Incremental Feature Learning for Cybersecurity under Evolving Feature Spaces

Rafael S. Lemos[1], Davyson S. Ribeiro[1], Francisco R. P. da Ponte[1], César Lincoln C. Mattos[1], Emanuel B. Rodrigues[1]

[1]Universidade Federal do Ceará (UFC)
Av. da Universidade, 2853 – CEP 60020-181 – Fortaleza – CE – Brasil

{rafael.lemos, davysonribeiro}@alu.ufc.br,
fco.rparente@gmail.com,
{cesarlincoln, emanuel}@dc.ufc.br

Abstract. *Cybersecurity systems continuously evolve by incorporating new data attributes, requiring machine learning models to adapt without costly retraining. This work proposes two Incremental Feature Learning (IFL) approaches for intrusion detection and vulnerability risk classification: a hybrid Naive Bayes model and a Multilayer Perceptron with partial weight freezing. Experimental results show that both approaches effectively adapt to progressively expanding feature spaces while maintaining competitive computational efficiency. The proposed MLP achieved over 93% accuracy for intrusion detection, whereas the incremental strategies reduced learning cost and memory consumption without full model retraining.*

1. Introduction

The rapid digitization of corporate environments and the growing integration of critical infrastructure into the global network have significantly expanded organizations' attack surface [Pinto et al. 2023]. The cybersecurity landscape is highly dynamic, characterized by continuously evolving threats and vulnerabilities. To maintain robust defense mechanisms, organizations rely on Vulnerability Management (VM) to identify, prioritize, and remediate systemic flaws [Secretaria de Governo Digital 2023], alongside Intrusion Detection Systems (IDS) as an essential layer of defense for real-time monitoring of network environments [Pinto et al. 2023]. However, as malicious actors rapidly shift their methodologies, static security solutions quickly become obsolete. Consequently, modern security mechanisms must not only detect known weaknesses but also dynamically adapt to emerging threat profiles and novel attack vectors [Dissanayake et al. 2022; Hozouri et al. 2025].

To achieve this required level of adaptability, Machine Learning (ML) has been widely adopted in both VM and IDS. Nevertheless, the operational reality of cybersecurity infrastructure introduces a significant challenge: as security tools and traffic analysis mechanisms are updated to capture new threat characteristics, the underlying data structures change. For instance, the deployment of a new vulnerability scanning engine or the introduction of advanced feature extraction tools in an IDS (such as CICFlowMeter[1] or

[1]A tool used for extracting network traffic features from PCAP files, widely applied in datasets such as CICIDS.

NetFlow[2]) generates novel attributes over time.

The core operational problem in this scenario is that legacy historical data possesses fewer or different attributes compared to newly collected data. Traditional ML models fail to handle this asymmetry natively, requiring complete retraining with the merged dataset to incorporate new features. In critical environments, this leads to increased computational costs and unacceptable delays in updating defense lines.

To address this structural evolution without the prohibitive cost of retraining, Incremental Learning (IL) emerges as a flexible paradigm [Van de Ven et al. 2022]. Within this domain, Incremental Feature Learning (IFL) specifically targets scenarios where the feature space expands dynamically. IFL enables a model to progressively assimilate new attributes while retaining the knowledge acquired from older, lower-dimensional data representations, ensuring continuous operation in expanding feature spaces [Kuang et al. 2024].

It is important to emphasize that standard algorithms must be structurally or mathematically adapted to perform IL and, more specifically, IFL. For instance, Naive Bayes (NB) can be adapted for incremental updates due to its assumption of conditional independence among features [Yang et al. 2023]. Moreover, Multilayer Perceptron (MLP) architectures require specific structural modifications—such as partial weight freezing strategies, as explored in prior literature [Sadreddin and Sadaoui 2022]—to accommodate new input nodes without suffering from catastrophic forgetting.

This study aims to evaluate the application of IFL techniques to enhance the adaptability of cybersecurity solutions, specifically targeting vulnerability risk classification and network attack detection. Due to the characteristics of the NB model, adapted update functions were applied to support the inclusion of new features across different data types, employing Gaussian NB for continuous data, Bernoulli NB for binary data, and Multinomial NB for categorical data. Furthermore, we employ an adapted MLP model utilizing a partial weight freezing mechanism to handle the expanding feature space. Both approaches are evaluated using the CVEjoin dataset [Ponte et al. 2023] and the NF-CSE-CIC-IDS2018-v2 dataset [Sarhan et al. 2022].

The main contributions of this work are highlighted as follows:

- The application and evaluation of Incremental Feature Learning (IFL) in two distinct, highly dynamic cybersecurity scenarios: Vulnerability Management and Intrusion Detection.
- The implementation of a hybrid incremental Naive Bayes architecture capable of dynamically incorporating new features across heterogeneous data types without retraining.
- The adaptation and application of an MLP model with partial weight freezing for expanding feature spaces in security datasets.
- A comparative benchmark demonstrating the efficacy of IFL approaches against standard static models when subjected to progressive attribute availability.

This work is organized as follows. Chapter 2 presents related works in the literature. Chapter 3 introduces the proposed approach, detailing the methodology adopted

[2]A Cisco-developed protocol for network traffic collection and analysis, used to generate flow-level statistics and communication metrics.

in this study. Chapter 4 describes the experimental setup, including datasets, evaluation metrics, and implementation details. Chapter 5 presents and discusses the obtained results, analyzing the performance of the proposed method. Finally, Chapter 6 concludes the work, summarizing the main contributions and suggesting directions for future research.

2. Related Work

This section reviews the literature on Machine Learning (ML) and Incremental Learning (IL) techniques applied across various domains using Naive Bayes (NB) and Multilayer Perceptron (MLP) models, highlighting that the application of Incremental Feature Learning (IFL) in information security contexts remains limited.

The study by [Yang et al. 2023] presents a classifier called 3WD-INB (Three-Way Incremental Naive Bayes), which adopts an approach based on three-way decisions (acceptance, rejection, and uncertainty region) combined with IL to update the model as new data is incorporated. The main contribution of this method lies in a decision mechanism that handles uncertainties during the classification process, improving robustness in scenarios with ambiguous data. The IL mechanism in 3WD-INB relies on a selective approach: only high-confidence samples are incorporated into the training set, dynamically updating the prior and conditional probabilities. However, the model treats all attributes uniformly, ignoring the heterogeneity of data types, and does not consider the introduction of novel attributes over time.

The work by [Kim et al. 2024] introduces Robust and Adaptive Incremental Learning (RAIL) to address the limitations of batch-based learning in continuously evolving feature spaces. RAIL utilizes an incremental NB classifier combined with a feature weighting mechanism, enabling model updates without requiring access to previous samples. A core requirement of this method is the discretization of continuous attributes to compute mutual information for feature selection. Consequently, its performance depends heavily on properly defined intervals; inadequate settings easily cause information loss or overfitting in highly variable scenarios. To avoid these issues, the approach proposed in our work bypasses this transformation entirely by employing specific probabilistic formulations tailored to each data type, thereby reducing preprocessing sensitivity while natively supporting new attributes.

The research in [Kuang et al. 2024] proposes an IFL approach using the Knowledge Distillation (KD) technique to improve learning efficiency when the feature space changes over time. By transferring knowledge incrementally from a “teacher” model to a “student” model, the approach allows the system to continuously learn new features without reprocessing historical data. This method proves useful in domains where data characteristics constantly evolve, such as Internet of Things (IoT) environments or healthcare records. Through various experiments, the authors demonstrate that incremental attribute distillation preserves previously learned knowledge while improving accuracy upon the introduction of new attributes.

Similarly focusing on healthcare data streams, the study in [Appasani et al. 2024] applies an incremental NB algorithm to address conceptual drift. The approach uses a sliding window to recalculate class and feature probabilities with each new chunk of data. When accuracy falls below a predefined threshold, a drift detection mechanism triggers a complete reset and retraining of the model using the most recent data. While demonstrating

computational efficiency in dynamic scenarios—such as survival prediction and fetal health classification—the method reveals critical limitations on small or imbalanced datasets. Furthermore, the reliance on full retraining upon drift detection deviates from pure incremental learning principles.

The work by [Sadreddin and Sadaoui 2022] presents an IFL framework for credit-card fraud detection using a neural network that expands its architecture horizontally as new data arrives. The model preserves prior knowledge by freezing previous neurons and layers, training only the newly added units for each data block. Although it shares the principle of weight freezing with this work to mitigate catastrophic forgetting, their framework differs by increasing the network complexity (number of hidden neurons) at each phase. In contrast, the approach proposed here strictly maintains a fixed internal structure.

Unlike the aforementioned works, which either focus exclusively on specific data types or linearly expand network complexity, this work investigates the feasibility of IFL in dynamic security environments (IDS and VM) through two distinct, optimized models. First, the proposed hybrid NB architecture inherently accommodates heterogeneous attributes (binary, categorical, and continuous) without relying on data transformation or binning. Second, the proposed MLP model introduces a selective input-layer expansion with partial weight freezing; this supports feature space expansion while keeping the hidden layers fixed, ensuring computational efficiency and constant inference times—critical requirements for real-time cybersecurity operations. Table 1 summarizes the related works and highlights the core contributions of this study.

Table 1: Security Models and Scenarios

Authors	Models	IFL	Security Scenario
[Yang et al. 2023]	3WD-INB	-	-
[Kim et al. 2024]	NB / Incremental NB / GLSC and ORF3V / RAIL	X	-
[Kuang et al. 2024]	MLP / IAL-KD	X	-
[Appasani et al. 2024]	Incremental NB / Adaptive Random Forest / Tree Hoeffding	-	-
[Sadreddin and Sadaoui 2022]	MLP	X	Credit-Card Fraud Data
This Work	Hybrid NB with IFL / MLP adapted for IFL	X	IDS and VM

3. Proposed Approach

The IFL model presented in this work aims to develop a machine learning model capable of learning without the need for retraining. Thus, a model is required that adapts to the data and maintains good classification performance even when newly received information contains a different set of attributes. The overall workflow of the proposed approach is illustrated in Figure 1 and Table 2.

As illustrated in the Figure 1, the initial dataset is divided into phases. The feature evolution process is represented in the feature evolution block, where an example scenario with three phases is shown: the feature set evolves progressively, starting with one-third of

the attributes, then two-thirds, and finally reaching the complete set of features. Each phase undergoes a preprocessing step to generate the corresponding input for the models, which are trained and evaluated at every stage. This procedure enables a comparative analysis of performance across all phases.

To address this scenario, two complementary incremental learning models are proposed and evaluated under the same experimental setting: a probabilistic approach based on NB and a neural-based approach using MLP. To evaluate and test the model, datasets were configured with scenarios involving attribute variation across phases, with samples randomly divided to enable incremental training. In this scenario, the attributes are partitioned in descending order of importance based on the Fisher Score. This configuration provides an evaluation setting for catastrophic forgetting, as the models must preserve the knowledge associated with the most relevant features introduced in the early phases while progressively incorporating attributes throughout the learning process.

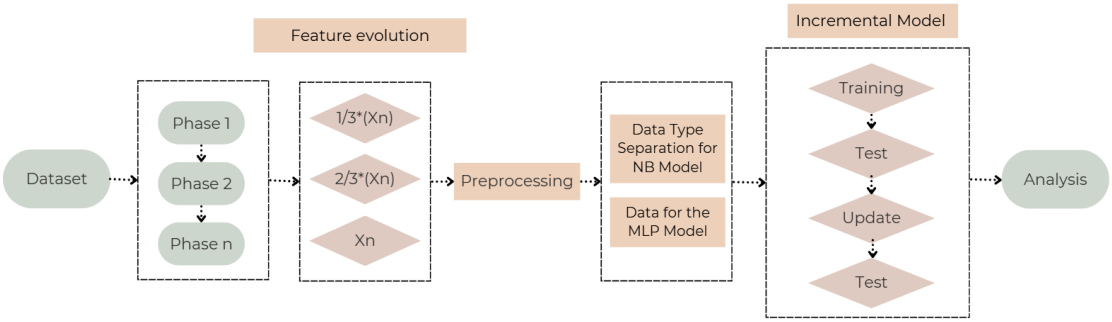

Figure 1: Data preprocessing workflow and model training.

3.1. Incremental Feature Learning

Table 2: IFL process algorithmic flowchart.

Algorithm Step	Pseudocode
1. Data segmentation	phases = [F1, F2, F3, ..., FN]
2. Feature scaling	sub_X = [0.1X_n, 0.2X_n, ..., X_n]
3. Preprocessing	X_proc = pre_process(dataset)
4. Initial Training (t=1)	model.fit(X_train[0], y_train)
5. Initial Validation	acc_1 = model.score(X_test[0], y_test)
6. Incremental Cycle	FOR t from 2 TO N_Phases:
7. Knowledge Injection	model.update(X_train[t], y_train)
8. Update Validation	acc$_t$ = $model.score(X\_test[t], y\_test)$
9. Results Analysis	plot_evolution(acc_history)

The first step in IFL is to train the model so that it can classify data based on the features initially presented to it. When new data is received, the model will be updated using new data, excluding previously observed samples while incorporating new attributes. At this stage, it is important that the model be able to classify the same classes using the new information without compromising the model's overall accuracy.

After each training phase, it is important to validate the model; for this, a test dataset is used. The test dataset is augmented with attributes in the same way as the

training dataset. Thus, the model is tested with the same attributes as in the training and test phases. To illustrate the test set: the initial number of attributes would be X_n; in the second phase, the number of attributes is X_{n+1}. This ensures that the training and test sets maintain the same number of attributes.

3.2. Naive Bayes

NB is based on Bayes's theorem, learns parameters by observing each feature individually, and collects simple statistics per class for each feature [Müller and Guido 2016]. Furthermore, NB performs well on small-scale datasets, is robust to missing values, and provides efficient training and inference even for large data volumes [Yang et al. 2023].

In conventional NB models, these sufficient statistics are estimated assuming a fixed feature space. However, under the proposed Incremental Feature Learning (IFL) setting, new attributes become available over time, requiring the feature space and the associated statistics to be progressively expanded while preserving the information learned in previous phases.

Let $\mathcal{F}_t$ denote the feature set available at incremental phase t, with $d_t = |\mathcal{F}_t|$. If q_t new features are introduced, then

$$d_t = d_{t-1} + q_t. \tag{1}$$

For every Naive Bayes model, the vector of sufficient statistics $\mathbf{s}_c$ is expanded as

$$\mathbf{s}_c^{(t,0)} = \begin{bmatrix} \mathbf{s}_c^{(t-1)} \\ \mathbf{0}_{q_t} \end{bmatrix}, \tag{2}$$

where $(t, 0)$ denotes the model state immediately after feature-space expansion and before processing the samples of phase t. The global count $N_c^{(t)}$ is used to estimate the class prior, whereas $N_{c,j}^{(t)}$ denotes the number of samples in which feature j was effectively available and is used whenever the conditional distribution of feature j requires feature-specific normalization.

The following subsections describe how this general feature-expansion mechanism is instantiated for the Bernoulli, Multinomial, and Gaussian Naive Bayes models.

3.2.1. Naive Bayes Bernoulli

For the Bernoulli NB model, $\mathbf{s}_c = \mathbf{f}_c$, where $\mathbf{f}_c$ stores the accumulated counts of active binary features. Following the feature-expansion mechanism defined in Equation (2), the feature counts are updated as

$$f_{c,j}^{(t)} = f_{c,j}^{(t-1)} + \sum_{\mathbf{x}_i \in \mathcal{X}_c^{(t)}} m_{i,j}^{(t)} x_{i,j}^{(t)}, \tag{3}$$

where $m_{i,j}^{(t)}$ indicates whether feature j is available in sample i.

The class prior is estimated from the global count

$$P^{(t)}(c) = \frac{N_c^{(t)}}{\sum_{c'} N_{c'}^{(t)}}, \tag{4}$$

whereas the conditional probability of an active feature is normalized by its feature-specific observation count,

$$P^{(t)}(X_j = 1|c) = \frac{f_{c,j}^{(t)} + \alpha}{N_{c,j}^{(t)} + 2\alpha}. \tag{5}$$

3.2.2. Multinomial Naive Bayes

The Multinomial NB model follows the same feature-expansion mechanism as the Bernoulli model, but considers $x_{i,j}^{(t)} \in \mathbb{N}_0$ as feature frequencies rather than binary activations. Consequently, the accumulated frequencies are updated according to Equation (3).

The class prior is estimated from $N_c^{(t)}$, whereas the conditional probability of feature j is computed as

$$P^{(t)}(X_j|c) = \frac{f_{c,j}^{(t)} + \alpha}{\sum_{k \in \mathcal{F}_t} f_{c,k}^{(t)} + \alpha d_t}. \tag{6}$$

Only features available in $\mathcal{F}_t$ contribute to the normalization term, ensuring that frequencies of newly introduced features are estimated exclusively from observations collected after their introduction.

3.2.3. Gaussian Naive Bayes

For the Gaussian NB model, $\mathbf{s}_c = \{\boldsymbol{\mu}_c, \mathbf{M}_{2,c}\}$, where $\mathbf{M}_{2,c}$ denotes the accumulated second central moments. Following the feature-expansion mechanism defined in Equation (2), Welford's algorithm updates the statistics of each available feature as

$$N'_{c,j} = N_{c,j} + 1, \tag{7}$$

$$\mu'_{c,j} = \mu_{c,j} + \frac{x_{i,j} - \mu_{c,j}}{N'_{c,j}}, \tag{8}$$

$$M'_{2,c,j} = M_{2,c,j} + (x_{i,j} - \mu_{c,j})(x_{i,j} - \mu'_{c,j}), \tag{9}$$

yielding

$$\sigma^2_{c,j} = \frac{M_{2,c,j}}{\max(N_{c,j} - 1, 1)} + \varepsilon. \tag{10}$$

The global count $N_c^{(t)}$ is used only to estimate the class prior, whereas $N_{c,j}^{(t)}$ is employed to estimate the conditional Gaussian distribution of feature j.

3.3. Multilayer Perceptron

The proposed MLP model is constructed with two hidden layers. At each new phase, a new model will be generated, which will inherit the weights from the input layer of the previous model, in addition to receiving the new attributes that emerge at each phase. This approach was adopted to avoid generating a model that is too complex. Thus, at each new phase, the model will be generated using part of the knowledge acquired from the previous phase, thereby preventing the loss of what was already learned by the previous model. The model configuration can be observed in Table 3.

Unlike full-replay approaches, the training uses only the data available in the current phase, while the test set remains fixed throughout all phases, ensuring consistency in evaluation.

Table 3: Settings for the incremental MLP model.

Parameter	Value
Architecture	MLP
Hidden Layers	(128, 64)
Activation Function	Rectified Linear Unit (ReLU)
Optimizer	Adam
Learning Rate	0.001
Epochs	50
Loss Function	Cross-Entropy
Training	Incremental (without replay)

4. Experiments

To conduct the experiments, the `sklearn`, `numpy`, and `pandas` libraries were used for data processing, transforming categorical variables, and creating the models used as baselines and for building the proposed models. Additionally, `PyTorch` was used to create the proposed incremental MLP model. To enable comparison with another incremental learning approach, the RAIL model proposed in [Kim et al. 2024] was implemented.

4.1. Datasets

This section describes the datasets used in the experiments, which are applied to train information security models in both vulnerability management and intrusion detection scenarios.

The CVEJoin dataset is an aggregation of multiple data sources centered around Common Vulnerabilities and Exposures (CVE), a standardized identifier system for publicly known cybersecurity vulnerabilities. CVEs play a fundamental role in cybersecurity by enabling consistent tracking, sharing, and management of security flaws across different tools and organizations[3]. The National Vulnerability Database (NVD) builds upon CVE records by providing standardized and enriched vulnerability information, including severity scores, exploitability metrics, and impact analysis, which support vulnerability

[3]National Institute of Standards and Technology (NIST), CVEs and the NVD Process. Available at: https://nvd.nist.gov/general/cve-process.

assessment and risk prioritization. Together, CVE and NVD form the foundation of modern vulnerability management and threat intelligence ecosystems.

The dataset contains more than 200,000 vulnerability records and can be used for various research purposes, such as analyzing the risk of exploitation and predicting vulnerability severity [Ponte et al. 2023]. In its processed form, CVEJoin includes 208 labeled samples, 29 attributes, and four classes corresponding to vulnerability risk levels: low, moderate, high, and critical. Additionally, it provides essential data for vulnerability management, including CVSS (Common Vulnerability Scoring System) scores, threat intelligence, and contextual security-related attributes.

In this study, we also used the NF-CSE-CIC-IDS2018-v2 dataset [Sarhan et al. 2022]. This dataset is a derivative of CSE-CIC-IDS2018 developed by the Canadian Institute for Cybersecurity (CIC) [Sharafaldin et al. 2018], that provides NetFlow-based representations, with a reduced and more structured set of attributes. This version was designed to facilitate the application of machine learning techniques in scenarios closer to operational environments, reducing redundancies and eliminating inconsistent or highly correlated attributes present in the original version. It contains 43 attributes and approximately 18.8 million samples. The CSE-CIC-IDS2018 dataset was developed to provide a realistic and labeled network traffic benchmark, aimed at supporting research on machine learning-based intrusion detection systems. The dataset includes multiple attack classes as well as benign traffic, enabling comprehensive evaluation of intrusion detection models under different types of malicious behavior and normal network activity.

A relevant aspect of this study is the difference in feature representation between the datasets. While CVEJoin provides structured attributes describing vulnerabilities from multiple information sources, NF-CSE-CIC-IDS2018-v2 represents network traffic through a compact NetFlow-based feature space. These complementary characteristics enable the evaluation of the proposed Incremental Feature Learning approach in distinct cybersecurity scenarios with progressively expanding attribute spaces, reflecting realistic environments in which new information sources and monitoring capabilities become available over time.

4.2. Application of the IFL models

The overall experimental configuration, including dataset splits, number of phases, and number of executions, is summarized in Table 4. For the experiment, the NF-CSE-CIC-IDS2018-v2 dataset was divided into 10 phases, ensuring that samples did not appear in more than one subset. The Fisher Score, a feature selection method for supervised models [Aggarwal 2020], was used to assist in the phase division, so that the attributes were distributed in descending order of relevance, from the most significant to the least significant. For the CVEJoin dataset, the division was performed into three phases due to the dataset size, as previously described. The phase construction was also guided by the Fisher Score.

The metrics calculated for the experiments were Accuracy, Precision, Recall, and F1-score for model validation. The experiment conducted on the two datasets was run 10 times to collect the metrics, and the average was calculated to enable comparisons. The metrics were calculated at the end of each phase to generate graphs for comparing learning performance across phases, as well as to verify the model's forgetting behavior.

Before the division into phases, each dataset was split into training and test sets.

For dataset with a larger number of samples, an 80/20 training-test split was adopted. For the dataset with a smaller amount of data, a 90/10 split was used in order to provide a larger number of samples for training without compromising the model evaluation capability. This strategy aims to balance the model's learning requirements and the statistical representativeness of the test set. Thus, after each model training phase, the model is tested to verify its classification performance. The same test set is used across all phases, ensuring the verification of the evolution of the model metrics, although with a different number of features depending on the characteristics of the training data for each respective phase.

Table 4: Datasets and Experimental Configuration

Dataset	Data Split	Phases	Executions	Samples Used
CVEJoin	90/10	3	10	208
NF-CSE-CIC-IDS2018-v2	80/20	10	10	1.09M

In the *NF-CSE-CIC-IDS2018-v2* dataset, a step was taken to remove irrelevant or redundant features. This decision aims to reduce noise, avoid redundancy, and potentially improve the performance of machine learning models. The dataset labels were binarized as follows: (i) instances of the majority class (benign) were assigned the value 0, and (ii) instances of all remaining classes (attacks) were assigned the value 1.

To mitigate the imbalance, an undersampling strategy was applied to the majority class. Specifically, random undersampling was performed on the benign class so that its size would be equal to the total number of instances in the other classes. This procedure ensures a balanced dataset, reducing the model's bias in favor of the predominant class.

5. Experimental results

After conducting the experiments and obtaining the prediction results from each model across the datasets, the different scenarios are analyzed and evaluated by comparing the performance metrics of RAIL from the related work, the baseline models (NB and MLP), and the proposed NB and MLP approaches.

The baseline was constructed using standard configurations of the NB and MLP models. At each phase, the models were trained from scratch using only the samples and attributes corresponding to the respective phase. Thus, at each stage, an independent model was built and evaluated exclusively on the data from that phase. In addition, a reference model (threshold) was defined, trained using the full set of attributes and evaluated in a standard setting. In this case, the train-test split was varied using different random seeds, and the experiment was repeated 10 times, with the final results reported as the average, representing an idealized scenario for the proposed models.

For the CVEJoin dataset, Table 5 and Figure 2 present the results obtained after applying the models. As observed from the variations, the models exhibited greater difficulty in learning the dataset for vulnerability risk classification. This dataset is smaller and presents higher complexity in terms of data types, including categorical, binary, and continuous features, as well as a reduced number of labeled samples. These factors are critical determinants of model performance. Furthermore, based on the metrics, it is

possible to observe the difficulty of the models even in terms of the decision threshold, where the MLP model achieved 73.81% accuracy and the NB model achieved 67.62%.

Table 5: Final metrics on dataset CVEJoin.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
RAIL	54.76 ± 8.32	50.34 ± 12.6	54.76 ± 8.32	50.07 ± 9.21
NB	54.76 ± 7.45	49.49 ± 13.43	54.76 ± 7.45	49.07 ± 11.2
Proposed NB	49.52 ± 15.09	47.80 ± 13.81	44.62 ± 9.77	39.98 ± 10.57
MLP	60.00 ± 6.46	56.07 ± 10.8	60.00 ± 6.46	54.83 ± 7.78
Proposed MLP	**70.16 ± 5.85**	**70.09 ± 6.38**	**70.16 ± 5.85**	**67.98 ± 6.64**

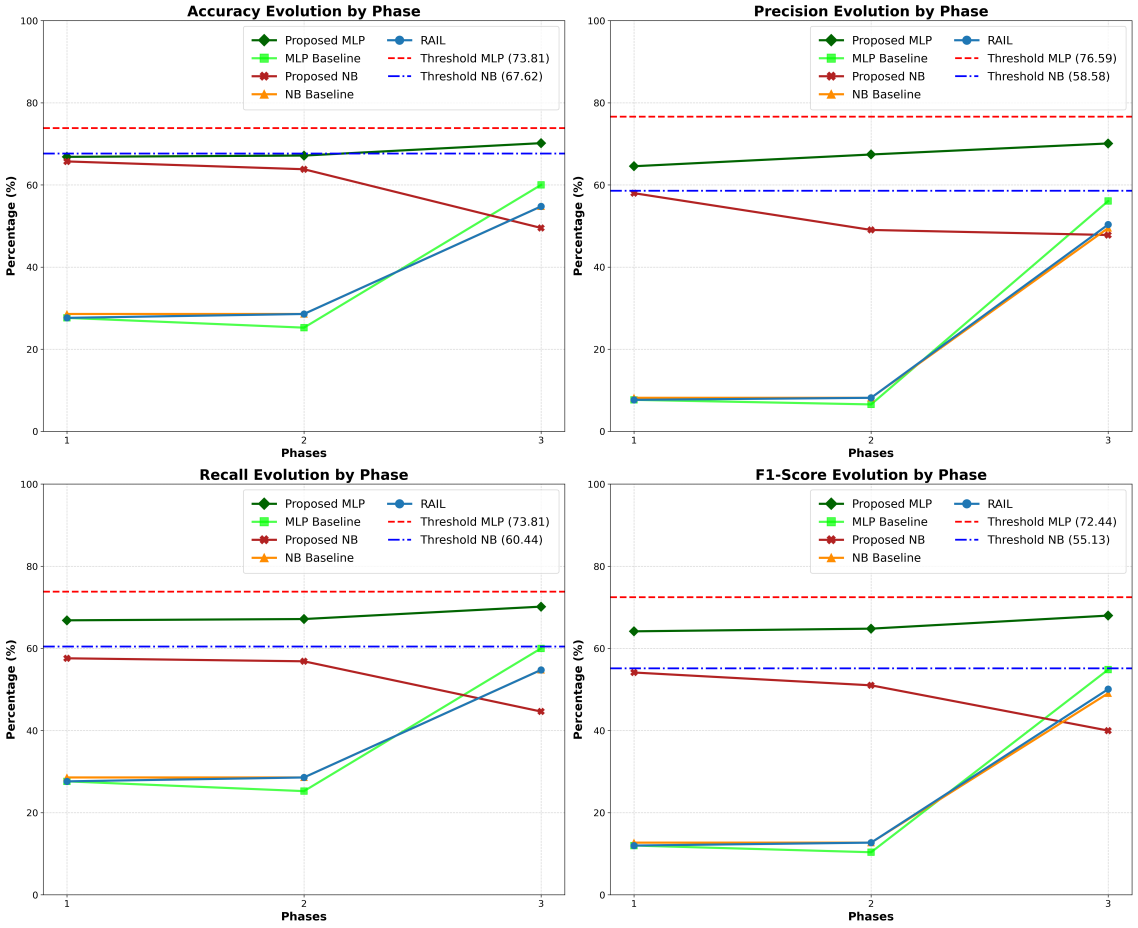

Figure 2: Comparison of metrics between models and baseline for the CVEJoin dataset.

The proposed NB model was unable to maintain its performance metrics across the phases, showing a decline in performance in phases 2 and 3. In contrast, the proposed MLP model was able to improve its performance, outperforming the baselines and achieving metrics above 70% in the final phase.

Table 6 presents the computational cost of the evaluated models on the CVEJoin dataset. Due to the relatively small size of the dataset, the computational costs were low

for all evaluated models. The Proposed NB achieved the lowest learning time (0.0407 ± 0.0049 s), reducing the training cost by approximately 60% compared with the NB baseline (0.1019 ± 0.0146 s). For the MLP, the baseline and incremental approaches exhibited nearly identical learning times, indicating that the proposed strategy introduced no significant training overhead.

Table 6: Computational cost of the evaluated models on the CVEJoin dataset.

Model	Total Learning Time (s)	Mean Inference Time (s)	Peak Memory (MB)
RAIL	1.8036 ± 0.0940	0.08257 ± 0.00606	23.2069 ± 0.0492
NB	0.1019 ± 0.0146	0.00299 ± 0.00027	22.7797 ± 0.0480
Proposed NB	0.0407 ± 0.0049	0.04419 ± 0.00455	22.7841 ± 0.0506
MLP	0.3421 ± 0.0244	0.000305 ± 0.000034	22.8108 ± 0.0480
Proposed MLP	0.3584 ± 0.0214	0.000306 ± 0.000026	22.8225 ± 0.0480

Both MLP models also achieved virtually identical inference times (approximately 0.3 ms), while the Proposed NB required a longer inference time than the baseline. Peak memory consumption remained nearly constant for all evaluated models, ranging from 22.78 MB to 23.21 MB. Overall, these results indicate that, for smaller datasets, the computational cost is not a limiting factor, making predictive performance the primary criterion for comparing the evaluated models.

For the NF-CSE-CIC-IDS2018-v2 dataset, the results obtained are shown in Table 7 and Figure 3. Based on the results obtained, it was observed that the proposed NB and MLP models achieved good metrics. Focusing on the results of the proposed NB model, it can be observed that the model occasionally exceeds the threshold of the NB; this occurs in scenarios where the dataset initially operates with the best attributes, meaning the model begins training with the most informative attributes for class classification. This does not happen with the proposed MLP model; it proved to be more consistent and maintained a classification level close to the MLP threshold.

In addition, the proposed NB model outperformed the NB baseline, demonstrating the ability to adapt to the addition of new features in the dataset and achieving the proposed objective, with the final average of the metrics in the last phase exceeding 88%, outperforming the other NB models. However, the model was also more sensitive to the addition of new features compared to both the baseline MLP and the proposed MLP, which achieved an accuracy of 93.04%.

Table 7: Final metrics on dataset NF-CSE-CIC-IDS2018-v2.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
RAIL	86.22 ± 0.07	86.23 ± 0.07	86.22 ± 0.07	86.22 ± 0.07
NB	86.66 ± 0.11	86.67 ± 0.11	86.66 ± 0.11	86.66 ± 0.11
Proposed NB	88.73 ± 0.11	88.87 ± 0.12	88.73 ± 0.11	88.72 ± 0.11
MLP	93.03 ± 0.07	93.82 ± 0.07	93.03 ± 0.07	93.00 ± 0.07
Proposed MLP	$\mathbf{93.04 \pm 0.01}$	$\mathbf{93.86 \pm 0.01}$	$\mathbf{93.04 \pm 0.01}$	$\mathbf{93.01 \pm 0.01}$

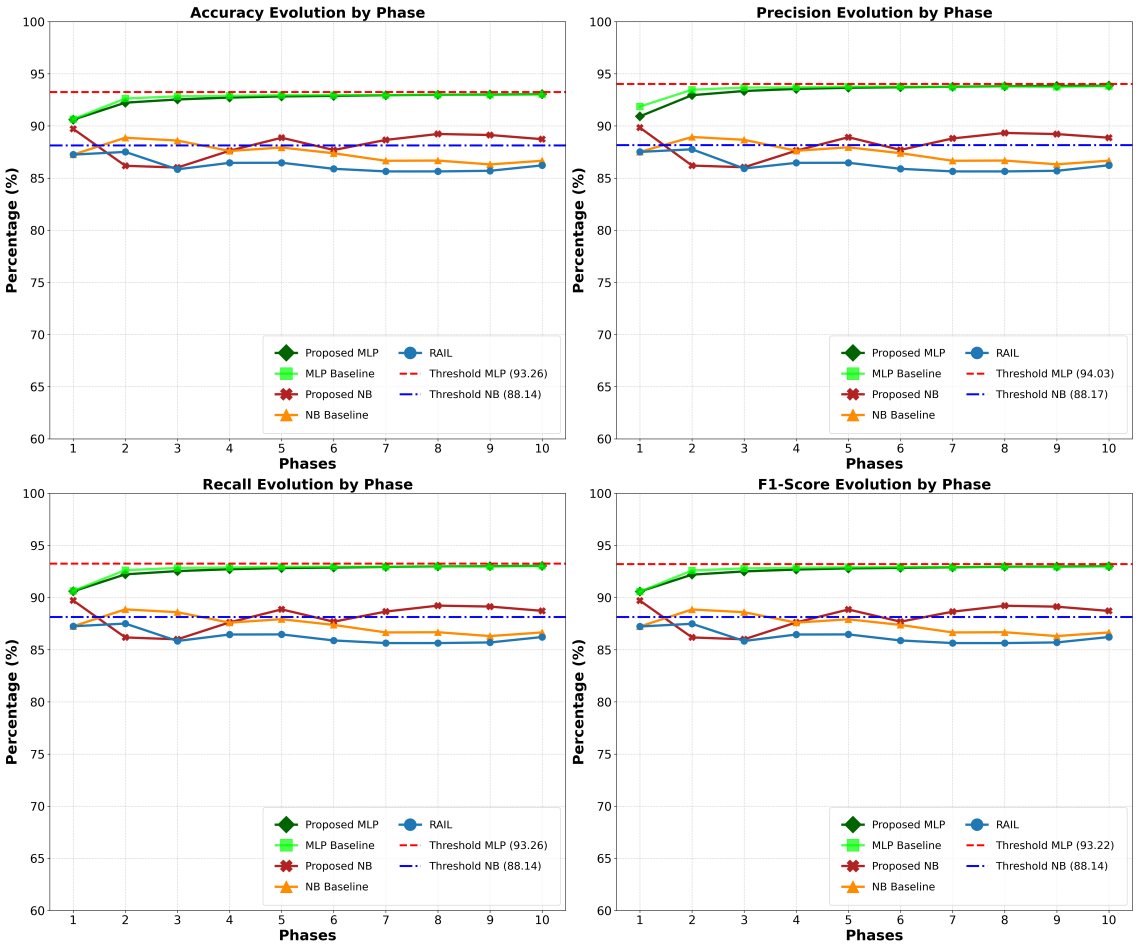

Figure 3: Comparison of Metrics Between Models and baseline for the NF-CSE-CIC-IDS2018-v2 Dataset.

As shown in Figure 3, the MLP baseline also achieved competitive performance throughout the incremental phases, reflecting its ability to adapt to the progressively expanding feature space. Although both MLP variants obtained similar average metrics, the proposed model consistently achieved the best performance while preserving previously acquired knowledge as new attributes were introduced. Furthermore, the proposed MLP exhibited greater stability across the 10 executions, with a lower standard deviation (± 0.01) than the baseline MLP (± 0.07).

Table 8: Computational cost of the evaluated models on the NF-CSE-CIC-IDS2018-v2 dataset.

Model	Learning Time (s)	Inference Time (s)	Peak Memory (MB)
RAIL	69.884 ± 3.515	0.8211 ± 0.0086	1568.29 ± 37.51
NB	19.053 ± 0.144	0.1271 ± 0.0020	307.88 ± 0.88
Proposed NB	6.922 ± 0.130	1.4046 ± 0.0115	213.46 ± 0.00
MLP	72.663 ± 1.339	0.0703 ± 0.0010	221.25 ± 17.39
Proposed MLP	72.632 ± 1.277	0.0701 ± 0.0017	184.44 ± 0.00

Table 8 presents the computational cost of the evaluated models on the NF-CSE-CIC-IDS2018-v2 dataset. Due to its larger size, the differences among the evaluated strategies are more pronounced. The Proposed NB achieved the lowest learning time (6.922 ± 0.130 s), reducing the training cost by approximately 64% compared with the NB baseline (19.053 ± 0.144 s). In contrast, the MLP baseline and Proposed MLP exhibited nearly identical learning times, indicating that the incremental strategy did not introduce additional training overhead.

Regarding inference, both MLP models achieved similar performance (approximately 70 ms), whereas the Proposed NB required a longer inference time than the baseline. In terms of memory consumption, the incremental models consistently reduced peak memory usage, with reductions of approximately 31% for Proposed NB and 17% for Proposed MLP relative to their respective baselines. Overall, these results indicate that the advantages of incremental learning become more evident on larger datasets, reducing learning time and memory consumption while maintaining the computational efficiency of the MLP.

6. Conclusion

This work investigated the application of IFL to cybersecurity scenarios in which the available feature space evolves over time. To address this problem, two incremental learning strategies were proposed: a NB model capable of expanding its statistical representation as new attributes become available and an MLP extended with a partial weight-freezing mechanism to accommodate progressively increasing input spaces without requiring complete model retraining.

The proposed approaches were evaluated on two distinct cybersecurity datasets representing vulnerability management (CVEJoin) and intrusion detection (NF-CSE-CIC-IDS2018-v2). The experimental results demonstrated that both models successfully adapted to the progressive introduction of new features while preserving previously acquired knowledge. The proposed MLP consistently achieved the best predictive performance, whereas the incremental Naive Bayes outperformed its conventional counterpart in most experimental settings, demonstrating the effectiveness of the proposed feature-expansion mechanism.

From a computational perspective, the proposed incremental strategies maintained competitive efficiency across both datasets. On the larger NF-CSE-CIC-IDS2018-v2 dataset, the incremental NB substantially reduced the total training time relative to the conventional baseline, while the proposed MLP preserved a training cost comparable to the baseline and reduced peak memory consumption. These results indicate that IFL enables continuous model adaptation while avoiding complete retraining and preserving computational efficiency.

Overall, the proposed methodology demonstrates that Incremental Feature Learning is a practical and effective strategy for machine learning models operating in dynamic cybersecurity environments with evolving feature spaces. Future work includes extending the proposed feature-expansion mechanism to additional learning algorithms, such as ensemble-based methods and gradient boosting models, as well as investigating hybrid architectures capable of improving robustness and generalization under continuously evolving feature spaces.

References

Aggarwal, C. (2020). *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery. CRC Press. ISBN: 9780367659141. URL: https://books.google.com.br/books?id=06q3zQEACAAJ.

Appasani, D., C. S. Bokkisam, and S. Surendran (2024). "An Incremental Naive Bayes Learner for Real-time Health Prediction". In: *Procedia Computer Science* 235, pp. 2942–2954.

Dissanayake, N. et al. (2022). "Software security patch management-A systematic literature review of challenges, approaches, tools and practices". In: *Information and Software Technology* 144, p. 106771.

Hozouri, A., A. Mirzaei, and M. Effatparvar (2025). "A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges". In: *Discover Artificial Intelligence* 5.1, p. 314.

Kim, C. H. et al. (2024). "Robust and Adaptive Incremental Learning for Varying Feature Space". In: *IEEE Access* 12, pp. 64177–64192.

Kuang, Z. et al. (2024). "Incremental attribute learning by knowledge distillation method". In: *Journal of Computational Design and Engineering* 11.5, pp. 259–283.

Müller, A. C. and S. Guido (2016). *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.".

Pinto, A. et al. (2023). "Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure". In: *Sensors* 23.5, p. 2415.

Ponte, F. R. P. da, E. B. Rodrigues, and C. L. C. Mattos (2023). "CVEjoin: An Information Security Vulnerability and Threat Intelligence Dataset". In: *International Conference on Advanced Information Networking and Applications*. Springer, pp. 380–392.

Sadreddin, A. and S. Sadaoui (2022). "Incremental Feature Learning for Fraud Data Stream". In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART)*, pp. 268–275.

Sarhan, M., S. Layeghy, and M. Portmann (2022). "Towards a standard feature set for network intrusion detection system datasets". In: *Mobile networks and applications* 27.1, pp. 357–370.

Secretaria de Governo Digital (2023). *Guia de Gerenciamento de Vulnerabilidades*. Versão 2.0. URL: https://www.gov.br/governodigital/pt-br/privacidade-e-seguranca/ppsi/guia_gerenciamento_vulnerabilidades.pdf.

Sharafaldin, I., A. H. Lashkari, and A. A. Ghorbani (2018). "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 108–116.

Van de Ven, G. M., T. Tuytelaars, and A. S. Tolias (2022). "Three types of incremental learning". In: *Nature Machine Intelligence* 4.12, pp. 1185–1197.

Yang, Z. et al. (2023). "A new three-way incremental naive bayes classifier". In: *Electronics* 12.7, p. 1730.