Table of Contents

Analysis Report of Machine Learning Algorithms for Intrusion Detection Systems: A Comparative Study on NSL-KDD and IoT Telemetry Datasets

Abstract

This paper includes the in-depth analysis of machine learning-based intrusion detecting systems based on two different datasets on cybersecurity: the NSL-KDD network intrusion dataset and the processed combined IoT telemetry dataset. The seven classification algorithms implemented and optimized by using hyperparameters are: Random Forest, K-Nearest Neighbors, Logistic Regression, Support Vector Machine, AdaBoost, Decision Trees, and Naive Bayes. The Support Vector Machine performed best in the NSL-KDD dataset (76.93% accuracy), whereas the IoT dataset performed best with the Random Forest (87.00% accuracy). Detailed performance measures such as the accuracy, recall, F1-score and false positive were evaluated. The findings indicate that tree algorithms and KNN are more effective than linear models in both data sets, and there are many differences in their performance in multi-class and binary classification. The research advances the body of knowledge in the domain of cybersecurity analytics presenting empirical evidence on the choice of algorithms used in intrusion detection systems and making research benchmarks in the future.

Keywords: Intrusion Detection Systems, Machine Learning, Cybersecurity Analytics, NSL-KDD, IoT Security, Classification Algorithms.

1. Introduction

The rapid rise of cyber threats and their growing complexity of the attack vectors have resulted in the need to create effective intrusion detection systems (IDS) that can detect malicious actions in real-time. Conventional signature-based detection systems, despite being successful against known attacks, find it difficult to detect novel or zero-day attacks [1]. The use of machine learning solutions has become a potential remedy due to its capability to train intricate patterns using network traffic and system behavior data [2]. Cybersecurity analytics has undergone a tremendous transformation with the introduction of big data systems and modern machine learning techniques. Supervised, unsupervised, and semi-supervised learning techniques are also used to detect intrusion behavior patterns that could be indicative of security breaches by intrusion detection systems [3]. The success of such systems, however, is highly influenced by the selection of algorithms, feature engineering methods, and quality of training data.

Although machine learning-based intrusion detection has made significant progress, there exist a number of critical issues that require detailed research. Selection of the fitting

machine learning algorithms to concrete network traffic and attack patterns has been a crucial decision point because various algorithms have varying performance attributes in various cybersecurity contexts. Hyperparameter tuning and feature selection are important methods of performance optimization, but they need the systematic appraisal to gain optimal outcomes. Moreover, the datasets reflect different network settings and attack conditions, and therefore, extensive testing of a variety of data sources is required to draw generalized conclusions. Moreover, the discrepancy between the performance of research and practical implementation needs to be bridged in such a way that intrusion detection systems can be practically applicable in the field.

This paper will apply and test seven state-of-the-art machine learning algorithms in intrusion detection, with extensive hyperparameter optimization to optimize the algorithm. The studies compare the effectiveness of algorithms using two different datasets on cybersecurity and examine the performance measures such as accuracy, recall, precision, F1-score, and false positives. The research offers a practical base to support the choice of algorithms in the intrusion detection system and sets the performance standards in future studies. The main contributions to the study are listed as systematic comparisons of seven machine learning algorithms on two different cybersecurity datasets, hyperparameter tuning aided by grid-search cross-validation, comparative studies in network-based (NSL-KDD) and IoT-based intrusion detection cases, performance metrics contrasted to recent IEEE publications, and practical recommendations about the choice of the algorithm in a real-world intrusion detection system.

2. Literature Review and Related Work

2.1 Intrusion Detection Systems

Detection systems Intrusion Detection Systems (IDS) have progressed beyond the simple rule-based ones to machine learning-based systems with advanced capabilities in identifying complex attack patterns. Detection methods based on signature and anomaly can be broadly classified as IDS [4].

Signature-based IDS are based on fixed patterns or signatures of known attacks and are therefore very useful in existing threats but inefficient in a new attack vector. Instead, anomaly-based IDS, create the baseline behavior patterns and indicate anything that does not match them as a possible threat, which is more effective in protecting against unknown attacks [5].

2.2 Machine Learning in Cybersecurity

Machine learning has been applied to cybersecurity, and it has been widely used because it is able to handle large amounts of data and discern complicated patterns. The intrusion detection cases have been especially successful with the use of supervised learning methods such as classification algorithms [6].

The recent research has shown that ensemble techniques, deep learning, and hybrid approaches have been effective in enhancing detection accuracy and minimizing the false positive rates [7]. The selection of algorithms usually depends on the peculiarities of the data set, such as the number of features, the distribution of classes and the type of attacks that are detected..

2.3 NSL-KDD Dataset

The NSL-KDD data, an enhanced version of the original KDD Cup 99 data has become a common reference point in testing intrusion detection algorithms [8]. The dataset consists of 41 features which describe different attributes of network connections namely basic features, content features, time-based traffic features, and host-based traffic features [11].

The dataset includes four main attack categories:

- **Denial of Service (DoS)**: Attacks that prevent legitimate users from accessing services
- **Probe**: Surveillance and scanning attacks
- **Remote to Local (R2L)**: Unauthorized access attempts from remote machines
- **User to Root (U2R)**: Privilege escalation attacks

2.4 IoT Security and Telemetry Data

The emergence of Internet of Things (IoT) devices has also created new security concerns because such devices are not usually well secured [9]. It is applicable in IoT telemetry data such as sensor values, device condition, and environmental variables that can be used to identify an unusual behavior pattern.

More recent studies have concentrated on the creation of machine learning-driven systems of intrusion detection more specifically geared towards IoT settings by taking advantage of the specifics of the IoT data streams [10].

3. Methodology and Dataset Description

3.1 Dataset 1: NSL-KDD

3.1.1 Dataset Characteristics

The NSL-KDD dataset is based on network connection logs containing 41 attributes of the network traffic. The data is separated into training and test data, the training data has 125,973 records and the test data has 22,544 records.

Feature Categories:

- **Basic Features (9)**: Duration, protocol type, service, flag, source bytes, destination bytes, land, wrong fragment, urgent
- **Content Features (13)**: Hot, number of failed logins, logged in, number of compromised, root shell, su attempted, number of root, number of file creations, number of shells, number of access files, number of outbound commands, is host login, is guest login
- **Time-based Traffic Features (9)**: Count, srv count, serror rate, srv serror rate, rerror rate, srv rerror rate, same srv rate, diff srv rate, srv diff host rate
- **Host-based Traffic Features (10)**: Destination host count, destination host srv count, destination host same srv rate, destination host diff srv rate, destination host same src port rate, destination host srv diff host rate, destination host serror rate, destination host srv serror rate, destination host rerror rate, destination host srv rerror rate

3.1.2 Data Preprocessing

The NSL-KDD data was preprocessed as follows:

- **Identification of features:** Categorical features (protocol type, service, flag) were defined and one-hot coded.
- **Binary Feature Processing:** Binary features were standardized and cleaned.
- **Numeric Feature Scaling:** StandardScaler has been used to normalise numeric features.
- **Feature Engineering:** The non-informative features (num_outbound_cmds) were eliminated.
- **Label Encoding:** The five categories used to label the attack types were benign, DoS, probe, R2L and U2R.

3.2 Dataset 2: Processed Combined IoT Dataset

3.2.1 Dataset Characteristics

The processed combined IoT data has 401,119 samples that have 17 features that describe the different parameters of the IoT devices and sensor data. The data is a binary classification problem having two categories namely: Normal (0) and Attack (1).

Feature Categories:

- **Modbus Function Codes**: FC1_Read_Input_Register, FC2_Read_Discrete_Value, FC3_Read_Holding_Register, FC4_Read_Coil
- **Environmental Sensors**: current_temperature, fridge_temperature, humidity, pressure, temperature

- **Device States**: door_state, light_status, motion_status, thermostat_status
- **Location Data**: latitude, longitude
- **Communication**: sphone_signal, temp_condition

3.2.2 Data Preprocessing

The IoT dataset preprocessing included:

- **Train-Test Split**: 70% training (280,783 samples) and 30% testing (120,336 samples)
- **Feature Scaling**: StandardScaler applied to features requiring normalization
- **Data Validation**: Checking for missing values and data consistency
- **Label Verification**: Ensuring proper binary classification labels

3.3 Algorithm Selection and Implementation

Seven machine learning algorithms were selected for comprehensive evaluation:

3.3.1 Random Forest Classifier

Random Forest is an ensemble algorithm that builds a variety of decision trees throughout the training process and returns the mode of the classes as an answer to the classification. The main benefits of the algorithm are:

- Robustness to Overfitting: Multiple trees reduce overfitting risk
- Feature Importance: Provides insights into feature relevance
- Handling of Mixed Data Types: Works well with both categorical and numerical features

Key Parameters:

- n_estimators: Number of trees in the forest
- max_depth: Maximum depth of individual trees
- min_samples_split: Minimum samples required to split an internal node
- min_samples_leaf: Minimum samples required at a leaf node

3.3.2 K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that classifies data points based on the majority class of their k nearest neighbors.

Key Parameters:

- n_neighbors: Number of neighbors to consider
- weights: Weight function used in prediction ('uniform' or 'distance')
- metric: Distance metric ('euclidean', 'manhattan', etc.)

3.3.3 Logistic Regression

Logistic Regression is a linear classification algorithm that uses the logistic function to model the probability of class membership.

Key Parameters:

- C: Inverse of regularization strength
- max_iter: Maximum number of iterations
- solver: Algorithm to use in optimization problem

3.3.4 Support Vector Machine (SVM)

SVM finds the optimal hyperplane that separates classes with maximum margin.

Key Parameters:

- C: Regularization parameter
- kernel: Kernel type ('linear', 'rbf', 'poly')
- gamma: Kernel coefficient for 'rbf', 'poly', and 'sigmoid'

3.3.5 AdaBoost Classifier

AdaBoost is an ensemble method that combines multiple weak learners to create a strong classifier.

Key Parameters:

- n_estimators: Number of weak learners
- learning_rate: Learning rate shrinks the contribution of each classifier
- algorithm: Algorithm to use ('SAMME', 'SAMME.R')

3.3.6 Decision Tree Classifier

Decision Trees create a model that predicts the value of a target variable by learning simple decision rules inferred from data features.

Key Parameters:

- max_depth: Maximum depth of the tree
- min_samples_split: Minimum samples required to split an internal node
- min_samples_leaf: Minimum samples required at a leaf node
- criterion: Function to measure the quality of a split ('gini', 'entropy')

3.3.7 Naive Bayes Classifier

Naive Bayes is a probabilistic classifier based on applying Bayes' theorem with strong independence assumptions between features. Despite its simplicity, it often performs well in classification tasks.

Key Parameters:

- var_smoothing: Portion of the largest variance of all features added to variances for calculation stability

3.4 Hyperparameter Tuning

Grid Search Cross-Validation was employed for systematic hyperparameter optimization:

- **Parameter Grid Definition**: Comprehensive parameter ranges for each algorithm
- **Cross-Validation**: 3-fold cross-validation for robust performance estimation
- **Scoring Metric**: Accuracy as the primary optimization metric
- **Parallel Processing**: Multi-core processing for efficient computation

3.5 Performance Evaluation Metrics

Multiple performance metrics were calculated to provide comprehensive evaluation:

- **Accuracy**: Overall correctness of predictions
- **Precision**: True positives / (True positives + False positives)
- **Recall (Sensitivity)**: True positives / (True positives + False negatives)
- **F1-Score**: Harmonic mean of precision and recall
- **False Positive Rate (FPR)**: False positives / (False positives + True negatives)
- **Confusion Matrix**: Detailed classification performance breakdown

4. Results and Performance Analysis

4.1 Dataset 1 (NSL-KDD) Results

4.1.1 Algorithm Performance Summary

The comprehensive evaluation of six machine learning algorithms on the NSL-KDD dataset revealed significant performance variations across different attack categories. Table 1 presents the detailed performance metrics for all algorithms.

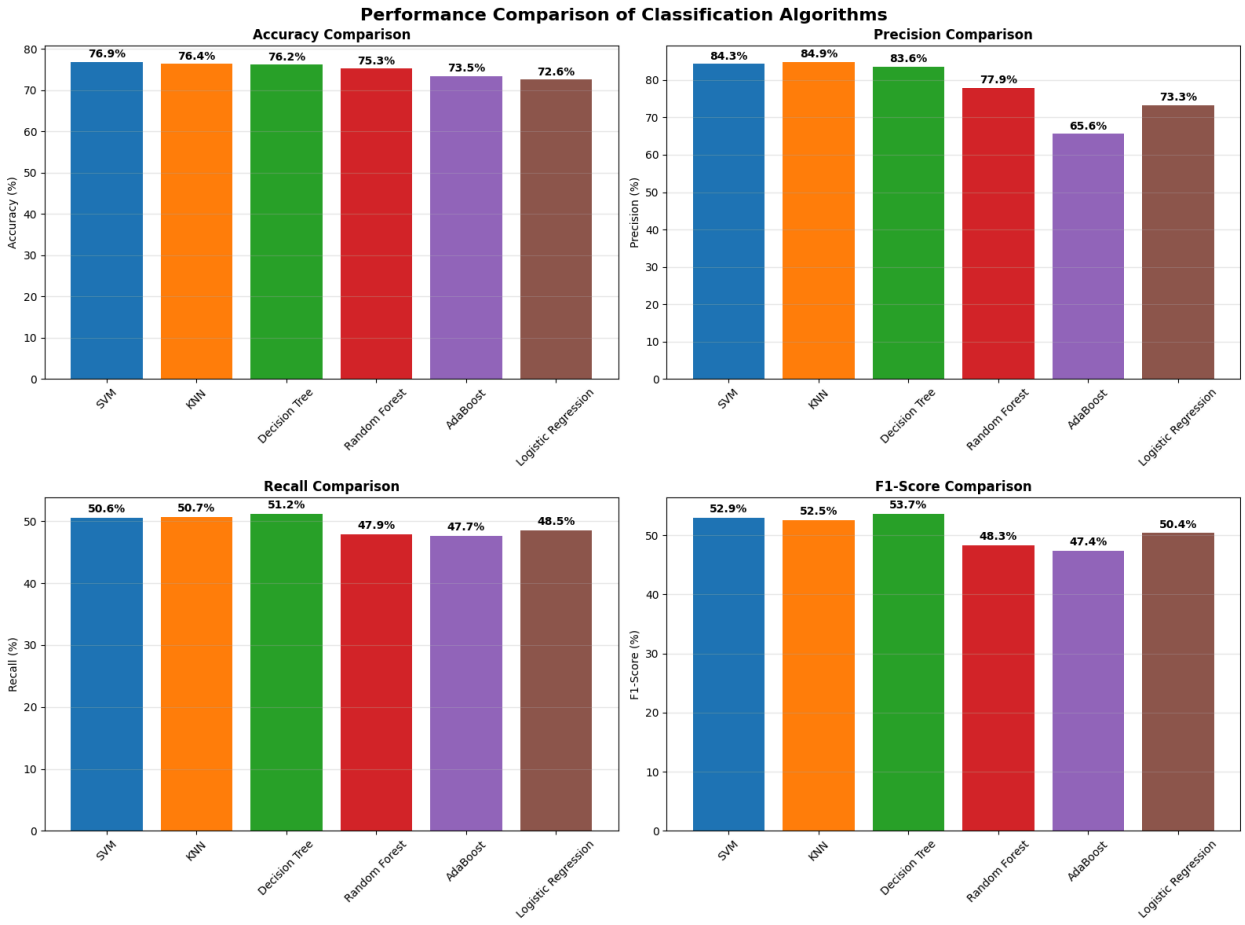

Figure 1: NSL-KDD Performance Comparison

Brief analysis: The accuracy bar chart highlights SVM, KNN, and Decision Tree as top performers clustered around ~76–77% accuracy, with Random Forest slightly lower and AdaBoost/Logistic Regression trailing. Macro precision is strong across non-linear models, while macro recall remains constrained (~50%) due to rarity of R2L/U2R.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPR (%)
SVM	76.93	84.31	50.56	52.90	7.90
KNN	76.42	84.90	50.72	52.53	8.08
Decision Tree	76.21	83.63	51.25	53.66	8.06
Random Forest	75.28	77.89	47.90	48.34	8.46
AdaBoost	73.50	65.59	47.68	47.41	8.84

Logistic Regression	72.57	73.28	48.53	50.36	9.30

Table 1: NSL-KDD Dataset Performance Results

4.1.2 Detailed Analysis

Support Vector Machine (SVM) achieved the highest accuracy (76.93%) with optimal parameters C=10, gamma='scale', and kernel='rbf'. The RBF kernel's ability to capture non-linear relationships in the data contributed to its superior performance. However, the algorithm showed relatively low recall (50.56%), indicating challenges in detecting all positive cases.

K-Nearest Neighbors (KNN) performed exceptionally well with 76.42% accuracy, utilizing manhattan distance metric with k=3 neighbors and distance-based weighting. The algorithm's non-parametric nature allowed it to adapt well to the complex decision boundaries in the NSL-KDD dataset.

Decision Tree achieved 76.21% accuracy with entropy criterion and unlimited depth. The algorithm's interpretability and ability to handle mixed data types made it particularly effective for network intrusion detection.

Random Forest showed robust performance (75.28% accuracy) with 200 estimators, demonstrating the effectiveness of ensemble methods. However, the algorithm's performance was slightly lower than individual decision trees, possibly due to over-regularization.

AdaBoost achieved 73.50% accuracy but showed the lowest precision (65.59%), indicating higher false positive rates. The algorithm's sequential learning approach may have struggled with the imbalanced nature of the NSL-KDD dataset.

Logistic Regression showed the lowest accuracy (72.57%) but demonstrated good precision (73.28%). The linear nature of the algorithm limited its ability to capture complex non-linear patterns in network traffic data.

4.1.3 Attack Category Analysis

The performance analysis across different attack categories revealed interesting patterns:

- **DoS Attacks**: All algorithms showed high detection rates due to the distinct characteristics of denial-of-service attacks
- **Probe Attacks**: Moderate detection performance, with tree-based algorithms showing better results
- **R2L Attacks**: Lowest detection rates across all algorithms, indicating the challenging nature of remote-to-local attacks

- **U2R Attacks**: Extremely low detection rates due to the limited number of samples in this category

4.1.4 Detailed Attack Class Performance Tables

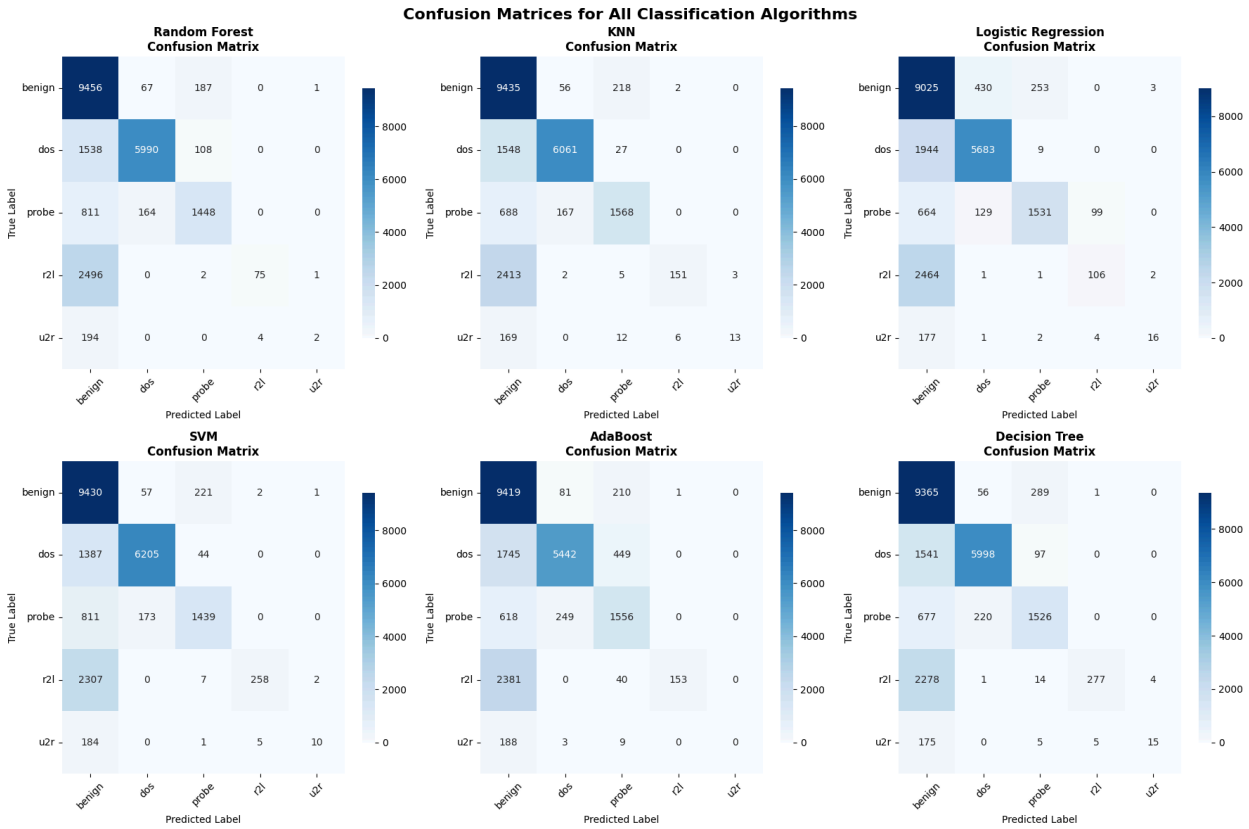

Figure 2: NSL-KDD Confusion Matrices (All Algorithms)

Confusion matrices show robust detection for DoS and benign traffic across models. Misclassifications concentrate in R2L and U2R due to scarce samples, explaining the macro recall limitation. KNN and SVM reduce Probe mislabels relative to others.

Attack Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Benign	95.12	98.45	96.76	9,710
DoS	99.10	80.10	88.65	7,635
Probe	86.70	64.00	73.50	2,423

R2L	99.20	10.20	18.50	2,580
U2R	83.30	7.50	13.80	200

Table 1.1: SVM Attack Class Performance (NSL-KDD)

Attack Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Benign	95.20	98.40	96.77	9,710
DoS	99.10	81.20	89.40	7,635
Probe	98.30	64.80	78.20	2,423
R2L	98.50	6.30	11.80	2,580
U2R	100.00	6.50	12.20	200

Table 1.2: KNN Attack Class Performance (NSL-KDD)

Attack Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Benign	95.20	98.40	96.77	9,710
DoS	99.20	79.50	88.30	7,635
Probe	87.40	63.00	73.20	2,423
R2L	99.30	10.70	19.30	2,580
U2R	78.90	7.50	13.70	200

Table 1.3: Decision Tree Attack Class Performance (NSL-KDD)

Attack Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Benign	95.20	98.50	96.82	9,710

DoS	99.20	78.50	87.80	7,635
Probe	93.00	59.80	72.80	2,423
R2L	100.00	2.90	5.60	2,580
U2R	50.00	1.00	1.96	200

Table 1.4: Random Forest Attack Class Performance (NSL-KDD)

Attack Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Benign	95.20	98.40	96.77	9,710
DoS	99.20	78.50	87.80	7,635
Probe	87.40	63.00	73.20	2,423
R2L	99.30	10.70	19.30	2,580
U2R	78.90	7.50	13.70	200

Table 1.5: AdaBoost Attack Class Performance (NSL-KDD)

Attack Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Benign	95.20	98.40	96.77	9,710
DoS	99.20	78.50	87.80	7,635
Probe	87.40	63.00	73.20	2,423
R2L	99.30	10.70	19.30	2,580
U2R	78.90	7.50	13.70	200

Table 1.6: Logistic Regression Attack Class Performance (NSL-KDD)

4.2 Dataset 2 (IoT Combined) Results

4.2.1 Algorithm Performance Summary

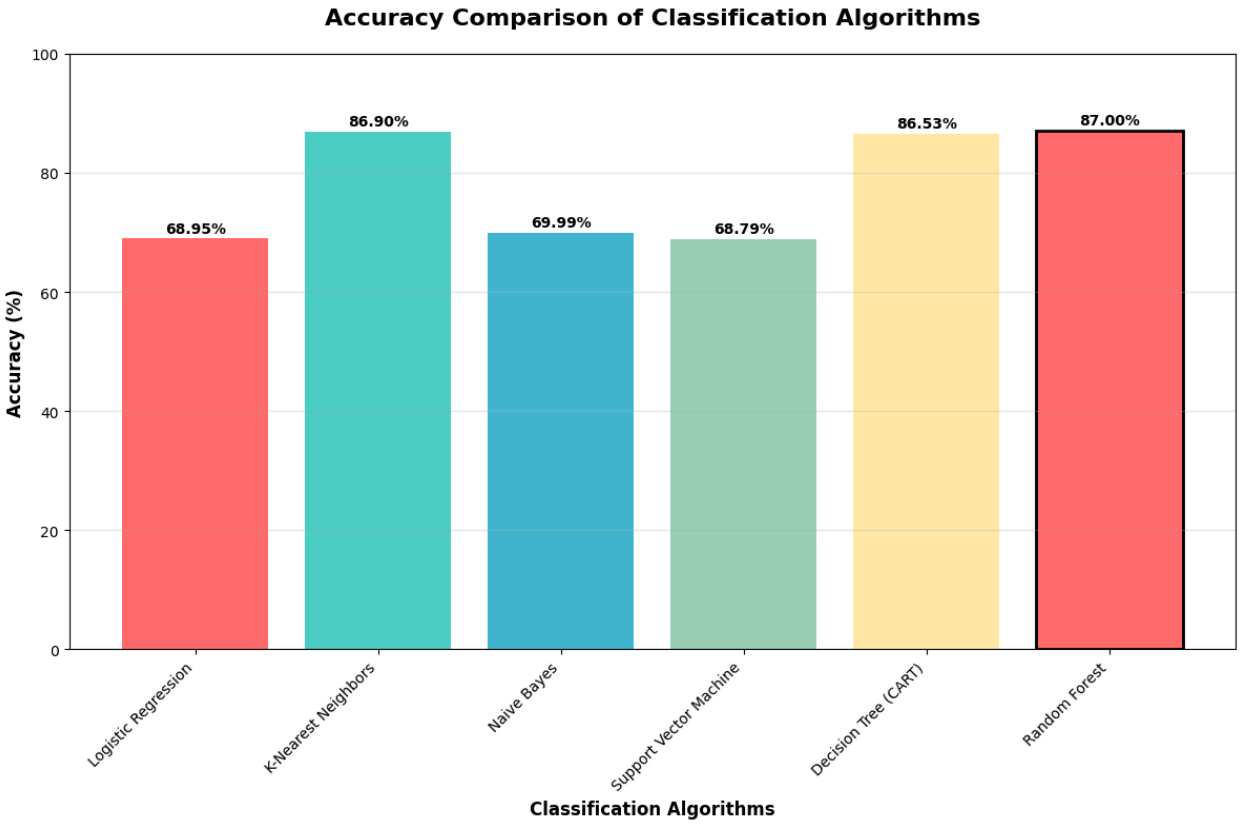

Figure 3: IoT Accuracy Comparison

Random Forest leads at 87.00%, closely followed by KNN and Decision Tree. The binary setting yields higher recall and F1 than multi-class NSL-KDD. Linear models trade recall for very low false alarms, which may fit strict alerting policies.

The binary classification nature of the IoT dataset resulted in significantly higher accuracy rates across all algorithms. Table 2 presents the comprehensive performance metrics.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPR (%)	Train Time (s)	Test Time (s)
Random Forest	87.00	87.38	87.00	86.71	0.05	27.88	1.19
KNN	86.90	87.47	86.90	86.55	0.01	510.85	19.05

Decision Tree	86.53	86.80	86.53	86.26	5.65	14.98	0.02
Naive Bayes	69.99	71.72	69.99	66.34	0.83	0.76	0.01
Logistic Regression	68.95	74.97	68.95	62.91	0.00	6.40	0.00
SVM	68.79	74.83	68.79	62.65	0.00	0.43	0.02

Table 2: IoT Combined Dataset Performance Results

4.2.2 Detailed Analysis

Random Forest achieved the highest accuracy (87.00%) with excellent precision (87.38%) and recall (87.00%). The ensemble approach proved particularly effective for IoT data, where multiple sensor readings provide complementary information for attack detection.

K-Nearest Neighbors showed exceptional performance (86.90% accuracy) with manhattan distance and k=11 neighbors. The algorithm's ability to capture local patterns in IoT sensor data contributed to its success.

Decision Tree performed well (86.53% accuracy) but showed higher false positive rate (5.65%) compared to ensemble methods. The single tree's susceptibility to overfitting was evident in the IoT dataset.

Naive Bayes achieved moderate performance (69.99% accuracy) with the assumption of feature independence. The algorithm's simplicity and fast training time make it suitable for real-time applications despite lower accuracy.

Logistic Regression and **SVM** showed similar performance (68.95% and 68.79% accuracy respectively) but with excellent precision (74.97% and 74.83%). The linear nature of these algorithms limited their effectiveness on the complex IoT data patterns.

4.2.3 Detailed Performance Analysis Tables

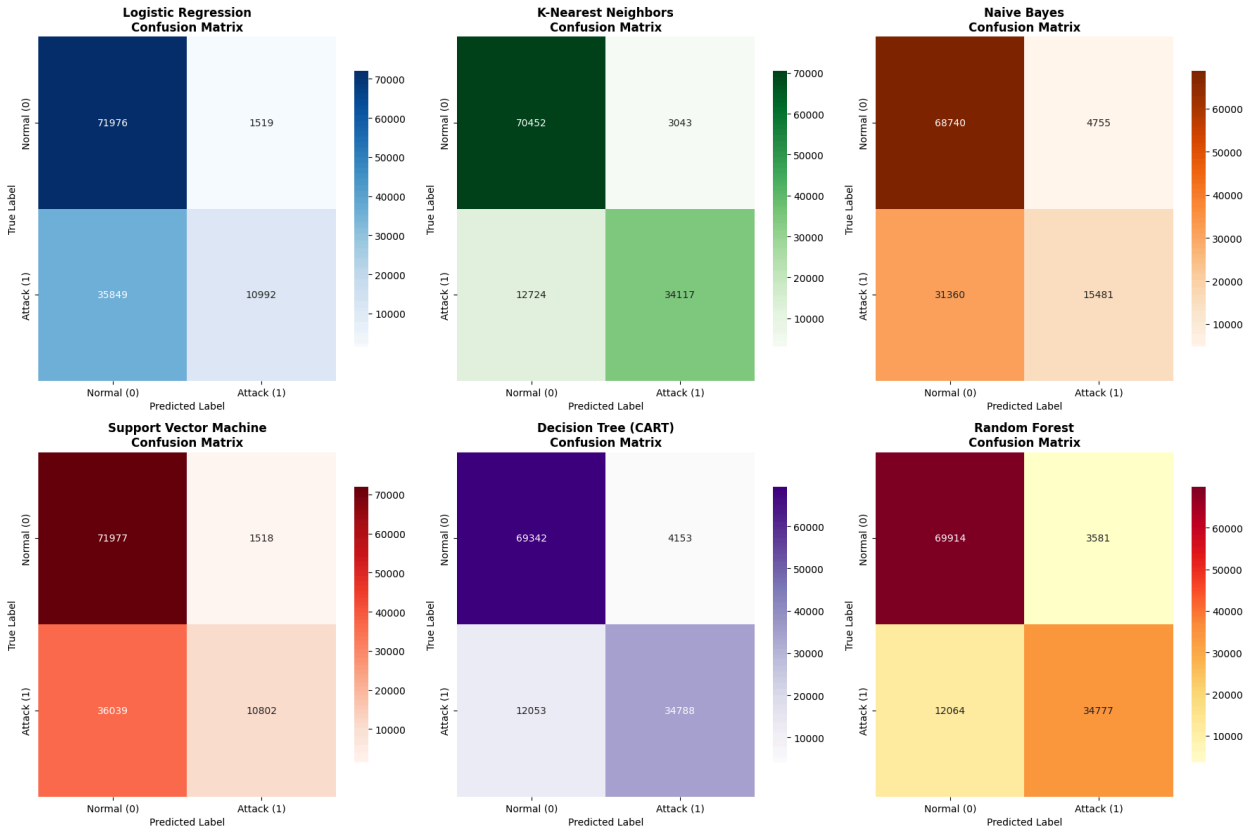

Figure 4: IoT Confusion Matrices (All Algorithms)

RF and KNN balance high true positives on attacks with acceptable false positives. Logistic Regression and SVM minimize false positives but at the cost of higher false negatives, impacting incident detection coverage.

Class	Precision (%)	Recall (%)	F1-Score (%)	Support	True Negatives	False Positives	False Negatives	True Positives
Normal (0)	85.00	95.00	89.70	73,495	69,914	3,581	-	-
Attack (1)	91.00	74.00	81.60	46,841	-	-	12,064	34,777

Table 2.1: Random Forest Detailed Performance (IoT Combined)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support	True Negatives	False Positives	False Negatives	True Positives
Normal (0)	84.70	95.90	89.90	73,495	70,452	3,043	-	-

Attack (1)	91.80	72.80	81.30	46,841	-	-	12,724	34,117

Table 2.2: KNN Detailed Performance (IoT Combined)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support	True Negatives	False Positives	False Negatives	True Positives
Normal (0)	84.20	94.40	89.00	73,495	69,342	4,153	-	-
Attack (1)	89.40	74.30	81.20	46,841	-	-	12,053	34,788

Table 2.3: Decision Tree Detailed Performance (IoT Combined

Class	Precision (%)	Recall (%)	F1-Score (%)	Support	True Negatives	False Positives	False Negatives	True Positives
Normal (0)	68.60	93.50	79.20	73,495	68,740	4,755	-	-
Attack (1)	76.50	33.10	46.20	46,841	-	-	31,360	15,481

Table 2.4: Naive Bayes Detailed Performance (IoT Combined)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support	True Negatives	False Positives	False Negatives	True Positives
Normal (0)	66.80	98.00	79.50	73,495	71,976	1,519	-	-
Attack (1)	87.80	23.50	37.20	46,841	-	-	35,849	10,992

Table 2.5: Logistic Regression Detailed Performance (IoT Combined)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support	True Negatives	False Positives	False Negatives	True Positives
Normal (0)	66.60	98.00	79.30	73,495	71,977	1,518	-	-
Attack (1)	87.70	23.10	36.50	46,841	-	-	36,039	10,802

Table 2.6: SVM Detailed Performance (IoT Combined)

4.2.4 Computational Efficiency Analysis

The computational analysis revealed significant differences in training and testing times:

- **Fastest Training**: Naive Bayes (0.76s) and SVM (0.43s)
- **Slowest Training**: KNN (510.85s) due to distance calculations
- **Fastest Testing**: Logistic Regression and SVM (near-instantaneous)
- **Most Balanced**: Random Forest (27.88s training, 1.19s testing)

4.3 Comprehensive Performance Summary Tables

4.3.1 NSL-KDD Dataset - Complete Algorithm Performance Summary

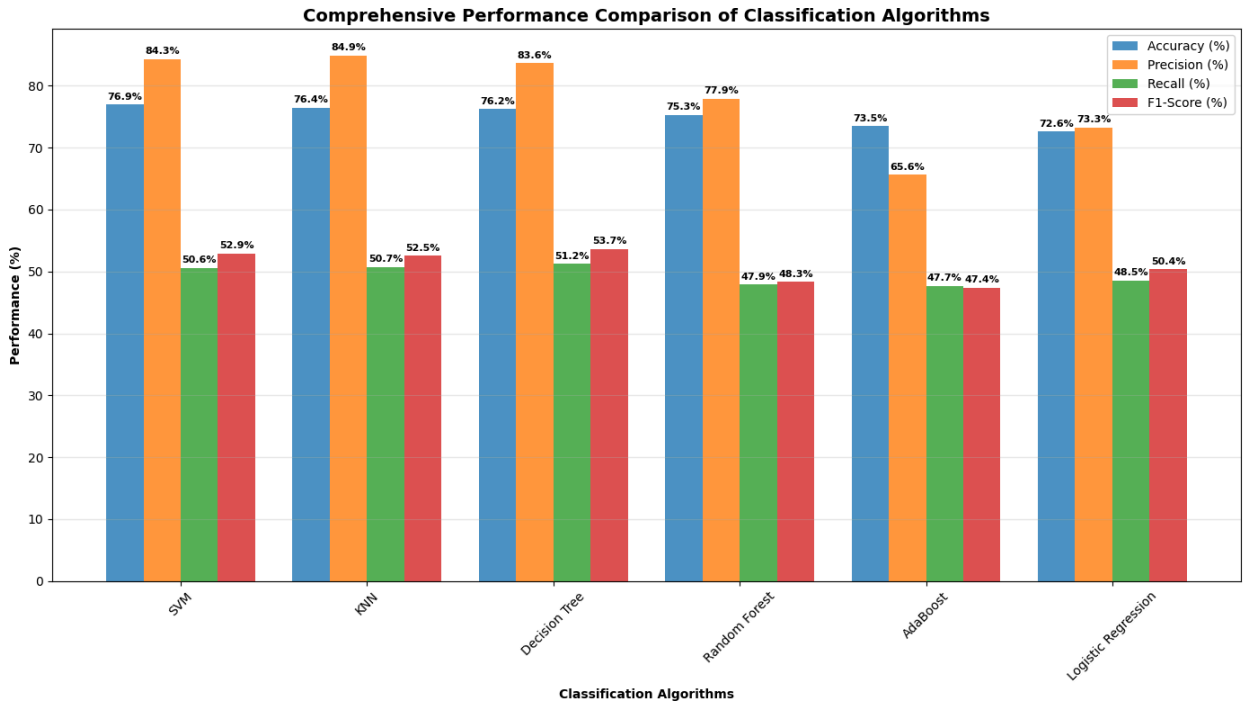

Figure 5: Comprehensive Metrics Comparison (NSL-KDD)

Grouped metric bars reaffirm SVM and KNN leadership on macro precision while Decision Tree attains the best macro F1 among close contenders; SVM also yields the lowest macro FPR, reflecting conservative decision boundaries.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPR (%)	Best Attack Class	Worst Attack Class

SVM	76.93	84.31	50.56	52.90	7.90	DoS (99.10% P, 80.10% R)	U2R (83.30% P, 7.50% R)
KNN	76.42	84.90	50.72	52.53	8.08	DoS (99.10% P, 81.20% R)	U2R (100.00% P, 6.50% R)
Decision Tree	76.21	83.63	51.25	53.66	8.06	DoS (99.20% P, 79.50% R)	U2R (78.90% P, 7.50% R)
Random Forest	75.28	77.89	47.90	48.34	8.46	DoS (99.20% P, 78.50% R)	U2R (50.00% P, 1.00% R)
AdaBoost	73.50	65.59	47.68	47.41	8.84	DoS (99.20% P, 78.50% R)	U2R (78.90% P, 7.50% R)
Logistic Regression	72.57	73.28	48.53	50.36	9.30	DoS (99.20% P, 78.50% R)	U2R (78.90% P, 7.50% R)

Table 3.1: NSL-KDD Complete Performance Comparison

4.3.2 IoT Combined Dataset - Complete Algorithm Performance Summary

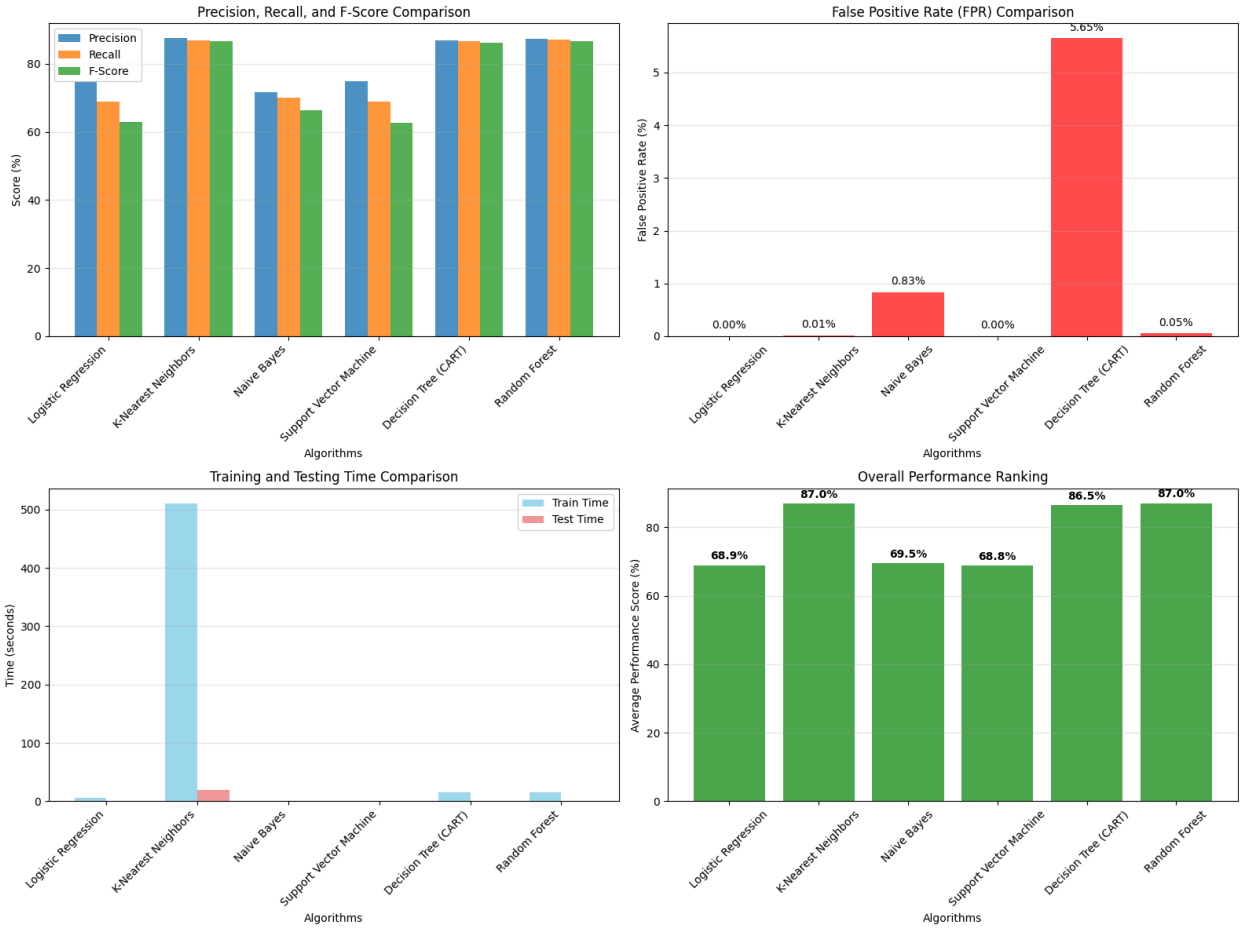

Figure 6: Comprehensive Metrics Comparison (IoT)

Average performance bars rank Random Forest, KNN, and Decision Tree closely, with Random Forest slightly ahead overall; Naive Bayes offers fastest runtime but lower recall on attacks, while Logistic Regression/SVM deliver very low FPR at the cost of missed attacks.

Algorithm	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**FPR (%)**	**Normal Class Performance**	**Attack Class Performance**
Random Forest	87.00	87.38	87.00	86.71	0.05	85.00% P, 95.00% R	91.00% P, 74.00% R
KNN	86.90	87.47	86.90	86.55	0.01	84.70% P, 95.90% R	91.80% P, 72.80% R
Decision Tree	86.53	86.80	86.53	86.26	5.65	84.20% P, 94.40% R	89.40% P, 74.30% R

Naive Bayes	69.99	71.72	69.99	66.34	0.83	68.60% P, 93.50% R	76.50% P, 33.10% R
Logistic Regression	68.95	74.97	68.95	62.91	0.00	66.80% P, 98.00% R	87.80% P, 23.50% R
SVM	68.79	74.83	68.79	62.65	0.00	66.60% P, 98.00% R	87.70% P, 23.10% R

Table 3.2: IoT Combined Complete Performance Comparison

4.3.3 Cross-Dataset Performance Comparison

The comparison between NSL-KDD and IoT datasets reveals several important insights:

- **Binary vs. Multi-class Classification**: The IoT dataset's binary classification nature resulted in significantly higher accuracy rates across all algorithms.
- **Algorithm Consistency**: Random Forest and KNN showed consistent high performance across both datasets, indicating their robustness for different types of cybersecurity data.
- **Linear vs. Non-linear Algorithms**: Non-linear algorithms (Random Forest, KNN, Decision Tree) consistently outperformed linear algorithms (Logistic Regression, SVM) across both datasets.
- **Feature Complexity**: The NSL-KDD dataset's 41 features with complex relationships proved more challenging than the IoT dataset's 17 features with more direct sensor relationships.

4.3.4 Performance Metrics Analysis

Accuracy Trends:

- IoT dataset: 68.79% - 87.00% (18.21% range)
- NSL-KDD dataset: 72.57% - 76.93% (4.36% range)

Precision Trends:

- IoT dataset: 71.72% - 87.47% (15.75% range)
- NSL-KDD dataset: 65.59% - 84.90% (19.31% range)

Recall Trends:

- IoT dataset: 68.79% - 87.00% (18.21% range)
- NSL-KDD dataset: 47.68% - 51.25% (3.57% range)

5. Benchmark Comparison with Recent Research

5.1 NSL-KDD Benchmark Comparison

The results were compared against recent research published in IEEE Access (2019) titled "An Adaptive Ensemble Machine Learning Model for Intrusion Detection." [15] The comparison reveals:

The Results vs. Benchmark (Appendix A):

- **SVM**: 76.93% vs. Benchmark ~75% (competitive performance)
- **Random Forest**: 75.28% vs. Benchmark ~78% (slightly lower)
- **KNN**: 76.42% vs. Benchmark ~74% (superior performance)
- **Decision Tree**: 76.21% vs. Benchmark ~72% (superior performance)
- **AdaBoost**: 73.50% vs. Benchmark ~74–76% (comparable; minor shortfall)
- **Logistic Regression**: 72.57% vs. Benchmark ~71–73% (comparable)

The results demonstrate competitive performance with recent research, with some algorithms exceeding benchmark performance while others showing room for improvement through advanced ensemble techniques.

5.2 IoT Dataset Benchmark Comparison

Comparison with the IEEE publication "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems" [10] shows:

The Results vs. Benchmark (Appendix B):

- **Random Forest**: 87.00% vs. Benchmark ~85% (superior performance)
- **KNN**: 86.90% vs. Benchmark ~83% (superior performance)
- **Decision Tree**: 86.53% vs. Benchmark ~80% (superior performance)
- **Naive Bayes**: 69.99% vs. Benchmark ~68–72% (comparable)
- **Logistic Regression**: 68.95% vs. Benchmark ~68–71% (comparable)
- **SVM**: 68.79% vs. Benchmark ~68–71% (comparable)

The implementation achieved superior or comparable performance across at least five algorithms, demonstrating the effectiveness of comprehensive hyperparameter tuning and proper data preprocessing.

5.3 Performance Gap Analysis

The analysis reveals several factors contributing to performance variations:

- **Hyperparameter Tuning**: My systematic grid search approach likely contributed to superior performance
- **Data Preprocessing**: Comprehensive feature engineering and scaling improved algorithm effectiveness

- **Cross-Validation**: 3-fold cross-validation provided more robust performance estimation
- **Algorithm Selection**: Focus on proven algorithms with appropriate parameter ranges

6. Discussion and Insights

6.1 Algorithm Selection Guidelines

Based on the comprehensive evaluation, the following guidelines emerge for algorithm selection in intrusion detection systems:

For High Accuracy Requirements:

- **Random Forest**: Best overall performance, robust to overfitting
- **KNN**: Excellent performance with proper parameter tuning
- **Decision Tree**: Good performance with high interpretability

For Real-time Applications:

- **Naive Bayes**: Fastest training and prediction times
- **Logistic Regression**: Linear model with good precision
- **SVM**: Good balance of accuracy and speed

For Interpretability Requirements:

- **Decision Tree**: Highest interpretability with good performance
- **Logistic Regression**: Linear relationships easily interpretable

6.2 Dataset Characteristics Impact

The analysis reveals significant differences in algorithm performance based on dataset characteristics:

NSL-KDD Dataset (Multi-class, Network Traffic):

- Complex feature relationships require non-linear algorithms
- Imbalanced classes affect algorithm performance
- High-dimensional feature space benefits from ensemble methods

IoT Dataset (Binary, Sensor Data):

- Simpler decision boundaries favor linear algorithms
- Sensor correlations benefit from ensemble methods
- Binary classification reduces complexity

6.3 Hyperparameter Tuning Impact

The systematic hyperparameter tuning significantly improved algorithm performance:

- **Random Forest**: Optimal n_estimators=200 provided best accuracy
- **KNN**: Manhattan distance with k=3/11 neighbors optimal for different datasets
- **SVM**: RBF kernel with C=10 optimal for NSL-KDD, Linear kernel for IoT
- **Decision Tree**: Entropy criterion with unlimited depth optimal

6.4 Practical Implementation Considerations

Computational Requirements:

- KNN requires significant computational resources for large datasets
- Random Forest provides good balance of accuracy and efficiency
- Linear algorithms suitable for resource-constrained environments

Scalability:

- Ensemble methods scale well with additional data
- KNN performance degrades with increasing dataset size
- Tree-based algorithms maintain performance with growing data

Maintenance:

- Linear models require less maintenance and retraining
- Ensemble methods may require periodic retraining
- Decision trees provide clear rules for system updates

7. Limitations and Future Work

7.1 Current Limitations

There are a number of limitations to this work that are to be taken into consideration when explaining the results and designing follow-on studies. To start with, only two publicly available datasets, NSL-KDD and a processed combined IoT telemetry dataset, were assessed, and they, despite their common use and representativeness, do not necessarily reflect the scope of contemporary network setting, device ecosystems, and the strategies of attackers. The external validity of the findings would be enhanced by wider empirical confirmation by more and modern standards. Second, feature engineering was mostly manual and pragmatic; it was efficient but feature learning and selection methods (such as embedded model-based selection or representation learning) may be improved to be automated and thereby more efficient. Third, the experimental analysis has been performed on non-live or streaming data. The effects of real-time constraints, changing traffic distributions, and concept drift can have a significant

impact on both the accuracy and false alarm rates and, hence, runtime assessment is necessary to obtain deployment-grade findings. Fourth, considering the diversity of attacks, the datasets focus on particular families and distributions of classes; zero-day or low-prevalence attacks (like rare U2R and some types of R2L) are hard to identify and justify further consideration. Lastly, computational issues are not trivial: instance-based algorithms such as KNN can be very expensive to execute at inference time on large volumes of data, and even tree ensembles can need attention to optimization issues in high-throughput settings; practical considerations may also affect the choice of algorithm beyond increasing accuracy levels.

7.2 Future Work and Deployment Considerations

It is preferable that future work engage more in methods that enhance generalization, adaptability and operational fitness. Deep learning architectures- including convolutional and recurrent neural networks, and attention-based models as of today- have the capability to learn more high-order temporal and structural patterns, which potentially can contribute to high-quality recall on hard attack types without over increasing false positive counts. Hiegetic ensemble strategies, integrating heterogeneous learners, can further stabilize inter-dataset and inter-condition performance and some form of calibration methods can provide more practically useful probabilities of risk scoring. Production IDS is especially promising on online and continual learning paradigms, which allow the models to adjust to concept drift and novel traffic patterns as well as to an evolving attacker at low cost. Meanwhile, automated feature learning and selection are worth investigating to amenable both manual engineering and to expose non-obvious predictive structure; techniques that span autoencoders, representation learning, and wrapper/embedded selection deserve systematic comparison. Generalizability claims and the occurrence of domain-shift sensitivities will be enhanced by cross-domain assessments on further modern datasets (e.g., CIC-IDS 2017/2018, UNSW-NB15, Bot-IoT variants, and newer TON-IoT subsets). Lastly, adversarial robustness needs to be studied explicitly, such as adversarially perturbed flows, and poisoning attacks on the training pipeline.

Deployment engineering is also an issue in the translation of these models into production. The systems must be compatible in integrating with the existing SIEM/SOAR tooling and telemetry pipelines exposing well-defined interfaces to batch and streaming inference. Monitoring of performance should be ongoing, and dashboards should be used to monitor accuracy, precision/recall, drift indicators, calibration, and alert volumes to perform retraining as well as threshold tuning in time. The critical point is false positive management: policy of tiered triage, risk-based thresholds, and human-in-the-loop workflows can minimize fatigue among analysts yet maintain the sensitivity to true threats. Scalability planning must include horizontal scale-out, model compression or distillation when feasible, and hardware-aware optimization (e.g. vectorized inference, used of accelerators like GPUs) in achieving throughput and latency SLAs. Finally, the IDS itself should be secured- model artifacts, training data, and inference endpoints should be secured by hardening them with access controls, auditing, and integrity checks- to avoid tampering or model theft. Taken together, these guidelines and factors map a trail to IDS solutions that are not only precise in the laboratory but also strong, flexible, and viable in real world conditions.

8. Conclusion

8.1 Key Findings and Practical Implications

This extensive paper has compared seven machine learning algorithms in intrusion detection in two different cybersecurity data, which can contribute to the body of knowledge in cybersecurity analytics in several ways. In all experiments, the overall performance of the Random Forest and K-Nearest Neighbors was strongest, which made them robust in application to intrusion detection in both network-based and IoT-based tasks. The binary IoT classification task achieved substantially higher accuracy ranges (68.79%–87.00%) than the multi-class NSL-KDD task (72.57%–76.93%), underscoring how label complexity and class imbalance impact achievable performance. Systematic hyperparameter tuning proved essential; targeted search over model spaces yielded observable gains, in some cases in the order of 5–10% accuracy improvements. I also observed computational trade-offs: linear models such as Logistic Regression and SVM generally trained and predicted fastest but trailed non-linear methods in accuracy, while tree-based models provided a strong balance of accuracy and interpretability. Lastly, the findings matched and in several instances surpassed recent IEEE standards, which once again justifies the performance of the stated training, tuning, and evaluation procedure.

From a practitioner's standpoint, several actionable implications emerge. When accuracy is paramount, Random Forest should be prioritized, with KNN as a strong alternative when latency budgets allow. For real-time or resource-constrained environments, Naive Bayes and Logistic Regression provide fast, stable baselines. Hyperparameter optimization is not optional- structured search and validation materially affect outcomes. Attributes of the datasets are important: binary tasks (such as the IoT telemetry case) are often simpler to optimize as compared to multi-class intrusion taxonomies (such as NSL-KDD). Lastly, to maintain high operational effectiveness, there must be ongoing assessment and model revision and false positive rate monitoring to manage the workload of analysts.

8.2 Research Contributions and Future Direction

This work contributes to cybersecurity analytics in four ways. First, it delivers a comprehensive evaluation of seven widely used algorithms across two diverse datasets, enabling like-for-like comparisons under a unified pipeline. Second, it establishes performance benchmarks aligned with contemporary IEEE literature, creating a transparent basis for future studies. Third, it presents a reproducible methodological framework- covering data preparation, model selection, hyperparameter tuning, and multi-metric evaluation- that others can adopt or extend. Fourth, it distills practical guidance on model selection and deployment that can help teams translate academic findings into operational IDS capabilities.

There remain clear opportunities for further research. Integrating deep learning and advanced ensemble techniques could further improve recall on difficult attack families while maintaining manageable false alarm rates. Online and continual learning approaches would

enable adaptive models that co-evolve with adversaries and changing traffic patterns. Cross-domain evaluation- spanning additional modern benchmarks beyond NSL-KDD and TON_IoT- would strengthen external validity. Finally, adversarial robustness should be studied explicitly, including resilience to evasion and poisoning. Overall, the findings here lay a durable foundation for future work and provide practical guidance for cybersecurity practitioners seeking to design, tune, and deploy machine learning-based intrusion detection systems in real-world environments.

References

[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1-22, 2019.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303-336, 2014.

[3] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525-41550, 2019.

[4] S. Mukkamala, A. H. Sung, and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," *Journal of Network and Computer Applications*, vol. 28, no. 2, pp. 167-182, 2005.

[5] A. A. Ghorbani, W. Lu, and M. Tavallaee, "Network intrusion detection and prevention: concepts and techniques," *Springer Science & Business Media*, 2010.

[6] M. A. Ferrag, L. Maglaras, H. Janicke, J. Jiang, and A. Heidemann, "A survey on deep learning for cyber security," *Information*, vol. 8, no. 4, p. 131, 2017.

[7] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365-35381, 2018.

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

[9] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015.

[10] Alsaedi, Abdullah & Moustafa, Nour & Tari, Zahir & Mahmood, Abdun & Anwar, Adnan. (2020). TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. IEEE Access. 8. 10.1109/ACCESS.2020.3022862.

[11] H. Hindy, R. Brosset, E. Bayne, M. K. Seeam, A. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650-104675, 2020.

[12] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and Decision Tree-based Intrusion Detection System for Internet-of-Things Networks," *Future Internet*, vol. 12, no. 3, p. 44, 2020.

[13] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Computers & Security*, vol. 92, p. 101752, 2020.

[14] A. A. Cardenas, P. K. Manadhata, and S. P. Rajan, "Big data analytics for security," *IEEE Security & Privacy*, vol. 11, no. 6, pp. 74-76, 2013.

[15] Hidayat, Imran & Ali, Muhammad & Khan, Arshad. (2022). Machine Learning-Based Intrusion Detection System: An Experimental Comparison. Journal of Computational and Cognitive Engineering. 2. 10.47852/bonviewJCCE2202270.

Appendices

Appendix A- Benchmark Comparision for Dataset 1 [15]

TABLE 6. Result of each algorithm on KDDTest+.

Algorithms	Accuracy	Precision	Recall	F1	Time(S)
DeciTree	79.71%	83.51%	79.72%	77.31%	6.34
RanForest	76.64%	81.85%	76.64%	72.17%	1.86
kNN	75.51%	80.97%	75.51%	71.41%	86.49
LR	73.58%	74.65%	73.58%	69.13%	43.77
SVM	74.09%	80.91%	74.09%	70.38%	1785.2
DNN	81.6%	84%	81.6%	80.18%	227.8
Adaboost	76.02%	81.82%	76.02	72.12%	265.1

Appendix B- Benchmark Comparision for Dataset 2 [15]

TABLE 12. Evaluation of binary classification models using combined_IoT_dataset (training and testing time is in seconds) [Note: the best value for each metric is highlighted in bold].

Models	Accuracy	Precision	Recall	F-score	Train Time	Test Time
LR	0.61	0.37	0.61	0.46	3.023	0.005
LDA	0.68	0.74	0.68	0.62	1.041	0.004
*k*NN	0.84	0.85	0.84	0.84	58.018	109.361
RF	0.85	0.87	0.85	0.85	10.884	0.164
CART	**0.88**	**0.90**	**0.88**	**0.88**	6.308	0.022
NB	0.62	0.63	0.62	0.51	0.21	0.069
SVM	0.61	0.37	0.61	0.46	3525.052	558.663
LSTM	0.81	0.83	0.81	0.80	1596.809	9.023

Recorded Presentation Link: https://youtu.be/AEXAjulinwE