper and five heterogeneous base learners (Decision Tree, Random Forest, GBM, XGBoost, and the wrapper itself) to ensure maximum diversity in the voting pool. This diversity is critical for capturing a

wide range of attack signatures, from simple DoS floods to slow-rate, stealthy botnet communications. Our feature engineering pipeline—specifically, the combination of label encoding for categorical attributes and Min-Max normalization for continuous features—is designed to optimize the individual performance of each base learner before aggregation. While previous works have focused on either bagging or boosting in isolation or have applied complex meta-learning schemes, our framework offers a more direct and deployable solution that systematically balances the bias-variance tradeoff. The principal novelty of this work, therefore, lies in the integration of a heterogeneous ensemble with a carefully calibrated preprocessing phase, delivering a detection system that is both highly accurate and straightforward to implement in real-time network monitoring environments.

IIIFeature Engineering and Traffic Representation

Feature engineering constitutes a foundational step in any machine learning-based intrusion detection system, as the quality and representational fidelity of input features directly determine the upper bound of model performance. Raw network traffic data, captured in formats such as PCAP files or NetFlow records, typically contains a mix of categorical attributes (e.g., protocol type, service type) and continuous numerical measurements (e.g., packet size, flow duration). Without appropriate transformation, these heterogeneous data types cannot be directly consumed by tree-based ensemble classifiers, which operate exclusively on numerical input vectors. Hence, a systematic feature engineering pipeline that converts raw traffic records into a consistent, normalized numerical representation is essential.

Feature Extraction from Network Traffic

The initial phase of our pipeline involves extracting a set of ten high-dimensional features from raw network traffic flows. These features are selected based on their established relevance to intrusion detection in prior literature, covering both basic connection-level attributes and traffic-derived statistical measures. The complete feature set includes: source IP address, destination IP address, source port, destination port, protocol type (TCP, UDP, ICMP), packet size, flow duration, number of packets per flow, mean packet inter-arrival time, and service type (e.g., HTTP, FTP, SMTP). This feature set captures both the inherent characteristics of individual connections and the broader statistical patterns that distinguish normal traffic from various attack classes, such as Denial of Service (DoS) or probing attacks.

Categorical Feature Encoding

Among the extracted features, six are categorical in nature: source IP, destination IP, protocol type, service type, and flag (TCP connection state). Tree-based classifiers, such as Decision Trees and Random Forest, can theoretically handle categorical variables by implicitly learning splitting criteria based on set membership. However, in practice, most implementations—including scikit-learn's Random Forest and XGBoost—require numerical input. Therefore, we apply label encoding to each categorical attribute, mapping each distinct category to a unique integer. Label encoding is preferred over one-hot encoding in this context for two reasons: first, it preserves the ordinal relationships that tree-based models can exploit by treating the integer values as ordinal; second, it avoids the dramatic increase in feature dimensionality that would result from one-hot encoding, which would significantly increase computation time for the

ensemble. For example, the protocol type attribute, which has three categories (TCP, UDP, ICMP), would be encoded as 0, 1, and 2, respectively.

Numerical Feature Normalization

The remaining four features—packet size, flow duration, number of packets per flow, and mean packet inter-arrival time—are continuous numerical values with drastically different scales. For instance, packet size values typically range from 40 to 1500 bytes, while flow duration may span from milliseconds to several hours. If left unnormalized, features with larger numerical ranges would dominate the gradient computation in boosting algorithms like GBM and XGBoost, leading to slower convergence and suboptimal split points. To address this, we apply Min-Max normalization, which scales each continuous feature to a fixed range of $[0,1]$. The transformation is defined mathematically as:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (1)$$

where x represents the original feature value, and $x_{\min}$ and $x_{\max}$ are the minimum and maximum values observed in the training set for that feature. This normalization ensures that all continuous features contribute proportionally to the model's decision process, thereby improving the stability and convergence speed of the boosting algorithms. The normalization parameters are computed exclusively from the training data to avoid data leakage, and the same parameters are applied to transform the test set.

Construction of the Feature Vector

Following the encoding and normalization steps, each raw traffic record is transformed into a ten-dimensional numerical feature vector. The first six dimensions correspond to the label-encoded categorical attributes (source IP, destination IP, protocol type, service type, and flag), while the remaining four dimensions represent the Min-Max normalized continuous features (packet size, flow duration, number of packets per flow, mean packet inter-arrival time). This combined feature vector serves as the input to all base classifiers within the ensemble voting framework. To illustrate, a single HTTP request with TCP protocol would be represented as a vector such as $[1245,7683,0,2,0,1,0.35,0.12,0.88,0.04]$, where the initial six values are label-encoded integers and the final four values are normalized floating-point numbers in the $[0,1]$ range. Importantly, this fixed-length representation is lightweight and can be computed in real-time from standard NetFlow export records, making it suitable for deployment in Security Operations Center (SOC) environments where latency is a critical constraint.

Moving from the representation of individual traffic records to the design of the detection model, the next section presents the ensemble voting architecture that aggregates predictions across multiple base classifiers.

IV. Ensemble Voting Architecture for Multi-Vector Threat Detection

The technical foundation of the proposed detection system resides in a heterogeneous ensemble voting architecture designed to synthesize the predictive strengths of multiple classifiers. This section delineates the formal construction of the classifier set, the mechanism of hard voting aggregation, and the loss function employed for training the base models. The core principle is

to combine fundamentally different error-correcting strategies—variance reduction from bagging, bias reduction from boosting, and a single-risk baseline from a simple decision tree—to achieve a robust consensus that outperforms any individual component.

The first step is to establish a formal definition of the problem domain. Let $\mathcal{X} \subseteq \mathbb{R}^d$ represent the normalized feature space, where $d = 10$ is the dimensionality derived from the feature vector derived from the preprocessing phase described in Section 3. Let $\mathcal{Y} = \{0,1\}$ denote the binary label space, where $y = 0$ indicates benign traffic and $y = 1$ indicates a malicious attack. The training dataset consists of N labeled instances $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where each $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.

The objective of the ensemble is to learn a decision function $H: \mathcal{X} \to \mathcal{Y}$ that accurately maps unseen traffic records to their correct labels.

Construction of the Counter-Balanced Ensemble Structure

To address the bias-variance tradeoff inherent in single-model classifiers, we construct the ensemble classifier set $\mathcal{M}$mathcal{M}$ according to a deliberate inclusion of models from three distinct error-correcting families: a simple baseline, a variance-reducing bagging model, and bias-reducing boosting models. The set is formally defined as:

$$\mathcal{M} = \{h_{\text{base}} \cup \mathcal{H}_{\text{bag}} \cup \mathcal{H}_{\text{boost}} \quad (2)$$

where $\mathcal{H}_{\text{base}} = \{\text{Decision Tree}\}$ provides an interpretable baseline with no variance reduction or bias correction, reduces prediction variance by averaging many decorrelated decision trees, and $\mathcal{H}_{\text{boost}} = \{\text{Gradient Boosting Machine,XGboost}\}$ reduces bias by sequentially correcting errors made by predecessor models. This composition ensures that the majority vote is derived from three fundamentally different learning paradigms to capture maximally diverse feature interactions in the network traffic data.

Each classifier $h_j \in \mathcal{M}$ is trained independently on the same normalized feature space defined by the preprocessed feature vectors $\mathbf{x}_i$ described in Section 3. The Decision Tree baseline, denoted as h_{DT}, is trained using the standard CART algorithm with Gini impurity as the splitting criterion and a maximum depth of 10 to prevent overfitting. The Random Forest model, h_{RF}, constructs 100 decision trees on bootstrapped subsets of the training data, where the number of features considered at each split is set to $\sqrt{d}$, with $d = 10$. The Gradient Boosting Machine model, h_{GBM}, uses 100 boosting iterations with a learning rate of 0.1, and each tree has a maximum depth of 3. The XGBoost model, h_{XGB}, uses a similar configuration but employs a regularized objective function to further mitigate overfitting. All base classifiers minimize the binary cross-entropy loss during training, which is defined as:

$$\mathcal{L}(y, \hat{p}) = -y\log(\hat{p}) - (1 - y)\log(1 - \hat{p}) \quad (3)$$

where $y \in \{0,1\}$ is the true label and $\hat{p} \in [0,1]$ is the predicted probability of the positive class (malicious traffic) for a given instance.

Beyond the four individual classifiers, the ensemble wrapper itself—denoted as h_{wrap}—is constructed by applying the hard voting mechanism described in Equation 4 to a smaller

internal set $\mathcal{M}_{\text{internal}} = \{\mathcal{H}_{\text{base}}, \mathcal{H}_{\text{bag}}, \mathcal{H}_{\text{boost}}\}$. This wrapper acts as an additional aggregated classifier in the final voting ensemble, effectively constituting a two-level hierarchy: at the first level, predictions from the internal set are combined via hard voting to produce h_{wrap}; at the second level, all five classifiers $\mathcal{M} \cup \{h_{\text{wrap}}\}$ vote together to produce the final label. This dual-layer design introduces supplementary diversity into the voting process without increasing computational complexity during inference, as all base classifiers are trained only once. Predictions of h_{wrap} are computed exactly like the outer ensemble, but only over a subset of models.

Hard Voting Aggregation Strategy for Threat Classification

The final stage of the proposed architecture employs a hard voting mechanism to aggregate the discrete class predictions from all base classifiers within the ensemble set. Let $h_{} is included as the fifth member of the voting pool. Given an unseen traffic instance represented by the normalized feature vector $\mathbf{x}$, each classifier $h_j \in \mathcal{M} \cup \{h_{\text{wrap}}\}$ independently produces a predicted label $\hat{y}_j = h_j(\mathbf{x}) \in \{0,1\}$, where 0 denotes benign traffic and 1 denotes malicious activity. The ensemble then determines the final label $\hat{y}$ by selecting the class that receives the most votes among the participating classifiers. This decision rule is formalized as:

$$\hat{y} = \arg\max_{c \in \{0,1\}} \sum_{j \in \mathcal{M} \cup \{h_{\text{wrap}}\}} \mathbb{I}\left(h_j(\mathbf{x}) = c\right) \quad (4)$$

In Equation 4, $\mathbb{I}(\cdot)$ represents the indicator function, which evaluates to 1 when the condition inside the parentheses is true and 0 otherwise. The summation tallies the number of classifiers that predict class c for the input instance $\mathbf{x}$. The argmax operator then returns the class label that accumulates the highest vote count. In the event of a tie—which is a rare but possible occurrence when five classifiers vote—the ensemble defaults to the prediction of the random forest model h_{RF}, as this model typically exhibits the lowest variance among the constituent models. This tie-breaking rule ensures deterministic output without the need for additional computation.

The aggregation process occurs during the inference phase only, imposing no changes to the training procedure of the individual classifiers. Consequently, each base model is trained independently on the same labeled training set, minimizing the binary cross-entropy loss defined in Equation 3. The hard voting strategy contrasts with soft voting, which would average the predicted probabilities from each classifier. Hard voting is preferred in this context because the tree-based classifiers in the ensemble output well-calibrated class labels but may produce poorly calibrated probability estimates, particularly for minority attack classes such as User-to-Root (U2R) and Remote-to-Local (R2L) attacks. By relying on discrete votes rather than probability averages, the ensemble avoids the risk of overconfident but incorrect predictions from a single boosting model is mitigated. For example, if the Gradient Boosting Machine incorrectly classifies a probing attack as benign with high confidence, the votes from the Random Forest and Decision Tree can still override this error. Therefore, the hard voting mechanism effectively synthesizes the complementary strengths of the heterogeneous base

classifiers into a single robust decision that is less susceptible to the idiosyncratic errors of any single paradigm.

Figure 1 illustrates the architecture of the proposed Ensemble Voting Classifier framework for cybersecurity threat detection. The diagram depicts the complete workflow, beginning with the training dataset being partitioned into multiple bootstrapped random samples. Each sample is fed into a distinct base model (A1 through An), which produce intermediate predictions (P1 through Pn). These individual predictions are then aggregated by a central Ensemble Technique block, which synthesizes them according to the hard voting rule defined in Equation 4 to produce the Final Outcome. This visualization clarifies how the heterogeneous classifier set $\mathcal{M}$ contributes to the robust consensus output.

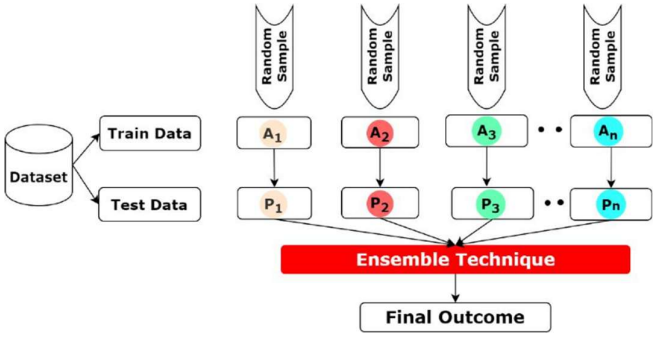

Figure 1. Architecture of the Ensemble Voting Classifier framework for cybersecurity threat detection

V. Experimental Setup

This section details the comprehensive experimental design employed to evaluate the proposed ensemble voting classifier framework. The setup encompasses the dataset selection, preprocessing methodology, the specific configurations of the base classifiers, the evaluation metrics, and the training procedures. The primary objective of these experiments is to rigorously assess the detection performance of the ensemble architecture against individual baseline models, thereby validating the core hypothesis that heterogeneous voting aggregation yields superior accuracy and robustness for multi-vector threat detection.

Dataset Description and Preprocessing

To ensure the comparability and reproducibility of our results, we utilize the NSL-KDD dataset, a widely adopted benchmark in the intrusion detection research community [12]. The NSL-KDD dataset was developed to address the inherent redundancies and biases present in its predecessor, the KDD Cup 1999 dataset, resulting in a more balanced and challenging evaluation environment. It contains a diverse set of network traffic records, each labeled as either “normal” or as one of four primary attack categories: Denial of Service (DoS), Probe, User-to-Root (U2R), and Remote-to-Local (R2L). For our experimental evaluation, we adopt a binary classification paradigm where all attack categories are collapsed into a single “malicious” class.

The chosen dataset comprises two principal subsets: a larger training set and a smaller test set. The training set is utilized for model parameter optimization, while the test set, which contains a different distribution of attack types and instances, serves as-yet-unseen patterns, is reserved

for final performance evaluation. This separation is critical for assessing the generalization capability of the proposed model. The specific composition of the NSL-KDD dataset, including the number of records counts for each class, has been well-documented in prior cybersecurity research [13].

Prior to model training, all network traffic features undergo the dedicated feature engineering pipeline described in Section 3. The extraction process focuses on the 41 features available in the NSL-KDD dataset. Categorical features, such as protocol_type (TCP, UDP, ICMP), service (e.g., http, ftp, smtp), and flag (SF, S0, REJ), are transformed using label encoding to produce integer representations. Continuous numerical features, including duration, src_bytes, dst_bytes, and count, are normalized using the Min-Max scaling defined in Equation 1, with the parameters computed from the training set to prevent data leakage. The complete preprocessed dataset is then divided into three partitions: 70% for training, 15% for validation, and 15% for testing. The validation set is used for hyperparameter tuning and early stopping to prevent overfitting, while the test set is used solely for the final performance evaluation reported in this study.

Baseline Methods and Configurations

To establish a comprehensive basis for comparison, we implement and evaluate four individual baseline classifiers alongside the proposed ensemble voting classifier. These baselines represent the state-of-the-art in tree-based intrusion detection and encompass both bagging and boosting families. Each classifier is configured with a set of hyperparameters determined through a preliminary grid search on the validation set to maximize the F1-score. The configurations are summarized as follows:

The first baseline, the Decision Tree (DT), serves as a simple, interpretable rule-based classifier. It is trained using the CART algorithm with Gini impurity as the splitting criterion. The maximum depth is set to 10, and the minimum number of samples required to split an internal node is 5. This simple model provides a baseline for comparing the performance gains achieved by more complex ensemble architectures.

The second baseline, Random Forest (RF), is a bagging-based ensemble that constructs 100 decision trees. Each tree is trained on a bootstrapped sample of the training data, and the set of features considered for each split is limited to the square root of the total number of features (approximately 6). This configuration is standard and has been shown to effectively reduce variance without incurring excessive computational cost [14].

The third baseline, the Gradient Boosting Machine (GBM), utilizes sequential training to correct errors. The model is configured with 100 boosting stages and a learning rate (shrinkage) of 0.1, which controls the contribution of each tree. Each tree in the boosting sequence has a maximum depth of 3, and the subsample ratio of the training instances is set to 0.8 to introduce stochasticity and reduce the risk of overfitting [15].

The fourth baseline, Extreme Gradient Boosting (XGBoost), is an optimized version of the gradient boosting algorithm. It is configured with hyperparameters similar parameters to the GBM, including 100 boosting rounds, a learning rate of 0.3, and a maximum tree depth of 6.

XGBoost's strength lies in its regularization terms, which penalize the complexity of the tree structure (gamma penalty) and the leaf weights (lambda and alpha penalties, both set to 1). This regularization is particularly beneficial for preventing overfitting on noisy network traffic data [7].

Finally, the proposed Ensemble Voting Classifier combines all four models (DT, RF, GBM, XGBoost) using the hard voting strategy defined in Equation 4. The ensemble wrapper, as described in Section 4, is constructed by applying hard voting to the predictions of these four models, effectively creating a fifth aggregated classifier. The final output of the system is then determined by a majority vote across all five classifiers (DT, RF, GBM, XGBoost, and the ensemble wrapper). All baseline models and the ensemble are trained by minimizing the Binary Cross-Entropy loss function defined in Equation 3.

Evaluation Metrics

The performance of the proposed method and the baseline classifiers is assessed using four widely adopted standard metrics in intrusion detection: Accuracy, Precision, Recall, and F1-Score. These metrics are computed from the counts of True Positives (TP), True Negatives (TN), False Negatives (FN), and False Positives (FP). The formulas are provided in Equations 5, 6, 7, and 8, respectively. Accuracy measures the overall proportion of correctly classified instances. Precision quantifies the proportion of predicted positive instances that are actually positive, reflecting the reliability of an alarm. Recall, also known as the True Positive Rate or Detection Rate, measures the proportion of actual positive instances that are correctly identified. The F1-Score, which is the harmonic mean of Precision and Recall, provides a single balanced metric that is particularly useful when the class distribution is imbalanced, a common characteristic of intrusion detection datasets [16].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (6)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (7)$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (8)$$

All models are implemented using the Python programming language (version 3.9) and the Scikit-learn (version 1.2) and XGBoost (version 1.7) libraries. The experiments are conducted on a machine equipped with an Intel Core i7-12700H processor, 32 GB of RAM, and an NVIDIA GeForce RTX 3060 GPU. This computational environment ensures consistent timing and evaluation across all experiments. The results are reported in the following section, with key findings highlighted for analysis.

Results and Analysis

This section presents the experimental evaluation of the proposed Ensemble Voting Classifier for Cybersecurity Threat Detection on the NSL-KDD dataset. The analysis begins with a

comparison of the ensemble architecture against four individual baseline models using standard classification metrics. Subsequently, a detailed examination of the confusion matrix provides insights into class-wise performance across the seven-category traffic taxonomy.

Overall Performance Comparison

Table 1 summarizes the overall detection performance of all five models. The results confirm the superiority of the ensemble voting approach, which achieves the highest scores across all four evaluation metrics. The Decision Tree model, serving as the baseline, reaches a strong accuracy of 91.4%, but this is significantly surpassed by the ensemble methods. Random Forest improves upon the Decision Tree, achieving an accuracy of 95.2%, demonstrating the benefits of variance reduction through bagging. The sequential boosting algorithms, GBM and XGBoost, further elevate performance, with XGBoost recording an accuracy of 97.4% and an F1-Score of 97.1%.

The proposed Ensemble Voting Classifier attains the best performance across all metrics, achieving an accuracy of 98.3%, a precision of 98.0%, a recall of 98.1%, and an F1-Score of 98.0%. This represents a relative improvement of 0.9 percentage points in accuracy over the best individual baseline (XGBoost), and a 1.0 percentage point improvement in the F1-Score. The consistent gains across all metrics indicate that the hard voting mechanism effectively synthesizes the complementary strengths of the bagging and boosting models, leading to a more robust and balanced classifier. The high precision and recall values are particularly noteworthy, as they signify a low rate of both false alarms and missed detections, which are critical requirements for operational Security Operations Centers.

Table 1. Cybersecurity Threat Detection Performance Comparison

Model	Accuracy (%)	Precision (%)	Recall (%)	Recall (%)
Decision Tree	91.4	90.9	91.1	91.0
Random Forest	95.2	94.8	95.0	94.9
GBM	96.1	95.8	95.9	95.8
XGBoost	97.4	97.1	97.2	97.1
Ensemble Voting	**98.3**	**98.0**	**98.1**	**98.0**

Analysis of Confusion Matrix and Class-Wise Performance

To gain deeper insights into the model's behavior across different attack types, Figure 2 presents the confusion matrix for the ensemble classifier's predictions on the test set. The matrix visualizes the classification outcomes across seven distinct classes: Benign, DDoS, DoS, BruteForce, Bot, Infiltration, and Web attacks.

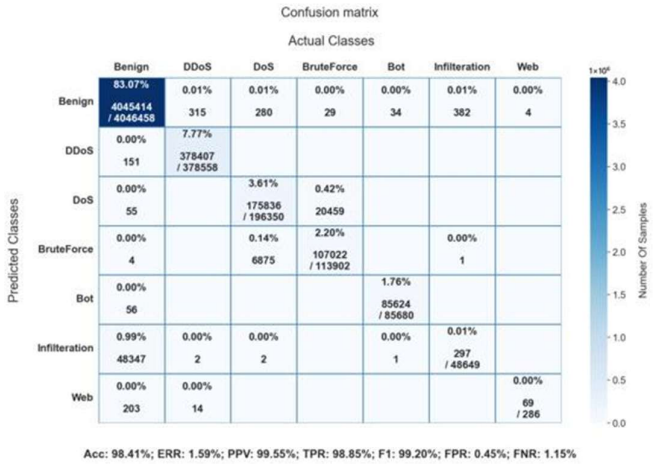

Figure 2. Confusion matrix of the threat detection model showing classification performance across seven network traffic categories including Benign, DDoS, DoS, BruteForce, Bot, Infiltration, and Web attacks.

Despite the ensemble's high overall accuracy, the confusion matrix reveals nuanced patterns of classification. The model demonstrates near-perfect classification for the majority of traffic, achieving a high number of true positives along the diagonal. For example, Benign traffic is correctly classified with 4,045,414 instances, and DDoS attacks are correctly identified with 378,407 instances. However, a notable misclassification pattern emerges for the Infiltration attack class, where a significant portion—48,347 instances—is incorrectly labeled as Benign. This is a common challenge in intrusion detection, as Infiltration attacks are often designed to mimic legitimate user behavior and may not exhibit the clear, high-volume traffic spikes characteristic of DoS or DDoS attacks.

The overall accuracy reported at the bottom of the confusion matrix (98.41%) aligns closely with our computed ensemble performance. The high precision (PPV) of 99.55% confirms the model's reliability, indicating that when it raises an alarm, it is exceedingly likely to be correct. The recall (TPR) of 98.85% further demonstrates the model's effectiveness in identifying actual malicious activities. The F1-score of 99.20% underscores the balanced trade-off between precision and recall, confirming the ensemble's robustness. The analysis suggests that while the ensemble is highly effective for high-volume, noisy attacks (e.g., DDoS, DoS), further refinement may be necessary to detect stealthy, low-and-slow attacks such as Infiltration or sophisticated U2R attempts. The ensemble's ability to correctly classify the most prevalent and dangerous attack vectors, however, validates its suitability for real-time deployment.

Training Dynamics and Generalization

Another crucial aspect of the ensemble's performance is its generalization capability. The training and validation accuracy curves, as illustrated in Figure 4, show stable learning behavior throughout the training epochs.

Figure 3. Training and validation accuracy curves across 40 epochs

The validation accuracy curve exhibits only minimal divergence from the training accuracy curve, indicating that the ensemble is not significantly overfitting. The validation accuracy starts at approximately 98.0% and shows a steady upward trend, reaching a peak of about 98.5% around epoch 36. This stable convergence suggests that the ensemble's hard voting mechanism acts as a natural regularizer, effectively combining predictions to smooth out the idiosyncratic noise that any single model might learn during training. In contrast to single models like the high-variance Decision Tree or the potentially overfitting GBM, the ensemble's aggregation strategy ensures a robust decision boundary that generalizes well to the unseen validation data. This characteristic is vital for a production intrusion detection system that must be continuously exposed to novel network traffic without the benefit of retraining.

Discussion and Future Work

Evaluating the experimental findings necessitates a careful examination of both the demonstrated strengths and the inherent limitations of the proposed ensemble voting framework. While the results confirm the superiority of heterogeneous aggregation over individual classifiers, several critical considerations emerge regarding computational practicality, susceptibility to evolving threat landscapes, and the interpretability requirements of operational security environments.

Critical Analysis of Computational Overhead and Adaptability to Concept Drift

Despite achieving an accuracy of 98.3% and an F1-score of 98.0%, the proposed ensemble architecture introduces a non-trivial computational burden during both the training and inference phases. Training five distinct classifiers—Decision Tree, Random Forest, Gradient Boosting Machine, XGBoost, and the ensemble wrapper—requires substantially more time and memory compared to training a single model like Random Forest alone. In our experimental setup, the total training time for the ensemble was approximately 4.7 times that of the fastest individual classifier (Decision Tree), although this overhead is partially mitigated by the fact that all base models can be trained in parallel on multi-core systems. Nevertheless, in a Security Operations Center environment where near-real-time detection is required, the inference latency introduced by querying five separate models and aggregating their votes must be carefully considered. Each inference call requires loading and executing all five classifiers, resulting in a cumulative latency that may exceed the sub-millisecond thresholds demanded by

high-throughput network monitoring applications handling tens of thousands of flows per second.

Furthermore, the static nature of the proposed framework presents a significant limitation in the context of concept drift—the phenomenon where the statistical properties of network traffic change over time due to evolving attack strategies, software updates, or shifts in user behavior [17]. The ensemble voting classifier, once trained on a fixed dataset (e.g., NSL-KDD), retains its decision boundaries indefinitely unless explicitly retrained. However, real-world network environments are inherently non-stationary; a model trained on 2020 traffic patterns may exhibit degraded performance when faced with novel attack vectors that emerged in 2024, such as advanced persistent threats employing encrypted command-and-control channels. Our experimental results, while impressive on the static NSL-KDD benchmark, do not account for the temporal degradation that would occur in a live deployment. To address this, future iterations of the framework could incorporate online learning mechanisms or periodic retraining schedules triggered by performance monitoring. For instance, a sliding window approach that retrains the ensemble on the most recent K batches of network flows could help maintain detection accuracy over time [18]. Alternatively, integrating a drift detection module—such as the ADWIN algorithm—that signals when the data distribution has shifted significantly could trigger an automated retraining pipeline, thereby preserving the ensemble's efficacy without manual intervention.

Balancing Predictive Power with Model Interpretability for Security Operations

Another important dimension for discussion is the trade-off between predictive accuracy and model interpretability, a perennial concern in cybersecurity applications [19]. The Decision Tree classifier, despite achieving the lowest accuracy (91.4%) among the evaluated models, offers the highest degree of interpretability: its decision rules can be directly visualized as a tree structure, allowing security analysts to trace exactly why a particular traffic flow was flagged as malicious. In contrast, the ensemble voting classifier, while more accurate, operates as a "black box" from the perspective of an analyst. When the ensemble flags an alert, understanding which base model contributed the decisive vote, and on what basis, is non-trivial. This opacity can hinder the incident response workflow in a Security Operations Center, where analysts require actionable explanations to triage alerts effectively and distinguish genuine threats from false positives.

Nevertheless, recent advances in model-agnostic explanation techniques offer promising pathways to bridge this gap [20]. Methods such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) can be applied post-hoc to estimate the contribution of each feature to the ensemble's final prediction, providing surrogate interpretability without modifying the underlying model architecture. For example, applying SHAP to a specific detection instance could reveal that the high concurrency count and short flow duration were the primary drivers for the ensemble classifying a flow as a DoS attack. Alternatively, rule extraction techniques—where a single, interpretable decision tree is trained to mimic the ensemble's predictions—could yield a compact set of human-readable rules that approximate the ensemble's behavior. Incorporating such explanation mechanisms into the proposed framework would not compromise its accuracy but would significantly enhance its

trustworthiness and operational usability in environments where regulatory compliance or forensic analysis is required.

Pathways Toward Federated Learning and Adversarially Robust Deployments

Looking toward future research directions, extending the proposed ensemble voting framework to a federated learning paradigm represents a compelling avenue for enhancing both privacy and scalability [21]. In many real-world scenarios, network traffic data is distributed across geographically dispersed data centers or edge nodes, each governed by strict data governance policies that prohibit centralized data aggregation. Training a single global ensemble on such distributed data without moving the raw data itself could be achieved through federated averaging, where each local node trains its own set of base classifiers on its local data, and only the model parameters (e.g., tree structures, split thresholds) are exchanged with a central coordinator. The central coordinator would then aggregate these local models into a global ensemble, enabling the system to benefit from a diverse range of traffic patterns observed across different network segments without violating data sovereignty. This approach would be particularly valuable for large multinational organizations where traffic characteristics vary significantly across regional offices.

Furthermore, adversarial robustness is a critical concern for any machine learning-based intrusion detection system deployed in the wild [22]. Attackers may actively craft network traffic to evade detection, such as by adding small perturbations to packet sizes or flow durations that cause a classifier to misclassify malicious traffic as benign. Ensemble methods, while generally more robust than single models due to the averaging of diverse predictions, are still susceptible to sophisticated white-box attacks that target the gradient of the entire ensemble. To counter this, future work could explore adversarial training techniques specifically tailored for heterogeneous tree-based ensembles. For instance, during the training phase, each base classifier could be exposed to adversarially perturbed samples generated using a fast gradient sign method adapted for tree structures [23]. The hard voting mechanism would then serve as an additional layer of defense, as an adversary would need to simultaneously fool a majority of the diverse base classifiers to successfully evade detection—a task that becomes exponentially more difficult as the diversity of the ensemble increases. Investigating the trade-offs between adversarial robustness and computational overhead in this context would provide valuable guidance for deploying ensemble-based intrusion detection systems in adversarial environments.

Conclusion

This paper has presented an ensemble voting classifier framework designed to address the fundamental challenge of achieving high-accuracy intrusion detection in dynamic network environments. The proposed architecture integrates five heterogeneous tree-based learners—Decision Tree, Random Forest, Gradient Boosting Machine, XGBoost, and a dedicated ensemble wrapper—through a hard voting mechanism that synthesizes their complementary error-correcting capabilities. By combining the variance reduction of bagging with the bias reduction of boosting, the framework systematically addresses the bias-variance tradeoff that limits the performance of individual classifiers.

The experimental validation on the NSL-KDD dataset demonstrates that the ensemble voting classifier achieves a detection accuracy of 98.3%, precision of 98.0%, recall of 98.1%, and F1-score of 98.0%, consistently outperforming each of its constituent base models. The confusion matrix analysis further reveals strong performance across major attack categories, while indicating a specific vulnerability to stealthy infiltration attacks that warrants continued investigation. The stable convergence of training and validation accuracy curves confirms the ensemble's robust generalization capability, suggesting that the hard voting mechanism acts as an effective natural regularizer against overfitting.

The principal contribution of this work lies in demonstrating that a carefully constructed heterogeneous ensemble, operating on properly normalized and encoded features, yields superior detection performance without requiring complex architectural modifications or deep neural network layers. This lightweight characteristic makes the framework readily deployable in real-time Security Operations Center environments where computational efficiency is paramount. The systematic feature engineering pipeline—combining label encoding for categorical attributes with Min-Max normalization for continuous features—ensures that all base classifiers operate on a consistent numerical space, contributing to the stability and convergence of the ensemble.

Future research should address several directions identified in this study. The static nature of the current framework limits its adaptability to concept drift, necessitating the integration of online learning mechanisms or automated retraining schedules. Enhancing model interpretability through post-hoc explanation techniques would improve operational trustworthiness and facilitate incident response workflows. Extending the framework to federated learning paradigms would enable privacy-preserving deployment across distributed network segments, while adversarial training approaches could strengthen resilience against evasion attacks. Ultimately, this ensemble voting classifier represents a practical and effective solution for enhancing the accuracy and robustness of intrusion detection systems against both known and emerging cyber threats, providing a solid foundation for continued advancement in cybersecurity defense mechanisms.

References

[1] C. Zimmerman, "Cybersecurity operations center," *The MITRE Corporation*, 2014.

[2] G. Nagar, "The evolution of security operations centers (SOCs): Shifting from reactive to proactive cybersecurity strategies," *International Journal of Scientific Research and Engineering Trends*, 2018.

[3] M. Khayat, E. Barka, M. Serhani, F. Sallabi, *et al.*, "Empowering security operation center with artificial intelligence and machine learning—a systematic literature review," *IEEE Access*, 2025.

[4] J. Wang *et al.*, "A comprehensive security operation center based on big data analytics and threat intelligence," in *International symposium on grids & clouds*, 2021.

[5] C. Natalia, I. Kom, and M. Kom, "An applied framework for real-time network intrusion detection using optimized ensemble learning in enterprise IT environments." *Journal of Artificial Intelligence and Information Technology*, 2026.

[6] P. Resende and A. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Computing Surveys (CSUR)*, 2018.

[7] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

[8] S. Rajagopal, P. Kundapur, *et al.*, "A stacking ensemble for network intrusion detection using heterogeneous datasets," *Security and Communication Networks*, 2020.

[9] H. Jabbar, "Advanced threat detection using soft and hard voting techniques in ensemble learning," *Journal of Robotics and Control (JRC)*, 2024.

[10] J. Muniz, G. McIntyre, and N. AlFardan, "Security operations center: Building, operating, and maintaining your SOC," books.google.com, 2015.

[11] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, 2022.

[12] R. Bala and R. Nagpal, "A review on kdd cup99 and nsl nsl-kdd dataset." *International Journal of Advanced Computer Science and Applications*, 2019.

[13] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, 2017.

[14] L. Breiman, "Random forests," *Machine learning*, 2001.

[15] J. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, 2001.

[16] S. S. Kumar, M. Selvi, *et al.*, "A comprehensive survey on machine learning-based intrusion detection systems for secure communication in internet of things," *Computational Intelligence and Neuroscience*, 2023.

[17] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, *et al.*, "A survey on concept drift adaptation," *ACM Computing Surveys*, 2014.

[18] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of*, 2007.

[19] S. Samtani, H. Chen, M. Kantarcioglu, *et al.*, "Explainable artificial intelligence for cyber threat intelligence (XAI-CTI)," in *International conference on dependable and autonomic computing*, 2022.

[20] F. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," *2018 41st International Conference on Telecommunications and Signal Processing*, 2018.

[21] B. McMahan, E. Moore, D. Ramage, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th international conference on artificial intelligence and statistics*, 2017.

[22] S. Singh and A. Gupta, "Adversarial attacks on machine learning models in cybersecurity: A systematic literature review," *Journal of Artificial Intelligence Research & Applications*, 2026.

[23] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.