Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 11, No. 03, September 2025.

IMPROVING IOT SECURITY: THE IMPACT OF DIMENSIONALITY AND SIZE REDUCTION ON INTRUSION-DETECTION PERFORMANCE

Remah Younisse[1], Amal Saif[1], Nailah Al-Madi[1], Sufyan Almajali[1] and Basel Mahafzah[2]

(Received: 16-Mar.-2025, Revised: 23-May-2025 and 24-Jun.-2025, Accepted: 27-Jun.-2025)

ABSTRACT

Intrusion detection in the Internet of Things (IoT) environments is essential to guarantee computer-network security. Machine-learning (ML) models are widely used to improve efficient detection systems. Meanwhile, with the increasing complexity and size of intrusion-detection data, analyzing vast datasets using ML models is becoming more challenging and demanding in terms of computational resources. Datasets related to IoT environments usually come in very large sizes. This study investigates the impact of dataset-reduction techniques on machine learning-based Intrusion Detection Systems (IDSs) regarding performance and efficiency. We propose a two-stage framework incorporating deep autoencoder-based feature reduction with stratified sampling to reduce the dimensionality and size of six publicly available IDS datasets, including BoT-IoT, CSE-CIC-IDS2018, and others. Multiple machine-learning models, such as Random Forest, XGBoost, K-Nearest Neighbors, SVM and AdaBoost, were evaluated using default parameters. Our results show that dataset reduction can decrease training time by up to 99% with minimal loss in F1-score, typically less than 1%. It is recognized that excessive size reduction can compromise detection accuracy for minority attack classes. However, employing a stratified sampling method can effectively maintain class distributions. The study highlights significant feature redundancy, particularly high correlation among features, across multiple IoT security-related datasets, motivating the use of dimensionality-reduction techniques. These findings support the feasibility of efficient, scalable IDS implementations for real-world environments, especially in resource-constrained or real-time settings. This work shows considerable redundancy in the datasets which questions the huge amount of these datasets, because, in many cases, the reduced datasets provide almost the same F1-score readings after data reduction. Rasing the alarm to notice the unnecessary massive amount of data used to build robust IDSs.

KEYWORDS

Dimensionality reduction, Data reduction, Autoencoders, Stratified sampling, Machine learning.

1. INTRODUCTION

Massive amounts of data are being generated due to digitization in different Internet of Things (IoT) domains, such as healthcare, vehicular networks [1]-[2], and Intrusion Detection Systems (IDSs) [3]. Two options are available for data reduction; reducing the number of features (feature reduction) or the number of tuples in the dataset (size reduction). Deep-learning (DL) techniques can deal with vast amounts of data. Still, DL only concerns some features in the data; thus, dimensionality reduction becomes an important step in best utilizing the resources [4]-[5].

Wearable devices, such as wearable healthcare devices, for example, generate a lot of features; it takes work to manage and store the generated data. It is hard to decide which features must be preserved for accurate diagnosis and which are not [6]. Due to the cost and computational resources needed to handle the enormous number of features, it becomes a challenge to reduce them without affecting the models' performance [7]. However, intrusion-detection datasets face unique issues. The extreme data imbalance is a major concern, where minority classes represent attack classes [8]. Hence, any reduction technique should consider the risk of eliminating them. Meanwhile, rapid learning and detection models are needed to enhance the detection process, because the sooner threats are detected, the less harmful the attacks are. Additionally, adversarial behaviors may intentionally mimic normal traffic, complicating feature learning. These challenges motivate the need for intelligent, attack-aware dataset-reduction strategies. Hence, the proposed approach in this study uses stratified sampling to maintain class balance and deep

1. R. Younisse, A. Saif, N. Al-Madi, S. Almajali are with Department of Computer Science, Princess Sumaya University for Technology, Amman, Jordan. Emails: r.baniyounisse@psut.edu.jo, ama20219010@std.psut.edu.jo, n.madi@psut.edu.jo and s.almajali@psut.edu.jo
2. Basel Mahafzah is with Computer Science Department, King Abdullah II School of Information Technology, The University of Jordan, Amman 11942, Jordan.

autoencoder-based feature extraction to preserve non-linear patterns and subtle feature dependencies critical for effective IDS performance.

Different dimensionality-reduction techniques could be used based on the data complexity, such as Principal Component Analysis (PCA), MDS, and Time-lagged independent component analysis (TICA) for linear manifolds, and Sketchmap, t-SNE, and deep methods for non-linear manifolds [9]. Principal Component Analysis (PCA) has been widely used in dimensionality reduction. It helps provide better data quality, improve classification, reduce the needed space and time, and remove irrelevant data [10].

At the same time, data reduction techniques are becoming popular and widely used for data visualization, simulation and analysis [11]. Stratified sampling is a famous method that divides data into similar groups known as strata [12]. Then, it selects a certain number of samples from each group, considering the data's distribution rate; any sample taken from the data should keep the same distribution in the original dataset. Stratified sampling was proven to be an efficient, unbiased sampling method and highly representative of the data being studied. The main drawback of stratified sampling is that it can only be applied when the data cannot be grouped in disjoint groups [13].

In recent years, many intrusion-detection datasets have been generated due to the rapid updates of the malware authors, and different attacks have been developed to maneuver different IDSs. It has been noticed that these datasets tend to be large, with millions of tuples and hundreds of features. Hence, different reduction techniques have to be studied and improved. The main motivation for this paper is to explore the value of using huge datasets to train machine learning (ML)-based IDSs and to assess the effect of reducing the size of the datasets used on these IDSs. Thus, this assessment work investigates the efficiency of different data reduction and feature-extraction techniques. Reducing the datasets' sizes will help improve the required ML-based IDS training time.

In the context of intrusion detection and dimensionality reduction, many works have focused on feature reduction techniques to speed up the ML models and enhance the outcomes of these models [14]-[15]. In comparison, the size-reduction aspect is not sufficient for the research work. One reason for this is the risk associated with removing potentially valuable information, primarily in class-imbalanced datasets where minority attack classes are already underrepresented. Unlike feature selection, which can often enhance generalization by removing noise and redundancy, size reduction, if not handled carefully, can negatively impact detection accuracy. Moreover, feature reduction methods compress dimensionality while keeping the overall event diversity. Our work aims to fill this gap by proposing a controlled size-reduction approach using stratified sampling, ensuring that data diversity and class proportions are preserved even in smaller training sets.The work in [16] explored how deep-learning models can be used as a feature-extraction tool aiming to remove redundant features from the dataset. The experiment was applied to an outdated balanced dataset with relatively small features. Meanwhile, information gain (IG-PCA) was also used as a dimensionality-reduction tool in [17]. In [18], two different feature-reduction methods were investigated with a more recent dataset than the dataset used in the previously mentioned works: the CISIDS2017 dataset.

This paper focuses on answering two questions. The first question is, "Is the large amount of data collected in IDS datasets needed to build robust IDS ML and AI systems?". The second question is, "What efficient reduction techniques can be used to reduce the size of IDS datasets, yet they can be used to build robust IDSs?".

Some works have focused on combining size and dimensionality-reduction techniques to extract the dataset's core value and enhance machine-learning (ML) model performance, but not in the context of intrusion-detection applications, such as the work in [19]. The primary contribution of this study is a practical framework for enhancing IDS performance through efficient data reduction rather than a novel detection algorithm. The proposed model combines deep feature extraction using autoencoders with stratified sampling to reduce the number of features and training samples without compromising classification performance. This two-stage reduction process significantly lowers computational costs and model complexity, making machine learning-based IDS solutions more scalable and suitable for real-time applications, especially in IoT scenarios. Experiments on six IDS-related datasets demonstrate that the proposed method preserves or even improves F1-scores while reducing training time by up to 99% in some cases. Therefore, this work's main contributions can be summarized as follows:

- We present a practical, two-stage dataset-reduction framework that combines stratified sampling with autoencoder-based feature selection to reduce both the size and dimensionality of IDS datasets. The first algorithm sorts the importance of the features in the dataset *via* an autoencoder. Then, the least important features are removed, followed by tuple reduction *via* stratified sampling. The second algorithm starts with stratified sampling, followed by feature ranking and selection.
- We empirically evaluate the trade-offs between different reduction percentages and their effects on training time and detection performance using multiple ML models across six public IDS datasets.
- We show that, when properly applied, dataset size and dimensionality reduction can achieve up to 99% decrease in training time with minimal performance loss (typically less than 1% drop in F1-score). The proposed reduction techniques prove that there is a notable degree of redundancy in the datasets. The huge amount of data should be questioned in these datasets, because, in many cases, the reduced datasets provide almost the same F1-score readings after data reduction. More attention should be paid to the unnecessary massive amount of data used to build robust IDSs.
- We provide a reproducible baseline for evaluating dataset-reduction strategies in IDSs, offering insights into scalability and efficiency for real-world deployment in resource-constrained environments.

The rest of this paper is organized as follows: Section 2 shows the related work. Preliminaries and methodology are presented in Section 3. Section 4 shows the results and assessments, and finally, the work is concluded in Section 5.

2. RELATED WORK

Data-reduction techniques are widely explored to address machine-learning datasets' growing complexity and size, specifically in intrusion detection systems (IDSs). These techniques typically fall into two categories: dimensionality reduction, which reduces the number of features (columns), and size reduction, which reduces the number of records (rows). This section critically investigates related works grouped by technique type and discusses their applicability to IDSs, mainly in IoT environments.

Linear techniques, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), have been broadly used to project data into lower-dimensional spaces. PCA is widely employed due to its computational efficiency and ability to decorrelate features. PCA has shown considerable performance with high-dimensional datasets, such as medical imaging and network traffic [20]-[21]. However, PCA supposes linear relationships between the features, which may not hold in complex IDS datasets. At the same time, LDA is useful for maximizing class separability, but suffers from scalability issues in large-scale, high-dimensional environments. Recent work has focused on autoencoders and their variants, including Deep Sparse Autoencoders (DSAEs), to handle these restrictions for non-linear and data-driven feature extraction. Unlike PCA, autoencoders do not suppose linearity and can model complicated feature relations [22]. This capability of modeling complex relationships makes them specifically suitable for IDS datasets with complex patterns and correlations. For instance, [23] used autoencoders to improve classification accuracy through feature selection. However, their work focused on general accuracy rather than on IDS-specific issues, like class imbalance or real-time deployment. Recently, Nabi and Zhou [24] explored using PCA and random projection for dimensionality reduction in intrusion-detection schemes using the NSL-KDD dataset. Their results emphasized random projection's computational efficiency and accuracy benefits over PCA. In contrast, this study explores deep autoencoders as a non-linear and data-driven technique for feature extraction and links this with a structured dataset size-reduction pipeline. Moreover, this evaluation spans multiple recent IDS datasets, addressing generalization, dataset redundancy, and attack-class preservation. In contrast to earlier studies that used traditional datasets, like NSL-KDD or outdated benchmark sets [16]-[17], our work leverages recent and large-scale IDS datasets, such as CSE-CIC-IDS 2018 and BoT-IoT. When integrated with stratified sampling, we confirm that autoencoders can preserve detection performance even under significant feature reduction.

Stratified sampling is a widely utilized technique for reducing dataset size while keeping class distributions, which is critical in class-imbalanced IDS contexts. Multiple studies [25][26][27][28] have examined its effect on handling large-scale datasets. For example, [28] proposed an enhanced stratified sampling framework with over-sampling of minority classes using Gaussian noise and clustering of

majority classes. However, these works frequently lacked comparative analysis of reduction order, sampling before *vs.* after feature reduction. Moreover, some prior works overlap in their discussion of stratified sampling without clearly distinguishing their contributions. We address this by systematically comparing each method's novelty and outcome: [25] applied stratified sampling in general big-data contexts, [26] optimized sampling with hash- based stratum construction, and [27] integrated stratified sampling with clustering for better illustration. Our method builds upon these by integrating sampling with deep feature selection, presenting a unified pipeline evaluated on multiple IDS datasets.

Recent developments have introduced scattering-based enhancements to graph neural networks for anomaly detection and feature learning. For instance, the STEG model [29] applies a wavelet-based scattering transform to edge features within an E-GraphSAGE architecture, significantly improving detection performance on network-intrusion datasets. STEG leverages multi-resolution edge encoding and node2vec embeddings to provide a fine-grained understanding of graph-structure anomalies, a strategy relevant to our anomaly-detection pipeline. In a related domain, the GeoScatt-GNN framework [30] combines geometric-scattering transforms with ANOVA-based statistical feature selection to predict Ames mutagenicity. While its application lies in bio-informatics, the architecture introduces a principled pipeline where meaningful features are extracted and filtered prior to GNN classification, highlighting the cross-domain effectiveness of scattering-transform approaches. Our work draws inspiration from these efforts, but focuses on reducing the dataset size, with a tailored architecture and feature-selection approach suited to network-level anomaly scenarios. We also emphasize the redundancy happening in the security-related dataset applied in the IoT environments.

A summary of the related works and methods is clarified in Table 1. Our approach closes this gap by employing a two-stage pipeline tested across six modern IDS datasets and comparing sampling-first *vs.* feature-first strategies. Additionally, we quantify training-time reduction and model resilience to aggressive reduction analysis, which previous studies often dismissed. The datasets related to security threats in IoT networks tend to be massive, hindering the detection models and requiring huge computational resources [31]. This work presents a methodology that can reduce dataset size while keeping the IDS performance high and accurate. Our work offers a more rigid, application-focused synthesis of dimensionality and size reduction in IDSs. It advances the field by addressing the interplay between reduction type and model performance using large-scale, recent IDS datasets. It also provides empirical proof across multiple classifiers and offers a reproducible framework for real-world deployment.

Table 1. Summary of data-reduction techniques in literature.

Technique	**Category**	**Dataset Used**	**Strengths**	**Limitations**
PCA [20]-[21], [24]	Dimensionality	NSL-KDD, CTG, DR	Fast, simple, linear separability	Fails on non-linear data
LDA [16]-[17]	Dimensionality	CTG, DR	Class separation-focused	Poor scalability
Deep Autoencoders [22]-[23]	Dimensionality	BoT-IoT, CSE- CIC-IDS2018	Handles non-linear features, scalable	Overfitting risk on small datasets
Stratified Sampling [25][26][27][28]	Size	KDD, CICIDS, financial	Maintains class balance	Requires stratification label
Sampling + Clustering [27]-[28]	Size	Big-data Clusters	Reduces outliers, enhances sample diversity	Adds clustering complexity
This Work	Size + Dimensionality	6 modern IDS datasets	Two-stage, flexible, efficient	Minor NB performance degradation noted

3. PRELIMINARIES AND METHODOLOGIES

This section introduces the datasets used in this study and the methodologies that are applied to reduce the size and dimensionality of the datasets. It also introduces the performance metrics that have been used to assess the efficiency of the methodologies used.

3.1 Datasets

Throughout this study, six intrusion-detection datasets were used: The Kitsune-ARP dataset [32], SNMP- MIB [33], the CSE-CIC-IDS2018 dataset [34], the BoTIoT dataset [35], the UNR-IDD dataset [36], and the credit-card fraud-detection dataset from [37]. The six datasets are all related to intrusion-detection applications, and they are collected from different hostile environments with different features and sizes.

What distinguishes the selected datasets in this study is that many datasets were recently collected from IoT environments. The datasets are challenging due to data imbalance, which is typical in intrusion-detection datasets in general. Meanwhile, many datasets are enormous, challenging for ML models, require a long time for training, and can result in very complex ML models. All non-binary datasets were transformed into binary datasets, such as the CSE-CIC-IDS2018 dataset.

A summary of the six datasets, including the size, dimension, and imbalance rate, is presented in Table.2. The datasets were renamed DS1-DS6 throughout this work, as shown in Table 2, to enhance the readability of the paper, especially the figures and tables.

Table 2. Summary of the datasets used in the study.

Dataset	Size (KB)	Records	Features	Imbalance Rate (%)
UNR-IDD (DS1)	267	2620	21	9.4
Kitsune-ARP (DS2)	15,300	15000	115	10
SNMP-MIB (DS3)	788	5000	34	10
CSE-CIC-IDS2018 (DS4)	315,233	1048576	80	50
BoTIoT (DS5)	620,600	2426574	24	27
Fraud detection (DS6)	100,500	248808	31	0.1724

3.2 The Techniques Used

We present here the main techniques used throughout this study. These techniques include sampling, dimensionality reduction, and ML techniques.

Sampling is selecting a representative set of items from a larger set. Sampling can be applied to select a specific number or percentage of samples. This work uses sampling with intrusion-detection datasets to select a certain percentage of the dataset to train the ML models, since many datasets are very large and contain hundreds of thousands of records, sometimes millions. Training machine-learning models with huge datasets requires high computational power and consumes time. The sampling process investigates the degree of redundancy existing in these datasets. When a half or a quarter of the data can be used to train the ML model and still give the same results as when the entire dataset was used, this can indicate that the dataset records include a noticeable degree of redundancy.

This work deploys stratified sampling to reduce the number of records in intrusion-detection datasets; meanwhile, it maintains the imbalance ratio. Since the data used is large and imbalanced, randomly selecting a small group from the data might alter the balance of the data; there is a more significant probability that a selected record belongs to the larger group. The datasets are also labeled, which makes stratified sampling a good choice for this presented work.

At the same time, dimensionality reduction is used for different purposes, such as having interpretable models or reducing the required computational time for ML-model training. PCA is commonly used for this task. The main difference between reducing the features using PCA and by autoencoders is that PCA can model linear structures. However, autoencoders do not assume linearity [18]. In [9], dimensionality-reduction techniques were divided into three main categories based on the data structure. Three main dimensionality-reduction methods are available in the literature: linear manifolds, non-linear manifolds and curved twisted manifolds.

Due to the high performance of autoencoders in reducing the intrusion detection features that outperform other methods, such as PCA and LDA [38]; dense autoencoders are used in the proposed methods in this paper. The main idea of the autoencoder is to have the ability to reconstruct the input after encoding it to a lower dimension. For dimensionality reduction, the most important part of the autoencoder is the latent space, the encoder's output, which has the most critical features of the input. Its size is a hyper-

parameter that can be controlled to define the desired number of features. In the proposed methods, features with the highest weights were selected after sorting all features based on their importance.

Eventually, the selected approaches to reduce the size and dimensionality of different intrusion-detection datasets are evaluated with nine different ML models. These ML models are the K-Nearest Neighbor algorithm (KNN), the Support Vector Machine Algorithm (SVM), Naive Bayes, linear regression, LDA, C5, XGBoost, Random Forest, and ADA. These ML models were selected throughout this study, because they are extensively used in the literature with similar datasets. The ML models were proven to be efficient and durable with tabular datasets. The random forest is a robust ensemble model that reduces overfitting and performs well on tabular data with noisy or redundant features. XGB is an ensemble model that proved its efficiency in many real problem-detection tasks. SVM model has a powerful feature which is called kernel trick that gives SVM the power to handle binary classification effectively. The KNN is a simple, non-parametric model that benefits from reduced feature spaces and works well for pattern recognition. AdaBoost is an ensemble technique that adapts to classification errors, making it more robust to decide on samples, which is very important with imbalanced datasets.

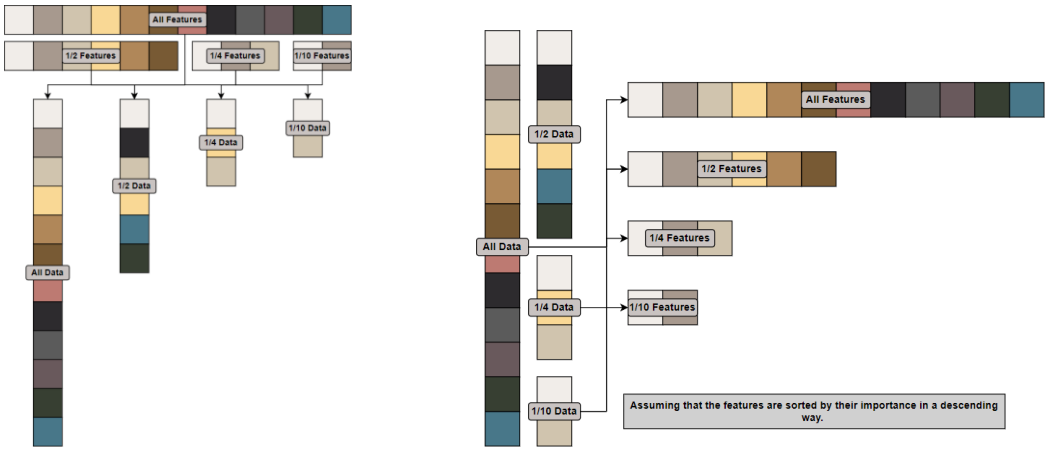
3.3 Proposed Methods

This work investigates how dataset size-reduction and feature-reduction methodologies affect machine-learning algorithms. The analysis is studied in the context of IDS systems and IoT environments. The method followed throughout the proposed work adapts two approaches clarified in Algorithms 1 and 2. In the first approach, data reduction is applied first, followed by size reduction, and then ML models are used with the data to build the IDS models. In the second approach, size reduction is applied before the feature-reduction step, and then ML models are used again to build the IDS models. Finally, the performance of the models built with the first approach is compared with the performance of those created with the second approach, as shown in Figure 1. The approach that produces better results is recommended for IDS datasets. The size-reduction method used throughout this study is the stratified sampling technique. Meanwhile, the feature reduction method used here is the dense autoencoder method.

Figure 1 illustrates the two-stage dataset-reduction strategies evaluated in this study. In the Feature Reduction First (FF) approach (Figure 2a), the full dataset is used to train an autoencoder, which ranks features based on their importance. The dimensionality of the dataset is then reduced by selecting the top-ranked features; 1/2, 1/4, or 1/10 of the full set. Finally, stratified sampling, 1/2, 1/4, or 1/10 of the full set, is applied on the reduced dataset to create reduced sub-sets for training, preserving class distribution. On the other hand, the Sampling First (SF) technique (Figure 2b) starts by applying stratified sampling directly to the full dataset, yielding sub-sets that are 1/2, 1/4, or 1/10 the dataset size. Each reduced sub-set is then passed to an autoencoder to perform feature reduction. The reduction at this stage is applied to extract 1/2, 1/4, 1/10, or full features. This order assumes that features are sorted in descending order of importance after training, as indicated in the figure. The main difference between the two methods is the timing of dimensionality reduction relative to volume reduction. FF (Feature reduction First) guarantees that the autoencoder is trained on the most complete data to capture more prosperous feature patterns. SF (Sampling First), meanwhile, reduces computational load earlier, but may lose important patterns due to early sub-sampling. This trade-off is critical when working with class-imbalanced and high-dimensional IDS datasets.

DS1, DS2, and DS3 were used through the investigation and steps mentioned in the previous paragraph. Meanwhile, DS4, DS5, and DS6 were used throughout the assessment process due to their large size where applying the reduction techniques is essential. During the assessment stage, we practically try to evaluate the performance of different ML models and the order of the reduction process. Time-analysis results, besides F-score measures, are recorded. In the second stage, we aim to prove the correctness of the conclusions made in the first stage. For example, time-reduction and close-to-perfect performance measures are used, despite using fewer data and features. In this study, we list only the F1-score as a suitable performance measure, which combines precision and recall. This allows us to evaluate the robustness of IDS performance across different levels of dataset reduction in a compact and interpretable way. Although additional metrics, such as recall, precision, and false-positive rate, were computed, their trends closely followed the F1-score. For clarity and space efficiency, only the F1-score is reported in

the main-results tables, as it sufficiently describes the robustness of IDS performance under dataset reduction.



(a) Feature Reduction First (FF) Approach (b) Size Reduction (SR) First Approach

Figure 1. The methodology followed to reduce the datasets.

Algorithm 1 Feature Extraction First (FF)

Require: Input dataset *DSi*
for each dataset *DSi*, $i \in [1,2,3]$ **do**
  for each *F* in $\{1, 2, 4, 10\}$ **do**
    Apply feature extraction on *DSi*, extracting the most important 1/*F* features from *DSi* and update *DSi*
    for each $F \in \{1, 2, 4, 10\}$ **do**
      Apply stratified sampling on *DSi* to extract 1/*S* of the data: *DSi* ←*DSi/S*
      Apply the machine learning methods to *DSi*
    end for
  end for
end for

Algorithm 2 Size Reduction First (SF)

Require: Input dataset *DSi*
for each dataset *DSi*, $I \in 1,2,3$ **do**
  for each *S* in $1, 2, 4, 10$ **do**
    DSi = StratifiedSampling(*DSi*, 1/*S*)
    for each $F \in \{1, 2, 4, 10\}$ **do**
      DSi = AutoencoderFeatureExtraction(*DSi*, 1/*F*)
      Apply the machine learning methods to *DSi*
    end for
  end for
end for

We investigate how the datasets' size and feature reduction can affect the performance of nine different ML models. The models were trained with the data prior to reduction, then trained with the reduced datasets. Size and dimensionality were reduced in different scenarios, and then a comparison was held to assess the different reduction scenarios. The method used for data reduction is the Stratified-sampling process which reduces the data size and keeps the data distribution untouched.

The technique used for the feature-reduction process is ranking the importance of all features of the datasets using a dense autoencoder. Every dataset was used to train the autoencoder and then, the encoder was used to explore and rank the importance of all features based on their weights. The features were then sorted, and the less critical features were dropped from the dataset. Many scenarios were examined; a half of the features were selected, and one-fourth and one-tenth of the features were selected in other scenarios. Selecting-all-features scenarios were also analyzed.

The encoder architecture with the bottleneck consists of three dense layers with the LeakyReLU activation function and two batch-normalization layers, as shown in Figure 2. The autoencoder designed in this study follows a symmetrical architecture tailored for reconstructing input features while capturing meaningful representations in its bottleneck layer. The input-layer size corresponds directly to the number of features in each dataset. The encoder consists of two fully connected layers: the first layer expands the dimensionality to twice the input size and applies a LeakyReLU activation function, followed by batch normalization. At the same time, the second layer reduces the dimensionality back to the original feature size using the same activation and normalization setup. The bottleneck layer maintains this same dimensionality, serving as the latent representation of the input data without applying compression, allowing for feature-importance extraction. The decoder mirrors the encoder in structure, reconstructing the data through symmetric dense layers and concluding with a linear activation function in the output layer. The architecture was selected to balance expressive power and computational efficiency, particularly for high-dimensional, imbalanced intrusion-detection datasets where non-linear patterns and feature interactions are prevalent. The model was trained using the Adam optimizer with a learning rate of 0.001, a batch size of 16, and 100 epochs. The trained encoder was used to extract latent feature weights; all feature weights were reported without reduction at this level, which were subsequently ranked to identify the most important features. While this work focuses on autoencoder-based feature extraction, we acknowledge the importance of traditional methods, such as ANOVA and chi-square [39]. However, these classical approaches rely on assumptions of linearity and independence among features, which are often violated in intrusion detection scenarios. Autoencoders, by contrast, provide the flexibility to model complex, non-linear, and correlated feature interactions more effectively. However, although autoencoders offer powerful non-linear feature-extraction capabilities, they also introduce certain limitations. One concern is the risk of overfitting when training deep models on reduced datasets. They also require high computational capabilities when very large datasets are used. Hence, they should be used with caution to deliver accepted results while requiring minimal computational power.

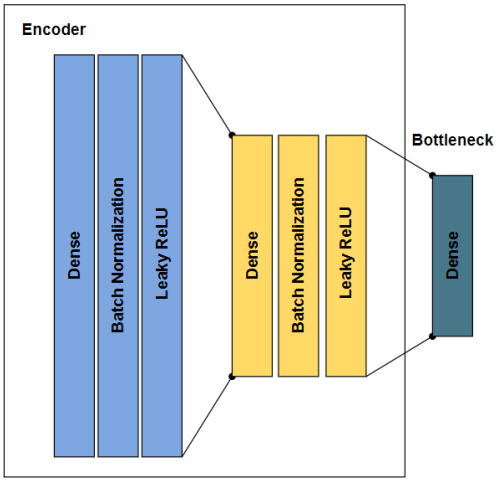


Figure 2. Encoder architecture.

Stratified sampling was used to reduce the size of the data. Every dataset was reduced to one half, one-fourth, and one-tenth; it was also analyzed without size reduction. The experiment goes through different steps, aiming to explore the efficiency of different reduction strategies. The whole data was analyzed with all features, a half of the features, one-fourth, and one-tenth using the nine ML models, which will be mentioned shortly. The exact process was repeated when one-fourth of the data was used, and one-tenth of the data was used. Reducing the data size followed by feature reduction is noted by (SF), which indicates "Sampling First " since the sampling method is applied to the data before the feature-reduction process. It is worth mentioning that when stratified sampling was applied before feature ranking, the importance ranks of some features changed due to the reduced dataset size. However, the most significant features showed minimal change in their ranking. During (FF) or "Feature First," the previously mentioned data-reduction process was applied, but feature reduction was applied first to the data, followed by the sampling step.

The used ML models are KNN, SVM, Naive Bayes, linear regression, LDA, C5, X-GBoost, Random Forest, and ADA. All models are evaluated with their default hyper-parameters as provided by sklearn Python libraries. Employing default parameters was to emphasize the practical applicability and effectiveness of the presented dataset-reduction techniques without requiring exhaustive hyper-parameter tuning. This setup demonstrates that meaningful improvements in computational efficiency and model performance can be achieved without additional optimization steps.

All datasets were normalized through a MinMaxScalar. During pre-processing, non-numerical features were dropped from the datasets, such as the Timestamp feature from the CSE-CIC-IDS2018 dataset and " Switch ID" and "Port Number" from DS1. Data pre-processing includes multiple steps, guaranteeing that the AI models will be fed with proper data values. During categorical feature encoding, all categorical features were encoded using One-Hot Encoding, transforming them into numerical formats suitable for machine-learning models. Moreover, features containing more than 50% of missing data were removed from the dataset. The remaining missing values were handled using mean imputation. Additionally, numerical features were scaled using Min-Max normalization, mapping feature values to a range between 0 and 1. This normalization improves model convergence and performance stability. Finally, stratified sampling was used explicitly to maintain the original class distribution, effectively managing dataset imbalance during data reduction. All datasets were divided into 64% for training and 36% for testing, while the 70/30 or 80/20 splits are widely used as standard practice. The slightly non-standard split in this study ensured that a representative portion of the minority class remained in the testing set, which is particularly important for performance evaluation on imbalanced datasets.

4. RESULTS AND ASSESSMENT

This section presents the results of the data-reduction techniques described in the previous section and investigates how combining different reduction techniques influences the ML models used. All the experiments were conducted using the Google Co-Lab platform based on Python 3. Google Co-Lab offers 12 GB RAM and 128 GB Disk. To rank feature importance, absolute weights from the first dense layer of the encoder were extracted. These weights reflect the strength of the connection between input features and their influence on the latent representation. We ranked in descending order based on the sum of absolute weights across all neurons in this layer. We then selected the top-k features: 1/2, 1/4, or 1/10 for further evaluation. Stratified sampling was applied using a class-wise sampling strategy to maintain class proportions. This was done *via* pandas.groupby('class').apply(lambda x: x.sample(frac=p)) in Python, where p is the target sampling fraction; 0.5, 0.25, 0.10. This method was used to generate progressively smaller, but balanced, datasets for training and testing. This step was either applied before or after feature selection based on the reduction strategy (SF or FF).

4.1 Machine-learning Model Results

To detail each model's performance, the F-score metric is used to represent the results as values for all steps of the two approaches in Tables 3, 4, 5, 6, 7, and 8, because F-score is sufficient measure for imbalanced data. The numbers at the top of the columns represent the feature percentage and the size percentage; F-S "0.5–0.25," in Table 4, for example, denotes the ML models' performance with a data sample retaining the top half of the features after ordering them according to their importance. If we have 20 features, for example, the top-10 features are used. Meanwhile, 0.25 means that one-fourth of the data tuples are used; for example, if we have 1000 tuples, 250 tuples are selected *via* stratified sampling and used through the training and testing processes. Features extraction precedes size reduction in this case where the "F" comes first. However, S-F "0.5–0.25," indicated using 50% of the tuples and 0.25 of the features where the size reduction precedes the feature extraction method.

In Table 3, all classifiers achieve a high F-score until the 0.1-1 reduction is applied, starting with size reduction. This is expected due to the small size and dimensions of DS1. However, when the reduction processes are swapped in Table 4, LR, SVM, and C4.5 can still produce high results. The conclusion that can be extracted from these results is that feature reduction first is better for small-sized and low-dimensional datasets. Moreover, RF, KNN, C4.5, and XGB are the best classifiers for DS2 based on the F-score when applying size reduction first, as shown in Table 5. XGB is the most stable classifier when feature reduction is applied first. At the same time, other models were unstable or could not achieve high F-scores in most data-reduction scenarios, as Table 6 demonstrates. As for the third dataset, KNN

is the best classifier for size reduction first, as shown in Table 7 and XGB is the best for feature reduction first, as shown in Table 8. Most of the classifiers performed well, and it was the most suitable dataset for NB. In all Tables 3-8, we have colored the highest values in each column in yellow to highlight the best results for each division, also to highlight the best-performing models. Our analysis shows that dataset and feature-reduction strategies exhibit multiple levels of performance degradation. Decreasing the dataset or feature set to 1/2 or 1/4 generally resulted in a less than 2% drop in F1-score. More aggressive reduction to 1/10 greatly affected detection accuracy, particularly for complex datasets, like BoT-IoT and CSE-CIC-IDS2018. Notably, KNN and AdaBoost shared larger performance drops under 1/10 feature reduction, due to their sensitivity to input dimensionality. In contrast, ensemble tree- based models, such as XGBoost and Random Forest, showed higher resilience, maintaining performance even when trained on only 10% of features or samples. This indicates that the model's robustness to feature sparsity and sample diversity plays an essential function in mitigating the effects of reduction. These trade-offs emphasize the significance of choosing the proper model and reduction level based on the dataset's complexity and attack distribution.

Table 3. DS1 sampling first F1-score results.

S–F	**1–1**	**1–0.5**	**1–0.25**	**1–0.1**	**0.5–1**	**0.5–0.5**	**0.5–0.25**	**0.5–0.1**	**0.25–1**	**0.25–0.5**	**0.25–.025**	**0.25–0.1**	**0.1–1**	**0.1–0.5**	**0.1–0.25**
KNN	**1.000**	**1.000**	**1.000**	0.923	**1.000**	**1.000**	**1.000**	0.885	0.999	0.976	**1.000**	0.923	0.560	0.498	0.500
SVM	0.995	**1.000**	**1.000**	0.885	**1.000**	0.976	**1.000**	0.885	0.999	0.999	**1.000**	0.923	0.500	0.500	0.500
NB	0.991	0.993	0.998	0.508	0.986	0.993	0.995	0.514	0.974	0.983	0.986	0.982	0.543	0.535	0.529
LR	0.999	**1.000**	**1.000**	0.846	0.999	0.976	0.962	0.885	0.999	0.988	0.885	0.692	0.500	0.500	0.500
LDA	0.999	**1.000**	**1.000**	1.000	0.999	0.976	**1.000**	1.000	0.999	0.998	**1.000**	1.000	0.500	0.500	0.500
C4.5	**1.000**	**1.000**	**1.000**	0.962	**1.000**	**1.000**	**1.000**	0.962	1.000	**1.000**	**1.000**	1.000	0.500	0.500	0.500
XGB	**1.000**	**1.000**	**1.000**	0.962	**1.000**	**1.000**	**1.000**	0.962	1.000	**1.000**	**1.000**	1.000	0.500	0.500	0.500
RF	**1.000**	**1.000**	**1.000**	1.000	**1.000**	0.976	**1.000**	1.000	1.000	**1.000**	**1.000**	1.000	0.500	0.500	0.500
Ada	**1.000**	**1.000**	**1.000**	1.000	**1.000**	**1.000**	**1.000**	1.000	1.000	**1.000**	**1.000**	1.000	0.500	0.500	0.500

Table 4. DS1 feature extraction first F1-score results.

F-S	**1–1**	**1–0.5**	**1–0.25**	**1–0.1**	**0.5–1**	**0.5–0.5**	**0.5–0.25**	**0.5–0.1**	**0.25–1**	**0.25–0.5**	**0.25–.025**	**0.25–0.1**	**0.1–1**	**0.1–0.5**	**0.1–0.25**	**0.1–0.1**
KNN	**0.999**	**0.999**	**0.999**	**0.999**	**1.000**	0.990	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.990	**1.000**	0.333
SVM	0.995	0.995	0.995	0.995	0.998	0.992	0.995	0.973	**1.000**	0.984	0.798	0.471	0.809	0.781	0.491	**1.000**
NB	0.918	0.918	0.918	0.918	0.844	0.836	0.814	0.791	0.885	0.843	0.983	**1.000**	0.764	0.708	0.565	**1.000**
LR	**1.000**	**1.000**	**1.000**	**1.000**	0.668	0.544	0.470	0.468	0.465	0.470	0.459	0.471	0.465	0.467	0.491	0.333
LDA	**1.000**	**1.000**	**1.000**	**1.000**	0.791	0.749	0.723	0.851	0.803	0.779	0.459	**1.000**	0.465	0.466	0.491	0.333
C4.5	**1.000**	**1.000**	**1.000**	**1.000**	0.996	**1.000**	**1.000**	0.931	0.998	0.973	0.964	**1.000**	**1.000**	0.942	**1.000**	**1.000**
XGB	**1.000**	**1.000**	**1.000**	**1.000**	0.996	**1.000**	**1.000**	0.944	0.998	0.978	0.982	0.818	0.999	0.980	0.491	0.000
RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.959	0.999	**1.000**	0.964	0.884	0.999	0.960	0.824	0.333
Ada	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.997	**1.000**	0.959	0.999	**1.000**	**1.000**	0.884	0.960	0.969	0.491	0.333

Table 5. DS2 sampling first F1-score results.

S-F	**1–1**	**1–0.5**	**1–0.25**	**1–0.1**	**0.5–1**	**0.5–0.5**	**0.5–0.25**	**0.5–0.1**	**0.25–1**	**0.25–0.5**	**0.25–0.25**	**0.25–0.1**	**0.1–1**	**0.1–0.5**	**0.1–0.25**	**0.1–0.1**
KNN	**0.999**	0.995	0.984	0.975	0.997	0.991	0.976	0.945	0.992	0.974	0.939	0.944	0.991	0.985	0.952	0.907
SVM	0.837	0.806	0.509	0.473	**0.839**	0.785	0.609	0.473	0.532	0.499	0.473	0.473	0.588	0.554	0.473	0.473
NB	0.250	0.253	0.249	0.255	0.233	0.234	0.232	0.240	0.235	0.240	0.169	0.179	0.412	0.417	**0.431**	0.407
LR	**0.882**	0.842	0.763	0.615	0.833	0.782	0.733	0.541	0.606	0.565	0.529	0.473	0.552	0.540	0.473	0.473
LDA	**0.978**	**0.978**	0.967	**0.990**	0.944	0.959	0.947	0.965	0.807	0.836	0.815	0.827	0.602	0.617	0.576	0.550
C4.5	**1.000**	**1.000**	0.998	0.985	0.998	0.995	0.992	0.995	**1.000**	0.995	0.994	**1.000**	0.995	0.977	0.955	0.946
XGB	**1.000**	0.998	0.998	0.995	**1.000**	0.998	0.996	0.995	**1.000**	0.999	0.994	0.995	0.998	0.995	0.975	0.985
RF	**1.000**	0.999	0.996	**1.000**	**1.000**	0.999	0.990	0.985	0.999	0.994	0.981	0.990	0.993	0.988	0.957	0.969
Ada	**0.999**	0.996	0.988	0.995	0.977	0.990	0.983	0.967	0.991	0.988	0.964	0.995	0.680	0.657	0.754	0.710

Table 6. DS2 feature extraction first F1-score results.

F-S	1–1	1–0.5	1–0.25	1–0.1	0.5–1	0.5–0.5	0.5–0.25	0.5–0.1	0.25–1	0.25–0.5	0.25–0.25	0.25–0.1	0.1–1	0.1–0.5	0.1–0.25	0.1–0.1
KNN	**0.999**	**0.999**	**0.999**	**0.999**	0.994	0.993	0.975	0.930	0.994	0.944	0.806	0.697	0.988	0.898	0.689	0.400
SVM	0.854	0.854	0.854	0.854	0.858	0.803	0.488	0.473	0.532	0.473	0.468	0.481	0.607	0.469	0.472	**1.000**
NB	0.255	0.255	0.255	0.255	0.231	0.238	0.247	0.259	0.232	0.235	0.243	0.184	0.408	0.429	0.382	**1.000**
LR	**0.899**	**0.899**	**0.899**	**0.899**	0.852	0.783	0.734	0.601	0.609	0.509	0.468	0.481	0.571	0.469	0.472	0.400
LDA	**0.987**	**0.987**	**0.987**	**0.987**	0.951	0.940	0.966	0.926	0.806	0.886	0.834	0.694	0.622	0.631	0.671	0.400
C4.5	**0.998**	**0.998**	**0.998**	**0.999**	0.996	0.994	0.996	0.985	0.996	0.986	0.993	**1.000**	0.985	0.955	0.689	0.400
XGB	**0.999**	**0.999**	**0.999**	**0.999**	**1.000**	0.994	0.998	0.990	**1.000**	**1.000**	**1.000**	**1.000**	0.995	0.964	0.817	**1.000**
RF	**0.999**	**0.999**	**1.000**	**0.999**	**1.000**	0.995	0.990	0.985	**1.000**	0.992	0.986	0.924	0.995	0.982	0.709	0.455
Ada	**0.997**	**0.997**	**0.997**	**0.997**	0.989	0.927	0.969	0.927	0.991	0.990	0.993	0.824	0.706	0.522	0.625	0.400

Table 7. DS3 sampling first F1-score results.

S-F	1–1	1–0.5	1–0.25	1–0.1	0.5–1	0.5–0.5	0.5–0.25	0.5–0.1	0.25–1	0.25–0.5	0.25–0.25	0.25–0.1	0.1–1	0.1–0.5	0.1–0.25	0.1–0.1
KNN	**1.000**	**1.000**	**1.000**	0.974	0.999	**1.000**	**1.000**	0.944	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.995	**1.000**
SVM	0.998	0.992	**1.000**	0.987	0.998	0.987	0.973	0.895	0.998	0.995	0.995	**1.000**	0.813	0.803	0.861	0.794
NB	0.912	0.923	0.922	0.874	0.852	0.866	0.853	0.807	0.880	0.883	0.860	0.856	0.738	0.757	0.751	0.763
LR	**1.000**	0.997	**1.000**	0.959	0.681	0.582	0.470	0.468	0.465	0.467	0.470	0.468	0.465	0.467	0.470	0.468
LDA	**1.000**	**1.000**	**1.000**	0.987	0.788	0.782	0.810	0.744	0.790	0.772	0.786	0.773	0.465	0.467	0.468	0.468
C4.5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.949	0.973	**1.000**	0.998	0.965	0.987	**1.000**	0.997	0.994	0.881
XGB	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.949	0.973	**1.000**	**1.000**	0.965	0.959	**1.000**	0.997	0.994	0.916
RF	**1.000**	**1.000**	0.971	**1.000**	**1.000**	**1.000**	0.965	**1.000**	**1.000**	**1.000**	0.959	**1.000**	**1.000**	0.997	0.989	0.899
Ada	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.949	0.973	**1.000**	**1.000**	0.965	0.973	0.968	0.966	0.941	0.859

Table 8. DS3 feature extraction first F1-score results.

F-S	1–1	1–0.5	1–0.25	1–0.1	0.5–1	0.5–0.5	0.5–0.25	0.5–0.1	0.25–1	0.25–0.5	0.25–0.25	0.25–0.1	0.1–1	0.1–0.5	0.1–0.25	0.1–0.1
KNN	0.999	0.999	0.999	0.999	**1.000**	0.990	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.990	**1.000**	0.333
SVM	0.995	0.995	0.995	0.995	0.998	0.992	0.995	0.973	**1.000**	0.984	0.798	0.471	0.809	0.781	0.491	**1.000**
NB	0.918	0.918	0.918	0.918	0.844	0.836	0.814	0.791	0.885	0.843	0.983	**1.000**	0.764	0.708	0.565	**1.000**
LR	**1.000**	**1.000**	**1.000**	**1.000**	0.668	0.544	0.470	0.468	0.465	0.470	0.459	0.471	0.465	0.467	0.491	0.333
LDA	**1.000**	**1.000**	**1.000**	**1.000**	0.791	0.749	0.723	0.851	0.803	0.779	0.459	**1.000**	0.465	0.466	0.491	0.333
C4.5	**1.000**	**1.000**	**1.000**	**1.000**	0.996	**1.000**	**1.000**	0.931	0.998	0.973	0.964	**1.000**	**1.000**	0.942	**1.000**	**1.000**
XGB	**1.000**	**1.000**	**1.000**	**1.000**	0.996	**1.000**	**1.000**	0.944	0.998	0.978	0.982	0.818	0.999	0.980	0.491	0.000
RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.959	0.999	**1.000**	0.964	0.884	0.999	0.960	0.824	0.333
Ada	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.997	**1.000**	0.959	0.999	**1.000**	**1.000**	0.884	0.960	0.969	0.491	0.333

After training the different ML models with different portions from different datasets, the following notes should be considered from the tables:

- In all cases, data can be reduced by a half regarding both size and dimensionality, yet the ML models' performance remains the same.
- Applying the proper process to select part of the data to train the model can give the same results when all the data is used.
- The data-reduction techniques used throughout this work can enhance the required time to train and test the models.
- The data-reduction techniques used throughout this work can also produce less complicated models with the same efficiency.

4.2 Evaluating the Proposed Methods

The assessment step is presented and explored in this sub-section, where multiple data-reduction scenarios are being applied to three huge datasets. Feeding these datasets into the ML models requires very high computational resources. Additionally, time-demanding processes should be considered.

For the datasets DS4, DS5 and DS6, the reduction techniques were applied to investigate how the precision, recall, and F1-score were affected. The required training time to train all models is also measured. DS4 is a huge dataset in size and dimension; by extracting 0.001 of the size and a half of the features, all the classifiers still have a high performance of F-score, especially the KNN. Nevertheless, NB classifier behavior is sensitive to this level of reduction, as shown in Figure 4a. In other experiments, the NB was the worst when applied to a vast dataset with a small dimension, such as (DS4). The LDA performance with DS5 degrades, compared with its performance when DS4 was used, while other algorithms were robust to the reduction, as shown in Figure 4b. A moderate dimension and size dataset (DS6) was used to investigate the proposed approach; NB was the worst in comparison, even without reducing the data, while the other algorithms performed well. RF and XGB classifiers are the best for this data, as shown in Figure 4c. Every experiment held to reduce the size or the dimensionality of DS4,

DS5, and DS6 datasets was repeated 10 times, and the ML model results were measured and averaged and then clarified in Figure 4. This step is necessary to examine the reduction techniques' effectiveness and confirm the derived conclusions.

The time required to train DS4 when all the data was used is 3750.40s. When the data was reduced to 0.05, 0.01, and 0.001, the required time to train all the models was reduced to 22.53s, 6.05s, and 3.74s, and when a half of the important features were selected from the 0.001 part of the data, the time was reduced to 3.2s. Yet, the ML classifiers still can detect anomaly behavior even when the dataset size is dramatically reduced (see Figure 4a).

As the experiment focuses on reducing the size of the datasets horizontally and vertically, this reduction is expected to affect the required time to train the ML models. DS4, Ds5, and DS6 are reduced in many ways to study how time is affected, and the time required to train all the mentioned ML models is reported. The time needed to train DS5 when all the data was used was 9779.81s, but when the size of the dataset was reduced to 0.01, the required time was 2.58s only, and the required time to train all the models was reduced to 1s when 0.001 of the dataset was used. Meanwhile, the ML models' performance measured in F-score are mostly close to 100% as shown in Figure 4b.

DS6 training time was 350.33s and reducing the size to the half made the training time become 118.4s. Reducing the features to the half made the training time become 221.41s, while combining both reductions made the time become 79.12s. ML models, such as KNN, SVM, XGB and RF, can still produce perfect results (see Figure 4c). Figure 3 lists the time required for training DS4, DS5 and DS6 and the required time when multiple reduction techniques were used. 0.5S means that a half the data was used, while 0.5F indicates the percentage of reduction applied to the features, where all the values in the figure are measured in seconds.

This reduction in computational time is due to the reduction in dataset rows and columns. The number of rows in each dataset is reduced *via* stratified sampling, while the number of columns is reduced via feature extraction carried out using the autoencoder model. Combining feature extraction with the size reduction process makes the dataset size shrink vertically and horizontally. The required processing time for ML models is a function of the number of rows and columns. Hence, if we can assume that the total computational time for these models is T = F (numOfRows,numOfCol,...), a function of the number of rows and the number of columns, then reducing the value of either numOfRows,numOfCol, or both will have a reducing impact on the required computational time.

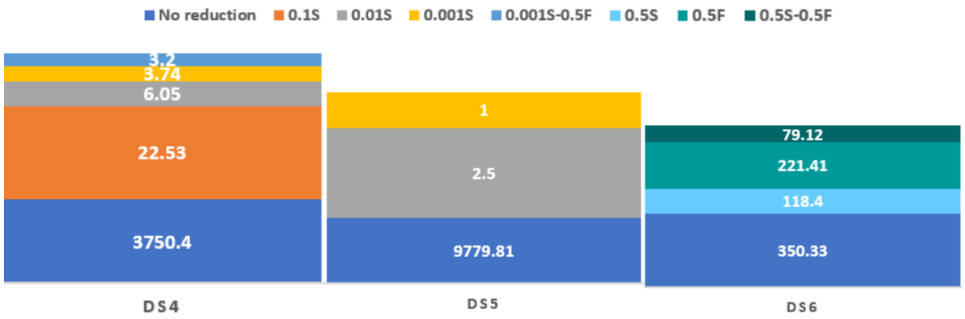


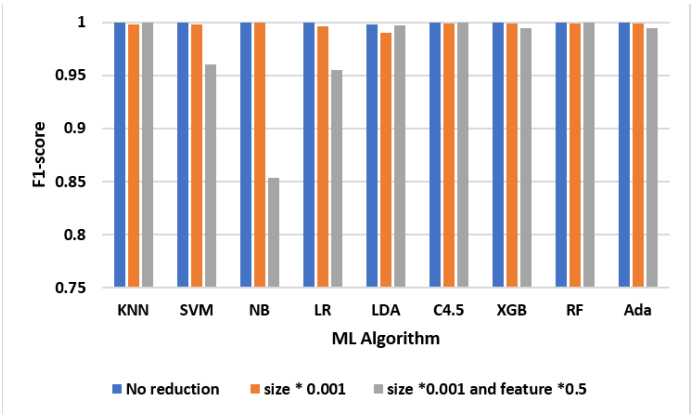
Figure 3. Time enhancement when large datasets were used.

4.3 Result Analysis and Recommendations

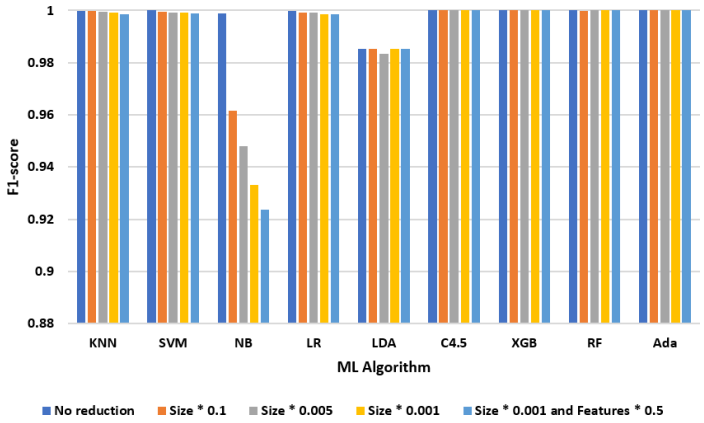
Simple reduction techniques, such as stratified sampling, can reduce the required time to build and train different ML models. Nevertheless, the performance of ML models is kept almost untouched. The huge amount of records stacked in different IDS datasets might be necessary, but not for IDS systems using ML models, such as those presented in this work. Some models can be less trusted, such as NB, and sometimes LR and LDA should be avoided, too. KNN, XGBoost, RF, and C-5 models are robust and can be trusted even when reduction methods are applied to the data.

When dealing with massive IDS datasets, reduction techniques, such as stratified sampling, and dimensionality-reduction techniques, such as autoencoders, are highly recommended to be used with the data to make it more usable. If the number of records in the dataset is small; i.e., < 20000, using the autoencoder first is highly recommended. For example, for a dataset similar to DS5, which is used here,

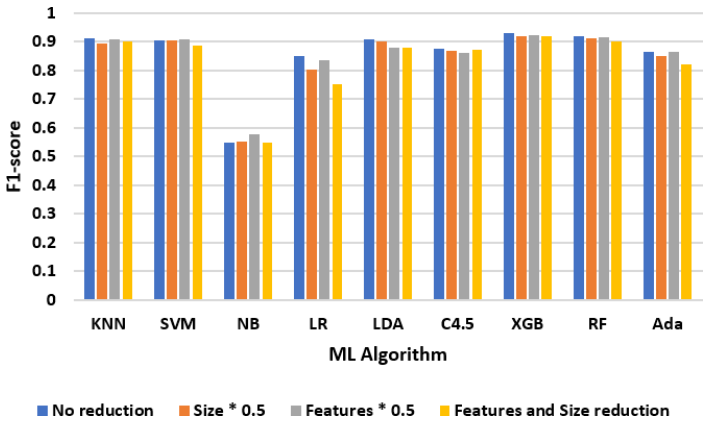
reducing the data size first is recommended, since training the autoencoder and getting the results from the encoder will take a very long time.



(a) DS4



(b) DS5



(c) DS6

Figure 4. ML model performance when large datasets were used.

If the dataset is already small, but has a large number of features, like DS2, which has 115 features, extracting the important features first is preferred since the autoencoder accuracy will be better with more data tuples to train it. Extracting the most important features from the dataset might enhance the performance of some ML models, like NB and SVM, with the DS2 results above.

The amount of the reduction to the data; i.e., how much data should be used to train the model, is a subject of experience and the logic of trial and error. The reduction tools are available and should be used with wisdom. For example, DS6 was reduced to the half to make the training time more efficient. While DS5 was reduced to one-tenth, considering that DS5 is almost five times the size of DS6, DS6 is a very unbalanced dataset.

The answer to the first research question is that huge IDS datasets are not necessarily needed, because the study results show that the ML models can produce sufficient results in many reduction cases, especially when certain ML models are used, such as Random Forest and KNN. The answer to the second question is that size reduction, feature reduction, and combining both reduction techniques can be used to reduce the size of the datasets while keeping the ML-model results sufficient. Although the proposed method does not introduce a new detection algorithm, it handles a crucial operational challenge in IDSs: the need for scalable and efficient model training on large, high-dimensional datasets. The framework shows that significant computational gains can be achieved through structured dataset reduction, allowing faster deployment and real-time responsiveness without degrading detection performance. This contribution supports more practical and cost-effective implementation of IDSs in environments where computational resources and latency are constrained.

4.4 Scalability Considerations and Real-world Deployment

The suggested dataset-reduction framework is developed to be modular and scalable, allowing it to adapt to diverse deployment scenarios. In cloud or cluster-based environments, the autoencoder training process can be parallelized and accelerated using GPU hardware, making it feasible to extract feature importance from even larger IDS datasets, such as real-time streaming logs or full network captures. The feature-selection step, once learned, can be reused across multiple time windows or data batches with minimal retraining.

Our sampling-first (SF) pipeline offers a practical compromise for edge-computing environments whose computational resources of which are limited. Applying stratified sampling before dimensionality reduction minimizes resource usage and preserves class distribution. Additionally, autoencoder-based feature selection lowers memory requirements and latency for deployed ML models. Thus, the discussed reduction methods are sufficient for academic evaluation and functional for real-world IDS applications where scalability, model-retraining efficiency, and system throughput are key considerations.

4.5 Comparison with Other Works

Table 9 demonstrates a comparison between our work and recent works with similar contributions.

The comparison of our work with recent contributions emphasizes key dissimilarities in dataset selection, feature-reduction methodologies, machine-learning models, and overall effectiveness in cyber-threat detection. One of the main strengths of our technique is the use of multiple datasets, including Kitsune- ARP, SNMP-MIB, CSE-CIC-IDS2018, BoTIoT, UNR-IDD, and Credit Card Fraud, which provides a more comprehensive evaluation of cyber threats. This contrasts studies, such as Behiry and Aly (2024), which focus on certain datasets, like NSL-KDD, UNSW-NB15 and CICIDS2017. Using various datasets in our study improves the generalizability of the results, although it presents sophistication in formalizing feature-selection techniques.

The data-reduction strategy used in our study combines autoencoders with stratified sampling, setting it apart from the principal component analysis (PCA) and singular value decomposition (SVD) approaches used in other studies. Autoencoders allow for non-linear feature extraction, which provides more robust dimensionality reduction, unlike traditional methods that assume linear relationships between variables. Compared to the Coot Optimization Algorithm (COA) used by Vallabhaneni et al. [42], our approach fulfills similar feature-reduction effectiveness, but significantly reduces the computational cost. Combining autoencoders with stratified sampling ensures that essential features are retained while reducing redundancy, making our method accurate and efficient. Another distinguishing factor is using stratified sampling instead of synthetic oversampling methods, like SMOTE, which Behiry & Aly [40] utilized. While SMOTE artificially generates new samples, stratified sampling preserves the natural distribution of data, preserving class balance without introducing synthetic artifacts. This approach ensures that minority-class instances, crucial for fraud and intrusion detection, remain well-represented while reducing data size. By leveraging stratified sampling, our method enhances dataset efficiency without sacrificing classification performance.

The selection of machine-learning models further distinguishes our work from previous studies. Our evaluation encloses a diverse set of algorithms, including K-Nearest Neighbors (KNN), Support Vector Machines (SVMs), Naïve Bayes, Linear Discriminant Analysis (LDA), C5, XGBoost, Random Forest,

and ADA, offering a comprehensive analysis of classification performance. In contrast, [40] and [42] mostly depend on deep-learning models, such as deep forward neural networks (DFNNs) and modified feedforward neural networks (FFNNs). While deep learning models perform well on high-dimensional data, they demand much more computational resources and training time. Our approach balances accuracy and computational efficiency by combining classical machine learning and ensemble methods, making it more suitable for real-time applications.

Table 9. Comparison of our work and recent works with similar contributions.

Criteria	Our work	Behiry & Aly [40]	Hossain et al. [41]	Vallabhaneni et al. [42]
Dataset Used	Kitsune-ARP, SNMP-MIB, CSE- CIC-IDS2018, BoTIoT, UNR-IDD, Credit Card Fraud	NSL-KDD, UNSW-NB15, CICIDS2017	Not specified (DDoS-related)	BotNet dataset
Dataset Size	Multiple large-scale datasets (ranging from 2,620 to 2,426,574 records)	175,466 samples (CICIDS2017)	Not provided	1,803,333 domain names
Feature-reduction Method	Autoencoders + Stratified Sampling	Singular Value De-composition (SVD) + PCA + KMC-IG	Hybrid Feature Selection	Coot Optimization Algorithm (COA)
Sampling Method	Stratified Sampling	SMOTE + ENN	Not specified	Not specified
Machine-learning Model	KNN, SVM, Naive Bayes, Linear Regression, LDA, C5, XGBoost, Random Forest, ADA	Deep Forward Neural Network (DFNN) + K-means Clustering (KMC)	Ensemble-based classifier	Modified Feed-forward Neural Network (FFNN)
Performance Metrics	Accuracy up to 99% (varies by dataset), F1-score analysis for different reduction strategies	Accuracy: 99.7%, F1-score: 98.8% (NSL-KDD)	Not specified	Accuracy: 97.56%, Precision: 96.76%
Computational Efficiency	Training time reduced significantly by applying size and feature reduction techniques	High efficiency due to hybrid feature selection	Not specified	Improved by using COA for feature selection
Real-time Applicability	Yes, reduces dataset size while maintaining accuracy for efficient IDS deployment	Yes, suitable for real-time WSN intrusion detection	Yes, aimed at robust DDoS mitigation	Yes, designed for Cybersecurity-attack prediction
Novelty	Combination of autoencoder-based feature selection and stratified sampling for dataset reduction	Hybrid feature reduction (SVD+PCA + KMC-IG) + deep learning	Hybrid feature selection + ensemble classification	COA-based feature selection with adaptive weight FFNN
Limitations	Some models (e.g., Naive Bayes) perform poorly on highly reduced datasets	Requires large labeled datasets	Requires further evaluation in real-world scenarios	Computational complexity in feature selection and training

The performance metrics indicate that our method achieves an accuracy of up to 99% across multiple datasets, comparable to the 99.7% accuracy reported in [40]. However, the key advantage of our approach lies in its computational efficiency. By reducing the dataset size while maintaining classification performance, our method enables faster training times, making it highly scalable for real-time intrusion-detection systems. In contrast, with a computationally expensive feature selection process [42], it achieved a slightly lower accuracy of 97.56%. Using autoencoder-based feature-selection in our work ensures optimal feature retention with minimal processing overhead, achieving a balance between performance and efficiency.

Real-time applicability is a critical aspect of intrusion-detection systems. Our study prioritizes this using

efficient data-reduction techniques and lightweight machine-learning models. While [40] and [41] argue real-time relevancy, their studies lack detailed evaluations of computational efficiency. Our work explicitly shows that dataset-size reduction leads to significantly lower training times, confirming that the model remains deployable in practical cyber-security environments. The novelty of our work lies in the hybrid combination of autoencoder-based feature selection with stratified sampling, which optimizes both dataset size and model performance. Unlike previous studies that rely only on statistical reduction techniques or heuristic optimization, our approach integrates deep feature extraction and data-selection strategies. This hybrid approach results in an efficient intrusion-detection system capable of handling large-scale datasets while maintaining high detection accuracy.

Despite the benefits, there are areas for additional improvement. Some models, such as Naïve Bayes, exhibit performance degradation when involved with highly-reduced datasets, suggesting that feature-selection techniques could be further purified to improve compatibility with a more expansive range of classifiers. Additionally, estimating the trade-off between dataset reduction and accuracy loss under extreme conditions would provide further insights into the scalability of our approach. Expanding the study to real-world cyber-security attack scenarios would further validate its functional applicability.

5. Conclusion and Future Work

This study presents and tests two methods to reduce the amount of data used to train and test IDSs. The first method depends on reducing the size of the datasets with very large tuples, followed by feature selection to improve the ML model's performance. The second method, which is more practical with relatively small datasets, aimed to select the most important features first and then reduce the number of used tuples; this method guarantees the selection of better features and also improves the ML-model performance. This emphasizes the redundancy happening in some datasets related to security attacks in IoT datasets, especially simulated datasets.

This study shows that careful dataset size and feature-dimensionality reduction can lower computational costs while maintaining equivalent intrusion-detection performance. Specifically, using only 25% of the original data or feature set resulted in a less than 2% reduction in F-score for most models and datasets. Even with a large reduction to 10%, the average F1-score declined by only 4%–6%, with ensemble models, such as XGBoost and Random Forest, showing more resilience compared to other simple classifiers, like KNN. The reduction framework is computationally efficient and robust across various IDS scenarios. Meanwhile, data-reduction processes should be taken with caution, because random or extreme data reduction might cause the models to produce unacceptable results, as seen in many scenarios throughout this study.

In the future, we plan to repeat the experiment with multi-class labeled datasets and check how the proposed reduction techniques would affect the ML models. We also wish to investigate and compare other multiple reduction techniques. Our plan also includes applying the reduction techniques to different convolutional neural-network architectures and employing XAI tools to explore the reasons behind feature-ranking results. It is also necessary to have methods to evaluate the redundancy level in a dataset to estimate the possible efficient reduction percentages that can be applied to the data.

References

[1] M. B. Younes and A. Boukerche, "A Performance Evaluation of a Context-aware Path Recommendation Protocol for Vehicular Ad-hoc Networks," Proc. of the 2013 IEEE Global Communications Conf. (GLOBE- COM), IEEE, pp. 516–521, Atlanta, USA, 2013.

[2] M. B. Younes, G. R. Alonso and A. Boukerche, "A Distributed Infrastructure-based Congestion Avoidance Protocol for Vehicular Ad Hoc Networks," Proc. of the 2012 IEEE Global Communications Conf. (GLOBECOM), pp. 73–78, Anaheim, USA, 2012.

[3] J. Al-Sawwa, M. Almseidin, M. Alkasassbeh, K. Alemerien and R. Younisse, "Spark-based Multi-verse Optimizer as Wrapper Features Selection Algorithm for Phishing Attack Challenge," Cluster Computing, vol. 27, no. 5, pp. 5799–5814, 2024.

[4] L. A. C. Ahakonye et al., "SCADA Intrusion Detection Scheme Exploiting the Fusion of Modified Decision Tree and Chi-square Feature Selection," Internet of Things, vol. 21, p. 100676, 2023.

[5] Y. Han, Y. Zhang and J. Wang, "Semantic-driven Dimension Reduction for Wireless Internet of Things," Internet of Things, vol. 25, p. 101138, 2024.

[6] F. Ali et al., "An Intelligent Healthcare Monitoring Framework Using Wearable Sensors and Social Networking Data," Future Generation Computer Systems, vol. 114, pp. 23–43, 2021.
[7] A. Shiravani, M. H. Sadreddini and H. N. Nahook, "Network Intrusion Detection Using Data Dimensions Reduction Techniques," Journal of Big Data, vol. 10, no. 1, p. 27, 2023.
[8] R. Younisse and M. AlKasassbeh, "SGID: A Semi-synthetic Dataset for Injection Attacks in Smart Grid Systems," Proc. of the 2024 15th IEEE Int. Conf. on Information and Communication Systems (ICICS), pp. 1–4, Irbid, Jordan, 2024.
[9] A. Glielmo, B. E. Husic, A. Rodriguez, C. Clementi, F. Noé and A. Laio, "Unsupervised Learning Methods for Molecular Simulation Data," Chemical Reviews, vol. 121, no. 16, pp. 9722–9758, 2021.
[10] B. M. S. Hasan and A. M. Abdulazeez, "A Review of Principal Component Analysis Algorithm for Dimensionality Reduction," Journal of Soft Computing and Data Mining, vol. 2, no. 1, pp. 20–30, 2021.
[11] S. Li, N. Marsaglia et al., "Data Reduction Techniques for Simulation, Visualization and Data Analysis," Computer Graphics Forum, vol. 37, pp. 422–447, Wiley Online Library, 2018.
[12] M. Dumelle, T. Kincaid, A. R. Olsen and M. Weber, "Spsurvey: Spatial sampling design and analysis in R," Journal of Statistical Software, vol. 105, no. 3, pp. 1–29, 2023.
[13] G. Sharma, "Pros and Cons of Different Sampling Techniques," International Journal of Applied Research, vol. 3, no. 7, pp. 749–752, 2017.
[14] Z. Ashi, L. Aburashed, M. Al-Qudah and A. Qusef, "Network Intrusion Detection Systems Using Supervised Machine Learning Classification and Dimensionality Reduction Techniques: A Systematic Review," Jordanian J. of Computers and Inform. Technol. (JJCIT), vol. 7, no. 4, pp. 373 – 390, 2021.
[15] N. Saran and N. Kesswani, "Intrusion Detection System for Internet of Medical Things Using GRU with Attention Mechanism-based Hybrid Deep Learning," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 11, no. 2, pp. 136-150, 2015.
[16] Y. Xiao, C. Xing, T. Zhang and Z. Zhao, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," IEEE Access, vol. 7, pp. 42210–42219, 2019.
[17] F. Salo, A. B. Nassif and A. Essex, "Dimensionality Reduction with IG-PCA and Ensemble Classifier for Network Intrusion Detection," Computer Networks, vol. 148, pp. 164–175, 2019.
[18] R. Abdulhammed et al., "Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection," Electronics, vol. 8, no. 3, p. 322, 2019.
[19] S. Ryu et al., "Convolutional Autoencoder Based Feature Extraction and Clustering for Customer Load Analysis," IEEE Trans. on Power Systems, vol. 35, no. 2, pp. 1048–1060, 2019.
[20] G. T. Reddy et al., "Analysis of Dimensionality Reduction Techniques on Big Data," IEEE Access, vol. 8, pp. 54776–54788, 2020.
[21] K. K. Pandey and D. Shukla, "Stratified Linear Systematic Sampling Based Clustering Approach for Detection of Financial Risk Group by Mining of Big Data," Int. J. of System Assurance Engineering and Management, vol. 13, pp. 1239–1253, 2021.
[22] K. Zhang et al., "History Matching of Naturally Fractured Reservoirs Using a Deep Sparse Autoencoder," SPE Journal, vol. 26, no. 4, pp. 1700– 1721, 2021.
[23] B. Manjunatha et al., "A Network Intrusion Detection Framework on Sparse Deep Denoising Autoencoder for Dimensionality Reduction," Soft Computing, vol. 28, no. 5, pp. 4503–4517, 2024.
[24] F. Nabi and X. Zhou, "Enhancing Intrusion Detection Systems through Dimensionality Reduction: A Comparative Study of Machine Learning Techniques for Cyber Security," Cyber Security and Applications, vol. 2, p. 100033, 2024.
[25] K. K. Pandey and D. Shukla, "Stratified Sampling-based Data Reduction and Categorization Model for Big Data Mining," Proc. of Communication and Intelligent Systems (ICCIS 2019), pp. 107–122, Springer, 2020.
[26] X. Zhao, J. Liang and C. Dang, "A Stratified Sampling Based Clustering Algorithm for Large-scale Data," Knowledge-based Systems, vol. 163, pp. 416–428, 2019.
[27] Y. Yang, J. Cai, H. Yang, Y. Li and X. Zhao, "ISBFK-means: A New Clustering Algorithm Based on Influence Space," Expert Systems with Applications, vol. 201, p. 117018, 2022.
[28] L. Cao and H. Shen, "CSS: Handling Imbalanced Data by Improved Clustering with Stratified Sampling," Concurrency and Computation: Practice and Experience, vol. 34, no. 2, p. e6071, 2022.
[29] A. Zoubir and B. Missaoui, "Graph Neural Networks with Scattering Transform for Network Anomaly Detection," Engineering Applications of Artificial Intelligence, vol. 150, p. 110546, 2025.
[30] A. Zoubir and B. Missaoui, "GeoScatt-GNN: A Geometric Scattering Transform-based Graph Neural Network Model for Ames Mutagenicity Prediction," arXiv preprint, arXiv: 2411.15331, 2024.
[31] M. Alqarqaz, M. Bani Younes and R. Qaddoura, "An Object Classification Approach for Autonomous Vehicles Using Machine Learning Techniques," World Electric Vehicle J., vol. 14, no. 2, p. 41, 2023.
[32] Y. Mirsky, T. Doitshman, Y. Elovici and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," arXiv preprint, arXiv: 1802.09089, 2018.
[33] M. Al-Kasassbeh et al., "Towards Generating Realistic SNMP-MIB Dataset for Network Anomaly Detection," Int. J. of Computer Science and Information Security, vol. 14, no. 9, p. 1162, 2016.

[34] UNB, "CSE-CIC-IDS2018 on AWS," [Online], Available: http://www.unb.ca/cic/datasets/ids-2018.html, Accessed on Apr. 25, 2023, 2018.

[35] N. Koroniotis et al., "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset," Future Generation Computer Systems, vol. 100, pp. 779–796, 2019.

[36] T. Das et al., "UNR-IDD: Intrusion Detection Dataset Using Network Port Statistics," Proc. of the 2023 IEEE 20th Consumer Comm. & Networking Conf. (CCNC), pp. 497–500, Las Vegas, USA, 2023.

[37] Kaggle, "Credit Card Fraud Detection," [Online], Available: www.kaggle.com/datasets/mlg-ulb/creditcardfraud, Accessed: June 1, 2023.

[38] Y. N. Kunang et al., "Automatic Features Extraction Using Autoencoder in Intrusion Detection System," Proc. of the 2018 Int. Conf. on Electrical Engineering and Computer Science (ICECOS), pp. 219–224, Pangkal, Indonesia, 2018.

[39] Z. Salah et al., "Optimizing Intrusion Detection in 5G Networks Using Dimensionality Reduction Techniques," Int. J. of Electrical & Computer Engineering, vol. 14, no. 5, pp. 2088-8708, 2024.

[40] M. H. Behiry and M. Aly, "Cyberattack Detection in Wireless Sensor Networks Using a Hybrid Feature Reduction Technique with AI and Machine Learning Methods," J. of Big Data, vol. 11, no. 1, 2024.

[41] M. A. Hossain and M. S. Islam, "Enhancing DDoS Attack Detection with Hybrid Feature Selection and Ensemble-based Classifier: A Promising Solution for Robust Cybersecurity," Measurement: Sensors, vol. 32, p. 101037, 2024.

[42] R. Vallabhaneni et al., "Feature Selection Using COA with Modified Feedforward Neural Network for Prediction of Attacks in Cyber-security," Proc. of ICDCOT, pp. 1–6, DOI: 10.1109/ICDCOT61034, 2024.10516044, 2024.

ملخص البحث:

يُعدّ كشف الاختراقات في بيئات إنترنت الأشياء أمراً أساسياً لضمان أمان شبكات الحاسوب. وتستخدم نماذج التّعلُّم الآلي على نطاقٍ واسعٍ لتحسين أنظمةٍ فعّالة للكشف عن الاختراقات. ومع التّزايد السّريع في تعقيد وحجم البيانات في أنظمة الكشف عن الاختراقات، فإنّ تحليل مجموعات بياناتٍ ضخمة باستخدام نماذج التّعلُّم الآلي باتَ ينطوي على المزيد من التّحدّيات والمتطلّبات المتعلّقة بمصادر الحوسبة. وتأتي مجموعات البيانات المتعلّقة ببيئات إنترنت الأشياء بأحجام ضخمة. وتبحث هذه الدّراسة في أثر تقنيات تقليل البيانات في مجموعات البيانات في فاعلية وأداء أنظمة الكشف عن الاختراقات.

نقترح في هذه الورقة إطار عملٍ ثنائي المراحل يدمج بين تقليل السِّمات وتقليل الأبعاد والحجم في مجموعات البيانات المتعلّقة ببيئات إنترنت الأشياء، وذلك على ستّ مجموعات بياناتٍ مُتاحة للعموم. وتمّ تقييم أداء عددٍ من نماذج التّعلُّم الآلي على مجموعات البيانات المدروسة. وتوضّح النّتائج الّتي حصلنا عليها أنّ تقليل البيانات من شأنه أن يؤدّي إلى تقليل زمن التّدريب بما يصل إلى 99% مع فقدانٍ هامشي في مؤشّرات الأداء لا يتجاوز 1%، وقد تبين أنّ التّقليل الزّائد للبيانات قد يؤثر سلباً في دقّة الكشف. وتسلط الدّراسة الضّوء على فوائد تقليل البيانات في مجموعات بيانات إنترنت الأشياء، وتدعم نّتائج الدّراسة جدوى تطبيقات الكشف الفعّال عن الاختراقات لبيئات العالم الحقيقي، وخاصّةً في الأوضاع الّتي تتّسم بمحدودية الموارد أو الّتي تتعلّق بالزّمن الحقيقي.