Journal of Cybersecurity and Information Management

◆ ISSN: 2690-6775 Vol: 18, No. 01, 2026 ◆ PP. 40–55
Article DOI: https://doi.org/10.54216/JCIM.180106

FULL LENGTH ARTICLE

Machine Learning-Driven Cyber Threat Prediction and Prevention: A Multi-Dataset Design and Comparative Evaluation

Krishneel Sundar[1,*] **Pritika Reddy**[1] **Kaylash C. Chaudhary**[2]

[1] Department of Computing Science and Information Systems, Fiji National University, Nasinu, Fiji
[2] Department of Information and Mathematical Sciences, the University of the South Pacific- Laucala Campus, Suva, Fiji
Emails: krishneelsundar143@gmail.com · pritikareddy26@gmail.com · kaylash.chaudhary@usp.ac.fj

Received: January 22, 2026 **Revised:** February 18, 2026 **Accepted:** March 30, 2026 ★ **Corresponding author**

ABSTRACT

As technology advances, the frequency and variety of intrusions and other security threats within network environments continue to grow. Intrusion detection systems (IDS) play a vital role in securing networks against unauthorized access and attacks on computer systems; however, traditional IDSs are very limited in their ability to recognize new, complex malicious threats because they rely on signature-based detection. Approaches based on machine learning have shown a promising alternative in identifying unknown malicious attacks. This study proposes a computationally efficient, generalizable machine-learning framework for robust cyber-threat prediction. Three benchmark datasets (HIKARI-2021, CIC-IDS2017, and KDDCup99) were used for full-pipeline evaluations, including preprocessing, feature selection, class-imbalance handling, hyperparameter optimization, and strict model validation. Eight classifiers were assessed, which included traditional classifiers and more modern ensemble methods. The results from this study showed that tree-based models, mainly both Random Forest and XGBoost achieved near-perfect performance across all datasets, reaching accuracy values up to 0.999 and F1-scores between 0.99 and 0.999. Additionally, the SHAP-based explainability analysis was applied to reveal features that drove predictions, enabling interpretability and transparency. Compared with prior studies, the proposed framework consistently delivers improved, more stable detection performance. The findings highlight that optimized ML models combined with balanced datasets and rigorous validation protocols can significantly enhance intrusion detection reliability. Furthermore, this approach provides a practical and scalable solution for strengthening cybersecurity defenses against evolving and emerging cyber threats.

Keywords: Network intrusion ▪ Machine Learning ▪ Intrusion Detection System ▪ Ensemble Methods ▪ Generalization ▪ SHAP Explainability

1. INTRODUCTION

Network intrusion is the general term for any malicious or unauthorized act that attempts to compromise the Confidentiality, Integrity, and Availability of a given networked system or its data. Recent research shows that cyber-crime is a macroeconomic risk and that by 2025, the worldwide cost of cybercrime is projected to be

approximately $10.5 trillion dollars each year. This figure will be driven by the proliferation of ransomware, data exfiltration, and an increase in automation of Attack Campaigns [1]. These incidents have affected individuals, corporations, as well as governmental entities through a direct loss of revenue, disruption of services, and regulatory penalties leading to damage to the corporation's reputation over time.

To mitigate such risks, intrusion detection systems (IDSs) have become an essential part of modern multi-layered security systems. A IDSs review network data and computer activities to find rule violations, detect intrusions, and alert security staff of any irregular activity [2]. An IDS can be either network-based (NIDS) or host-based (HIDS); NIDS collect packets and flow data from multiple networks, while HIDS observe logs and events. Both are typically deployed with firewalls and host endpoint security (i.e., McAfee/VirusScan) so that the organization has visibility into and awareness of any more elaborate forms of multi-stage intrusions of data [3]. Traditional IDSs that work on signature matching and rules-based detection cannot detect new attack patterns, encrypted transmissions, or high-speed data streams, meaning they can produce a lot of "false positive" alerts and can miss real intrusions [3]. Alert fatigue and missed detections have illustrated the requirement for a more adaptable and data-driven detection strategy.

Machine-learning-based IDS (ML-IDS) systems are rapidly gaining popularity due to their ability to automatically and rapidly adapt to the dynamic nature of network traffic [4]. ML-IDS systems use machine-learning algorithms to create complex classifiers based on previous traffic, and they offer greater flexibility than traditional signature-based IDS [4]. Recent studies have demonstrated that classical machine-learning models (such as Random Forest, Support Vector Machine, K-Nearest Neighbors) and modern machine-learning models (such as gradient boosting and DNNs) have the potential to achieve very high detection rates when tested using standard datasets and they can also detect both abnormal and normal behavior in computers and networks [3]. For example, Hossain and Islam [5] created an ensemble or hybrid ML-IDS which combines multiple tree-based classifiers and achieved over 99% accuracy on several public datasets. Similarly, multi-dataset ML-IDS systems tested on NSL-KDD, CIC-IDS2017, and UNSW-NB15 have all demonstrated impressive performance in laboratory environments [6].

The development of a stable intrusion detection system (IDS) in the realm of digital cyberspace presents many challenges. Cyber threats continue to evolve daily, which complicates the job of developing an IDS that will accurately identify and effectively respond to all types of attack on modern computer network infrastructures [7]. Also, the high volume of network traffic created by a digitized society today makes detecting unusual events and patterns of behavior difficult [8]. False positives and false negatives are two of the biggest obstacles to creating a robust IDS. A false positive occurs when an IDS alerts an administrator of a potential attack, but it's actually a false alarm. A false negative occurs when an IDS fails to detect a real attack. Both of these situations have serious implications for organizations; false positives lead to excess system downtime, while false negatives can create an exploit and cause catastrophic damage to an organization [9]. Furthermore, the building of stable Intrusion Detection Systems (IDS) requires a combination of both cybersecurity and machine learning knowledge. Cyber security professionals have a difficult time keeping up-to-date on all of the most advanced attack strategies and techniques due to ongoing changes in both the complexity of modern computer networks and the fast-paced nature of evolving cyber threats. In addition, machine learning is an area of constant evolution, which means being familiar with the current cutting-edge techniques and the ability to leverage them properly for security purposes is an ongoing educational process [10]. This article provides a robust and computationally efficient technique for intrusion detection utilizing a machine learning algorithm to build a robust machine learning IDS. The contributions of this research are:

- A unified, end-to-end reproducible IDS framework that includes a complete, ready-to-implement machine learning pipeline for network intrusion detection that consists of: data cleaning, type-aware pre-processing, class imbalance handling, feature selection, model training, hyperparameter optimization with K-fold cross-validation, final evaluation on held-out test data, and SHAP explainability.
- Comparative evaluation and design using multiple datasets. This research involves systematically comparing different benchmark datasets, including both legacy and modern, such as KDDCup99, CIC-IDS2017 and Hikari-2021, to provide a coherent understanding on how the different models behave in dissimilar feature spaces, attack types and traffic distributions; moreover, it highlights the strengths and weakness of generalization of these models.
- Rigorous optimization of tabular ML models for IDS. Through the application of Bayesian optimization in conjunction with stratified cross-validation, robust hyperparameters were discovered for a selection of state-of-the-art ensemble and tree-based model families while providing comprehensive evaluation metrics (including accuracy, precision, recall, F1-score, AUC, etc.) that allow for a rigorous comparison against previously published literature.
- Integrated SHAP-based explainability. SHAP is applied to the optimized models to find out which features have the highest influence on the optimized models.

This work aims to enhance cyber threat prediction and prevention by developing a robust machine-learning-based intrusion detection framework evaluated across multiple benchmark datasets. The methodology and findings presented in this paper can be applied by cybersecurity researchers and practitioners, like network administrators and system designers, to create reliable, generalizable and interpretable intrusion detection systems. The proposed

leakage-safe, multi-dataset-based machine learning (ML) framework allows for reliable model selection and deployment in a variety of different networking environments. Additionally, practitioners are provided with resources for balancing the performance of intrusion detection systems with computational efficiency, which leads to the development of scalable and practical cybersecurity solutions that can be used in both real-world and resource-constrained environments.

The rest of this article is organized as follows. In section 2 provides an overview of the relevant literature regarding ML. Section 3 describes proposed pipeline, which includes Pre-Processing, Feature Selection, Model Families, and Optimization Strategies. In section 4, a discussion about the performed experiments and findings is conducted. Finally, the section 5 contains the conclusion of this paper.

2. RELATED WORK

This section presents earlier research on machine learning and detection systems for intrusions, exclusively dealing with HIKARI-2021, CIC-IDS2017, and KDDCup99 datasets. The purpose of this section is to reviews and summarizes the major methodologies gaps in the field of IDS development, which provide the basis for the current study. Earlier work used classical datasets such as KDDCup99 and NSL-KDD to benchmark algorithms and select features. As the network environment and attack behaviors became more complex, the research community began moving towards modern datasets such as CIC-IDS2017 and HIKARI-2021, which include more realistic representations of traffic patterns, encrypted sessions, and modern attack vector [11], [12]. In parallel, the research community was advancing ensemble learning, adversarial ML, and explainable AI (XAI), which shifted the focus of IDS research from accuracy to interpretability, robustness, and generalizability [13].

In the context of NIDS robustness, Tariq et al. [14] developed an "Intelligent Cyber Security Framework" based on an ensemble meta-classifier comprising a selection of traditional ML and DL models, specifically Random Forest, Decision Tree, Gaussian Naive Bayes, K-Nearest Neighbors, Logistic Regression, MLP and CNN, trained on HIKARI-2021 in their study. The methodology included pre-processing (encoding to deal with categorical features, an 80/20 train-test split and hyperparameter tuning prior to ensemble learning). The result was a reported performance improvement over single learners, with the ensemble achieving 96.3% accuracy (and good precision/recall/F1 scores)—this suggests that ensemble learning will yield a more robust IDS for modern encrypted traffic.

Similarly, Vitorino et al. [15], in a company-centric robustness analysis, used tree-ensemble models (Random Forest (RF) XGBoost (XGB) LightGBM (LGBM) Explainable Boosting Machine (EBM)) and evaluated them on CICIDS2017 through a corrected "NewCICIDS" and HIKARI. Their methods set a standardized benchmark as indicated by (i) normal training and (ii) adversarial training with the A2PM constrained-perturbation method and evaluated clean as well as model-target adversarial hold-out datasets (5-fold CV and grid search for tuning). The results indicated that NewCICIDS improved performance generally (XGB and EBM especially), adversarial training improved robustness without injury to generalization nor improved false positive rates, whereas robustness of the more recent traffic HIKARI was more difficult—XGB/EBM retained better precision through RF/LGBM was less robust—indicating the difficulties within the domain of modern enterprise encrypted traffic patterns.

In contrast, Ferriyan et al. [12], in their work, constructed a fresh benchmark dataset for NIDS that combined real encrypted benign/background traffic with scripted application-layered attacks, derivatively producing ~86 Zeek/CICFlowmeter-style features, designating flows into benign/background and four distinct "attack" varieties (Bruteforce, Bruteforce-XML, Probing, XMRIGCC). The method articulated the "content" and "process" requirements requisite for usable and reproducible datasets (inclusive capture with pcaps + flows, payload availability granted with privacy preserving anonymization, explicit ground truth, and encryption data). The result was the openly available HIKARI-2021 dataset and a comparative study (analysis) which explicitly showed how it complemented/updated legacy dataset's values such as KDD99, UNSW-NB15, and CIC-IDS2017.

In a study by Soumik [16], the CIC-IDS2017 dataset was implemented using a full AI/ML pipeline (data cleaning, SMOTE class balancing, Min-Max normalization feat. selection) and performance analyses were conducted over various learners, such as (RF, DT, Stacked LSTM and AdaBoost). Methodologically, the data were partitioned 80/20 (train/test) after pre-processing, and a comparative study was conducted across the models. Results showed that Random Forest was the best-performing learner, achieving approximately 99.90% accuracy, good precision/recall/F1, and suggesting future exploration of hybrid AI models and feature engineering for IDS.

In the paper by Farhat et al. [17], the target of DoS/DDoS detection in cloud-deployed environments is addressed together with the use of CIC-IDS2017 data sets (Wednesday and Friday traces) to perform comparative tests on various ML algorithms in particular XGBoost. The aims and methodology employed consisted of the definition of the DoS/DDoS subsets, appropriate features for the specific attack family used, definition of the common IDS parameters (i.e., accuracy, precision, recall, F1, false alarm rate), and experiments to allow a balance between speed and quality of detection. In summary, it was established that XGBoost could achieve a 99.11% detection accuracy and ~0.011% false alarm rate indicating high performance for detecting credible efficiencies. An indication of the computational trade-offs to be made when seeking detection quality and the important characteristics of careful choice of features.

Cantone et al. [18] focused at the matter of cross-dataset generalization in ML-based NIDS. They set four supervised classifiers, Linear Discriminant Analysis, Decision Tree, Random Forest and XGBoost, to work on four different datasets, namely CIC-IDS2017, CSE-CIC-IDS2018, LycoS-IDS2017 and a new dataset they called LycoS-Unicas-IDS2018. The methodology examined between within-dataset splits as well as cross-dataset testing, binary (benign vs. malicious) and single-attack testing, mRMR feature-selection and extensive visualization were adopted to demonstrate the differences in feature/label heterogeneity. The outcome was interesting: almost perfect performance in the same dataset but accuracy results were almost close to random for most cross-dataset settings. The authors attributed these failures to dataset anomalies/heterogeneity and stressed the necessity of taking into account the differences between datasets in real-world NIDS, while donating the LycoS-Unicas-IDS2018 resource.

In a study by Bitra et al. [19], the KDDCup99 dataset was used to employ four classifiers: K-nearest neighbor (KNN) classifier, support vector machine (SVM), random forest (RF), and LightGBM (LGBM). The methodology of Bitra et al. included the selection of a dataset, cleansing and normalization of the dataset, feature selection based on a correlation matrix, model training through train/test splits, and evaluation of results by corresponding accuracy, precision, recall, F1 and confusion matrices to select the best model for the IDS. The results showed that RF outperformed the others (LGBM and SVM ranked second), while KNN performed the worst; thus, from the authors' point of view, RF would be the most efficient model to use for intrusions in KDDCup99.

Li et al. [20] utilized a 10% stratified subsample of KDD'99 for detection of Denial-of-Service traffic with three supervised models of Logistic Regression, Naïve Bayes, and also an Artificial Neural Network (ANN). They identified, one-hot encoding categorical features and data split into training/testing subsamples, 75/25 as well as 70/30 and 80/20 variations, comparing the classifiers using accuracy, ROC, and balanced accuracy metrics on a more or less imbalanced setting. The results indicated that ANN outperformed Naïve Bayes and Logistic Regression across all evaluated metrics.

In contrast, Hossain and Islam [5] proposed an ensemble-based IDS assessed on several public datasets (e.g., NSL-KDD, UNSW-NB15, CSE–CIC–IDS2018, WSN-DS, NF-UQ-NIDS, NF-ToN-IoT, CCC) that used pre-processing features and feature selection (correlation analysis, mutual information and PCA) and ensembles (Random Forest, Bagging, AdaBoost, Gradient Boosting, XGBoost, Simple Stacking) that were then evaluated. Their techniques were described in detail, including preprocessing stages and testing methods, as well as multi-metric evaluation metrics (accuracy, precision, recall, F1, AUC, balanced accuracy, and Cohen's κ), with comprehensive results tables. It was shown that the Random-Forest-based algorithms generally outperformed others, often achieved accuracies greater than 99% with small FPR and the authors stated that the ensembles were effective across datasets.

Tripathy and Behera [21] explored intrusion detection on the KDD Cup '99 dataset by implementing and comparing a broad range of ML classifiers (e.g., Logistic regression, Decision tree, KNN, naïve Bayes variants, SVM, Random Forest, AdaBoost, XGBoost, Passive–Aggressive, Ridge, Rocchio, Perceptron/ANN, SGD/BPN) The methodology is organized into a five-stage IDS pipeline, comprising data gathering, pre-processing, feature extraction, feature selection, detection. The evaluation is based on accuracy, precision, recall and F1. The result is a comparative performance evaluation to reduce false positives and provide guidance for selecting models. In terms of accuracy Support Vector Machine gives the highest accuracy rate 98.08%. Table 1 presents previously published related works and highlights the methodological gaps identified in each study.

Table 1: List of Methodological Gaps Identified in Existing Intrusion Detection Literature

Papers and Reference	Dataset Used	Feature Selection	k-fold CV	Resampling	Hyperparameter tuning	Explainability
An Adversarial Robustness Benchmark for Enterprise Network Intrusion Detection. (2023) [15]	Hikari-2021 and CICIDS2017	YES	YES (5-fold CV)	NO (Just adversarial data augmentation)	YES (Grid search)	NO
Generating Network Intrusion Detection Dataset Based on Real and Encrypted Synthetic Attack Traffic. (2021) [12]	Hikari-2021	NO	NO	NO	NO	NO
Intelligent Cyber Security Framework for Threat Detection using Ensemble Learning Techniques. (2025) [14]	Hikari-2021	NO	NO	NO	YES	NO
A comparative analysis of Network Intrusion Detection (NID) using Artificial Intelligence techniques for increase network security. (2024) [16]	CIC-IDS-2017 FridayDDos	YES	NO	YES (SMOTE)	NO	YES (Simple feature-importance explainability, but not advanced XAI)
Evaluation of DoS/DDoS Attack Detection with ML Techniques on CIC-IDS2017 Dataset. (2023) [17]	CIC-IDS-2017 FridayDDos	YES	NO	NO	NO	NO
On the Cross-Dataset Generalization of Machine Learning for Network Intrusion Detection. (2024) [18]	CIC-IDS2017, CSE-CIC-IDS2018, LycoS-IDS2017 and LycoS-Unicas-IDS2018	YES	NO	YES (Down sampling)	YES (Grid search)	YES (Dataset/feature visualizations as explainability, not SHAP/LIME.)
Comparative analysis on intrusion detection system using machine learning approach. (2024) [19]	kddcup99	YES	NO	NO	NO	NO
Denial of Service (DoS) Attack Detection: Performance Comparison of Supervised Machine Learning Algorithms. [20]	kddcup99	NO	NO	NO	NO	NO
Ensuring network security with a robust intrusion detection system using ensemble-based machine learning. (2023) [5]	NSL-KDD, UNSW-NB15, CSE–CIC–IDS2018, WSN-DS, NF-UQ-NIDS, NF-ToN-IoT, CCC	YES	NO	NO	NO	NO
Performance evaluation of machine learning algorithms for Intrusion Detection Systems. (2023) [21]	kddcup99	YES	YES (10-fold CV)	NO	NO	NO
This Work	Hikari-2021, CICIDS2017 and kddcup99 (Legacy + Modern)	YES	YES (5-fold cross-validation)	YES (Over-sampling (SMOTE))	YES (Bayesian search)	YES (SHAP)

The existing literature on intrusion detection has many limitations. Most of this research relies on either single (where most IDSs have static models and lack an effective way of adapting them to changing threats) or legacy datasets creating poor cross-dataset generalizability with limited applicability to real-world settings [12], [14], [16], [17], [19], [20], [21].

In addition, the evaluation of these models is often poorly executed methodologically, particularly with insufficient use of either stratified or nested methods for conducting

cross-validation, which can lead to an increased potential for data leakage and inflated performance estimates [21], [12], [14], [16-20].

Moreover, many models suffer from class imbalance and thus fail to recognize attacks against minorities [21], [12], [14], [15], [17], [19], [20], [21].

Furthermore, the majority of research in this area has relied on default or ad-hoc hyperparameter tuning rather than using systematic optimization techniques such as Bayesian search [21], [12], [16], [17], [19-21].

Finally, a major consideration for many applications of IDS technology is that many do not adequately balance computational efficiency with interpretability, which prevents them from being deployed successfully in both operational and resource-limited environments [21], [12], [14-21].

This study addresses these gaps by jointly incorporating dataset diversity, leak-proof validation, imbalance handling, Bayesian optimization, scalable feature reduction, and explainable ML within a unified evaluation framework.

3. PROPOSED METHODOLOGY

This section provides a detailed description of the proposed machine learning-based model development pipeline for detecting intrusions. Figure 1 shows the proposed architecture of the network intrusion detection system (NIDS), which has been evaluated using three benchmark datasets: (1) HIKARI-2021; (2) CICIDS-2017; and (3) KDDCUP99. The raw network traffic flows are pre-processed and filtered to include only those features that provide the most informative attributes. Eight machine learning classifiers were trained using stratified 5-fold cross-validation with SMOTE to handle class imbalance, and optimized using Bayesian hyperparameter tuning. Then, the trained models classify the network traffic as either benign or malicious, with performance assessed according to common performance metrics. Lastly, SHAP is used to provide transparency and interpretability into how these models make their predictions.

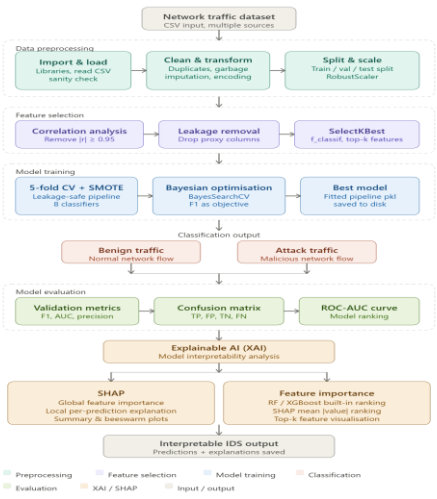

Figure 1: System architecture of our proposed framework

Algorithm 1. Overall Pipeline of the Proposed Network Intrusion Detection System

Require: Network traffic dataset D with feature matrix X and binary or multi-class labels y, stored in CSV format.

Ensure: Trained and optimized classifier ensemble M, evaluation metrics (Accuracy, Precision, Recall, F1-score, ROC-AUC), ranked model comparison, and serialized best model artifact.

1. **Data loading:** Read D into a DataFrame; display rows. Set global seed S = 42 for reproducibility.
2. **Duplicate and garbage value treatment:** Remove exact duplicate rows. Identify categorical columns and scan for garbage tokens (e.g., "unknown", "?", "-", "null", "N/A", ""); replace each with the column mode. For numerical columns, convert any physically impossible negative values to their absolute values.
3. **Exploratory data analysis (EDA):** Plot KDE histograms and boxplots for numerical features; count the IQR-based outliers per column. Compute a full pairwise Pearson correlation matrix. For all feature pairs with $|r| \geq \tau$ (e.g., $\tau = 0.95$).
4. **Leakage column removal:** Drop index artifact columns, direct label-proxy columns (e.g., traffic category labels that subsume the target label y), and all highly correlated redundant features identified in Step 4. Record the final retained feature set.
5. **Train / validation / test split:** Apply stratified splitting to preserve class proportions across all three partitions: train (70%), validation (15%), test (15%). Verify that class ratios are consistent across splits.
6. **Missing value imputation:** Replace $\pm\infty$ with NaN. Fit a median imputer on X_train for numerical columns and a mode imputer for categorical columns; apply the fitted imputers to validation and test sets without re-fitting. Confirm zero missing values across all splits post-treatment.
7. **Feature encoding:** Rule-based encoding. For high-cardinality categorical columns, apply frequency encoding. For binary categoricals, apply label encoding. For low-cardinality nominal columns (2–9 unique values), apply one-hot encoding with column alignment across splits.
8. **Feature selection:** Fit SelectKBest with a univariate statistical scoring function (e.g., f-statistic for classification) on X_train, retaining the top k features.
9. **Feature scaling:** Fit a robust scaler (IQR-based, quantile range 25–75) on X_train; transform validation and test sets using the fitted scaler only.
0. **Save preprocessed artefacts:** Write the scaled train, validation, and test sets (with labels) to disk.
1. **Model configuration:** Instantiate a set of classifiers (e.g., Decision Tree, Random Forest, XGBoost, MLP). Apply regularization-aware default hyperparameters to each. Define a Bayesian search space per classifier.
2. **K-fold stratified cross-validation:** For each classifier, imblearn.Pipeline: SMOTE(sampling_strategy = 0.30, k_neighbors = 2).→ Classifier. Run cross_validate - CV = 5-fold stratified and with n_jobs=8.
3. **Bayesian hyperparameter optimization:** For each classifier, wrap the same pipeline in BayesSearchCV (RF surrogate, n_iter = 10 general / 5 for MLP, inner CV = 3-fold stratified). Apply DeltaYStopper(δ = 1e-4, n_best = 3) as early-stop callback.
4. **Validation evaluation:** For each optimized pipeline, compute Accuracy, Precision, Recall, F1, and ROC-AUC. Plot confusion matrix and ROC curve per model.
5. **Final test evaluation:** Repeat Step 15 on the held-out X_test. Save per-model metrics to a summary CSV. Generate confusion matrices and ROC curves for all models.
6. **Results analysis and ranking:** Merge CV, validation, and test metrics into a unified comparison table. Rank models by Test-F1 → Test-ROC-AUC → Test-Accuracy.

3.1 Datasets Description

This research makes use of three benchmark intrusion detection datasets, HIKARI- 2021, CIC-IDS2017, and KDDCup99, to assess the robustness of the presented machine learning architecture as well as the generalizability of its results.

3.1.1 HIKARI-2021

The HIKARI-2021 was created by Ferriyan et al. [12] which is a modern intrusion detection dataset meant to provide encrypted and realistic network traffic data, including a mix of real and synthetic attacks, over HTTPS (TLS 1.2). The data consists of over 555,278 flow records with 86 features obtained through the use of the Zeek framework. The attack types include brute force login attacks, XML-RPC variants, probing attacks, and crypto mining sessions, among the anonymized benign traffic. The distribution of classes of about 93.6% normal and 6.4% attack. Together with processed PCAP files, labelled flows, and many applications layer attack types, HIKARI-2021 provides a complete benchmark for testing machine learning models that aim at detection of encrypted network intrusion detection.

3.1.2 CIC-IDS-2017 FridayDDos

The Friday-DDoS subset of CIC-IDS-2017 was created at the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick by Sharafaldin et al. [11], [22] contains labelled bidirectional flow records reflecting some benign and DDoS traffic samples collected during a controlled enterprise testbed of July 7, 2017. The Friday Working Hours Afternoon DDoS file encompasses 225,745 flows with 79 produced features extracted using the

CICFlowMeter, which is commonly used as compact binary classification benchmark (BENIGN vs DDoS) with the class distribution of DDoS samples being approx. 57.42% and benign samples being approx. 42.58%, and is hosted on Kaggle thus widely available for reproducible machine learning based intrusion detection research.

3.1.3 Kddcup99

KDDCup99 dataset was created in 1999 by Stolfo et al. [23] which is a legacy benchmark for intrusion detection that was created from the 1998 DARPA offline evaluation. The data consists of TCP/IP connections that have been engineered to yield 41 features with a binary label of normal or attack. There are 494,020 samples in the dataset with a distribution of classes of about 60.3% normal and 39.7% attack. Although the dataset is commonly used, KDD-99 is also known to contain duplicate and redundant entries that may be responsible for inflated model performance; therefore, NSL-KDD is much preferred in current research. The dataset is available at UCI and Kaggle for reproducible machine learning experiments.

3.2 Experimental Setup

For the experiments of this study, the following configuration was used. These tools, Library, and strategy allowed for a robust and reproducible machine learning pipeline to perform intrusion detection and cyber-threat prediction.

Table 1: Experimental Setup — Hardware, Software, and Parallelization Configuration

Hardware	The models have been trained on a platform with Intel Core i7 processor, 16GB of RAM, and Nvidia GEFORCE RTX 4050 graphic card which is enough to handle the computational power of deep-learning networks.
Library and software	The Python 3.13.7 was utilized in a Jupyter Notebook environment from Anaconda distribution to manage dependencies properly and utilize a controlled execution space. The large datasets of network traffic were handled, cleaned, and manipulated using libraries for data processing such as Pandas and NumPy. The pipeline was developed with the use of modules from scikit-learn "(sklearn)" for pre-processing, feature selection, model training and evaluation. The performance of gradient boosting was made possible with XGBoost, while visual analytics and correlation heatmaps made used of Matplotlib and Seaborn. For hyperparameter optimization, Bayesian Optimization (BayesSearchCV) was employed from the Scikit-Optimize library and the Imbalanced-learn (SMOTE) package was employed to address class imbalance in the training data.
Parallel Strategy	OUTER_PARALLEL. OUTER_N_JOBS = 8. Avoiding nested threading the environment variable is set to: (-os.environ["OMP_NUM_THREADS"]="1"), (-os.environ["MKL_NUM_THREADS"]="1"), (-os.environ["NUMEXPR_NUM_THREADS"]="1"), (-os.environ["OPENBLAS_NUM_THREADS"]="1"), (-os.environ["VECLIB_MAXIMUM_THREADS"] = "1"),

3.3 Data Pre-processing

The cleaning and transforming of data are an essential step before using it with machine learning algorithms, this step is referred to as data pre-processing. Many aspects of machine learning model results depend on the quality of the input data therefore, the pre-processing of input data ensures the consistency, accuracy and value of that input data to create a better machine learning model. Data pre-processing can eliminate inconsistencies and errors in input data, can standardize input data, making it easier to compare and analyze input data, and can make input data more easily manageable and usable by machine learning models [24].

A methodological approach consisting of a structured pre-processing pipeline was utilized for cleaning and transforming the dataset prior to training a model as the main technical method of the study. The pipeline consists of a series of sequentially ordered steps that are shown in Table 3 below**.**

Table 2: Data Pre-processing Pipeline Steps and Actions

Step	Action
Duplicate removal	Using the Pandas library, duplicate entries were identified and then removed from the dataset to achieve a unique dataset.
Invalid Value Handling	Infinite values, extremely large values, and any other invalid or inconsistent data were replaced with "NaN" through the use of Pandas and NumPy utilities.
Missing Value Removal	Any row that contained at least one Missing Value was also removed from the dataset so that the resulting dataset was clean and trustworthy.
Feature Type Separation	The cleaned dataset was separated into Numerical and Categorical data types to allow appropriate processing for each.
Numerical Standardization	The Standard Scaler from Scikit-Learn was applied to all numerical features to ensure uniform scale across all numerical variables.
Categorical Encoding	The Categorical variables were converted from categorical to numerical formats through the Scikit-Learn library: Label Encoding for simple categories, and One-Hot Encoding or Frequency Encoding was applied for High Cardinality Features.
Target Label Transformation	The Label column was transformed into a binary number - 0 for benign traffic and 1 for attack traffic - in order to have a constant target class for the supervised classification process.
Feature-Target Split	The dataset was partitioned into feature variables (X) and the target variable (y), with the Label column serving as the output class.

Because the Label field was already binary (0 = benign, 1 = attack), no additional discretization was required, and the models were trained directly using these original classification labels. The systematic preprocessing pipeline utilizing Pandas, NumPy, and Scikit-Learn has produced a clean, consistent, and comprehensive dataset for the machine learning models used in this study.

3.4 Dataset Splitting

The dataset is divided into training, validation, and test subsets using the function train_test_split from the sklearn (scikit-learn) library. The split was done using stratified sampling. First 70% of the data is taken for training purposes while the other 15-15% is taken for the initial validation and test set, immediately after cleaning and duplicate removal, but before any feature engineering and selection, in order to reduce the risk of data leak and provide a more realistic measure of how well the model generalizes. The purpose of this three-way split is to create separate training, optimization, and testing dataset for the proposed intrusion detection models, in order to prevent data leakage, and ensure an accurate evaluation of the effectiveness of the pipeline.

3.5 Feature selection

Feature selection is an important phase while developing a machine learning model to detect network intrusion. Not every feature available in the dataset may be usable, nor will they equally contribute to predicting network intrusion [25]. In this proposed method, two different Filter-Based Feature Selection methods were used: Correlation Analysis and SelectKBest with ANOVA F-Score for their application. The combination of these feature selection techniques enables the creation of a more compact, efficient and high-performance feature space which save computational efficiency and increase predictive stability of intrusion detection systems [25], [26]. A brief overview of the two approaches is provided below.

3.5.1 Correlation Analysis

Correlation analysis is important for feature selection in machine learning because it helps identify highly correlated features that may cause overfitting or redundancies in the model [27]. During the implementation of our predictive model, we conducted a correlation analysis to determine the set of potential input features to use. This is accomplished by computing and analyzing a "correlation matrix," or "corr," of input features from the input data set and determining those features that possess an absolute correlation coefficient greater than 0.95.

The correlation coefficient helps determine how strongly or weakly two variables (X and Y) are linearly related. The correlation coefficient can be defined mathematically as the covariance between input data variables X and Y divided by the total of their respective standard deviations or standard deviation sum. This is represented in its mathematical form by the following equation.

$$corr(X,Y) = \frac{cov(X,Y)}{(std(X) * std(Y))} \quad (1)$$

Where:

- Corr(X,Y) represents the correlation coefficient between variables X and Y.
- cov(X,Y) represents the covariance between X and Y.
- std(X) represents the standard deviation of variable X.
- std(Y) represents the standard deviation of variable Y.

When two features are highly correlated, one of those features was removed to help reduce multicollinearity and improve the generalization of the model.

3.5.2 ANOVA F-Score

Choosing the best K numerical features which are the most strongly related to a categorical target (Label) variable. This is used in classification problems in which targets are categorical and features are continuous. It checks whether the means are significantly different across the groups. ANOVA is a fast efficient way of selecting the most discriminative features for classification problems, especially for continuous features when the target variable is categorical [28].

The ANOVA F-score is calculated as the ratio of between-group variability to within-group variability, as represented by the following equation:

$$F = \frac{Between - group\ variability}{Within - group\ variability} \quad (2)$$

where:

- F represents the ANOVA test statistic used to measure feature significance.
- Between-group variability refers to the variation of feature means between different target classes.
- Within-group variability refers to the variation of feature values within the same target class.

A feature that has a higher F-score is more relevant than a feature with a lower F-score because it helps to discriminate between target classes.

3.6 Handling of Class Imbalance

To resolve the imbalance between normal and attack traffic in the datasets, a number of specific steps are required in order to ensure a fair representation of all classes. The Synthetic Minority Oversampling Technique (SMOTE) is one of the techniques that will be used to generate synthetic instances of the minority (attack) classes, which is done by interpolation with the existing nearest neighbors [29]. In this way, it is possible to increase the percentage of the minority samples without making duplicates. SMOTE enhances recall for minority classes and reduces overfitting when compared to random oversampling, especially when applied after robust data cleaning has occurred [30].

3.7 Hyperparameter Optimizations

To optimize the models, the Bayesian Optimization is more intelligent and efficient method that explore the hyperparameter space by adjusting the prior distributions of the hyperparameters according to previously evaluated configurations so that optimal values may be reached quickly [31]. Bayesian Optimization, using this technology together with Stratified K-Fold cross-validation, will optimize hyperparameters efficiently for better generalization of the models than those produced by standard grid or random search techniques. Stratified K-Fold CV will preserve the original class distribution across the folds which translates to being able to carry out performance estimations without bias and in a reliable way [32]. This combined approach will thus lead to an effective trade-off between efficiency of exploration, computational cost and robustness of the model.

3.8 Training using k-fold stratified cross validation

The method k-fold cross validation is a resampling technique and a validation method in which the data divides into k equal folds, preserving the arrangement of distributions of each of the classes in the partitions. One-fold for validation plus other left-over folds were for training in each iteration, with the process repeated once all folds had served as the validation set [33]. This research utilized K-fold stratified cross-validation (with 5 folds) alongside Bayesian optimization of hyperparameters to find the best values for the learning rate, dropout rate, and gradient clipping threshold. SMOTE is applied inside each

training fold as it generated synthetic samples of the minority class by interpolating between existing instances. This design rigorously and also in an unbiased way evaluated all of the candidate classifiers under conditions that were balanced for training.

3.9 Evaluating and Comparing Model Performance

Machine Learning's ultimate goal is to produce a model that accurately predicts future results based on new Data (unseen Data). The methods we use to evaluate a model's success in accomplishing its objective are referred to as Evaluation Metrics. The evaluation metrics of the model will be evaluated based on key classification metrics which contain Accuracy (ACC), Precision (PRE), Recall (REC), and the given F1-Score (F1). Additionally, the area under the ROC model (AUC) will be computed. All combined this gives a robust model for classification effectiveness in the balanced and imbalanced state [5], [34]. These measures are briefly described as follows:

Accuracy: Overall fraction of correct predictions [5], [34].

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad (3)$$

Precision: Of all of the events the model flagged as being a threat, what was the alert purity percentage.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4)$$

Recall: Of all actual threats, how many actual threats did the model catch with sensitivity/true positive rate.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (5)$$

F1: Harmonic mean of Precision and Recall.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 * (\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (6)$$

ROC AUC: Area under the Receiver Operating Characteristic curve, plotting TPR (= Recall) against FPR (= FP/(FP+TN)) over all thresholds.

$$AUC - ROC = \int TPR\,(FPR) dFPR \quad (7)$$

For Equations 3, 4, 5, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

3.10 SHAP (SHapley Additive exPlanations) Explainability

The use SHAP explainability adds an additional way to interpret machine learning models based on a theory-driven approach. Using a model of cooperative games theory, SHAP assigns a contribution value for each feature so that the model can be described in a clear and accurate way [35]. Summary and dependence plots were also developed using SHAP for the trained classifiers to find the most significant features and to see how those features impacted the prediction of an intrusion. Using SHAP increases the transparency of the models (e.g., XGBoost, Random Forest and neural network) and provides better explanations than traditional importance scores, particularly on large datasets [35]. This study integrates the SHAP explainabiltiy to enhance the reliability and analytical capability of the IDS pipeline.

3.11 Algorithms

Eight supervised machine learning classification algorithms were used in this study to detect and classify network intrusion traffic. Selection of each algorithm was based upon its suitability as a binary classification technique, its ability to manage large quantities of network traffic data, and the established use of each within the body of literature associated with intrusion detection. Additionally, Supervised Machine Learning works better on tabular datasets than unsupervised machine learning because of the presence of labeled targets, allowing supervised machine learning algorithms (like XGBoost, LightGBM, and Random Forest) to establish clear input-output relationships. The predictive accuracy of an unsupervised learning method on tabular data is lower than supervised learning because unsupervised methods do not rely on external targets for training and thus have limited ability to discover patterns within the data [36]. A very brief description of the algorithms is given below in Table 3:

Table 3: Comparative Overview of Machine Learning Classification Algorithms Employed in this Study

Algorithms	Type	Key Mechanism
Random Forest (RF)	Ensemble Learning	Utilizes bagging as well as random selection of features to create a collection of uncorrelated decision trees, the final classification of which will result from the majority vote across all trees [37].
Decision Tree (DT)	Tree-based Learning	Recursively divides the feature space from top to bottom in order to conduct a greedy search process to find optimal split points for classifying samples based on either Gini impurity or information gain, and continue until all samples are classified into label classes. [31].
Extreme Gradient Boosting (XGB)	Ensemble Learning (Gradient Boosting)	XGBoost works by building an ensemble of decision trees sequentially, with each new tree correcting the errors of the previous one. XGBoost is a supervised learning algorithm that increases the gradient descent algorithm. The model performance increases because using regularization techniques and advanced optimization techniques, it reduces overfitting [38].
Artificial Neural Network (ANN; Multilayer Perceptron)	Deep Learning	An artificial neural network (ANN) is produced by imitating the structure and functions of the interlinked network of neurons of the human being's brain. By using a series of inter-related neuron layers, the neural network can take input signals in, process them and send them on through to the next layer. The neural network has the ability to discover very complex non-linear relationships between features and output classes [31].
Logistic Regression (LR)	Statistical / Linear	The algorithm classifies i.e., it predicts the outcome of either discrete or categorical outcome. The prediction here is probability of the classification of the input whereas linear regression used to predict values. The classification is done using the binary data such as Yes / No, True / False, or 0 / 1. It is assigning a predicted value of the input into a value between 0 and 1 using the sigmoid function [37].
K-Nearest Neighbors (KNN)	Instance-Based Learning	The KNN method assumes nothing at all with respect to the distribution of the underlying data set, and thus learns in a non-parametric manner and instance-based way. K-Nearest Neighbors is also known as lazy learning algorithm as it does not actually learn from the training data set immediately but simply keeps the data set with itself and when it has to classify then applies itself on the data set [31].
Linear Support Vector Machine (LSVM)	Kernel-Based Learning	A Support vector machine (SVM) finds the best boundary available. This boundary is exactly a hyperplane which distinguishes between the different classes in all of the data given. SVMs have a maximization goal of that margin between the two classes. The wider the margin is the better that the model will perform on new and unseen data. Linear SVMs work with data which is separable linearly. Thus, there is no need for transformations of the data to be made to separate it into classes [31].
Naïve Bayes (NB)	Probabilistic Learning	Naïve Bayes uses principles of probability for classification tasks. It is part of the family of generative learning algorithms, this means that it attempts to model the distribution of inputs of each class of some kind. The basis of Naïve Bayes is in fact Bayes' Theorem which is probably better known as a probability classifier. It is important to arrive at a basic description of Bayesian statistics. Without this is only possible to understand with great difficulty this algorithm [39].

4. RESULT

In this section, we evaluate the performance of the proposed framework for IDS on the three datasets that have been employed in this study.

The parameter tuning shown in Table 5 and Table 6 was done by observing which yields the good results with the models. In order to assess the model stability and robustness across dataset, the same parameter settings were used in HIKARI-2021, CIC-IDS2017, and KDDCup99 dataset. This strategy takes away tuning bias and makes it easier to evaluate how consistent the predictive behavior of the model is, across a multitude of network environments.

Table 4: Default Parameters for each Classifier

Classifiers	Parameters
Random Forest	random_state=SEED, n_jobs=INNER_N_JOBS, n_estimators=450, max_depth=16, min_samples_leaf=2, min_samples_split=4, max_features="sqrt", bootstrap=True, max_samples=0.8, warm_start=False
XGBoost	random_state=SEED, eval_metric="logloss", n_jobs=1 if USE_OUTER_PARALLEL else 18, tree_method="hist", verbosity=0, objective="binary:logistic", n_estimators=350, max_depth=5, learning_rate=0.12, subsample=0.8, colsample_bytree=0.8, colsample_bynode=1.0, min_child_weight=5, gamma=0.0, max_bin=255, reg_lambda=5.0, reg_alpha=0.1
Decision Tree	random_state=SEED, max_depth=10, splitter="best", min_samples_split=6, min_samples_leaf=3, criterion="gini"
Neural Network	random_state=SEED, solver="adam", early_stopping=True, validation_fraction=0.15 n_iter_no_change=15, max_iter=1000, alpha=1e-3, hidden_layer_sizes=(64, 32), learning_rate_init=1e-3, batch_size=1024, activation="relu", shuffle=True
KNN	n_neighbors=7, weights="uniform", p=2, algorithm="brute", n_jobs=INNER_N_JOBS
Linear SVM	random_state=SEED, max_iter=1000, C=0.5, dual=False, tol=1e-3, penalty="l2", loss="squared_hinge"
Logistic Regression	random_state=SEED, solver="saga", penalty="l2", C=0.5, max_iter=1000, tol=1e-4, fit_intercept=True, n_jobs=None
Naïve Bayes	var_smoothing=1e-9

Table 5: Custom Bayesian search space Parameter for each Classifiers

Classifiers	Parameters
Random Forest	"n_estimators": Integer(300, 700), "max_depth": Integer(12, 28), "min_samples_split": Integer(2, 32), "min_samples_leaf": Integer(1, 16), "max_samples": Real(0.5, 0.9), "ccp_alpha": Real(1e-7, 1e-3, prior="log-uniform"), "max_features": Categorical(["sqrt", "log2", 0.3, 0.5, 0.8]), "criterion": Categorical(["gini", "entropy", "log_loss"]), "min_impurity_decrease": Real(1e-9, 1e-3, prior="log-uniform"), "max_leaf_nodes": Integer(64, 2048
XGBoost	"n_estimators": Integer(300, 700), "max_depth": Integer(4, 7), "learning_rate": Real(0.06, 0.18, prior="log-uniform"), "subsample": Real(0.6, 1.0), "colsample_bytree": Real(0.5, 1.0), "min_child_weight": Integer(1, 12), "gamma": Real(0.0, 8.0), "reg_lambda": Real(1.0, 20.0, prior="log-uniform"), "reg_alpha": Real(1e-3, 10.0, prior="log-uniform"), "max_bin": Integer(128, 512),
Decision Tree	"max_depth": Integer(10, 24), "min_samples_split": Integer(4, 24), "min_samples_leaf": Integer(2, 8), "max_leaf_nodes": Integer(16, 1024), "max_features": Categorical([None, "sqrt", "log2", 0.5, 0.8]), "criterion": Categorical(["gini", "entropy"]), "min_impurity_decrease": Real(1e-9, 1e-3, prior="log-uniform"), "ccp_alpha": Real(1e-6, 1e-2, prior="log-uniform")
Neural Network	"hidden_layer_sizes": Categorical(_HLS, transform="identity"), "alpha": Real(1e-6, 1e-2, prior="log-uniform"), "learning_rate_init": Real(1e-4, 3e-3, prior="log-uniform"), "learning_rate": Categorical(["constant", "adaptive"]), "activation": Categorical(["relu", "tanh"], transform="identity"), # Optional (uncomment if you want to tune it): "batch_size": Categorical([256, 512, 1024]),
KNN	"n_neighbors": Integer(3, 11), "weights": Categorical(["uniform"]), "p": Categorical([2]),
Linear SVM	"C": Real(1e-3, 8.0, prior="log-uniform"), "penalty": Categorical(["l2"]), "loss": Categorical(["squared_hinge"]), "dual": Categorical([True, False])
Logistic Regression	"C": Real(0.1, 2, prior="log-uniform"), "tol": Real(1e-4, 2e-3, prior="log-uniform"), "penalty": Categorical(["l2", "l1"]), "fit_intercept": Categorical([True, False])
Naïve Bayes	"var_smoothing": Real(1e-12, 1e-6, prior="log-uniform")

4.1 Result of the Evaluation Metrics

Table 7 outlines the performance metrics obtained from the evaluated machine learning models. The eight machine learning models were evaluated on the HIKARI 2021 to obtain the performance metrics using the intrusion detection pipeline proposed in this paper. The metrics obtained were Accuracy, Precision, Recall, F1-Score and ROC AUC metrics. Unlike accuracy, which can be misleading in highly imbalanced datasets, the evaluation primarily focused on the F1-score for ranking the models, as it provides a balanced measure of both precision and recall. The Random Forest classifier had the best performance of all models with a F1 of 0.9969, ROC AUC of 1.0000 and Accuracy of 0.9996. These values indicate that the model is capable of correctly classifying the normal and attack traffic with extremely high reliability. The Random Forest model also has an excellent balance in precision and recall with very little misclassification of the two classes.

Table 6: HIKARI-2021 Evaluation Metrics

Model	Rank	Accuracy	F1	Precision	Recall	ROC AUC
Random Forest	1	0.99958	0.996913	0.994195	0.999646	0.999998
XGBoost	2	0.99952	0.99647	0.994538	0.998408	0.999997
Decision Tree	3	0.999508	0.996382	0.994363	0.998408	0.999701
Neural Network	4	0.966179	0.756967	0.738925	0.775911	0.844584
KNN	5	0.959768	0.742528	0.656433	0.854616	0.972131
Linear SVM	6	0.913509	0.60474	0.438355	0.974708	0.935866
Logistic Regression	7	0.716695	0.198716	0.122967	0.51751	0.686147
Naive Bayes	8	0.138945	0.135218	0.072555	0.991687	0.686292

Moreover, Table 7 shows the evaluation of the eight machine learning models developed using the proposed method for intrusion detection on the CIC-IDS2017 Friday-DDoS dataset. It was found that XGBoost exhibited overall the best results among all classifiers with a F1 of 0.9998 a perfect ROC AUC of 1.0000, and an Accuracy of 0.9982 thereby proving its extreme proficiency in distinguishing DDoS attacks from normal traffic. The high F1-Score is highly significant as it presents a good balance between Precision and Recall which verifies that the model detects attack traffic with great accuracy while the incidence of false positives and false negatives is extremely low. This balance is very important in the area of cyber security applications where both undetected intrusions and unnecessary alerts can create operational difficulties. In a similar way, the perfect ROC AUC score shows that XGBoost achieves an excellent degree of class separability across all thresholds ensuring consistently, that the attack traffic has a higher probability than that given to the normal traffic. These figures prove that XGBoost has proved the most reliable and effective classifier for the detection of DDoS activity in this dataset.

Table 7: CIC-IDS-2017 (Friday DDoS) Evaluation Metrics

Model	Rank	Accuracy	F1	Precision	Recall	ROC AUC
XGBoost	1	0.999821	0.99979	0.99965	0.99993	0.999954
Random Forest	2	0.999761	0.99972	0.99965	0.99979	0.999973
Decision Tree	3	0.999731	0.999685	0.999649	0.99972	0.999834
Neural Network	4	0.999044	0.998878	0.998738	0.999019	0.999069
KNN	5	0.998506	0.998246	0.998947	0.997546	0.998967
Linear SVM	6	0.983775	0.981222	0.968198	0.994602	0.996298
Naive Bayes	7	0.792213	0.678115	0.997956	0.513531	0.830951
Logistic Regression	8	0.599367	0.639415	0.518674	0.833427	0.851097

Furthermore, Table 8 shows the results evaluated on the KDDCup99 dataset using the proposed Intrusion Detection pipeline. When considering all classifiers, it was found that XGBoost was the strongest, yielding a F1 value of 0.9994, a perfect ROC AUC of 1.0000, and an Accuracy of 0.9995. This indicates excellent ability to discriminate between normal and attack traffic. The high F1-Score of this classifier indicates a good balance between Precision and Recall performance with respect to normalizing false alarm rates and missed intrusions. Similarly, the near perfect ROC AUC indicates strong class separability for all threshold settings. Despite the fact that KDDCup99 is old, and has known disadvantages (such as redundant records, old attack profiles, bad traffic distribution), the proposed pipeline performs reasonably well, giving a consistent benchmark for the assessment of model performance. When trying all models such as Random Forest, Decision Tree, Neural Network, KNN etc., XGBoost was found to be the best performing classifier, which adds to its attractiveness on old intrusion detection datasets.

Table 8: KDD Cup 1999 Evaluation Metrics

Model	Rank	Accuracy	F1	Precision	Recall	ROC AUC
XGBoost	1	0.999542	0.999423	0.999654	0.999192	0.999999
Random Forest	2	0.999359	0.999192	0.999192	0.999192	0.999997
Decision Tree	3	0.998809	0.998499	0.99896	0.998038	0.999441
Neural Network	4	0.99771	0.99711	0.99838	0.995844	0.998118
KNN	5	0.997161	0.996419	0.99711	0.995729	0.997936
Linear SVM	6	0.989651	0.986974	0.985611	0.988341	0.996867
Naive Bayes	7	0.976143	0.969491	0.983836	0.955558	0.978876
Logistic Regression	8	0.932503	0.916676	0.898194	0.935934	0.935309

4.2 Cross-Dataset Synthesis

Through a comparative analysis of HIKARI-2021, CIC-IDS2017 Friday-DDoS and KDDCup99, this work is able to demonstrate that an effective intrusion detection system can be built using common techniques and methods. Although there are some differences between the three datasets in terms of data structures, feature dimensions, traffic flow characteristics and attack types, the same machine-learning pipeline was applied uniformly, enabling fair cross-dataset comparison.

Both XGBoost and Random Forest achieved the highest levels of accuracy, F1-Score, and ROC-AUC across all datasets. This demonstrates a strong ability for these classifiers to perform well against both modern and legacy intrusions. Their high F1-scores represent a good balance between precision and recall, minimizing false positives and providing high detection rates. The high ROC-AUC values confirm that these two classifiers provide excellent separation of normal, as well as attack traffic. While KDDCup99 serves mainly as a baseline, results from HIKARI-2021 and CIC-IDS2017 validate the framework under more realistic and contemporary network conditions. Overall, the results confirm the reliability and adaptability of the proposed pipeline across diverse network environments.

4.3 Confusion Matrix for the Best Model for each Dataset

Figure 2 shows the confusion matrix obtained for the HIKARI-2021 dataset illustrates how the Random Forest Classifier exhibits close to perfect performance, classifying correctly almost all samples of normal (0) and attack (1) instances. Only 35 misclassifications occurred for the more than 83,292 samples considered, illustrating most spectacularly the ability to generalize over such a modern trendsetting dataset representative of IoT application. The subsequent ROC curve analysis gives further confirmation of this behavior, producing an AUC of 1.00, that is perfect separability of normal and malicious traffic. Such a model displays sensitivity (small number of false negatives) as well as precision (small number of false positives) suitable for real-world intrusion detection within real IoT environments.

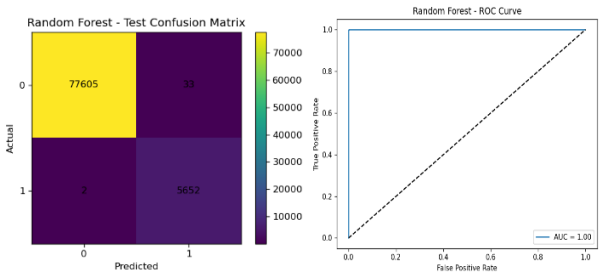

Figure 2: Confusion matrix of the Random Forest classifier on the HIKARI-2021 dataset.

Moreover, for the CIC-IDS2017 Friday-DDoS dataset as shown in Figure 3, the XGBoost classifier yields superior predictive performance with only 6 misclassified samples from over 33,467 samples in the testing set. The confusion matrix shows a good and quite accurate detection of normal traffic and attacks traffic DDoS, showing that the classifier is quite able to predict the statistical modes of this behavior of denial-of-service with high traffic volumes. The ROC curve also reaches AUC of 1.00, indicating that the classifier is excellent in the discrimination of both classes in relation to the threshold. Such results demonstrate the capacity of XGBoost to process complex distributions of traffic and features of high dimensionality typical of modern scenarios of DDos.

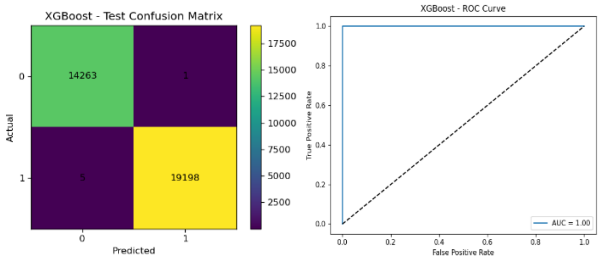

Figure 3: Confusion matrix of the XGBoost classifier on the CIC-IDS-2017 FridayDDos dataset

Furthermore, Figure 4 shows the confusion matrix for KDDCup99 indicates that the XGBoost model again performs very well on this legacy data set and has a very high accuracy with just 10 misclassifications between the normal and attack classes. While the data set is old and has known problems (for instance, redundant records, old attack patterns, etc.), the model is performing almost perfectly because the data is simpler in structure and has much lower feature complexity than the modern IDS data sets. The ROC curve produces again an AUC of 1.00 which indicates perfect separability of the classes. These results show that newer ensemble methods have no problem learning on older data sets but they show the deficiency of KDDCup99 as a representation of modern-day cyber threats.

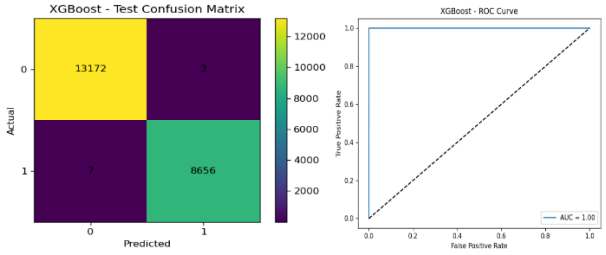

Figure 4: Confusion matrix of the XGBoost classifier on the Kddcup99 dataset

4.4 SHAP Analysis for the Best Model for each Dataset

In order to explain the behavior of the top-performing models on each dataset, we computed SHapley Additive exPlanations (SHAP) to determine both global feature importance and feature value impact on model output. In short, SHAP provides a game-theoretic attribution of each feature in the individual prediction, facilitating the ranking of features according to their influence and inspection of how low vs. high feature values can drive the decision toward “attack” or “benign”. The best-performing machine learning model for each dataset is described below.

The SHAP analysis of the Random Forest model, based on the HIKARI-2021 dataset as shown in Figure 5, indicates that traffic-volume and payload-based features are the strongest drivers of prediction, including forward subflow bytes, forward subflow packets, response frequency, and payload features (for example, flow packets payload total/average). These features are clearly well-suited to indicating abnormal flow size, abnormal flow rate, and their payload distribution, behaviors which indicates offense malicious traffic on the dataset. The importance of the Unnamed: 0 feature is high, but it is not a true security feature, but an artefact of the index. The plots of the SHAP values also indicate that extreme values of originp and origin_freq push the prediction strongly towards the attack class, meaning that an unusual source-endpoint was an important discriminator between benign flows and malicious flows.

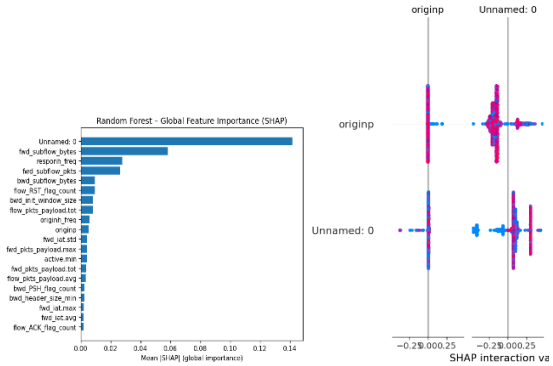

Figure 5: SHAP Analysis of the Random Forest classifier on the HIKARI-2021 dataset.

Moreover, in regard to the XGBoost model applied to the CIC-IDS-2017 Friday-DDoS data set as shown in Figure 6, the SHAP results indicate that the packet-length e.g., Fwd Packet Length Mean, Fwd Packet Length Max, Total Length of Fwd Packets and flow-rate features dominate the prediction process, along with transport-layer and header features like Destination Port, Bwd Packet Length Std and Bwd Header Length. These features account for the size of packets, variation in packet size, and total amount of data being sent to the target, all of which are consistent with the patterns exhibited by higher-volume DDoS traffic. SHAP value distributions indicate that very high forward packet-length values and total bytes-forwarded strongly bias predictions toward the attack class, while lower values are more apt to correlate with benign flows. Other contributing features e.g., act_data_pkt_fwd, Init_Win_bytes_forward/backward, and Flow Packets/s support the model’s detection of the flow bursty nature and increases, all of which suggest that volume- and rate-based indicators are essential for effective DDoS detection.

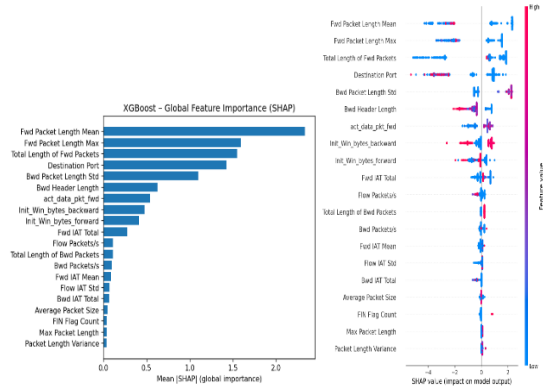

Figure 6: SHAP Analysis of the XGBoost classifier on the CIC-IDS-2017 FridayDDos dataset

Furthermore, in the case of the KDDCup99 dataset as shown in Figure 7, the XGBoost model points to connection-level and host-based statistics as the critical predictors, as for example, same_srv_rate, src_bytes, service frequency, dst_bytes, and count lead to the most important features. These features mostly capture how often a host/service is accessed, how many bytes were sent back/forth, and whether consecutive connections were similar, which are all indicative of scanning and probing behavior. The SHAP distributions clearly depict how very high same_srv_rate and count lead the model to clearly predict the intrusion class and that balanced patterns of connection histories represent benign traffic. Other features such as serror_rate, srv_count, and duration also help discriminate the different types of attacks, capturing aspects of error patterns and the persistence of connections that are tied to malicious attacks.

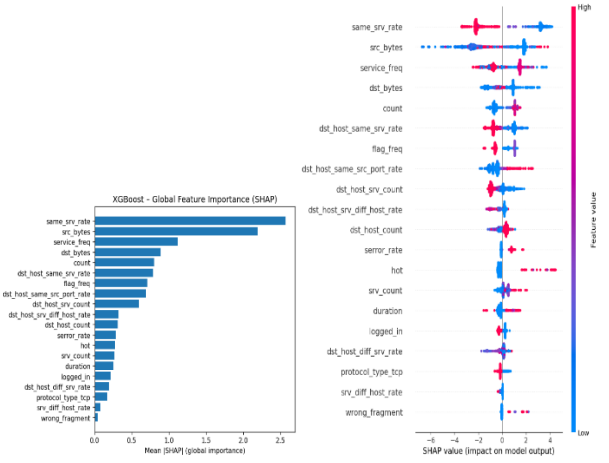

Figure 7: SHAP Analysis of the XGBoost classifier on the Kddcup99 dataset

4.5 Comparing Machine Learning Result with other researchers

The findings across the three datasets imply that our study, particularly consistently outperformed the past IDS (machine-learning) studies largely due to systematic imbalance correction, stratified cross-validation, Bayesian hyperparameter optimization and principled feature selection. For HIKARI-2021 as shown in Table 9, using our tuned Random Forest and XGBoost modelling, we achieved near perfect performance (F1 = 0.999, ROC-AUC = 0.9999) which outperformed the prior studies that addressed imbalance and tuning.

Similarly, for the CIC-IDS2017 Friday-DDoS subsets as shown in Table 10, our models again showed an almost perfect trade-off between precision and recall, with XGBoost and RF outscoring scores reported in prior studies (usually, F1 = 0.97 to 0.99) by a considerable margin. This has further improved the reliable outcomes and reduced the false negatives.

In the evaluation of KDDCup99, sometimes good performance is reported but tends to be inflated due to model tuning and validation not clearly established and specified, whereas in our case both the XGBoost and RF models as shown in

Table ***11***, reflected strong improvement in F1 and recall rate through a more stringent evaluation that demonstrated a clear improvement in detecting the minority class. Overall, the consistent evidence of improvement across datasets indicate it is the methodological designs, and not only SMOTE, that lead to achieving greater accuracy, reducing false alarms, and more credible and generalizable IDS outcomes than previously reported baseline benchmarks.

Table 9: HIKARI-2021 Performance Matric of other researcher's vs Performance Matric of current study

Algorithm	Accuracy	Precision	Recall	F1	Reference
Previous Work					
KNN	0.98	0.86	0.90	0.88	Ferriyan et al. [1].
MLP	0.99	0.99	0.99	0.99	
SVM	0.99	0.99	0.98	0.99	
RF	0.99	0.99	0.99	0.99	
GB	0.9335	0.9114	0.9335	0.9097	Tariq et al. [2].
RF	0.8815	0.8764	0.8815	0.8789	
DT	0.8782	0.8755	0.8769	0.8782	
MLP	0.9321	0.8957	0.9321	0.9009	
KNN	0.9232	0.8904	0.9232	0.9028	
LR	0.9322	0.8698	0.8999	0.9325	
NB	0.7221	0.9438	0.7221	0.7913	
RF	98.34	99.79	71.84	83.54	Vitorino et al. [3].
XGB	98.35	99.83	71.91	83.60	
Results of the Present Study					
RF	0.99958	0.994195	0.999646	0.996913	Machine Learning-Driven Cyber Threat Prediction and Prevention: A Multi-Dataset Design and Comparative Evaluation.
XGB	0.99952	0.994538	0.998408	0.99647	
DT	0.999508	0.994363	0.998408	0.996382	
NN (MLP)	0.966179	0.738925	0.775911	0.756967	
KNN	0.959768	0.656433	0.854616	0.742528	
LSVM	0.913509	0.438355	0.974708	0.60474	
LR	0.716695	0.122967	0.51751	0.198716	
NB	0.138945	0.072555	0.991687	0.135218	

Table 10: CIC-IDS-2017 FridayDDos Performance Matric of other researcher's vs Performance Matric of current study

Algorithm	Accuracy	Precision	Recall	F1	AUROC	Reference
Previous Work						
NB	0.8084	0.8112	0.8084	0.8043	-	Farhat et al. [4].
LR	0.8413	0.8604	0.8413	0.8345	-	
RF	0.9896	0.9897	0.9896	0.9896	-	
XGB	0.9911	0.9912	0.9911	0.9912	-	
DT	-	-	-	0.9978	0.9997	Cantone et al. [5].
RF	-	-	-	0.9979	0.9998	
XGB	-	-	-	0.9979	1.0000	
RF	0.9990	0.9778	0.9708	0.9741	-	MD Shadman Soumik [6].
DT	0.9959	0.9987	0.9959	0.9932	-	
Results of the Present Study						
XGB	0.999821	0.99965	0.99993	0.99979	0.999954	Machine Learning-Driven Cyber Threat Prediction and Prevention: A Multi-Dataset Design and Comparative Evaluation.
RF	0.999761	0.99965	0.99979	0.99972	0.999973	
DT	0.999731	0.999649	0.99972	0.999685	0.999834	
NN (MLP)	0.999044	0.998738	0.999019	0.998878	0.999069	
KNN	0.998506	0.998947	0.997546	0.998246	0.998967	
LSVM	0.983775	0.968198	0.994602	0.981222	0.996298	
NB	0.792213	0.997956	0.513531	0.678115	0.830951	
LR	0.599367	0.518674	0.833427	0.639415	0.851097	

Table 11: Kddcup99 Performance Matric of other researcher's vs Performance Matric of current study

Algorithm	Accuracy	Precision	Recall	F1	AUROC	Reference
Previous Work						
KNN	0.9724	0.9806	0.9724	0.9741	-	Tripathy et al. [7].
DT	0.9713	0.9809	0.9713	0.9731	-	
RF	0.9714	0.9811	0.9714	0.9733	-	
SVM	0.9808	0.9815	0.9808	0.9808	-	
LR	0.9667	0.9777	0.9667	0.9689	-	
XGB	0.9464	0.9646	0.9464	0.9496	-	
BNB	0.9473	0.9594	0.9473	0.9487	-	
RF	1.00000	1.00000	1.00000	1.00000	-	Hossain et al. [8].
XGB	0.99963	0.99938	0.99963	0.99950	-	
NN	0.99855	-	-	-	0.999	Li et al. [9].
LR	0.99665	-	-	-	0.997	
NB	0.97887	-	-	-	0.979	
RF	0.9874	0.8819	0.9862	0.8797	-	Bitra et al. [10].
KNN	0.9298	0.6928	0.8382	0.7936	-	
SVM	0.9491	0.7238	0.8061	0.7142	-	
LGBM	0.9541	0.7948	0.9542	0.8035	-	
Results of the Present Study						
XGB	0.999542	0.999654	0.999192	0.999423	0.999999	Machine Learning-Driven Cyber Threat Prediction and Prevention: A Multi-Dataset Design and Comparative Evaluation.
RF	0.999359	0.999192	0.999192	0.999192	0.999997	
DT	0.998809	0.99896	0.998038	0.998499	0.999441	
NN (MLP)	0.99771	0.99838	0.995844	0.99711	0.998118	
KNN	0.997161	0.99711	0.995729	0.996419	0.997936	
LSVM	0.989651	0.985611	0.988341	0.986974	0.996867	
NB	0.976143	0.983836	0.955558	0.969491	0.978876	
LR	0.932503	0.898194	0.935934	0.916676	0.935309	

5. DISCUSSION

The obtained results of the experiment highlight the effectiveness of the ensemble-based and deep learning methods (Random Forest, XGBoost, and MLP), which consistently outperform traditional classifiers when evaluated across all metrics. In particular, XGBoost achieved greater than 99.9% accuracy and F1-score between 99% to 99.9% on two of three datasets. Equally important was the methodological pipeline; for example, the F1-score for the Linear SVM raised from 0 to 0.605 due to using the techniques such as feature selection, SMOTE, and Bayesian tuning steps on HIKARI-2021. Thus, both model choice and pipeline design are equally important. By using SHAP analysis for post-hoc decision justification, it was proven that all decisions were based on significant network features, providing the level of transparency required for analyst confidence and compliance with regulatory bodies.

The high levels of accuracy obtained from the CIC-IDS2017 and KDD Cup 99 datasets align with other literature in this area — previous studies have reported accuracies of between 98% and 99.99% for these same datasets (Table 11 and Table 12), confirming that our findings are not unusual. In terms of data integrity, SMOTE was applied exclusively to the training data after a stratified train/test split, ensuring that there was no data leakage, and that the test dataset did not influence any preprocessing or model tuning processes. It is well established in the literature that benchmark datasets such as CIC-IDS2017 and KDD Cup 99 are generally more structured and cleaner than real world network traffic, leading to a natural inflation of performance metrics - this is a well-documented limitation in the field. The combined use of 5-fold CV, SMOTE to balance training data, and Bayesian hyperparameter optimization provided methodological assurance that the results presented show genuine model generalization, and not overfitting to the training dataset. Importantly, the relatively lower results on HIKARI-2021 demonstrate that the framework does not produce near-perfect outputs in all cases, suggests the high quality of performance found in the legacy datasets is due to their simplicity of structure rather than a systematic inflation of results.

In developing the framework, practical considerations regarding accessibility were paramount; all of the experimentation was done using standard CPU hardware, which makes the pipeline a reasonable option for all organizations with resource constraints including small corporations, as well as national cybersecurity teams in developing regions such as the Pacific Islands, where there is an urgent need for scalable and low-cost methods of threat detection. The high precision of top-performing models minimizes the fatigue of analysts associated with false alarms, whilst the high recall of these same models prevents legitimate threats from being missed; both of these features are critical for any solution implemented in a live security environment.

In addition to offline evaluations, it is necessary to also consider the practicality of using XGBoost and MLP models in a real-time, high-speed network environment. Due to its ability to optimize its entire predictive path through a fixed order of decision trees, XGBoost is a very good candidate for real-time inference; latency per sample is usually microseconds, allowing it to be realistically implemented in inline network monitoring systems that can run at gigabits per second. Although the MLP model requires slightly more computational resources than the XGBoost model, the use of GPU acceleration during production deployment of the MLP model (using batched inference of concurrent flows) allows for greatly increased throughput. Both the XGBoost and MLP models can use modern production-serving technology/frameworks (e.g., ONNX Runtime, TensorFlow Serving, and NVIDIA Triton Inference Server) that provide low latency, microservice-based infrastructure for enterprise quality network environments. A pragmatic consideration is that both models require that the features for each flow be pre-extracted, rather than extracted from raw packet streams, and require an upstream feature extraction process (e.g., using CICFlowMeter or nDPI),

which adds a small amount of latency; however, the amount of latency is manageable. Lightweight surrogate models and model distillation techniques are a potential speed vs. accuracy trade-off under ultra-low-latency conditions such as those present in the financial and industrial control network environments. Real-time deployment was not the focus of this research; however, both models indicate capability to be considered for integration into production NIDS pipelines given appropriate telemetry infrastructure due to their computational properties.

Nevertheless, there are still significant limitations present in this study. First, the benchmark datasets that are used do not capture all of the types of diversity found in the real-world. Additionally, because there are no region-specific benchmark datasets that were used in this study, there is a restriction to how these findings can be applied to particular geographical areas.

6. CONCLUSION

Due to the growing complexity and frequency of cyber threats, the demand for intelligent, adaptive, and scalable intrusion detection systems continues to increase. This study demonstrated that machine learning can serve as an effective and accessible foundation for network intrusion detection across diverse network environments. By evaluating a structured pipeline including feature selection, class balancing, hyperparameter optimization, explainability analysis, and using three benchmark datasets, the results indicated that ensemble and deep learning models, especially the XGBoost, Random Forest, and MLP models, outperformed traditional classification methods. Notably, the framework was developed with real-world constraints in consideration and uses a standard CPU hardware requirement, which makes it usable by organizations with few resources and emerging cybersecurity ecosystems such as those in the Pacific Islands region.

Although the framework establishes strong generalizability, its evaluation on benchmark datasets and CPU-based hardware represents boundaries that future work should seek to expand. Future studies should focus on collecting live traffic, using GPU accelerated architectures, developing datasets that are representative of a particular geographical area and developing a real-time retraining pipeline as new threats emerge. Overall, this study has created a reproducible, transparent, and practical ML-based IDS development framework, thus establishing a strong basis to further progress in developing intelligent cybersecurity systems and expanding on the work of previous researchers in this field.

CONFLICT OF INTEREST

Authors confirm that there is no conflict of interest associated with this publication.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] S. Morgan, “Cybercrime To Cost The World $10.5 Trillion Annually By 2025.” Accessed: Feb. 07, 2026. [Online]. Available: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/

[2] IBM, “What is an Intrusion Detection System (IDS)?” Accessed: Feb. 07, 2026. [Online]. Available: https://www.ibm.com/think/topics/intrusion-detection-system

[3] A. Hozouri, A. Mirzaei, and M. Effatparvar, “A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges,” Discover Artificial Intelligence, vol. 5, no. 1, Dec. 2025, doi: 10.1007/s44163-025-00578-1.

[4] M. L. Ali, K. Thakur, S. Schmeelk, J. Debello, and D. Dragos, “Deep Learning vs. Machine Learning for Intrusion Detection in Computer Networks: A Comparative Study,” Applied Sciences (Switzerland), vol. 15, no. 4, Feb. 2025, doi: 10.3390/app15041903.

[5] M. A. Hossain and M. S. Islam, “Ensuring network security with a robust intrusion detection system using ensemble-based machine learning,” Array, vol. 19, Sep. 2023, doi: 10.1016/j.array.2023.100306.

[6] P. Waghmode, M. Kanumuri, H. El-Ocla, and T. Boyle, “Intrusion detection system based on machine learning using least square support vector machine,” Sci. Rep., vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-95621-7.

[7] R. O. Arogundade, “Network Security Concepts, Dangers, and Defense Best Practical,” Computer Engineering and Intelligent Systems, Mar. 2023, doi: 10.7176/ceis/14-2-03.

[8] D. H. Jeong, B. K. Jeong, and S. Y. Ji, “Multi-Resolution Analysis with Visualization to Determine Network Attack Patterns,” Applied Sciences (Switzerland), vol. 13, no. 6, Mar. 2023, doi: 10.3390/app13063792.

[9] F. Hachmi, K. Boujenfa, and M. Limam, “Enhancing the Accuracy of Intrusion Detection Systems by Reducing the Rates of False Positives and False Negatives Through Multi-objective Optimization,” Journal of Network and Systems Management 2018 27:1, vol. 27, no. 1, pp. 93–120, May 2018, doi: 10.1007/s10922-018-9459-y.

[10] Md. R. Ahmed, salekul Islam, S. Shatabda, A. K. M. M. Islam, and Md. T. I. Robin, “Intrusion Detection System in Software-Defined Networks Using Machine Learning and Deep Learning Techniques –A Comprehensive Survey,” Nov. 21, 2022. doi: 10.36227/techrxiv.17153213.v2.

[11] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy,

SciTePress, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[12] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, "Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic [dataset]," Applied Sciences (Switzerland), vol. 11, no. 17, Sep. 2021, doi: 10.3390/app11177868.

[13] K. He, D. D. Kim, and M. R. Asghar, "Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey," IEEE Communications Surveys and Tutorials, vol. 25, no. 1, pp. 538–566, 2023, doi: 10.1109/COMST.2022.3233793.

[14] T. Bin Tariq et al., "Intelligent Cyber Security Framework for Threat Detection using Ensemble Learning Techniques", doi: 10.56979/802/2025.

[15] J. Vitorino, M. Silva, E. Maia, and I. Praça, "An Adversarial Robustness Benchmark for Enterprise Network Intrusion Detection," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 14551 LNCS, pp. 3–17, 2024, doi: 10.1007/978-3-031-57537-2_1.

[16] MD Shadman Soumik, "A comparative analysis of Network Intrusion Detection (NID) using Artificial Intelligence techniques for increase network security," International Journal of Science and Research Archive, vol. 13, no. 2, pp. 4014–4025, Dec. 2024, doi: 10.30574/ijsra.2024.13.2.2664.

[17] S. Farhat, M. Abdelkader, A. Meddeb-Makhlouf, and F. Zarai, "Evaluation of DoS/DDoS Attack Detection with ML Techniques on CIC-IDS2017 Dataset," in International Conference on Information Systems Security and Privacy, Science and Technology Publications, Lda, 2023, pp. 287–295. doi: 10.5220/0011605700003405.

[18] M. Cantone, C. Marrocco, and A. Bria, "On the Cross-Dataset Generalization of Machine Learning for Network Intrusion Detection," Feb. 2024, doi: 10.1109/ACCESS.2024.3472907.

[19] Venu Gopal Bitra, Ajay Kumar, Seshagiri Rao, Prakash, and Md. Shakeel Ahmed, "Comparative analysis on intrusion detection system using machine learning approach," World Journal of Advanced Research and Reviews, vol. 21, no. 3, pp. 2555–2562, Mar. 2024, doi: 10.30574/wjarr.2024.21.3.0983.

[20] Z. Li et al., "Denial of Service (DoS) Attack Detection: Performance Comparison of Supervised Machine Learning Algorithms," in 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, Aug. 2020, pp. 469–474. doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00088.

[21] S. S. Tripathy and B. Behera, "PERFORMANCE EVALUATION OF MACHINE LEARNING ALGORITHMS FOR INTRUSION DETECTION SYSTEM," 2023, doi: 10.17605/OSF.IO/WX6CS.

[22] "Intrusion detection evaluation dataset (CIC-IDS2017) [dataset]." Accessed: Feb. 07, 2026. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[23] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "KDD Cup 1999 Data - UCI Machine Learning Repository [dataset]." Accessed: Feb. 07, 2026. [Online]. Available: https://archive.ics.uci.edu/dataset/130/kdd+cup+1999+data

[24] A. Subasi, "Chapter 2 - Data preprocessing," Practical Machine Learning for Data Analysis Using Python, pp. 27–89, 2020, doi: 10.1016/b978-0-12-821379-7.00002-3.

[25] S. Walling and S. Lodh, "Network intrusion detection system for IoT security using machine learning and statistical based hybrid feature selection," Security and Privacy, vol. 7, no. 6, p. e429, Nov. 2024, doi: 10.1002/spy2.429.

[26] M. B. Musthafa et al., "Optimizing IoT Intrusion Detection Using Balanced Class Distribution, Feature Selection, and Ensemble Machine Learning Techniques," Sensors, vol. 24, no. 13, Jul. 2024, doi: 10.3390/s24134293.

[27] R. Duangsoithong and T. Windeatt, "Correlation-Based and Causal Feature Selection Analysis for Ensemble Classifiers," Artificial Neural Networks in Pattern Recognition, vol. 5998, pp. 25–36, 2010, doi: 10.1007/978-3-642-12159-3_3.

[28] N. Verma, N. Kumar, K. Singh, A. Aljohani, A. Sinha, and S. A. Hussain, "A novel univariate feature selection with ANOVA F-test-based machine learning model for Intrusion Detection Framework of Robotics system," Applied Artificial Intelligence, vol. 39, no. 1, Dec. 2025, doi: 10.1080/08839514.2025.2539395.

[29] A. H. Ali, M. Charfeddine, B. Ammar, and B. Ben Hamed, "Intrusion Detection Schemes Based on Synthetic Minority Oversampling Technique and Machine Learning Models," Proceedings - 2024 IEEE 27th International Symposium on Real-Time Distributed Computing, ISORC 2024, 2024, doi: 10.1109/ISORC61049.2024.10551335.

[30] M. Khairy, T. M. Mahmoud, and T. Abd-El-Hafeez, "The effect of rebalancing techniques on the classification performance in cyberbullying datasets," Jan. 01, 2024, Springer Science and Business Media Deutschland GmbH. doi: 10.1007/s00521-023-09084-w.

[31] D. Bisen, A. Ghanghoria, P. Saurabh, D. Rohith, and U. Singh, "Optimizing Intrusion Detection in Software-Defined Networks Through Automated Machine Learning and Intelligent Feature Engineering," IEEE Access, vol. 13, pp. 194097–194114, 2025, doi: 10.1109/ACCESS.2025.3632116.

[32] P. Heidari and A. Milan, "Combining K-fold cross validation with bayesian hyperparameter optimization for accuracy enhancement of land cover and land use classification," Sci. Rep., vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-23336-w.

[33] "3.1. Cross-validation: evaluating estimator performance — scikit-learn 1.8.0 documentation." Accessed: May 06, 2026. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html

[34] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” International Journal of Data Mining & Knowledge Management Process, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[35] A. A. Awan, “An Introduction to SHAP Values and Machine Learning Interpretability | DataCamp.” Accessed: Feb. 07, 2026. [Online]. Available: https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability

[36] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on tabular data?,” Jul. 2022, [Online]. Available: http://arxiv.org/abs/2207.08815

[37] S. Chalichalamala, N. Govindan, and R. Kasarapu, “Logistic Regression Ensemble Classifier for Intrusion Detection System in Internet of Things,” Sensors 2023, Vol. 23, vol. 23, no. 23, pp. 1–19, Dec. 2023, doi: 10.3390/S23239583.

[38] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, “XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud,” Proceedings - 2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018, pp. 251–256, May 2018, doi: 10.1109/BIGCOMP.2018.00044.

[39] Y. Huang, “Network Intrusion Detection Method Based on Naive Bayes Algorithm,” Proceedings of 2022 6th Asian Conference on Artificial Intelligence Technology, ACAIT 2022, 2022, doi: 10.1109/ACAIT56212.2022.10137846.