Evaluating Long-Term Performance of ML-based Intrusion Detection Systems under Temporal and Structural Changes

Minh Nhat Vo
Department of Information Security Academy of Cryptography Techniques
Ho Chi Minh, Vietnam
vonhat010@gmail.com

Xuan Hung Truong
Academy of Cryptography Techniques
Hanoi, Vietnam
hungtx.ncs@actvn.edu.vn

Abstract— While Machine Learning (ML) has advanced Intrusion Detection Systems (IDS), validating its long-term reliability remains a challenge. Standard protocols, which typically test models on held-out attack patterns, often neglect the temporal drift inherent in live networks. To address this limitation, we introduce an evaluation framework that enforces a chronological separation between training and testing datasets, thereby simulating realistic structural changes in traffic. Our study implemented six prevalent models (including DT, RF, SVM, ANN, and DNN) using benchmark datasets such as CIC-IDS2017 and CSE-CIC-IDS2018. Contrary to expectations based on training metrics, results indicate that complex tree-based models (DT, RF) suffer significantly from overfitting when facing temporal shifts. In contrast, SVM and ANN exhibited greater stability. Data from LUFlow further confirms that minimizing the structural divergence between time-stamped datasets is key to maintaining model robustness.

Keywords: Detection System (IDS); Machine Learning (ML); Network Security; CIC-IDS2017; CSE-CIC-IDS2018; LUFlow Dataset

I. INTRODUCTION

Cybersecurity has become increasingly critical as the frequency and sophistication of cyberattacks continue to grow, targeting vital infrastructures such as Supervisory Control and Data Acquisition (SCADA) systems, power grids, hospital. To defend against such threats, organizations deploy various protective measures, among which Intrusion Detection Systems (IDS) play a central role. IDS can be categorized as either Network-based IDS (NIDS) or Host-based IDS (HIDS), and they operate by matching known attack signatures or by identifying anomalous behaviors. While signature-based IDS provides high accuracy for known threats, it cannot detect novel attacks. In contrast, anomaly-based IDS is more flexible and capable of identifying new threats, but it often suffers from high false-positive rates.

In recent years, Machine Learning (ML) has been widely applied to IDS to enhance detection capabilities. By learning traffic patterns, ML-based IDS can identify both known and previously unseen attacks, including zero-day threats. Algorithms such as Support Vector Machines (SVM) and Artificial Neural Networks (ANN) have shown promising results in this context. Despite these advancements, most IDS evaluation studies rely on a single dataset for both training and testing. This practice, while convenient, does not reflect realistic network conditions where traffic patterns and attack strategies evolve over time.

To address the limitations of traditional evaluation methods, this study proposes a temporal-based evaluation framework. The main contributions of this paper are summarized as follows:

- We propose a chronological evaluation methodology where training and testing data are strictly separated by time to simulate realistic concept drift, moving beyond standard random k-fold cross-validation.
- We provide a scientific justification for feature selection based on Gini Impurity (Random Forest), demonstrating its superiority in preserving feature interpretability compared to transformation-based methods like PCA.
- We conduct a comprehensive empirical analysis of six ML/DL models, identifying that while tree-based models (DT, RF) excel in static environments, SVM and ANN exhibit superior robustness against temporal shifts in network traffic

II. LITERATURE REVIEW

Intrusion Detection Systems (IDS) have evolved from traditional signature-based approaches, which are effective for known threats but fail against novel attacks, to anomaly-based techniques that can identify new behaviors but often suffer from high false positives. With the advancement of machine learning (ML), models such as Decision Trees, Random Forests, Support Vector Machines (SVM), and Artificial Neural Networks (ANN) have been

https://doi.org/10.65153/bw356376
Journal of Science and Technology of East Asia University of Technology

widely applied to improve detection accuracy. Several studies using benchmark datasets like KDD99, CIC-IDS2017, and CSE-CIC-IDS2018 have reported high detection rates, particularly with SVM and ANN models[15] [16]. More recently, deep learning (DL) methods, including Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), have shown the ability to extract hierarchical features from raw traffic data, improving scalability and robustness in intrusion detection. Hybrid approaches that integrate ML/DL models with statistical or rule-based methods have also been proposed to balance accuracy and interpretability.

Despite these advances, most evaluations still rely on training and testing within a single dataset. This practice often leads to overfitting and does not reflect real-world scenarios where network conditions and attack strategies evolve over time. Recent research highlights the importance of cross-dataset evaluation and temporal analysis, underscoring the need for methodologies that can assess the long-term effectiveness of IDS in dynamic environments.

III. THEORETICAL BACKGROUND

A. Dataset

Datasets for IDS are almost always very difficult to find as organizations rarely share them publicly, making the search quite challenging. In this study, we utilized the CIC-IDS2017, CIC-IDS2018, and LUFlow datasets, which are among the few publicly available and widely accepted benchmarks in the intrusion detection community. These datasets provide diverse and realistic network traffic scenarios, enabling a comprehensive evaluation of the proposed machine learning models.

CIC-IDS2017 dataset [1] is a comprehensive benchmark for evaluating Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). It includes up-to-date benign traffic and diverse common attack types such as Brute Force, DoS, DDoS, Heartbleed, and infiltration, captured over five days with realistic user behavior simulation. The dataset meets 11 key criteria for reliability, including complete network configuration, labeled flows, multiple protocols, and detailed metadata. CIC-IDS2017 enables robust evaluation of machine learning models for intrusion detection and is publicly available for research purposes.

CSE-CIC-IDS2018 dataset [2] [3] is a realistic cyber defense dataset created through collaboration between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). It contains network traffic data collected from a victim organization with 5 departments, 420 PCs, and 30 servers, alongside an attacking infrastructure of 50 machines. The dataset includes seven attack scenarios: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and internal network infiltration. Network traffic and system log files were captured, from which 80 network traffic features were extracted using CICFlowMeter-V3. This dataset simulates diverse and realistic network conditions with detailed intrusion profiles, making it valuable for research and development of intrusion detection systems and cybersecurity analytics.

LUFlow dataset [4], developed by Lancaster University in 2020, is a flow-based dataset designed for intrusion detection research. It consists of real-world network traffic captured via honeypots deployed in Lancaster University's public network space. This dataset labels network flows as benign, malicious, or outlier, with outliers representing suspicious traffic that cannot be definitively classified as attacks. The real-world nature of the dataset allows it to reflect emerging threats, but it lacks detailed attack type labels due to the inherent uncertainty of intent in live network data. LUFlow contains 16 features per flow and includes telemetry correlated with external cyber threat intelligence to provide a reliable ground truth for malicious behavior.

B. Meaching Learning Models

Machine learning (ML) techniques have demonstrated considerable success in data prediction and classification across various research areas. Within intrusion detection systems (IDS), ML methods are commonly used to distinguish between normal and malicious network traffic. This section provides a brief overview of some of the most commonly utilized ML algorithms in the IDS field. [5] [6]

Decision Tree(DT) [7] classifier organizes data classification using a tree structure composed of nodes and leaves. Nodes represent decision conditions, while leaves indicate the classification

results. During training, the tree is built iteratively by selecting features to split the data at each node, aiming to create homogeneous groups. Feature selection relies on metrics such as Gini impurity or Entropy to evaluate split quality. Features near the root node contribute more to the classification outcome. In testing, each sample travels from the root through the tree according to node conditions until reaching a leaf, which determines its class label.

Random Forest (RF) [8] is an ensemble learning method in artificial intelligence that combines multiple decision trees. Each tree is trained on a bootstrapped dataset, which is created by randomly sampling with replacement from the original training data. This process ensures that each tree differs slightly from the others. When making predictions, each tree votes on the class, and the class with the most votes becomes the final prediction. By leveraging multiple trees, Random Forest reduces overfitting and is more robust to noise, while maintaining the simplicity and interpretability of individual decision trees.

Support Vector Machine (SVM) [9] is a supervised machine learning model used primarily for classification tasks. The main goal of SVM is to find a hyperplane that best separates data points of different classes with the maximum margin. In cases where data cannot be linearly separated, SVM uses kernel functions to map data into higher-dimensional spaces to achieve separability. Despite being computationally intensive, SVM is favored for its effectiveness and lower tendency to overfit compared to other models.

Naïve Bayes (NB) [10] is a simple probabilistic classifier based on Bayes' theorem. It calculates the probability of a data instance belonging to each class under the assumption that all features are conditionally independent. During training, the model estimates these probabilities, which are then used to classify new data. Due to its straightforward probability calculations, Naïve Bayes is computationally efficient. Although the independence assumption is often unrealistic, this method can still achieve competitive accuracy compared to more complex models.

Artificial Neural Network (ANN) [11] [12] are computational models inspired by the structure and function of the human brain. They consist of layers of interconnected nodes called neurons. Data passes from the input layer through one or more hidden layers to the output layer, with each neuron applying a weighted sum of inputs followed by a nonlinear activation function. The model learns by optimizing these weights and biases during training to minimize prediction errors. ANNs can capture complex patterns in raw data without requiring manual feature engineering. However, training them can be computationally intensive and typically requires large datasets. Due to their ability to learn hierarchical representations, ANNs have become essential in various AI applications such as image and speech recognition.

Deep Neural Network (DNN) [13] [14] are multilayer artificial neural networks, typically consisting of two or more hidden layers. Since their successful training by Hinton et al. in 2006, DNNs have gained significant traction, especially in intrusion detection systems (IDS). Compared to single hidden layer networks, DNNs can learn more complex patterns due to their depth, yet they require more computational resources and larger datasets for effective training. The increased number of layers enables DNNs to capture hierarchical features, improving performance in diverse applications.

IV. RESEARCH DESIGN

The experimental workflow of this research include five phase. First, the datasets underwent preprocessing, including data cleaning and preparation. Second, feature selection was applied to improve the efficiency of the models. The third step, involved hyperparameter optimization to accuracy. Then, cross-validation was performed to validate the models. Finally, the trained models were evaluated on the testing dataset using multiple performance metrics.

In this study, the experiments were conducted twice, each employing a distinct pair of datasets. The first pair included CIC-IDS2017 and CSE-CIC-IDS2018, both developed by the Canadian Institute for Cybersecurity (CIC). Since these datasets were produced by the same organization, their feature sets match. In the experiments, CIC-IDS2017 was used for model training, while CSE-CIC-IDS2018 served as the test set.

A. Data pre-processing

The first step in the experiment involved preprocessing the dataset. This included cleaning by removing unwanted records such as those with

missing or infinite values and eliminating duplicates to provide unique samples for model training. Ensuring the dataset is balanced between normal and abnormal network traffic is critical to prevent bias during training. To address class imbalance, the majority classes were downsampled randomly, and minor classes were merged to create balanced categories. Additionally, both training and testing datasets were aligned to have identical columns in the same order and were relabeled if necessary to maintain consistent class labels.

B. Feature Selection

This rephrasing keeps the original meaning while avoiding direct copying.Feature selection is key to improving IDS performance by enhancing accuracy and efficiency . With around 80 features in CIC-IDS2017, training without selection is slow and may include noisy, less useful features. As Aksu et al. noted, using over 40 features often lowers accuracy.

Feature selection was conducted using the Random Forest (RF) algorithm, which evaluates feature importance based on Information Theoretic principles, specifically Gini Impurity. Unlike Principal Component Analysis (PCA), which transforms features into a new lower-dimensional space and results in a loss of interpretability, the RF-based method preserves the original network attributes (e.g., Destination Port, Flow Duration). This is crucial for cybersecurity analysts to identify specific attack vectors. The importance of a feature j is calculated by averaging the decrease in node impurity across all trees in the forest:

$$I(j) = \frac{1}{N_T} \sum_{t=1}^{N_T} \sum_{s \in S_t : v(s)=j} \Delta i(s,t) \quad (1)$$

where N_T is the number of trees, v(s) is the variable used at split s, and Δi is the impurity decrease. This approach allows us to scientifically identify the most relevant features without losing semantic meaning.

We used scikit-learn's RandomForestClassifier to train a random forest on the training data, selecting the top n features by importance. This method is common, as shown by prior studies . After initial reduction, we further narrowed features using brute force, adding features incrementally to build models and track accuracy. Based on this, a concise feature set was chosen, with some removed manually. Features like source IP, though correlated, were excluded to avoid overfitting and improve model generalizability.

C. ML Training

After selecting features, we trained the models using the training dataset. We optimized the model hyperparameters with grid search, using scikit-learn's GridSearchCV, which tests different hyperparameter combinations to find the best accuracy. For example, in a deep neural network (DNN), the number of hidden layers is a hyperparameter, while weights and biases are model parameters. Because tuning requires multiple trainings, only part of the data was used. While modern heuristic optimization algorithms (such as Genetic Algorithms or PSO) are efficient for vast search spaces, they provide approximate solutions. In this study, we employed Grid Search over a carefully defined bounded space to ensure reproducibility and guarantee finding the global optimum within the specified parameters.

To ensure reproducibility and identify the global optimum, the Grid Search explored the following hyperparameter spaces based on standard practices and preliminary tests:

- SVM: We examined the penalty parameter C ∈ {0.1, 1, 10, 100} and kernel functions ∈ {'linear', 'rbf', 'sigmoid'}. The result indicated that a higher C value (100) with the RBF kernel yielded the best generalization.
- Random Forest (RF) & Decision Tree (DT): We varied the splitting criteria ∈ {'gini', 'entropy'}, maximum depth ∈ {10, 20, None}, and for RF specifically, the number of estimators ∈ {50, 100, 200}.
- ANN & DNN: The search included variations in activation functions ∈ {'relu', 'tanh', 'logistic'} and solvers ∈ {'adam', 'sgd'}. For the architecture, we tested neuron configurations ranging from 1 hidden layer (30 to 50 neurons) for ANN, to 3 – 4 hidden layers (e.g., structures like [64, 32, 16] or uniform layers) for DNN.
- Naïve Bayes (NB): We tuned the portion of the largest variance of all features that is added to variances for calculation stability (var_smooth) within the range {10^-9, 10^-8, 10^-7}.

The final optimal parameters presented in the experimental setup were selected based on the highest validation accuracy derived from these combinations.

For the Deep Learning models (ANN and DNN), we designed a specific architecture to balance complexity and performance. The configuration is as follows:

https://doi.org/10.65153/bw356376

ANN: A Multi-Layer Perceptron (MLP) with one hidden layer containing 50 neurons.

DNN: A deep architecture with three hidden layers consisting of [64, 32, 16] neurons, respectively.

Optimization: Both models utilize the Adam optimizer (learning rate $\alpha = 0.001$) and ReLU activation functions for all hidden layers to mitigate the vanishing gradient problem. Dropout regularization (rate = 0.2) was applied to prevent overfitting.

We implemented decision tree (DT), random forest (RF), support vector machine (SVM), naïve Bayes (NB), artificial neural network (ANN), and deep neural network (DNN).

- The DT used cost complexity pruning (ccp_alpha) to avoid overfitting by cutting less important branches.
- The RF tuned tree depth, minimum samples for leaf and split, splitting criteria (Gini or entropy), and number of trees.
- The SVM used different kernels (linear, Gaussian RBF, sigmoid) defined as:

$$K(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}, \mathbf{x_j}) \tag{2}$$

$$K(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\gamma \left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2 + C\right) \tag{3}$$

$$K(\mathbf{x_i}, \mathbf{x_j}) = \tanh\left(\gamma(\mathbf{x_i}, \mathbf{x_j}) + r\right) \tag{4}$$

where γ,C,rγ,C,r are parameters optimized during training.

- NB tuned the Gaussian smoothing parameter (var_smooth).
- ANN, built with MLPClassifier, tuned hidden layers, neurons, activation functions (Logistic, Tanh, ReLU), solvers (SGD, Adam), and penalty alpha to prevent overfitting. Activation functions used were:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

$$f(x) = \tanh(x) \tag{6}$$

$$f(x) = \max(0, x) \tag{7}$$

- DNN was similar to ANN but had 3-4 hidden layers with equal neurons per layer with the same hyperparameter tuning as ANN.

D. Validate Model Accuracy

After training the models, we used k-fold cross-validation to evaluate their performance on the training dataset. This process involved splitting the data into k parts, using one part for testing and the rest for training in each iteration. The main goal was to ensure the accuracy remained consistent across all folds, which helped to detect overfitting. We then compared the model's accuracy to results from existing literature. If the accuracy was comparable, we moved on to the next step, otherwise, we revised the experiment to improve the model.

E. Model Evaluation

After validating the models, the final step was training and testing. Specifically, 70% of the training set was used for model training, the remaining 30% for validation, and the testing set for final evaluation. The study aimed to compare the accuracy and efficiency of different models across datasets. The performance metrics included Accuracy, Precision, Recall, and F1-score, defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F_1.Score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

Here, TP (true positive) and TN (true negative) represent correctly classified samples, while FP (false positive) and FN (false negative) are misclassified samples. A confusion matrix was also used for visualization. Furthermore, time complexity was measured in terms of training and prediction time, though no comparison with prior studies was made due to dependency on implementation, dataset size, and hardware.

V. RESULT AND ANAYLYST

This section presents in detail the experimental results and analysis, along with the environment used in this study. It discusses the findings derived from the CIC-IDS2017 and CSE-CIC-IDS2018.All experimental source code and related details are publicly accessible on a GitHub repository. [17] [18]

https://doi.org/10.65153/bw356376

A. *Enviroment*

The models for this study were implemented using the Python programming language within *a* Google Colab environment. Key libraries such as scikit-learn, pandas, NumPy, and Matplotlib were used for all experiments. The hardware specifications for this work included an Intel(R) Xeon(R) CPU @ 2.20 GHz, 12.7 GB of RAM, and an NVIDIA Tesla T4 GPU with 15 GB of VRAM, running on the Ubuntu 22.04.4 LTS operating system.

B. *Conduct experiments*

The CIC-IDS2017 (≈800 MB) and CSE-CIC-IDS2018 (≈6.4 GB) datasets are relatively large, and due to hardware constraints, only 10% of the latter was used. Both datasets required preprocessing, including the removal of records with missing or infinite values and duplicates. To address the severe class imbalance, resampling was applied to the training set. While we acknowledge that altering the class distribution may reduce the naturalness of the network traffic profile, training on the original highly imbalanced data would lead to a strong bias towards the benign class, causing the model to miss-detect minority attack types. Therefore, balancing was performed strictly during the training phase to ensure the models learn attack features effectively, while the validation relied on metrics that account for these distributions.

In CIC-IDS2017, attack classes with more than 100,000 samples and the benign class were downsampled to reach a 1:1 ratio between benign and malicious samples. For CSE-CIC-IDS2018, all malicious samples were retained, while the column structure was unified with CIC-IDS2017 by removing two unique columns.

Although these steps improved the class distribution, minority attack classes were not oversampled to avoid model bias. Instead, all malicious traffic was merged into a single "malicious" label, ensuring that the models would not overfit specific attack categories.

Feature selection plays a crucial role in training models with the CIC-IDS2017 dataset, which contains 78 features. In the experiments, only the CIC-IDS2017 dataset was utilized for this process. Due to the large number of features, manual inspection was not performed.

During feature selection with the Random Forest algorithm, 10% of the dataset was used for model training. The importance scores generated in this process were then applied to rank the features. The resulting ranking is illustrated in Fig.1

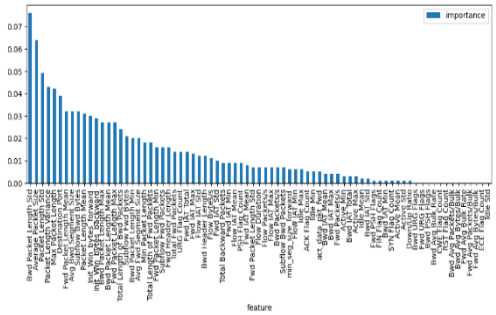

Figure 1. Feature relevance scores for cic-ids2017 dataset

A brute-force strategy was applied to further reduce features. Starting with Bwd Packet Length Std, features were sequentially added according to their ranking. As shown in Fig.2, model accuracy increased with additional features and peaked with the top 11.

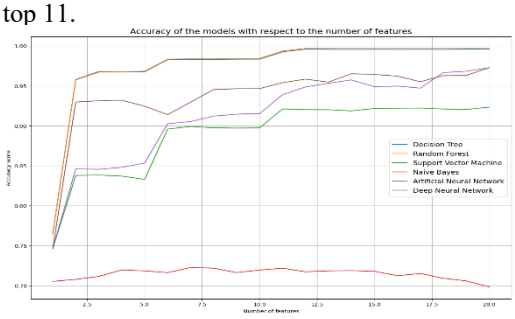

Figure 2. Model performance by number of features

After feature selection, the models were trained and tuned using GridSearchCV. Initial estimates suggested DT and RF could reach about 99% accuracy, ANN and DNN around 96%, SVM 93%, and NB 70%. Tuning had little effect on DT, RF, and NB, while SVM improved to 96% with C = 100, γ = 1, and the RBF kernel. It is noted that a high penalty parameter (C = 100) was selected for the SVM.

While, in theory, a large C can lead to overfitting by enforcing a hard margin, our empirical results on the independent test set (CSE-CIC-IDS2018) demonstrated that this configuration achieved the highest generalization accuracy (75.49%) compared to other models, including ANN (71.25%) and DT (70.41%) as shown in Table IV. Contrary to the common assumption that high C values degrade testing performance due to overfitting, in this specific cross-dataset evaluation, the stricter decision boundary provided by SVM (C = 100) proved more effective in distinguishing malicious flows under

temporal structural changes than the probabilistic boundaries of ANN. This suggests that, for the complex decision boundaries present in network traffic data, a stricter margin was necessary to effectively separate malicious flows from benign traffic.

ANN showed no gain as 40 neurons were already near the optimal 50. For DNN, three hidden layers with 15 neurons each raised accuracy to 97%. Both ANN and DNN performed best with the Tanh activation and Adam solver, showing that most models worked well with default settings, except ANN and DNN which required fine-tuning of hidden layers.

The validation results show that most models achieved high accuracy. DT and RF performed best with about 99%, followed by DNN at 96.98%. SVM and ANN reached around 95%, while NB had the lowest accuracy at 73.73%, indicating weak performance on the CIC-IDS2017 dataset in Table I.

Table I. Cross-Validation Accuracy Results (CIC-IDS2017)

Model	Fold	Accuracy	Mean Accuracy	Standard Deviation
DT	Fold 1	0.99476735	0.9947	0.0001
	Fold 2	0.99481865		
	Fold 3	0.99456215		
	Fold 4	0.99466475		
	Fold 5	0.99486982		
RF	Fold 1	0.99471605	0.9948	0.0003
	Fold 2	0.99530601		
	Fold 3	0.9949469		
	Fold 4	0.9944339		
	Fold 5	0.99469027		
SVM	Fold 1	0.95885703	0.9583	0.0012
	Fold 2	0.95690761		
	Fold 3	0.95954958		
	Fold 4	0.95677936		
	Fold 5	0.95918943		
NB	Fold 1	0.73067255	0.7301	0.0004
	Fold 2	0.73010824		
	Fold 3	0.72933874		
	Fold 4	0.73018519		
	Fold 5	0.73040913		
ANN	Fold 1	0.9669625	0.9640	0.0020
	Fold 2	0.96480788		
	Fold 3	0.96426922		
	Fold 4	0.9613451		
	Fold 5	0.96239579		
DNN	Fold 1	0.97440107	0.9727	0.0011
	Fold 2	0.97129739		
	Fold 3	0.97286205		
	Fold 4	0.97332376		
	Fold 5	0.97168142		

The validated accuracy of the models was compared with the works of Kostas [19] and Vinayakumar et al.[20], both using the CIC-IDS2017 dataset with binary classification. Results show that DT, RF, and SVM achieved higher accuracy than previous studies, while NB performed better than Vinayakumar et al. but worse than Kostas. Table II

TABLE II. ACCURACY COMPARISON

ML Models	Accuracy		
	This Work	Kostas	Vinayakumar et al
DT	0.9947	0.95	0.94
RF	0.9948	0.94	0.94
SVM	0.9583	-	0.80
NB	0.7301	0.87	0.31
ANN	0.9640	0.97	0.96
DNN	0.9727	-	0.94

This study compares the performance of machine learning models with other research. Kostas achieved higher accuracy by using model-specific feature selection. In contrast, Vinayakumar et al. had lower accuracy (except for their ANN) because their focus was on optimizing DNNs, not other models. Compared to Hossain et al., this research used a progressive dataset instead of a random split, which provides a more realistic evaluation of model performance as the network environment changes over time. Overall, the models in this paper not only achieved better accuracy but also offered a deeper insight into their performance on progressive datasets.

To begin, we evaluated the models' performance on the training dataset using optimal hyperparameters. Table III presents the accuracy and F1-scores, while the confusion matrices in Fig.3 provide a visual representation. The results clearly indicate that DT and RF demonstrated the highest accuracy, followed by SVM, ANN, and DNN. NB, however, showed a bias towards the "benign" class.

TABLE III. EVALUATION METRICS OF MODELS ON THE CIC-IDS2017 DATASET

Models	Accuracy	EvaluationMetrics			
		Precision	Recall	F1-Score	Class
DT	**0.993833**	0.9987 0.9890	0.9890 0.9987	0.9938 0.9938	Benign malicious
RF	0.9944	0.9987 0.9901	0.9901 0.9987	0.9944 0.9944	Benign malicious
SVM	0.9525	0.9874 0.9206	0.9155	0.9501	Benign maliciou

https://doi.org/10.65153/bw356376

			0.9883	0.9532	s
NB	0.7459	0.6568 0.9353	0.9663 0.4911	0.7821 0.6440	Benign malicious
ANN	0.9497	0.9857 0.9168	0.9112 0.9867	0.9470 0.9505	Benign malicious
DNN	0.9707	0.9937 0.9489	0.9469 0.9939	0.9697 0.9709	Benign malicious

Figure 3. Confusion matrices on the CIC-IDS2017

Next, we conducted a crucial evaluation using an entirely new dataset, CSE-CIC-IDS2018, which the models had not been trained on. The results, as shown in Table IV and the corresponding confusion matrices in Fig.4, revealed a significant drop in performance for all models except for SVM. This poor generalization performance suggests that the models overfitted the training data to varying degrees.

TABLE IV. EVALUATION METRICS OF MODELS ON THE CSE-CIC-IDS2018

Models	**Accuracy**	**EvaluationMetrics**			
		Precision	Recall	F1-Score	Class
DT	**0.70413**	0.6079 0.9970	0.9990 0.3485	0.7559 0.5165	Benign malicious
RF	**0.6804**	0.5896 0.9976	0.9993 0.2967	0.7417 0.4573	Benign malicious
SVM	**0.7549**	0.6979 0.8427	0.8871 0.6116	0.7812 0.7088	Benign malicious
NB	**0.3645**	0.5008 0.0279	0.9919 0.0002	0.6656 0.0005	Benign malicious
ANN	0.7125	0.6474 0.8387	0.9046 0.501	0.7547 0.627	Benign malicious
DNN	0.6985	0.6275 0.8592	0.9283 0.4427	0.7488 0.5843	Benign malicious

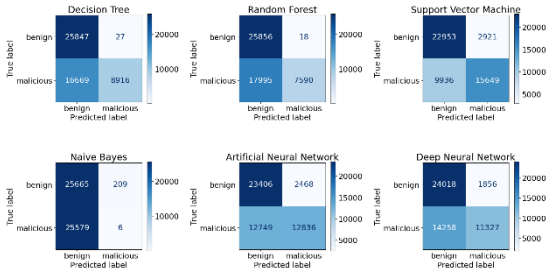

Figure 4. Confusion matrices on the CSE-CIC-IDS2018

Among them, SVM and ANN proved to be more robust. With an accuracy of 76% and 71% respectively, they were the top performers on the new dataset. The confusion matrices illustrate a common issue: a bias towards classifying samples as "benign." While this resulted in misclassifying more benign samples as malicious (indicated by a low recall score of around 0.9 for the benign class), it also enabled them to correctly identify a greater number of malicious samples, yielding a significantly higher recall score for that class compared to other models.

Accuracy results Fig.5 show SVM with the smallest drop, followed by ANN and NB, though NB declines sharply in F1-score Fig.6. DT and RF exhibit the largest decreases in both accuracy and F1-score, indicating that SVM and ANN are less affected by overfitting, while DT and RF are most impacted.

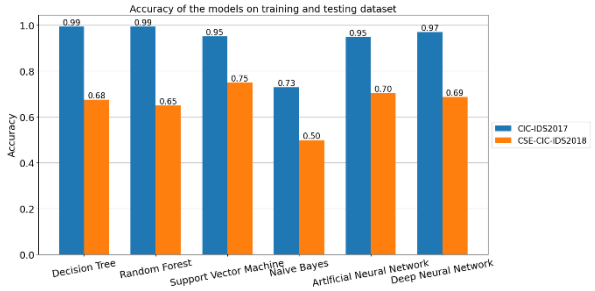

Figure 5. Accuracy of the models on the CIC's

https://doi.org/10.65153/bw356376

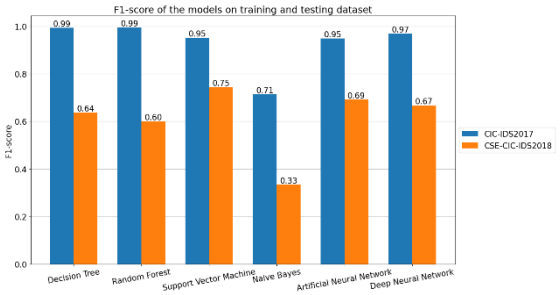

Figure 6. . F1-score of the models on CIC's

VI. CONCLUSIONS

The experimental results indicate that most ML models achieved high accuracy when both training and testing were conducted on CIC-IDS2017. However, their performance dropped significantly when evaluated on the independent CSE-CIC-IDS2018 dataset, revealing a strong tendency toward overfitting. Among all classifiers, Support Vector Machine (SVM) and Artificial Neural Network (ANN) demonstrated the best generalization ability, maintaining relatively stable performance across datasets.

REFERENCES

[1] C. I. f. Cybersecurity, "IDS 2017 | Datasets | Research | Canadian Instute for Cybersecurity | UNB," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html.

[2] U. Canadian Institute for Cybersecutiry, "IDS 2018 | Datasets | Research | Canadian Instute for Cybersecurity | UNB," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html.

[3] C. I. f. Cybersecurity, "A Realistic Cyber Defense Dataset," [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018/.

[4] R. Mills, "LUFlow Network Intrusion Detection Data Set," [Online]. Available: https://www.kaggle.com/datasets/mryanm/luflow-network-intrusion-detection-data-set/data.

[5] S. S. C. D. T. D. C. M. K. K. T. Saranya, "Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review," Elsevier B.V., Third International Conference on Computing and Network Communications (CoCoNet'19), 2020.

[6] I. C. O. Vincent Zibi Mohale, "Evaluating machine learning-based intrusion detection systems with explainable AI: enhancing transparency and interpretability," Frontiers in Computer Science, 2025.

[7] T. Nguyễn, "Decision Tree algorithm," [Online]. Available: https://machinelearningcoban.com/tabml_book/ch_model/decision_tree.html.

[8] GeeksforGeeks, "Random Forest Algorithm in Meaching Learning," 23 Jul 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/.

[9] scikit-lear, "Support Vector Machines," [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#implementation-details.

[10] A. N. Abid Ali Awan, "Naive Bayes Classification Tutorial using Scikit-learn," 3 Mar 2023. [Online]. Available: https://www.datacamp.com/tutorial/naive-bayes-scikit-learn.

[11] IBM, "What is a Nerual Network?," 6 October 2021. [Online]. Available: https://www.ibm.com/think/topics/neural-networks.

[12] S. S. K. A. I. S. G. E. P. K. M. Lydia, "A comprehensive review on wind turbine power curve modeling techniques," 2014.

[13] M. Mercier, "What is a deep neural network?," 14 Jan 2025. [Online]. Available: https://botpress.com/blog/deep-neural-network.

[14] Y. S. Sydney Mambwe Kasongo, "A Deep Learning Method With Filter Based Feature Engineering for Wireless Intrusion Detection System," IEEE, 2019.

[15] Y. L. Zhaoyang Xu, "Robust Anomaly Detection in Network Traffic: Evaluating Machine Learning Models on CICIDS2017," 2025.

[16] T. P. J. V. Kanimozhi, "Artificial Intelligence based Network Intrusion Detection with Hyper-Parameter Optimization Tuning on the Realistic Cyber Dataset CSE-CIC-IDS2018 using Cloud Computing," IEEE, 2019.

[17] I. S. Tuan-Hong Chua, " Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection Using Progressive Dataset," Symmetry , 2023.

[18] C. T. Hong, "Evaluation-of-Machine-Learning-Algorithm-in-Network-Based-Intrusion-Detection-System," [Online]. Available: https://github.com/tuanhong3498/Evaluation-of-Machine-Learning-Algorithm-in-Network-Based-Intrusion-Detection-System?tab=readme-ov-file.

[19] A. K. M. M. B. N. R. Ryan Mills, "Practical Intrusion Detection of Emerging Threats," IEEE, 2021.

[20] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," IEEE, 2019 .

https://doi.org/10.65153/bw356376
Journal of Science and Technology of East Asia University of Technology