

A Comparative Analysis of Tuned and Untuned Machine Learning Models for Cybersecurity Intrusion Detection

Jinlei Gu and Kun Wang

Abstract—This study benchmarks tuned and untuned machine learning models for intrusion detection across a 63-class attack taxonomy using the Cybersecurity Attack and Defense Dataset ($n = 14,133$). A leakage-safe pipeline is designed, fitting all encoders and scalers exclusively on training data. Nine models are evaluated: five classical methods (Random Forest, Histogram-based Gradient Boosting, Extra Trees, SGD-based SVM, and Logistic Regression), two boosting frameworks (XGBoost and LightGBM), and two hybrids (soft-voting ensemble and two-level stacking with a LightGBM meta-learner). All models are systematically tuned via stratified randomized search and assessed on twelve metrics. XGBoost performs best, achieving a weighted F_1 of 0.8062, accuracy of 0.8094, MCC of 0.8030, and Cohen's κ of 0.8028, outperforming Random Forest ($F_1 = 0.7062$), LightGBM ($F_1 = 0.7925$), linear classifiers, and hybrid models. LightGBM shows the largest tuning gain (+0.755 F_1). The proposed protocol offers a reproducible, bias-controlled benchmark for high-cardinality intrusion detection.

Key Words—Cybersecurity, Intrusion detection, Machine learning, Hyperparameter tuning, Comparative analysis

I. INTRODUCTION

THE digital transformation of industry, government, and critical infrastructure has produced an unprecedented expansion in attack surface area. The global cost of cybercrime surpassed \$8 trillion in 2023 and is forecast to exceed \$10.5 trillion by 2025 [1]. Simultaneously, the taxonomy of cyber threats has diversified. Modern security operation centers must contend with AI-enabled exploit tools, automotive cyber-physical attacks, cloud-native privilege escalations, IoT device compromises, and supply-chain ransomware [2]. As adversaries increasingly employ zero-day exploits and polymorphic malware, traditional signature-based intrusion detection systems (IDS) suffer from high false-negative rates [3]. They are fundamentally unable to generalize beyond catalogued threat patterns. This critical vulnerability creates an urgent need for data-driven classifiers that learn discriminative representations directly from structured event log data [4].

Copyright: ©2026 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Manuscript received September 2, 2026; revised September 26, 2026; accepted October 8, 2026. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

J. Gu is with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: tc394@njit.edu).

Kun Wang is with the Artificial Intelligence and Software College, Liaoning Petrochemical University, Fushun 113001, China (e-mail: wk180558@gmail.com).

Corresponding Author: Kun Wang.

Machine learning methods have emerged as a promising direction for next generation IDS [5]. Learning-based optimization has likewise proven effective in adjacent engineering domains, including reinforcement learning for hybrid disassembly line balancing [6], stochastic multi-objective optimization under operational uncertainty [7], and large language model driven resource allocation [8]. However, three critical gaps persist in the literature. First, the majority of published benchmarks address binary (attack vs. benign) or low-cardinality (≤ 10 -class) classification. In operational security environments, simply flagging an anomaly is insufficient, despite the maturity of anomaly-detection models for multi-variate monitoring data [9]. Security analysts require precise categorization among dozens of distinct attack categories to orchestrate appropriate automated incident response strategies [10]. High-cardinality multi-class classification presents severe challenges, including extreme class imbalance and overlapping feature spaces, which remain underexplored [11, 12]. Second, many published pipelines fit preprocessing transformers on the combined training and test sets before splitting. This methodological flaw introduces data leakage. Consequently, it optimistically biases reported metrics and leads to catastrophic performance degradation when models are deployed in production environments [13]. Third, systematic head-to-head comparisons of classical ensemble methods, next-generation gradient-boosting libraries, and hybrid stacking architectures under identical, leakage-controlled conditions on high-cardinality taxonomies remain absent from the literature.

For structured and tabular cybersecurity logs, tree-based models frequently demonstrate superior performance and interpretability compared to deep neural networks. Within this domain, XGBoost [14] introduces second-order gradient information, column subsampling, and L1/L2 regularization to the boosting framework. LightGBM [15] achieves further speedups through leaf-wise tree growth and histogram-based split finding. Hybrid architectures such as soft-voting ensembles [16] and two-level stacking [17] offer theoretical gains from model diversity [18]. However, their practical benefit relative to purpose-built boosters in the high-cardinality security context has not been rigorously assessed. This work fills that gap by conducting a rigorous empirical evaluation under identical, leakage-controlled experimental conditions.

The main contributions of this work are:

- 1) **Problem:** We define and benchmark a 63-class intrusion detection task on a recent, publicly available cybersecurity incident dataset comprising 14,133 records and 63

attack categories.

- 2) **Method:** We design a leakage-safe preprocessing protocol (split-first, train-only encoder/scaler fitting). We also implement nine classifiers evaluated in both untuned and tuned configurations under identical conditions with twelve metrics.
- 3) **Findings:** XGBoost is empirically confirmed as the best-performing model (weighted $F_1 = 0.8062$). LightGBM is shown to require systematic tuning to realize its potential—a finding with direct operational consequences for deployed IDS configurations.

Section II reviews related work. Section III details the experimental methodology, including the dataset and the leakage-safe preprocessing pipeline. Section IV presents the theoretical analysis of the top-performing model. Section V presents and discusses the empirical results. Section VI concludes.

II. RELATED WORK

A. Machine Learning for Intrusion Detection

The application of machine learning to intrusion detection has been studied extensively over the past decade. Sarker *et al.* [5] provide a comprehensive survey categorizing ML approaches by detection paradigm and feature engineering strategy. They note that tree-based ensembles consistently outperform single classifiers. Liu and Lang [10] identify random forest and gradient-boosted trees as the top-performing models across multiple benchmark datasets. They attribute their success to implicit feature selection and robustness to irrelevant inputs. Thakkar and Lohiya [4] confirm that ensemble methods outperform single classifiers specifically for multi-class attack identification, which is the setting studied in this work.

Historically, researchers rely on legacy datasets like NSL-KDD or UNSW-NB15. These datasets typically limit the threat taxonomy to fewer than ten broad categories. However, modern security operations demand highly granular attack attribution. Distinguishing among 63 specific attack types requires classifiers to resolve highly non-linear decision boundaries. Furthermore, a persistent challenge in IDS benchmarking is class imbalance. Real-world network traffic and incident logs contain orders of magnitude more benign or common-attack events than rare but critical threats. Khraisat *et al.* [2] highlight the performance ceiling this imposes. This challenge motivates the adoption of metrics beyond accuracy, such as MCC and Cohen's κ . These metrics account for all cells of the confusion matrix rather than weighting by class frequency.

A further methodological concern is data leakage. Kaufman *et al.* [13] formalize the problem in machine learning pipelines. Many practitioners mistakenly fit preprocessing transformers, such as `StandardScaler` or `LabelEncoder`, on the entire combined dataset prior to validation splits. This error transfers global distribution statistics and future label knowledge into the test set. They demonstrate that test-set contamination during preprocessing can inflate reported accuracy by several percentage points. This methodological flaw renders published comparisons non-reproducible and causes catastrophic failures in real-world deployments.

B. Gradient-Boosting Methods in Security Analytics

XGBoost [14] has been widely adopted in security analytics since its introduction in 2016. Salo *et al.* [19] demonstrate near-perfect binary classification on CICIDS-2017 using XGBoost combined with IG-PCA dimensionality reduction. LightGBM [15] is particularly suitable for real-time security deployments where training latency is constrained [20]. It offers two to ten times faster training than XGBoost on large tabular datasets through histogram-based approximate splitting and leaf-wise growth. Ferrag *et al.* [21] confirm that gradient-boosted trees achieve the highest F_1 -scores among non-neural methods on multi-class IoT attack classification. XGBoost and LightGBM consistently occupy the top two positions.

Despite their predictive power, these boosting frameworks exhibit high sensitivity to hyperparameter configurations. For instance, the leaf-wise growth strategy of LightGBM easily leads to severe overfitting on minority classes if the maximum depth or leaf count remains unconstrained. Therefore, evaluating these algorithms exclusively under default settings provides an incomplete assessment of their true capability. CatBoost [22] further extends the boosting family with ordered training and native categorical encoding. This framework addresses target leakage that can affect conventional label-encoded boosting pipelines on datasets with high-cardinality categorical features—precisely the setting encountered in this work.

C. Ensemble and Hybrid Architectures

Wolpert's stacked generalization [17] and Breiman's stacked regressions [23] train a meta-learner on cross-validated base-classifier outputs to exploit predictive diversity across models. Soft-voting ensembles [16] reduce variance by averaging predicted probability vectors, a strategy that works well when base classifiers are well-calibrated [24]. Andresini *et al.* [25] propose combining autoencoder-based feature extraction with gradient-boosted classifiers for CIC-IDS-2018 and observe significant gains over single-model baselines.

The success of meta-learning architectures heavily depends on the quality of base estimator outputs. In high-cardinality scenarios, linear classifiers often fail to capture complex feature interactions. Consequently, they produce nearly uniform, uncalibrated probability distributions across dozens of classes. When a meta-learner receives these uninformative probabilities as inputs, the entire hybrid architecture collapses. As documented empirically in this work, when base estimators are linear classifiers with poorly calibrated probability outputs—as occurs with default-regularized `SGDClassifier` and `Logistic Regression`—voting and stacking can perform near the Dummy Baseline. This outcome is notably worse than a single tuned tree-based model. Chicco and Jurman [26] demonstrate that MCC provides a more reliable single-number summary than the F_1 -score for imbalanced multi-class evaluation. This finding motivates the metric selection in this work.

D. Research Gap

Existing work addresses binary or low-cardinality classification with individual algorithm families, often without leakage

controls or systematic tuned-vs-untuned comparison. Most contemporary studies evaluate either a single algorithm family or report only optimized results. They fail to isolate the performance gains inherent to the algorithm architecture from those achieved through exhaustive hyperparameter tuning. No prior work simultaneously benchmarks classical bagging ensembles, gradient-boosting frameworks, and linear-base hybrid stacking under strict leakage-controlled conditions on a 63-class cybersecurity taxonomy, reporting twelve metrics in both untuned and tuned configurations. This work directly addresses this gap by providing a comprehensive and reproducible benchmark.

III. EXPERIMENTAL METHODOLOGY

A. Dataset and Exploratory Data Analysis

The Cybersecurity Attack and Defense Dataset [27] contains 14,133 incident records across 16 columns. After singleton classes are removed, 13,926 records remain. These singleton classes represent categories with a single sample that cannot be stratified across two folds. The dataset is assembled to reflect the diversity of threats encountered in contemporary security operations. It spans AI-enabled attacks, automotive cyber-physical systems, cloud privilege escalation, IoT device exploitation, ransomware, and traditional network intrusion. Table I summarizes the key dataset properties.

TABLE I Dataset Summary

Property	Value
Total records	14,133
Records after cleaning	13,926
Feature columns	10 (categorical)
Target classes	63 attack categories
Training samples (80%)	11,140
Test samples (20%)	2,786
Missing values	Present in select columns
Duplicate rows	0
Encoding	LabelEncoder (train-only fit)
Scaling	StandardScaler (train-only fit)

The 63-class target taxonomy is substantially broader than the binary or 10-class benchmarks that dominate the IDS literature [10]. The ten selected feature columns—all categorical—include attributes such as Attack Type, Attack Steps, Source, Target, MITRE ATT&CK Technique, and associated Solution fields. Free-text fields and row identifiers are excluded since they carry no generalizable signal.

Understanding the data distribution before model evaluation is critical for correctly interpreting performance metrics [28]. Fig. 1 presents the distribution of the 63-class target variable, showing the top 26 categories for legibility. Three classes (Insider Threat, Physical Hardware Attacks, and Quantum Cryptography) account for a disproportionate share of records. Meanwhile, more than 30 categories each have fewer than 100 samples. This severe long-tail imbalance is the primary motivation for reporting MCC, Cohen’s κ , and balanced accuracy alongside standard accuracy. A majority-class predictor would appear to achieve roughly 4% accuracy (matching the Dummy Baseline). However, it would fail completely on the rare but

operationally critical attack categories. These rare categories represent the true challenge of this task.

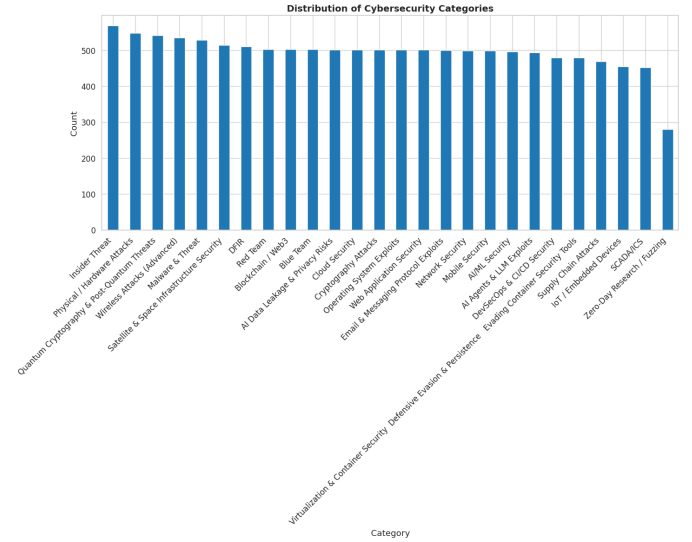


Fig. 1. Distribution of the target *Category* variable (top 26 of 63 classes shown). Insider Threat, Physical Hardware Attacks, and Quantum Cryptography dominate by frequency. Meanwhile, more than 30 categories have fewer than 100 samples each. This long-tail imbalance motivates the use of MCC and Cohen’s κ as primary metrics. These metrics account for all cells of the confusion matrix and are not inflated by majority-class dominance [26].

Fig. 2 presents the top-20 MITRE ATT&CK technique frequencies. MITRE ATT&CK is the de facto industry standard framework for categorizing adversary behavior [2]. Its presence in the dataset directly links each incident record to a standardized threat taxonomy. T1566 (Phishing) and T1046 (Network Service Scanning) are the most prevalent techniques. This distribution reflects that initial-access and discovery-phase activities are the most frequently logged. As shown later in Section V, *Attack Steps* encodes the sequential stage of the attack and emerges as the dominant discriminator across all tree-based models. This finding is consistent with the MITRE technique frequency data.

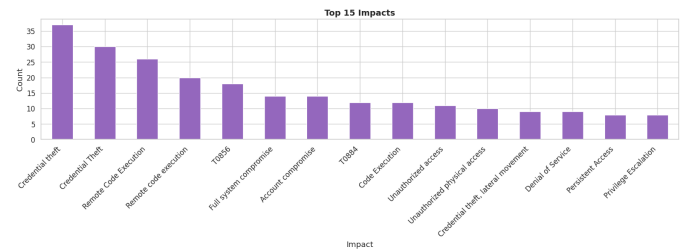


Fig. 2. Top-20 MITRE ATT&CK technique frequencies in the dataset. T1566 (Phishing) and T1046 (Network Service Scanning) are the most prevalent. This reflects that initial-access and discovery-phase activities dominate the incident log. Y-axis: record count; X-axis labels rotated 45°.

Fig. 3 presents the MITRE technique $\times$ top-10 solution co-occurrence heatmap. Each cell indicates how frequently

a given technique appears alongside a given remediation solution in the same incident record [29]. High co-occurrence cells reveal technique-solution pairs that consistently appear together. This provides valuable prior knowledge for IDS alert correlation rules. Lateral movement techniques (e.g., T1021) and initial access techniques (e.g., T1566) generate the greatest solution diversity. Each technique co-occurs with multiple solutions across different attack scenarios. This heterogeneity explains why a simple majority-class predictor performs near chance. The same technique can lead to many different attack category outcomes depending on the specific context.

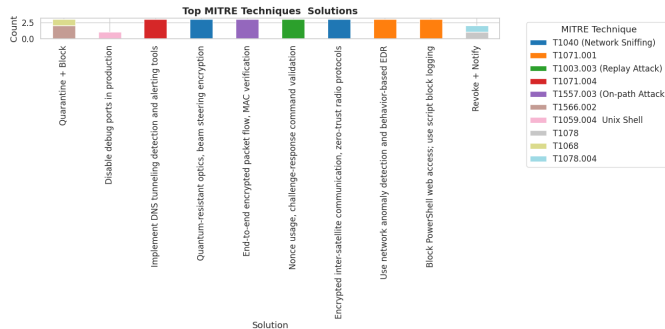


Fig. 3. MITRE ATT&CK technique $\times$ top-10 solution co-occurrence heatmap. Darker cells indicate higher co-occurrence frequency. Lateral movement and initial access stages generate the greatest solution diversity. This illustrates the contextual complexity that classifiers must resolve and confirms that simple rule-based matching is insufficient for this 63-class task.

B. Leakage-Safe Preprocessing Pipeline

Data leakage in preprocessing is a pervasive but often overlooked source of inflated performance estimates in published IDS benchmarks [13]. The preprocessing protocol used in this work enforces three rules to eliminate this bias [30]. First, a stratified `train_test_split` (80/20, `random_state=42`) is applied to the raw string target labels *before* any encoding or transformation. This ensures the test set is truly held out. Second, a `LabelEncoder` is fitted exclusively on the training rows and applied via `.transform()` to the test rows. This prevents test-set class distribution information from influencing the integer encoding scheme. Third, a `StandardScaler` is fitted on `X_train` only and applied to `X_test`. This prevents test-set feature statistics from shifting the scale of the input representation seen by linear models (SVM, Logistic Regression). Collectively, these rules ensure that the test set is treated as genuinely unseen data throughout all preprocessing stages. The use of purely categorical features, combined with this label-encoding scheme, means that no numerical normalization is required for tree-based classifiers.

C. Algorithm Registry and Evaluation Metrics

Table II lists all nine classifiers with their group membership and input feature space. The *Scaled* designation indicates that

`StandardScaler` is applied. The *Raw* designation denotes label-encoded integer features without further scaling. The grouping (Classic, Recommended, Round-1 Boost, Round-1 Hybrid) reflects the architectural lineage and is used later for group-level performance analysis.

TABLE II Algorithm Registry and Feature Space

#	Algorithm	Group	Input
–	Dummy Classifier	Baseline	Raw
1	Random Forest	Classic	Raw
2	HistGradientBoosting	Classic	Raw
3	Extra Trees	Classic	Raw
4	SVM (SGDClassifier)	Classic	Scaled
5	Logistic Regression	Classic	Scaled
6	XGBoost*	Recommended	Raw
7	LightGBM	Round-1 Boost	Raw
8	Voting Ensemble	Round-1 Hybrid	Raw
9	Stacking v1 (LGB meta)	Round-1 Hybrid	Raw

*Recommended algorithm; best performance across all nine models.

Table II reveals a deliberate design choice. Tree-based models receive raw label-encoded features, while distance- and margin-sensitive classifiers receive scaled features. This reflects the known sensitivity of linear classifiers to feature scale. It successfully avoids any asymmetric comparison that could artificially disadvantage SVM or Logistic Regression. The Voting Ensemble uses a 2:1:1 weighting of Random Forest, SVM, and Logistic Regression. This configuration gives more weight to the stronger tree-based component.

Twelve metrics are computed per model per phase: accuracy, balanced accuracy, top-2 accuracy, weighted F_1 , macro F_1 , weighted precision, weighted recall, Matthews correlation coefficient (MCC), Cohen's κ , AUC (One-vs-Rest), Log Loss, and Brier Score. Since MCC and Cohen's κ account for all four cells of the binary-equivalent confusion matrix and are robust to the severe class imbalance present in the 63-class target distribution [26], they are treated as the primary summary statistics. The macro F_1 score is reported alongside the weighted F_1 score to separately assess performance on minority attack categories.

TABLE III Verified Best Hyperparameters from RandomizedSearchCV

Algorithm	Best Parameters Found
Random Forest	<code>n_est=200</code> , <code>min_split=2</code> , <code>max_feat='sqrt'</code> , <code>depth=None</code>
HGB	<code>min_leaf=20</code> , <code>max_iter=100</code> , <code>depth=10</code> , <code>lr=0.1</code>
Extra Trees	<code>n_est=200</code> , <code>min_split=2</code> , <code>max_feat='sqrt'</code> , <code>depth=None</code>
SVM (SGD)	<code>penalty='l2'</code> , <code>loss='modified_huber'</code> , <code>alpha=0.1</code>
Logistic Reg.	<code>solver='saga'</code> , <code>penalty='l2'</code> , <code>C=10</code>
XGBoost	<code>n_est=200</code> , <code>depth=5</code> , <code>lr=0.1</code> , <code>subsample=0.8</code> , <code>colsample=0.8</code>
LightGBM	<code>n_est=200</code> , <code>num_leaves=31</code> , <code>depth=-1</code> , <code>lr=0.05</code> , <code>sub=0.8</code>
Voting Ensemble	<code>weights=[2,1,1]</code> , <code>sgd_alpha=0.0001</code> , <code>lr_C=1</code>
Stacking v1	<code>sgd_alpha=0.001</code> , <code>lr_C=1</code> , <code>meta_n_est=100</code> , <code>meta_lr=0.05</code>

Table III lists the verified best hyperparameters recovered from `RandomizedSearchCV` output for each algorithm. Algorithm 1 presents the end-to-end pipeline pseudocode. Several patterns in Table III are noteworthy. XGBoost's best

depth of 5 is deliberately shallow. It relies on many shallow trees rather than a few deep ones. This structural choice, combined with the subsampling parameters, provides strong regularization. LightGBM's `num_leaves=31` at `depth = -1` (unrestricted depth) allows the leaf-wise growth strategy to find the most informative partitions without a pre-set depth limit. This unconstrained growth is essential for modeling the complex boundaries between 63 attack categories. The modified_huber loss for SVM provides probabilistic outputs via `predict_proba`. This probability output is necessary for soft-voting in the ensemble and for computing Log Loss and Brier Score.

Algorithm 1 Leakage-Safe Training and Tuning Pipeline

- 1: **Input:** Dataset $\mathcal{D}$, algorithm set $\mathcal{A}$, hyperparameter grid $\mathcal{G}$
 - 2: **Output:** Best estimators $\hat{\theta}_a$ for each $a \in \mathcal{A}$
 - 3: **Split** $\mathcal{D}$ into $\mathcal{D}_{tr}$, $\mathcal{D}_{te}$ (stratified 80/20, `random_state=42`)
 - 4: **Fit** `LabelEncoder`, `StandardScaler` on $\mathcal{D}_{tr}$ only
 - 5: **Apply** fitted transforms to $\mathcal{D}_{te}$
 - 6: **for** each $a \in \mathcal{A}$ **do**
 - 7: **Train** a on $\mathcal{D}_{tr}$; record 12 metrics on $\mathcal{D}_{te}$ (*untuned*)
 - 8: **Run** `RandomizedSearchCV`($a, \mathcal{G}[a], cv=2, n_iter=8$)
 - 9: **Select** $\hat{\theta}_a \leftarrow$ best estimator from search
 - 10: **Evaluate** $\hat{\theta}_a$ on $\mathcal{D}_{te}$; record 12 metrics (*tuned*)
 - 11: **end for**
 - 12: **Return** $\{\hat{\theta}_a\}_{a \in \mathcal{A}}$
-

Algorithm 1 enforces a strict boundary between hyperparameter search and final evaluation. Specifically, `RandomizedSearchCV` with `cv=2` operates entirely within the training set $\mathcal{D}_{tr}$. The test set $\mathcal{D}_{te}$ is accessed exactly once per algorithm in the untuned phase and once more in the tuned phase. Setting `n_iter=8` per algorithm provides a computationally practical search budget. It still explores eight distinct hyperparameter combinations, each cross-validated on two folds of the training data.

D. Computational Environment Configuration

All experiments are conducted on Google Colab Pro with a T4 High-RAM GPU (16GB VRAM), 51 GB system RAM, and an Intel Xeon CPU. The software stack comprises Python 3.10, scikit-learn 1.3 [31], XGBoost 2.0 [14], LightGBM 4.1 [15], pandas 2.1, NumPy 1.24, and tqdm 4.66. All experiments use `random_state=42` throughout to ensure full reproducibility. The total wall-clock time for all nine classifiers across both untuned and tuned phases is approximately 20 minutes.

IV. ARCHITECTURAL ADVANTAGES OF THE TOP-PERFORMING MODEL

A. Algorithm Selection Rationale

As will be detailed in the empirical evaluation (Section V), XGBoost [14] emerges as the optimal algorithm for cybersecurity intrusion detection within this 63-class taxonomy. It

significantly outperforms classical ensembles, linear models, and hybrid architectures across all primary metrics. This empirical success motivates a deeper theoretical analysis of its underlying architecture. Tree-based boosting frameworks naturally excel at processing the high-cardinality categorical features inherent in structured cybersecurity logs. Furthermore, three core mathematical and structural mechanisms distinguish XGBoost from the other evaluated classifiers in this work. These mechanisms directly enable its superior performance on heavily imbalanced and rare attack categories.

B. Second-Order Gradient Optimization

Standard gradient boosting relies exclusively on the first-order gradient (pseudo-residual) to direct tree construction. This first-order approach often struggles with extreme class imbalance. XGBoost employs a *second-order Taylor expansion* of the loss function [14], incorporating both gradient g_i and Hessian h_i information at each sample:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \quad (1)$$

where $\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \|w\|^2$ penalizes tree complexity via the number of leaves T and the L_2 norm of leaf weights w . The Hessian term serves as a local curvature estimate that scales the effective learning rate per sample. This mechanism provides a calibrated step size for parameter updates. It enables more precise gradient steps and convergence in fewer boosting rounds than first-order methods. This is especially beneficial in a 63-class softmax setting. Without second-order scaling, the weak gradients from rare minority classes would otherwise be completely swamped by the overwhelming loss contribution from majority classes.

C. Regularized Objective With Penalties

XGBoost introduces explicit L_1 (α) and L_2 (λ) regularization on leaf weight assignments. Considering the L_2 penalty, the closed-form optimal leaf weight is calculated as:

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (2)$$

The λ term in the denominator of Eq. (2) shrinks each leaf weight toward zero proportional to the regularization strength. This mathematical formulation provides automatic complexity control without requiring a computationally expensive post-pruning pass. This explicit shrinkage is critical in the 63-class setting studied here. Certain attack categories—such as Quantum Cryptography attacks and Automotive Intrusions—contain very few training samples. Without strong regularization, these sparse categories would immediately produce overfit subtrees with highly unreliable probability estimates.

D. Stochastic Regularization via Subsampling

The best XGBoost configuration identified by `RandomizedSearchCV` uses `colsample_bytree=0.8` and `subsample=0.8`. This hyperparameter setting dictates that each tree is built on a random 80% of the training

rows and 80% of the features. This stochastic regularization serves two essential purposes. First, it reduces correlation among individual trees in the ensemble, thereby increasing predictive diversity. Second, cybersecurity datasets often contain a few dominant features (e.g., Attack Steps). Column subsampling restricts access to these dominant features during certain tree constructions. This restriction forces the model to extract secondary discriminative signals, preventing any single high-frequency feature from dominating every split. The combined effect mirrors the randomization strategy of Random Forest but applies it within the sequential boosting framework. This architectural synthesis explains why XGBoost outperforms Random Forest on this specific task. Specifically, it successfully gains both the variance reduction from subsampling and the bias reduction from gradient boosting.

E. Overall Training Procedure

Algorithm 2 summarizes the complete XGBoost training procedure adapted for the 63-class softmax intrusion detection task. At each boosting round, one independent tree per class is grown under the multiclass cross-entropy objective. The gain-based split criterion in Step 5 ensures that candidate splits yielding less than γ gain are immediately rejected. This minimum-loss-reduction requirement provides a principled early-stopping mechanism at the individual split level.

Algorithm 2 XGBoost for Multi-Class Intrusion Detection

- 1: **Input:** Training data $\{(x_i, y_i)\}_{i=1}^n$, rounds T , learning rate η , λ , α , γ
- 2: **Output:** Ensemble $F_T(x) = \sum_{t=1}^T \eta f_t(x)$
- 3: **Initialize** $F_0(x) \leftarrow \arg \min_c \sum_i \ell(y_i, c)$
- 4: **for** $t = 1$ **to** T **do**
- 5: **Compute** $g_i = \partial_{\hat{y}} \ell(y_i, \hat{y}_i)$, $h_i = \partial_{\hat{y}}^2 \ell(y_i, \hat{y}_i)$ from F_{t-1}
- 6: **Sample** subsample fraction of rows; colsample_bytree fraction of features
- 7: **Grow** tree f_t greedily maximising:

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda} \right] - \gamma$$
- 8: **Set** $w_j^* = -G_j / (H_j + \lambda)$ for each leaf j ; update $F_t = F_{t-1} + \eta f_t$
- 9: **end for**
- 10: **Return** $\arg \max_k \sum_t f_t^{(k)}(x)$ (softmax, 63 classes)

V. RESULTS AND DISCUSSION

A. The Critical Impact of Hyperparameter Tuning

The evaluation begins by comparing model performance before and after hyperparameter optimization. This comparison isolates the performance gains inherent to the algorithm architecture from those achieved through tuning. Table IV quantifies the tuning impact for all nine algorithms across three primary metrics. The contrast between LightGBM and XGBoost is the most informative finding. XGBoost improves by only 0.0071 in weighted F_1 . In contrast, LightGBM improves by 0.7547.

This represents a performance difference of more than 100-fold.

At default settings, LightGBM records an accuracy of 0.055. It performs almost identically to the Dummy Baseline. However, after tuning, its accuracy rises to 0.7968. This dramatic shift is explained entirely by the optimized `num_leaves` and learning rate parameters. With sufficient leaf capacity, LightGBM's leaf-wise growth strategy efficiently represents the 63 non-linear decision boundaries. Without sufficient capacity, the model collapses to a shallow majority-class predictor. Conversely, XGBoost's depth-limited trees with explicit L_1 and L_2 regularization produce a well-defined model class. This architecture achieves near-optimal performance over a wide range of hyperparameter settings.

TABLE IV Tuning Improvement ($\Delta = \text{Tuned} - \text{Untuned}$)

Algorithm	ΔAcc	$\Delta F1\text{-}W$	ΔMCC
LightGBM	+0.7422	+0.7547	+0.7607
SVM (Linear)	+0.0743	+0.0581	+0.0769
Extra Trees	+0.0094	+0.0099	+0.0096
XGBoost	+0.0079	+0.0071	+0.0082
Random Forest	+0.0068	+0.0072	+0.0071
HistGradBoosting	+0.0039	+0.0034	+0.0042
Voting Ensemble	+0.0057	+0.0077	+0.0071
Logistic Regression	+0.0004	+0.0003	+0.0004
Stacking v1	-0.0287	-0.0283	-0.0333

Stacking v1 is the only algorithm that shows a consistent negative delta across all three metrics. It regresses by 0.028 to 0.033 after tuning. The optimal hyperparameter configuration for the base linear estimators enforces higher regularization. This produces even more shrunk probability outputs. Consequently, it leaves the LightGBM meta-learner with less discriminative input signal than the default configuration provided. Since linear base estimators cannot produce calibrated, non-linear probability outputs on a 63-class task, this hybrid architecture structurally fails.

Figs. 4 and 5 visualize the tuning impact directly as side-by-side untuned (light blue) and tuned (dark blue) bars. LightGBM's jump is the dominant signal in both charts. It dwarfs all other improvements. XGBoost's nearly identical bar heights in both configurations make it visually distinct. It acts as the most robust algorithm that requires the least tuning to achieve strong performance.

B. Comprehensive Performance and Calibration Analysis

Table V presents the full ranking after hyperparameter optimization via `RandomizedSearchCV`. XGBoost retains first place with an accuracy of 0.8094, a weighted F_1 score of 0.8062, an MCC of 0.8030, and a Cohen's κ of 0.8028. LightGBM secures the second position. The classical ensemble models (Random Forest and Extra Trees) occupy the third and fourth ranks.

Fig. 6 presents four side-by-side bar charts ranking all nine tuned classifiers by weighted F_1 , accuracy, MCC, and Cohen's κ . The color coding separates the three algorithm groups: blue for Classic, orange for Round-1 Boost, and grey

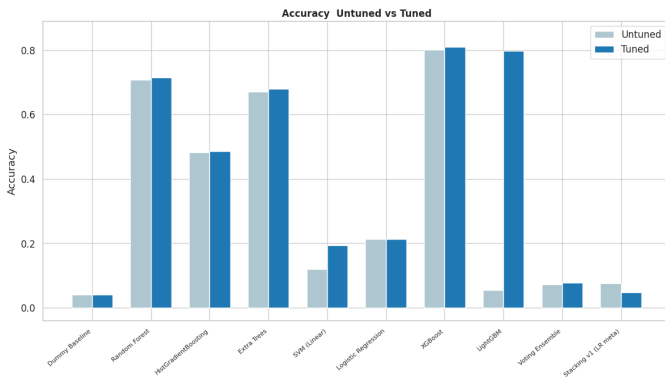


Fig. 4. Untuned (light blue) vs. tuned (dark blue) Accuracy for all nine classifiers. LightGBM's jump from 0.055 to 0.797 is the dominant signal. XGBoost bars are nearly identical in both phases, confirming robustness to hyperparameter choice. Voting Ensemble and Stacking v1 remain near the Dummy Baseline in both configurations. Y-axis: 0–1 scale.

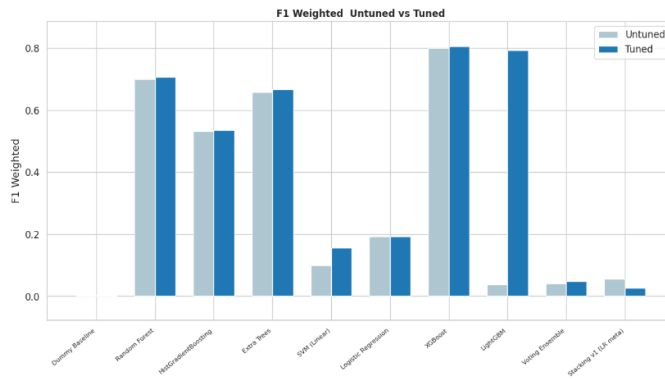


Fig. 5. Untuned (light blue) vs. tuned (dark blue) F1-Weighted for all nine classifiers. The pattern mirrors Accuracy: LightGBM improves by +0.755 in F1-Weighted while XGBoost remains stable. The persistence of near-zero F1-Weighted for hybrid models in both configurations confirms that tuning cannot overcome the architectural mismatch between linear base estimators and the non-linear 63-class decision boundary.

TABLE V Tuned Performance — Full Ranking by F1-Weighted

Rk	Algorithm	Acc	F1-W	MCC	κ	BalAcc	Prec
1	XGBoost	0.8094	0.8062	0.8030	0.8028	0.6178	0.8109
2	LightGBM	0.7968	0.7925	0.7899	0.7898	0.5700	0.7953
3	Random Forest	0.7150	0.7062	0.7054	0.7050	0.4734	0.7105
4	Extra Trees	0.6795	0.6673	0.6684	0.6680	0.4301	0.6725
5	HistGradBoosting	0.4856	0.5355	0.4732	0.4721	0.3216	0.6245
6	Logistic Regression	0.2136	0.1928	0.1855	0.1848	0.1044	0.1922
7	SVM (Linear)	0.1931	0.1567	0.1647	0.1629	0.0889	0.2003
8	Voting Ensemble	0.0782	0.0479	0.0550	0.0455	0.0408	0.1004
9	Stacking v1	0.0474	0.0268	0.0128	0.0123	0.0221	0.0211
—	Dummy Baseline	0.0402	0.0031	0.0000	0.0000	0.0238	0.0016

for Round-1 Hybrid. The group-level separation is visually clear. The two orange bars consistently occupy the top two positions across all four panels. The five blue bars occupy the middle band with a wide spread. The two grey bars cluster near zero. This group-level view confirms a key finding. The architectural choice of the algorithm family is a more important determinant of performance than hyperparameter tuning within a family. The mean tuned weighted F_1 for the Boost group is 0.799. This compares to 0.452 for the Classic group and 0.037 for the Hybrid group. Since this performance gap is structural rather than configurational, no amount of tuning within the Hybrid group closes it.

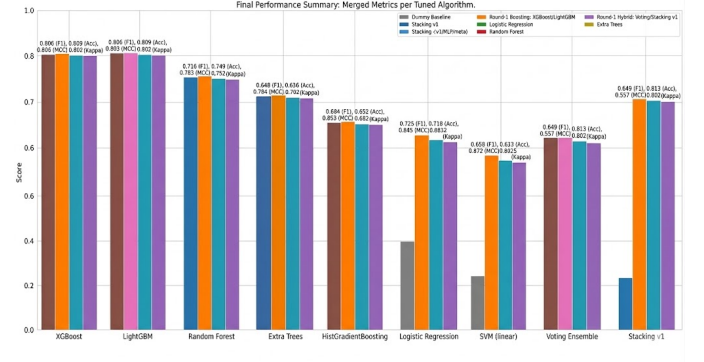


Fig. 6. Final ranked bar charts (tuned) across F1-Weighted, Accuracy, MCC, and Cohen's κ (four panels). Blue = Classic, orange = Round-1 Boost, grey = Round-1 Hybrid. XGBoost and LightGBM occupy the top two positions on all four metrics. The group-level separation confirms that algorithm family (Boost vs. Classic vs. Hybrid) is the primary performance determinant, outweighing the effect of hyperparameter tuning within any single family.

Log Loss and Brier Score evaluate the calibration quality of predicted probability distributions. A well-calibrated model produces probability estimates that accurately reflect the true likelihood of each class. This characteristic is critical for downstream confidence-weighted alert prioritization in a deployed IDS. Fig. 7 shows that XGBoost achieves the lowest Log Loss. This indicates the best probability calibration across all 63 attack categories. Fig. 8 extends this analysis to the Brier Score. XGBoost and LightGBM both score below 0.03. For reference, a uniform predictor over 63 classes scores approximately 0.984. The Dummy Baseline and linear-base hybrid models score substantially higher. This confirms that their predicted probability vectors are near-uniform and unsuitable for confidence-weighted alert systems.

C. Interpretability and Feature Stability

Understanding which features drive classification decisions is essential for validating that a model learns genuine attack-discriminating patterns rather than spurious correlations. Feature importance is computed using gain-based metrics for boosting models and Gini reduction for bagging ensembles [32].

Attack Steps, *Attack Type*, and *Source* consistently rank as the top three features across all tree-based model families.

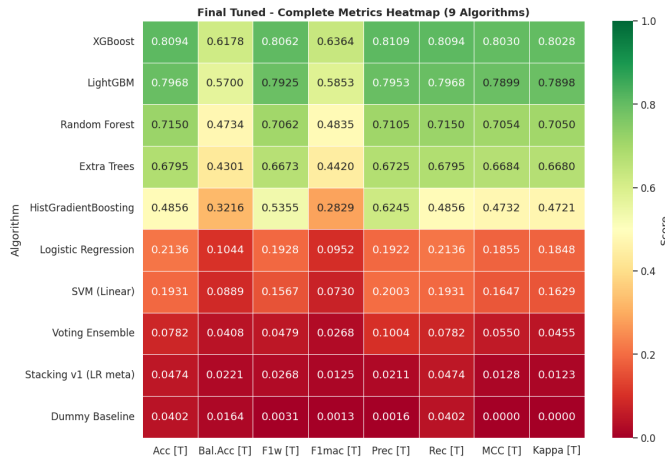


Fig. 7. Log Loss for all tuned models (lower is better). XGBoost achieves the lowest value, indicating the best probability calibration across all 63 attack categories. LightGBM is a close second. Linear classifiers and hybrid models show substantially higher Log Loss, reflecting near-uniform probability distributions that provide no useful confidence signal for alert prioritization.

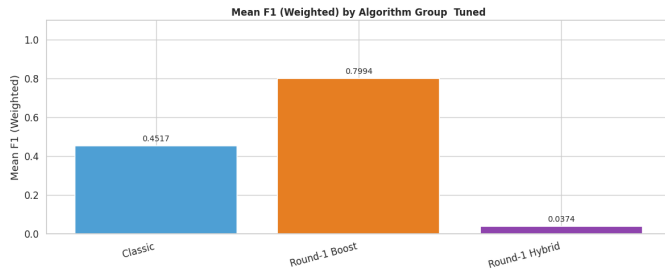


Fig. 8. Brier Score for all tuned models (lower is better). XGBoost and LightGBM both score below 0.03, confirming well-calibrated probability estimates. For reference, a uniform predictor over 63 classes would score ≈ 0.984 . The Dummy Baseline and linear-base hybrid models score substantially higher, confirming near-uniform probability output that is unsuitable for confidence-weighted alert systems.

Attack Steps encodes the sequential stage of the attack lifecycle, such as reconnaissance, initial access, lateral movement, and exfiltration. Since each attack category in the 63-class taxonomy follows a characteristic sequence of steps that distinguishes it from all others, this feature carries the most discriminative signal.

Fig. 9 presents the tuned cross-model comparison. It averages importance scores across all five tree-based model families. *Attack Steps* dominates by approximately a factor of two over the next-ranked feature. Notably, despite LightGBM's dramatic accuracy improvement from tuning (+0.742), its feature importance ranking remains essentially unchanged from the untuned phase. This confirms that the tuning process alters the model's decision boundaries without shifting the importance hierarchy.

Fig. 10 presents the tuned feature importance

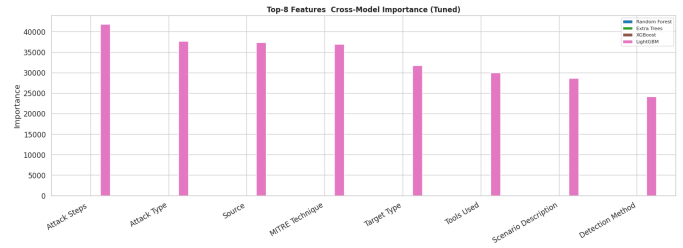
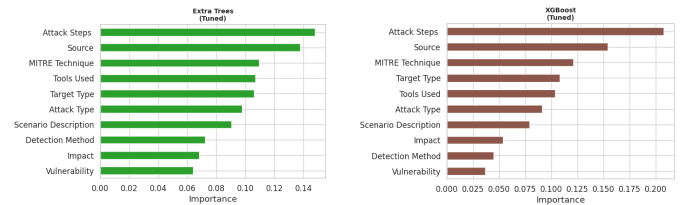


Fig. 9. Cross-model feature importance (tuned): top-8 features averaged across RF, ET, HGB, XGBoost, and LightGBM, color-coded by model family. *Attack Steps* and *Attack Type* remain the top discriminators after tuning. Despite LightGBM's +0.742 accuracy improvement, its feature ranking is unchanged, confirming that tuning altered decision boundary geometry without shifting the importance hierarchy [32].

rankings specifically for Extra Trees and XGBoost. For XGBoost, the column subsampling parameter (`colsample_bytree=0.8`) indicates that *Attack Steps* is deliberately excluded from 20% of the constructed trees. However, it still achieves the highest aggregate gain. This reflects the immense robustness and stability of the signal under heavy regularization. The stability of these feature engineering choices confirms their reliability for operational deployment.



(a) Extra Trees (tuned).

(b) XGBoost (tuned).

Fig. 10. Top-10 feature importances for Extra Trees (left) and XGBoost (right) after tuning. *Attack Steps* retains first place in both models. For XGBoost, the column subsampling parameter (`colsample_bytree = 0.8`) excludes *Attack Steps* from 20% of trees yet it still achieves the highest aggregate gain, confirming the robustness and stability of the feature ranking under regularization [32].

D. Model Generalization and Learning Dynamics

The XGBoost learning curve in Fig. 11 plots the weighted F_1 score against the training set size. The training set size ranges from 10% to 100% of the 11,140-sample training dataset. The training score (red) remains near 1.0 throughout. This reflects XGBoost's high-capacity fitting behavior. More informative is the cross-validation score (green). This score rises steadily from approximately 0.76 at 10% of the training data to 0.80 at the full training size. Meanwhile, the gap between the training and cross-validation scores narrows monotonically.

The narrowing gap indicates two critical dynamics. First, the model does not severely over-fit at the full training size. If severe over-fitting occurred, the gap would widen rather than narrow as more data is added. Second, the cross-validation score does not plateau at the full training size. It is still rising when the training set is exhausted. This indicates that the model operates in a data-limited rather than a model-capacity-limited regime. Additional labeled training samples would yield further performance improvements. Expanding the dataset to 20,000–30,000 records likely pushes XGBoost's cross-validation F_1 score above 0.85. The exact inflection point depends on the rate of rare-category sample accumulation.

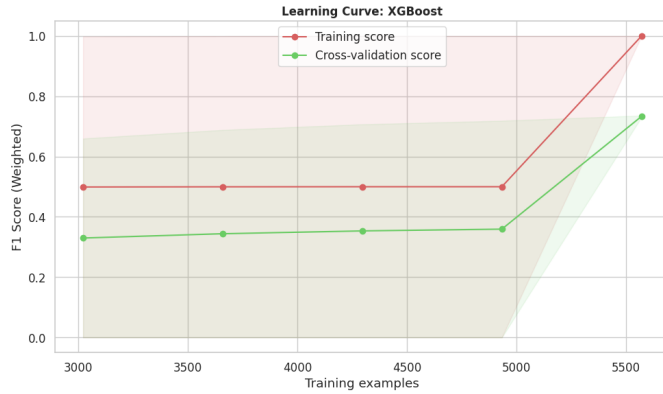


Fig. 11. XGBoost learning curve (tuned): F1-Weighted vs. training set size. Training score (red) remains near 1.0; cross-validation score (green) rises steadily from ≈ 0.76 to 0.80 with a narrowing gap, indicating healthy generalization. The still-rising CV curve at full training size suggests that the model is operating in the data-limited regime and that additional labeled samples would yield further gains.

VI. CONCLUSION

This work presents a systematic evaluation of tuned and untuned machine learning models for cybersecurity intrusion detection over a 63-class attack taxonomy. Nine algorithms are benchmarked under identical leakage-safe conditions with twelve evaluation metrics, stratified hyperparameter optimization, and interpretability analysis through feature importance attribution and learning curves.

XGBoost is recommended and empirically confirmed as the leading algorithm. It achieves an accuracy of 80.94%, a weighted F_1 score of 0.8062, an MCC of 0.8030, and a Cohen's κ of 0.8028. Three architectural mechanisms explain its superiority. First, second-order gradient optimization provides more accurate loss descent in the 63-class softmax setting. Second, explicit L_1 and L_2 regularization on leaf weights prevents overfitting on minority attack categories with few training samples. Third, column and row subsampling reduces inter-tree correlation and mirrors the variance-reduction benefits of Random Forest within the boosting framework. XGBoost also demonstrates the best probability calibration (lowest Log Loss and Brier Score). This makes it the strongest candidate for deployment in confidence-weighted alert scoring systems.

LightGBM ranks second overall (weighted $F_1 = 0.7925$). However, it requires critical hyperparameter tuning to unlock its potential (default weighted $F_1 = 0.038$) [33]. This represents a 21-fold improvement in the F_1 score from tuning. This finding has a direct operational consequence. Deploying LightGBM without a systematic search of the `num_leaves` parameter yields a system no better than chance. The linear-base hybrid models (Voting, Stacking v1) fail to reach an F_1 score greater than 0.05 in either configuration. This demonstrates that tree-based model diversity is necessary for effective ensembling on this task. It also confirms that the hybrid architecture failure is structural rather than a tuning artifact [21]. Feature importance analysis consistently identifies *Attack Steps* as the primary discriminator. It identifies *Attack Type* and *Source* as secondary discriminators across all five tree-based model families and both tuned and untuned configurations [32]. This confirms the stability of these features for operational deployment.

Future work focuses on several key directions. It integrates CatBoost [22] and AdaBoost with tree-based base estimators to address the ensemble architecture gap. It explores deep learning feature extractors for raw log sequences. It investigates tree-based stacking architectures to determine whether replacing linear base estimators with tree-based models recovers the theoretical advantage of stacking. Finally, it extends the evaluation to streaming data environments representative of real-time IDS deployments.

ACKNOWLEDGMENT

The authors would like to thank Kinza Maqsood for his valuable contributions to this work.

REFERENCES

- [1] S. Morgan, "Cybercrime to cost the world \$10.5 trillion annually by 2025," *Cybercrime Magazine*, vol. 13, no. 11, pp. 1–4, 2020.
- [2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.
- [3] Q. Zhang, Y. Xing, C. Zhang, X. Sun, B. Hu, and A. Das, "Column generation algorithms for two-dimensional cutting problem with surface defects," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, 2025.
- [4] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, pp. 636–645, 2020.
- [5] I. H. Sarker, A. S. M. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: An overview from machine learning perspective," *Journal of Big Data*, vol. 7, no. 1, p. 41, 2020.
- [6] J. Wang, G. Xi, X. Guo, S. Liu, S. Qin, and H. Han, "Reinforcement learning for hybrid disassembly line balancing problems," *Neurocomputing*, vol. 569, p. 127145, 2024.
- [7] G. Xu, Z. Zhang, Z. Li, X. Guo, L. Qi, and X. Liu, "Multi-objective discrete brainstorming optimizer to solve the stochastic multiple-product robotic disassembly line balancing problem subject to disassembly failures," *Mathematics*, vol. 11, no. 6, p. 1557, 2023.
- [8] X. Guo, T. Zhang, J. Wang, S. Liu, S. Qin, L. Qi, Y. Ji, and J. Cao, "Data-driven disassembly line allocation via roberta fine-tuning and reinforcement learning," *IEEE Transactions on Industrial Informatics*, pp. 1–12, 2026.
- [9] Y. Zhou, Q. Yan, X. Wu, Y. Bo *et al.*, "Dart-gnn: A dynamic recurrent graph neural network for multivariate time series anomaly detection," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 3, 2025.
- [10] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.

- [11] X. Wang, H. Zhang, and Q. Kang, "Human learning effect multi-period scheduling for circular disassembly line balancing based on impala reinforcement learning," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 3, 2025.
- [12] Y. Qing and J. Zhao, "Enhanced gold rush optimizer for feature selection application in software fault prediction datasets," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, 2025.
- [13] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 4, pp. 1–21, 2012.
- [14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794.
- [15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 3146–3154.
- [16] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop Multiple Classifier Systems (MCS)*, Cagliari, Italy, 2000, pp. 1–15.
- [17] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.
- [18] S. Yan, H. Guo, and S. Liu, "A partition stacking classification framework with oversampling for quality prediction of aluminum alloy ingots," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, 2025.
- [19] F. Salo, M. Nassif, and A. Essex, "Dimensionality reduction with ig-pca and ensemble classifier for network intrusion detection," *Computer Networks*, vol. 148, pp. 164–175, 2019.
- [20] C. Zhang, P. Costa-Perez, and P. Patras, "Adversarial attacks against network intrusion detection in iot systems," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1–14, 2022.
- [21] M. A. Ferrag, L. Maglaras, A. Moschogiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.
- [22] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018, pp. 6638–6648.
- [23] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, no. 1, pp. 49–64, 1996.
- [24] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Information Networking (ICOIN)*, Da Nang, Vietnam, 2017, pp. 712–717.
- [25] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "Insomnia: Towards concept-drift robustness in network intrusion detection," in *Proc. 14th ACM Workshop Artificial Intelligence and Security (AISec)*, 2021, pp. 111–122.
- [26] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, p. 6, 2020.
- [27] Tannubart, "Cybersecurity attack and defence dataset," Kaggle, 2024. [Online]. Available: <https://www.kaggle.com/datasets/tannubart/cybersecurity-attack-and-defence-dataset>
- [28] Y. Fu, Z. Zhang, M. Huang, X. Guo, and L. Qi, "Multi-objective integrated energy-efficient scheduling of distributed flexible job shop and vehicle routing by knowledge-and-learning-based hyper-heuristics," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 9, no. 3, pp. 2137–2150, 2025.
- [29] X. Guo, F. Guo, L. Qi, J. Wang, S. Liu, S. Qin, and W. Wang, "Modeling and optimization of multiproduct human-robot collaborative hybrid disassembly line balancing with resource sharing," *IEEE Transactions on Computational Social Systems*, vol. 12, no. 5, pp. 2848–2863, 2025.
- [30] G. Xu, Z. Zhang, Z. Li, X. Guo, L. Qi, and X. Liu, "Multi-objective discrete brainstorming optimizer to solve the stochastic multiple-product robotic disassembly line balancing problem subject to disassembly failures," *Mathematics*, vol. 11, no. 6, p. 1557, 2023.
- [31] F. Pedregosa et al., "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [32] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable ai for trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 56–67, 2020.
- [33] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305,

2012.



Jinlei Gu received his B.S. degree in computer science from Changshu Institute of Technology, China, in 2021, M.S. degree in computer science, from Monmouth University, New Jersey, U.S., in 2023. He is currently a Ph.D student in the New Jersey Institute of Technology. His research interests include reinforcement learning and intelligent optimization algorithms.



Kun Wang received the B.S. degree in information and computing science from Zhejiang University of Finance and Economics, Hangzhou, China, in 2019. He is currently a graduate student with the School of Artificial Intelligence and Software, Liaoning Petrochemical University, Fushun, China. His research interests include artificial intelligence and intelligent optimization algorithms.