Feature Selection Enhancement to Evaluate Attack Detection in the Internet of Things Environment

Khawlah Harahsheh *, Rami Al-Naimat, Chung-Hao Chen

Posted Date: 5 March 2024

doi: 10.20944/preprints202403.0211.v1

Keywords: Machine Learning; Feature Selection; Intrusion Detection System; IoT; Cybersecurity; Wireless Security; SDN

 doi:10.20944/preprints202403.0211.v1

Article

Feature Selection Enhancement to Evaluate Attack Detection in the Internet of Things Environment

Khawlah Harahsheh [1], Rami Al-Naimat [2] and Chung-Hao Chen [3]

[1] Ph.D. Student, Department of Electrical & Computer Engineering, Old Dominion University, Norfolk, VA, 23529 USA; khara001@odu.edu
[2] Karak, Jordan; ramiplan@yahoo.com
[3] Department of Electrical & Computer Engineering, Old Dominion University, Norfolk, VA, 23529 USA; CXCHEN@odu.edu
* Correspondence: Khawlah_moh@hotmail.com

Abstract: The rapid evolution of technology has given rise to a connected world, where billions of devices interact seamlessly, forming what is known as the Internet of Things (IoT). While IoT offers incredible convenience and efficiency, it presents a significant challenge in cybersecurity and is characterized by various power, capacity, and computational process limitations. Machine learning techniques are employed within Intrusion Detection Systems (IDS) to enhance their capabilities in identifying and responding to security threats. The key features selected to improve IDS efficiency and reduce dataset size, thereby decreasing the time required for attack detection, are drawn from the extensive network dataset. This paper introduces an enhanced feature selection method designed to reduce the computational overhead on IoT resources while simultaneously strengthening intrusion detection capabilities within the IoT environment. Experimental results based on the InSDN dataset demonstrate that our proposed methodology achieves the highest accuracy with the fewest number of features and low computational cost. Specifically, we attain a 99.99% accuracy with 11 features and a computational time of 0.8599 seconds.

Keywords: machine learning; feature selection; intrusion detection system; IoT; cybersecurity; wireless security; SDN

1. Introduction

As the Internet of Things (IoT) continues to proliferate, ensuring the security of interconnected devices and networks becomes increasingly crucial. A complete IoT system includes devices, sensors, networks, software, and other essential components necessary for operation and interconnection. Devices and sensors of this nature often have low resource requirements and multiple security vulnerabilities from manufacturers [1]. It is forecasted that by 2030, there will be approximately 500 billion IoT devices connected to the internet [2]. Edge devices, gateways, cloud servers, data-analysis tools, and user interfaces represent the main components of an IoT system. In addition to having sensors and controllers, edge devices can also perform initial data reviews before transmitting it to the cloud.

Cybersecurity is a critical concern in today's digital age due to the increasing number of cyber-attacks and the growing sophistication of cybercriminals. Ensuring the security of computer systems, networks, and data is of utmost importance, particularly for IoT devices. IoT environments are vulnerable to various cyber threats and attacks, making intrusion detection a fundamental aspect of IoT security. There are many security flaws in IoT devices, often originating from the manufacturer, along with insufficient defense against cyberattacks in the edge network areas. Additionally, the importance of the data collected by IoT devices contributes to the increase in cyberattacks in IoT systems.

 doi:10.20944/preprints202403.0211.v1

Traditional rule-based IDS have limitations in detecting emerging and unknown attacks. This has led to the integration of machine learning techniques into IDS, enabling them to adapt and evolve to ever-changing security threats. The complexity of high-dimensional features poses challenges for the speed and performance of intrusion detection classification [3]. An Intrusion Detection System is a security mechanism designed to monitor network traffic, detect suspicious activities, and alert system administrators of potential intrusions or security breaches. Analyzing data traffic patterns within a network enables intrusion detection systems (IDS) to detect potential attacks. Feature extraction becomes essential to reduce the computational burden associated with processing raw data in IDS.

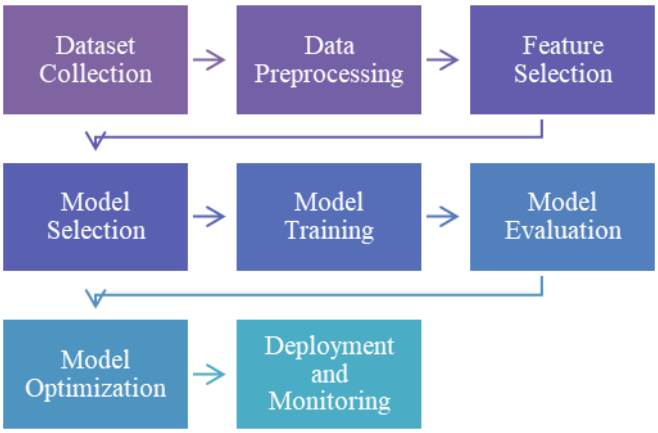

Figure 1. Intrusion Detection Steps Using Machine Learning.

It becomes imperative to identify and select the most relevant data features to enhance the performance of a machine learning model, particularly for IDS systems in IoT devices where securing these environments with conventional algorithms and techniques poses a challenge. IoT devices often have limitations such as low computing capabilities, limited power sources, restricted storage, or memory capacity. Traditional feature selection methods face optimization difficulties and have high computational complexity [2]. The limitations of IoT devices lead to a higher likelihood of attacks or flaws in their security protocols [4]. These limitations and challenges of IoT encompass various categories including hardware constraints, software issues, network concerns, and security vulnerabilities. It is crucial to thoroughly consider all these limitations and challenges before embarking on the development of any systems and security solutions.

Feature selection involves searching within the training data to enhance the classification accuracy of a machine learning classifier. It is crucial to identify and select the most relevant features in the data to improve the performance of the ML model, particularly in anomaly-based IDS [5]. This process entails selecting a subset of input features from the original set of features in the training data. Feature selection holds significant importance, especially as data scale increases, to improve the accuracy and efficiency of machine learning models and prevent overfitting [6]. One of the main challenges in ML is to identify the group of relevant features that strike a balance between high accuracy and low time complexity [7].

The contributions of this work are as follows:

- Innovative Approach to Feature Selection: Introduced novel feature selection techniques specifically tailored for Intrusion Detection Systems (IDS) in IoT environments. This approach not only significantly reduces the dimensionality of the dataset but also ensures the selection of the most relevant features, enhancing the overall efficiency and effectiveness of intrusion detection.
- Enhanced Intrusion Detection Performance: Demonstrated through rigorous experiments that our methodology outperforms existing methods in extracting optimal feature subsets. It achieves superior classification accuracy with fewer features, significantly reducing

 doi:10.20944/preprints202403.0211.v1

computational time and power consumption, thereby addressing the critical constraints of IoT-based IDS.

- Benchmarking Against Outdated Practices: Highlighted the limitations of using non-compatible and outdated datasets, like the KDD'99, in current IDS research. Our results underscore the importance of updated and relevant datasets for developing more accurate and efficient IDS solutions tailored to modern IoT ecosystems.

In this study, the latest InSDN dataset published in 2020 is used. The rest of this paper is organized as follows: Section 2 reviews related research on enhancing feature selection. Section 3 provides an overview of the proposed method, the dataset used, and the four stages for enhancing feature selection. Section 4 presents the experimental results and discussion, followed by section 5 which discusses future work and challenges. Finally, Section 6 concludes this paper.

2. Related Work

Many researchers are interested in securing IoT and wireless networks, and in this section, the focus is on the research that uses the InSDN dataset, and it compares their results with the results of this study. Numerous published studies rely on datasets that are either incompatible or outdated. For example, the KDDCup99 dataset, which was published in 1999, has been used in recent research [9,10]. Additionally, the Nsl-KDD dataset, published in 2009, has been used in research such as [3,5,11–14]. However, using outdated datasets in an Intrusion Detection System (IDS) for machine learning may not accurately reflect the current threat landscape. New types of attacks and vulnerabilities may have emerged since the dataset was created, making the model less effective in identifying contemporary threats. Outdated datasets can also introduce disadvantages such as limited relevance to current threats, obsolete features, lack of diversity, mismatch with real-world scenarios, and reduced model performance, which can lead to a false sense of security. Therefore, in this section, we discuss several methodologies proposed in recent research that utilize the InSDN dataset, which was published in 2020, such as [8,15– 20].

D. Firdaus et al. [17] proposed a DDoS detection method using Machine Learning with Ensemble Algorithm. Their study consists of two methodologies: clustering and classification, and Ensemble Algorithm clustering and classification. The researchers utilize Ensemble Algorithm K-means++ and Random Forest to achieve high detection accuracy and efficiency, obtaining a remarkable accuracy of 100% using 15 features out of the original dataset's 84. Conversely, other researchers in [8,19,20] also achieved very high accuracies of 99.42%, 99.9961%, and 99.94% respectively, but they employed 48 or 56 features. A. Ibrahimy et al. [19] used feature correlation to reduce the number of features in the InSDN dataset, while V. Hnamte and J. Hussain [20] employed two 1D convolutional layers to capture important features from the input data. These layers are connected to a flattened layer, which converts the output of the convolutional layers into a 1D array. Subsequently, four fully connected dense layers are utilized, activated by the rectified linear unit (ReLU) activation function.

The researchers in [15,16] achieved an accuracy of 98.98% with 30 features and 98% with 20 features respectively. A. Zainudin et al. [15] employed the LightGBM feature selection technique, which utilized three main modules: pre-block, depth-wiseConv, and stacked GRU. The depth-wiseConv module utilized a residual connection to enhance training model performance and address the issue of gradient vanishing. Additionally, a factorized convolution architecture was utilized to create a lightweight model structure.

3. Proposed Methodology

The number of features in an Intrusion Detection System (IDS) significantly influences its accuracy and performance. A larger feature set can provide a more detailed representation of the data, potentially leading to better detection of complex intrusion patterns. However, it can also introduce noise and irrelevant information, which can confuse the model, leading to overfitting and reduced generalization capabilities. Conversely, a smaller, well-curated feature set can result in a more streamlined and interpretable model, reducing the computational load and enabling the IDS to operate more efficiently, especially for IDS in IoT environments and resource limitations.

 doi:10.20944/preprints202403.0211.v1

In this paper, we present a sophisticated hybrid feature selection approach specifically devised for the nuanced requirements of IoT environments. The proposed methodology unfolds across several defined stages:

1. Initial Stage (Data Preprocessing).
2. Second Stage (Dimension Reduction).
3. Feature Selection Stage.
4. Final Stage (Model Evaluation).
5. Caching mechanism.

Each stage contributes to the overarching goal of selecting the most pertinent features for use in the IDS. The resulting features are of pivotal importance in machine learning, directly influencing the model's learning efficacy and predictive accuracy. Our methodological process, depicted in Figure 2, underscores the critical nature of feature selection in enhancing IDS performance within IoT contexts.

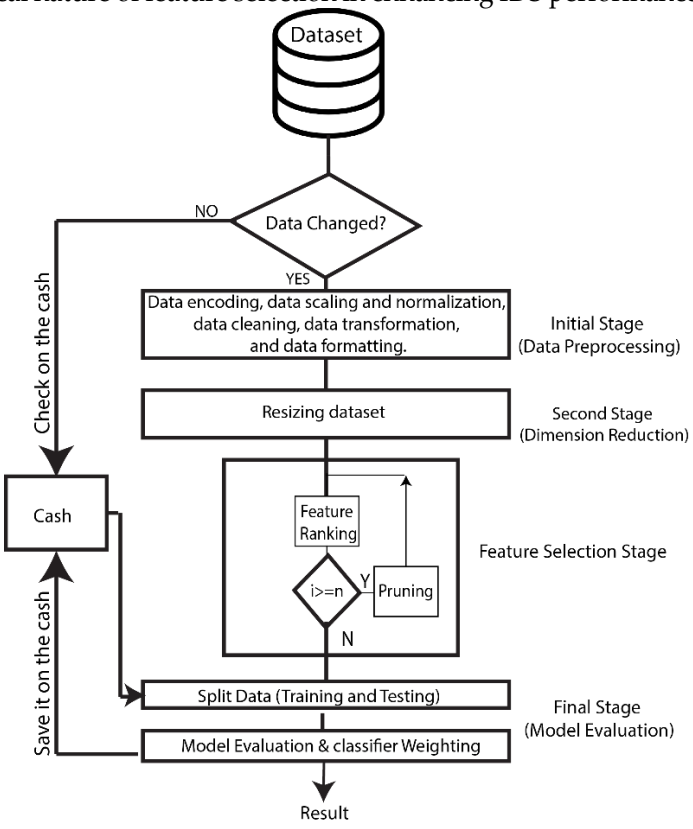

Figure 2. The whole process of the proposed method.

Incorporating a caching mechanism significantly improves the efficiency of our feature selection methodology. Throughout our proposed approach, the system meticulously identifies and caches the most significant features for subsequent iterations. After each iteration, the system checks the dataset's feature count. Unchanged counts prompt the retrieval of previously selected features from the cache, streamlining the process to the final evaluation stage. Conversely, any change in feature count mandates a fresh execution through all four stages, commencing with Data Preprocessing and culminating in Model Evaluation.

A. Initial Stage (Data Preprocessing).

The data processing stage—data encoding, scaling, normalization, cleaning, transformation, and formatting—is essential for preparing the raw dataset and creating a refined dataset where the intrinsic patterns can be more easily and accurately discerned by the IDS, leading to improved detection accuracy and performance. Data encoding converts categorical data into a numerical format, which is necessary because most machine learning algorithms can only interpret numeric values. This ensures that valuable categorical information, such as protocol types or service names, is retained and can be used in the detection process. Data scaling and normalization are critical steps

 doi:10.20944/preprints202403.0211.v1

that adjust the range of the feature data. Scaling alters the range of the data to a common scale without distorting differences in the ranges of values, while normalization adjusts the data in such a way that its distribution will have a mean value of 0 and a standard deviation of 1. These steps are crucial to prevent features with larger scales from dominating those with smaller scales, thus ensuring that the IDS model treats all features equally.

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset, which might otherwise lead to incorrect conclusions. Data transformation involves converting data into a suitable format or structure for analysis, which could include generating polynomial features or interaction terms if needed. Finally, data formatting ensures that all data is consistently and correctly formatted, such as ensuring data times are in a uniform format, which is fundamental for time-series analysis in intrusion detection. One major challenge in IDS is dealing with high-dimensional and imbalanced data, which increases the cost of machine learning [21]. Well-preprocessed data lowers the chance of overfitting, aids in the model's learning of pertinent patterns, and produces predictions that are more reliable and accurate.

B. Second Stage (Dimension Reduction).

The second stage is the dataset resizing stage, which is a crucial step in data preprocessing, there are various methods used for dimensionality reduction such as Principal Component Analysis (PCA) to suit different needs. Linear Discriminant Analysis (LDA) excels in classification tasks by using label information to maintain class distinctions. t-Distributed Stochastic Neighbor Embedding (t-SNE) shines in visualizing complex data but can be computationally demanding and sensitive to hyperparameters. Uniform Manifold Approximation and Projection (UMAP) offers a faster alternative to t-SNE, suitable for large datasets while preserving data structure. Autoencoders (AE), leveraging neural networks, are powerful for non-linear transformations but require substantial data and computational resources. Random Projection provides a speedy, simple approach though it may yield less precise results due to its stochastic nature. Finally, Isomap is effective for unfolding manifolds by maintaining geodesic distances, albeit at a higher computational cost. The choice among these techniques hinges on the specific dataset and analytical objectives, balancing factors like visualization clarity, computational efficiency, and interpretability. In this stage, we use PCA because it gives us faster speed in data resizing compared with the other methods, while other methods gave us good results but it took more computational time.

After employing the dimension reduction algorithm in our proposed method, the preprocessed data will be fed into the Feature Selection Stage. Each line will undergo feature ranking and pruning after a specific number of iterations. The combination process may involve selecting the best features based on scoring metrics or aggregating results from different workers. The outcome of the Feature Selection stage will be cached, which offers a time-saving advantage for subsequent runs on similar datasets or during iterative model development.

C. Feature Selection Stage.

The act of choosing a subset of the most pertinent features (input variables or attributes) from the initial collection of features is known as feature selection in machine learning. Improving the machine learning model's efficiency, decreasing overfitting, and improving performance are the goals. Three primary approaches can be used to classify the many feature selection techniques into general categories:

- Filter Methods: Filter approaches don't need to train a machine learning model because they rely on statistical metrics. They evaluate each feature's importance separately, without reference to the learning algorithm. Common filter methods include Correlation-based feature selection, Chi-squared test, and Information gain and mutual information.
- Wrapper Methods: By training and evaluating machine learning models on various feature combinations, wrapper approaches assess feature subsets. Although they require more computing power than filter approaches, they may produce feature subsets that are better. Common wrapper methods include Forward selection, Backward elimination, and Recursive feature elimination (RFE).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2024 doi:10.20944/preprints202403.0211.v1

- Embedded Methods: Feature selection is a crucial step in the model training process that is carried out by embedded methods. Usually, they are employed with algorithms that facilitate feature selection by default. Common embedded methods include L1 Regularization (Lasso), Tree-based methods, and Feature importance from Support Vector Machines (SVMs).

In our proposed method, we utilize the filter method because we aim to enhance the speed of our IDS. Higher speed translates to lower power usage, reduced capacity requirements, and faster computational time. Filter methods are generally faster compared to wrapper methods. They preprocess the data independently of the learning algorithm. Common filter methods include mutual information, chi-squared test, and correlation-based feature selection. These methods can be highly efficient, particularly when dealing with high-dimensional data. They employ statistical or correlation-based techniques to rapidly evaluate the relevance of individual features without the need to train a machine learning model.

Common metrics for feature selection include mutual information, chi-squared, and correlation coefficients. In a study by researchers [23], both filter and wrapper methods were compared, and it was found that wrapper methods often require more computing resources but provide higher accuracy. On the other hand, filter methods require fewer resources but lack optimization capabilities. Filter methods are known for their ease of implementation and faster execution compared to wrapper and hybrid methods. However, wrapper methods outperform filter methods in terms of accuracy, albeit at the cost of slower execution [24]. Another study by Z. Zhanget. al [11] also compared these feature selection methods and found that wrapper methods are closely tied to the classifier, which uses the performance of the classifier as an objective function to evaluate the current feature subset. Wrapper methods train a new model for each feature subset, resulting in a relatively higher computational cost.

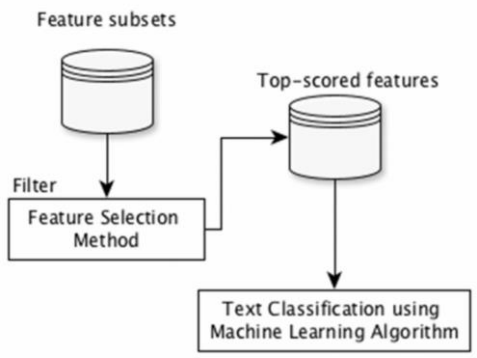

Figure 3. A schema of filter method [24].

Algorithm 1: Rank features using a Random Forest classifier pseudocode.

```
# Step 1: Initialize the VarianceThreshold object with a threshold value
threshold_value = 0.1
variance_selector = VarianceThreshold(threshold=threshold_value)
# Step 2: Fit the selector to your data
variance_selector.fit(X)
# Step 3: Get the indices of the features selected
selected_feature_indices = variance_selector.get_support(indices=True)
# Step 4: Use the selected features
selected_features = X[:, selected_feature_indices]
```

The provided Rank feature pseudo code outlines a procedure for ranking features using a combination of a variance threshold method and a Random Forest classifier. The pseudo-code stops short of detailing the application of the Random Forest classifier; however, in typical usage, the Random Forest would be employed post-variance threshold application to rank the features

 doi:10.20944/preprints202403.0211.v1

according to their importance. This importance is gleaned from how much the features contribute to the accuracy of the Random Forest model. By combining the variance threshold with Random Forest ranking, the algorithm aims to enhance the feature selection process, potentially improving the predictive power of the subsequent machine learning models and offering a deeper understanding of the features' influence within the dataset. The algorithm commences by initializing a VarianceThreshold object with a predefined threshold value, set here at 0.1. This threshold acts as a filter to remove features with low variance, under the assumption that features with a lower variance may contain less information. The variance_selector is then fitted to the dataset denoted as X, where it computes the variance for each feature.

Subsequently, the algorithm retrieves the indices of the features that surpass the variance threshold by invoking the get_support method with indices=True, which returns an array of indices. These indices represent the features that have been deemed important enough to retain based on their variance. In the final step, the algorithm selects the features from the dataset X using the obtained indices, creating a subset of X that only includes features with significant variance. This subset, denoted as selected_features, can be used for further processing or directly in the Random Forest classifier for ranking based on feature importance.

Use filter methods to select features based on their statistical properties. Common filter methods include:

- Correlation: Calculate the correlation between features and remove highly correlated or redundant ones.
- Variance Threshold: Remove features with low variance as they often contain little information.
- Statistical Tests: Utilize statistical tests (e.g., chi-squared, ANOVA) to select features that significantly contribute to the classification.

You can use several different methods and approaches to increase the feature selection process speed. Choosing features can be computationally costly, particularly when working with big datasets or intricate feature spaces. It is challenging for IoT characteristics to exploit the features and attributes of IDS to ensure self-protection [5]. Here are some methods used in the proposed method (Figure 2) to enhance the speed of your feature selection process:

- Feature Ranking: In this manner, you might not need to assess every feature in the dataset and instead concentrate on the most promising features first.
- Pruning: Early in the feature subset evaluation process, prune or remove any subsets that don't seem promising. This expedites the feature selection process and shrinks the search space. In the proposed methodology, feature ranking is performed in each round, while pruning is conducted every n number of iterations.
- Caching: Implementing a caching mechanism can significantly enhance the efficiency of our feature selection methodology. During our proposed method journey, the system meticulously identifies and selects the most significant features, which are then cached for subsequent iterations. The system checks the dataset's feature count after each iteration. If there is no change in the number of features, the system will swiftly retrieve the previously selected features from the cache, skipping unnecessary steps and advancing directly to the final evaluation stage. Conversely, any alteration in feature count necessitates a fresh execution through all four stages—beginning with Data Preprocessing and culminating in Model Evaluation.

D. Final Stage (Model Evaluation).

It is important to note that while these methods are generally faster, they may not always yield the best subset of features for a specific problem. The choice of feature selection method should consider factors such as the nature of the data, the complexity of the problem, the desired trade-off between speed and model performance, and the specific goals of the analysis. Sometimes, a slightly slower method may yield better results in terms of model accuracy and generalization.

Using Random Forest (RF) or L1 regularization (Lasso) can offer faster feature selection compared to other techniques. With Random Forest, you can assess feature importance, which is often faster than running more computationally intensive wrapper methods like recursive feature elimination with cross-validation (RFECV). Random Forest models can achieve high prediction

 doi:10.20944/preprints202403.0211.v1

accuracy with a small overhead in terms of computational resource usage [25]. In the proposed method, we use Random Forest as a supervised ML model for detecting attacks. Its advantages, such as handling both continuous and categorical data, addressing missing and outlier values, and shorter training time, led us to choose it as the classifier [26].

L1 regularization (Lasso) is commonly used in linear models to encourage sparsity in feature selection. It can be relatively fast, especially when combined with optimization techniques like coordinate descent.

4. Experimental Results and Discussion

This paper introduces an improved feature selection methodology aimed at achieving high accuracy while utilizing a reduced number of features. The focus on a lower feature count is particularly relevant for wireless networks, which often have resource constraints. In this section, we present a hybrid feature selection approach designed to enhance the performance of intrusion detection system (IDS) classification. We demonstrate the application of our methodology on the InSDN dataset. Software-defined networking (SDN) has been developed to reduce network complexity by centralizing control and management of the entire network [8]. SDN-based Industrial Internet of Things (IIoT) networks utilize a centralized controller, which can make them vulnerable to DoS/DDoS attacks and susceptible to single points of failure [18].

A. Dataset

The dataset used in this research is called the InSDN dataset, which was published in 2020 by M. Elsayd et al. [8]. The purpose of creating the InSDN dataset was to reduce its size compared to other IDS datasets. The researchers collected real-world network traffic from an SDN environment and classified it based on the presence of DDoS attacks. Unlike NSL-KDD and KDD99, InSDN provides a realistic representation of traffic in an SDN environment [17].

The dataset consists of a total of 343,889 data records with 84 features. Of these, 127,828 records correspond to ordinary traffic, while 216,061 records correspond to attack traffic. InSDN includes various attack scenarios, such as SYN floods and TCP, UDP, and ICMP floods, which are all included in the dataset. Table 1 presents the distribution of samples within the InSDN dataset.

Table 1. Distribution of the Samples Insdn Dataset [18].

No	Class	Description	Sample
1	Benign	Benign traffic is generated, which includes several well-established application services.	53,616
2	DDoS	Some DDoS attack scenarios, including ICMP flood, UDP flood, and TCP-SYN flood attacks, are considered in the InSDN dataset. Utilize the Hping3 tool to generate these DDoS attacks. The Hping3 tool runs on the virtual host using the Mininet SDN platform.	53,616
3	Probe	The main goal of a probing attack is to learn the target's information and supporting infrastructure, such as open ports, operating systems, databases, and web servers. An attacker may attempt to exploit known vulnerabilities, gain access to sample files, and utilize default user accounts if they are familiar with the specific type of web server (particularly for its default configuration).	53,616
4	DoS	This attack could rapidly exhaust the SDN controller's resources, exposing the entire system to inaccessibility by authorized users. A DoS attack can overwhelm the victim system with numerous fraudulent packets that lack matching rules in flow tables and switches. The two primary DoS attack types that typically impact an SDN architecture are network-based and application-based DoS attacks.	53,616
5	BFA	A brute-force attack is also known as a password-guessing attack. This attack involves compromising the username and password credentials to obtain access to the victim's server. This attack was launched to obtain the login and password credentials using the Burp suite and Hydra tools.	1,405
6	Web-attack	This dataset considers the most common web application attacks, including cross-site scripting (XSS) and SQL injection attacks. The XSS attack can circumvent the client machine's access protections by inserting malicious code into the trusted website. The SQL injection attack can provide the attacker access to the SQL database's contents by manipulating the database underlying the web application with malicious quarries.	192
		Total	**216,061**

The dataset includes a variety of attack scenarios, such as TCP, UDP, and ICMP floods, as well as SYN flood and Slowloris attacks. The InSDN dataset was generated using multiple virtual machines with an SDN network architecture. The standard Ubuntu system represents regular users, while the Kali system represents attackers performing various types of attacks on the SDN network [20].

B. Experimental Result and Analysis

 doi:10.20944/preprints202403.0211.v1

In this section, we present the outcomes of our extensive investigation into the effectiveness of the proposed methodology. Our experiments involve extracting crucial features from the InSDN dataset through the suggested stages. We evaluate the performance using classification matrices, which include metrics such as accuracy, loss, and AUC score.

In our proposed system, we have different stages: the Initial Stage (Data Preprocessing), Second Stage (Dimension Reduction), Feature Selection Stage, and Final Stage (Model Evaluation). After each stage, we preserved the dataset for size comparison, as illustrated in Figure 4. The original InSDN dataset consists of 84 features. However, after applying the dimension reduction algorithm in the second stage, the dataset is reduced to only 18 columns. The ultimate dataset is further refined, retaining the top 11 features. The most important features are shown in Figure 5.

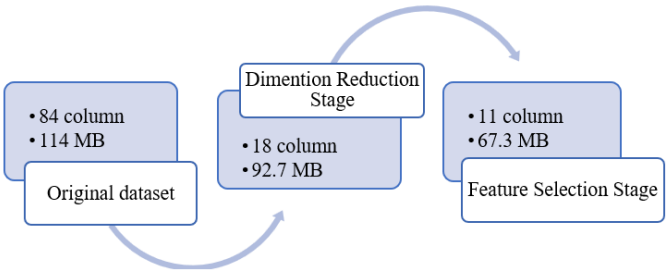

Figure 4. Saved data Size and number of features after each stage.

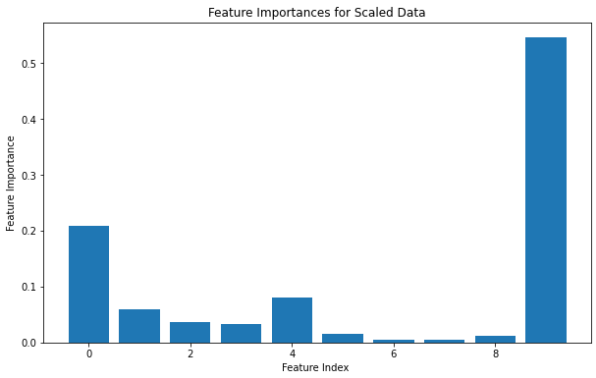

Figure 5. Top 8 important features of the InSDN dataset.

C. Evaluation Metrics

By selecting the most relevant features and eliminating irrelevant or redundant ones, you can enhance the performance of the IDS. This reduces computational complexity and improves its ability to accurately detect and classify intrusions. Metrics for evaluating Intrusion Detection Systems (IDS) are used to assess the performance of IDS solutions in detecting and preventing network or system intrusions and security breaches. These metrics help security professionals understand the effectiveness and efficiency of their IDS solutions. Here are some common IDS evaluation metrics:

- True Positive (TP): The number of correctly detected intrusions or attacks by the IDS.
- True Negative (TN): The number of correctly identified normal or non-malicious network activities by the IDS.
- False Positive (FP): The number of normal activities incorrectly classified as intrusions or attacks by the IDS. This is also known as a "false alarm."
- False Negative (FN): The number of intrusions or attacks that the IDS failed to detect or classify as normal. This is also known as a "missed detection."
- Accuracy: The ratio of correctly identified instances (TP + TN) to the total number of instances. It provides an overall measure of the IDS's correctness.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

 doi:10.20944/preprints202403.0211.v1

- Precision: Precision is defined as the ratio of true positives (TP) to the total number of instances classified as positive by the IDS (TP + FP). It measures the ability of the IDS to avoid false positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

- Recall (Sensitivity or True Positive Rate): The ratio of true positives (TP) to the total number of actual positive instances (TP + FN). It measures the IDS's ability to identify all positive instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \text{Recall}_{\text{Positive}} \quad (3)$$

- F1 Score: The harmonic mean of precision and recall. It balances the trade-off between false positives and false negatives [1].

$$\text{F1 Score} = 2 * \frac{\text{precision} * \text{Recall}}{\text{precision} + \text{Recall}} \quad (4)$$

The proposed model achieved excellent results with an accuracy of 99.99%, precision of 99.99%, recall of 99.99%, ROC score of 0.99997, and a computation cost of 0.8599 seconds when using 11 features. Figure 6 depicts the module results with actual versus predicted values.

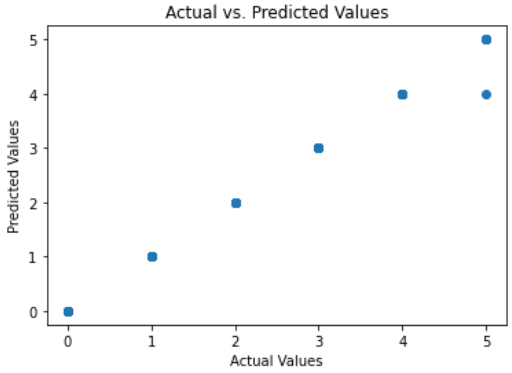

Figure 6. Actual VS Predicted Values.

Table 2 presents a comparison between our methodology and existing approaches. Our methodology achieves a remarkably high accuracy of 99.99% with a minimal number of features, using only 11. Additionally, we achieve a high computational speed of less than a second, specifically 0.8599 seconds. This makes our methodology suitable for wireless networks, particularly in limited resource environments such as IoT.

Table 2. Comparison Between Our Method and Other Research.

Ref #	ML Tech.	Accuracy	# of features	Time (second)
[8]	RF	99.42%	48	226.151
[15]	CNN	98.98%	30	0.164
[16]	RF	98%	20	20.50
[17]	KNN and RF	100%	15	1.5
[18]	CNN	90.83%	19	21.0506
[19]	KNN	99.9961%	56	18.69
[20]	DCNN	99.94%	-	-
Our paper	**Hybrid Feature Selection**	**99.99%**	**11**	**0.8599**

 doi:10.20944/preprints202403.0211.v1

5. Future Work and Challenges

Our methodology incorporates a caching mechanism to optimize the feature selection process. After each round, we check the number of columns in the dataset. If the number of columns remains constant, the methodology retrieves features from the cache, bypassing intermediary stages and sending them directly to the final stage. However, if there is a change in the column counts, the proposed approach proceeds through all four stages. It extracts the most influential features, sends them to the Model Evaluation stage, and stores them in the cache for future use.

In our future work, we aim to enhance our methodology by conducting a comprehensive examination of the data within these columns, rather than solely relying on the column count. This adjustment is necessary because there may be cases where the number of columns remains unchanged, but the content within those columns is entirely replaced. Additionally, the methodology might encounter a new dataset with a coincidental matching number of columns. Therefore, we plan to assess the data itself to ensure accurate feature selection.

Our next plan is to further improve the methodology by incorporating lightweight techniques. We intend to evaluate and test these enhancements on larger datasets to ensure their effectiveness and scalability. This way, we can address the power limitations of IoT devices while achieving optimal feature selection performance.

6. Conclusions

This paper introduces an advanced feature selection method to address the need for efficiency and reduced dataset size in IDS. The method combines multiple stages to optimize secure communication in wireless environments. The main objectives include improving model performance, reducing overfitting, and enhancing the efficiency and speed of the machine-learning model. The ultimate goal is to achieve high accuracy with a minimal number of features to make the IDS more compatible with IoT environments.

The proposed methodology is applied to the InSDN dataset, and experimental results demonstrate its effectiveness. The methodology successfully achieves the objective of high accuracy with a minimal feature set and low computational cost. Notably, the results show an impressive 99.99% accuracy using only 11 features, with a computational time of 0.8599 seconds. These findings highlight the potential of the methodology to advance secure communication in wireless IoT environments.

References

1. N. T. Cam and N. G. Trung, “An Intelligent Approach to Improving the Performance of Threat Detection in IoT,” in IEEE Access, vol. 11, pp. 44319-44334, 2023, doi: 10.1109/ACCESS.2023.3273160.
2. H. Chen, X. Ma and S. Huang, “A Feature Selection Method for Intrusion Detection Based on Parallel Sparrow Search Algorithm,” 2021 16th International Conference on Computer Science & Education (ICCSE), Lancaster, United Kingdom, 2021, pp. 685-690, doi: 10.1109/ICCSE51940.2021.9569597.
3. R. Zhao, Y. Mu, L. Zou, and X. Wen, “A Hybrid Intrusion Detection System Based on Feature Selection and Weighted Stacking Classifier,” in IEEE Access, vol. 10, pp. 71414-71426, 2022, doi: 10.1109/ACCESS.2022.3186975.
4. Harahsheh, K. M., & Chen, C. H. (2023). A Survey of Using Machine Learning in IoT Security and the Challenges Faced by Researchers. Informatica, 47(6).
5. B. Natarajan, S. Bose, N. Maheswaran, G. Logeswari and T. Anitha, “A New High-Performance Feature Selection Method for Machine Learning-Based IOT Intrusion Detection,” 2023 12th International Conference on Advanced Computing (ICoAC), Chennai, India, 2023, pp. 1-8, doi: 10.1109/ICoAC59537.2023.10249916.
6. A. M. Yesaswini and K. Annapurna, “A Hybrid Approach for Intrusion Detection System to Enhance Feature Selection,” 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), Trichy, India, 2023, pp. 1301-1307, doi: 10.1109/ICAISS58487.2023.10250515.
7. N. Abbas, Y. Nasser, M. Shehab and S. Sharafeddine, “Attack-Specific Feature Selection for Anomaly Detection in Software-Defined Networks,” 2021 3rd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), Agadir, Morocco, 2021, pp. 142-146, doi: 10.1109/MENACOMM50742.2021.9678279.

8. M. S. Elsayed, N. -A. Le-Khac and A. D. Jurcut, "InSDN: A Novel SDN Intrusion Dataset," in IEEE Access, vol. 8, pp. 165263-165284, 2020, doi: 10.1109/ACCESS.2020.3022633.
9. Y. Li, K. Shi, F. Qiao, and H. Luo, "A Feature Subset Selection Method Based on the Combination of PCA and Improved GA," 2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), Taiyuan, China, 2020, pp. 191-194, doi: 10.1109/MLBDBI51377.2020.00042.
10. D. P. Hostiadi, Y. P. Atmojo, R. R. Huizen, I. M. D. Susila, G. A. Pradipta and I. M. Liandana, "A New Approach Feature Selection for Intrusion Detection System Using Correlation Analysis," 2022 4th International Conference on Cybernetics and Intelligent System (ICORIS), Prapat, Indonesia, 2022, pp. 1-6, doi: 10.1109/ICORIS56080.2022.10031468.
11. Z. Zhang, J. Wen, J. Zhang, X. Cai, and L. Xie, "A Many Objective-Based Feature Selection Model for Anomaly Detection in Cloud Environment," in IEEE Access, vol. 8, pp. 60218-60231, 2020, doi: 10.1109/ACCESS.2020.2981373.
12. Yu, T., Liu, Z., Liu, Y., Wang, H., & Adilov, N. (2020, November). A New Feature Selection Method for Intrusion Detection System Dataset–TSDR method. In 2020 16th International Conference on Computational Intelligence and Security (CIS) (pp. 362-365). IEEE.
13. G. Parimala and R. Kayalvizhi, "An Effective Intrusion Detection System for Securing IoT Using Feature Selection and Deep Learning," 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2021, pp. 1-4, doi: 10.1109/ICCCI50826.2021.9402562.
14. S. Sarvari, N. F. Mohd Sani, Z. Mohd Hanapi and M. T. Abdullah, "An Efficient Anomaly Intrusion Detection Method With Feature Selection and Evolutionary Neural Network," in IEEE Access, vol. 8, pp. 70651-70663, 2020, doi: 10.1109/ACCESS.2020.2986217.
15. A. Zainudin, R. Akter, D. -S. Kim and J. -M. Lee, "Towards Lightweight Intrusion Identification in SDN-based Industrial Cyber-Physical Systems," 2022 27th Asia Pacific Conference on Communications (APCC), Jeju Island, Korea, Republic of, 2022, pp. 610-614, doi: 10.1109/APCC55198.2022.9943641.
16. M. N. Yilmaz and B. Bardak, "An Explainable Anomaly Detection Benchmark of Gradient Boosting Algorithms for Network Intrusion Detection Systems," 2022 Innovations in Intelligent Systems and Applications Conference (ASYU), Antalya, Turkey, 2022, pp. 1-6, doi: 10.1109/ASYU56188.2022.9925451.
17. D. Firdaus, R. Munadi and Y. Purwanto, "DDoS Attack Detection in Software Defined Network using Ensemble K-means++ and Random Forest," 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2020, pp. 164-169, doi: 10.1109/ISRITI51436.2020.9315521.
18. A. Zainudin, R. Akter, D. -S. Kim and J. -M. Lee, "Federated Learning Inspired Low-Complexity Intrusion Detection and Classification Technique for SDN-Based Industrial CPS," in IEEE Transactions on Network and Service Management, vol. 20, no. 3, pp. 2442-2459, Sept. 2023, doi: 10.1109/TNSM.2023.3299606.
19. A. Maulana Ibrahimy, F. Dewanta, and M. Erza Aminanto, "Lightweight Machine Learning Prediction Algorithm for Network Attack on Software Defined Network," 2022 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), Bandung, Indonesia, 2022, pp. 1-6, doi: 10.1109/APWiMob56856.2022.10014244.
20. V. Hnamte and J. Hussain, "Network Intrusion Detection using Deep Convolution Neural Network," 2023 4th International Conference for Emerging Technology (INCET), Belgaum, India, 2023, pp. 1-6, doi: 10.1109/INCET57972.2023.10170202.
21. G. Fu, B. Li, Y. Yang, and Q. Wei, "A Multi-Distance Ensemble and Feature Clustering Based Feature Selection Approach for Network Intrusion Detection," 2022 International Symposium on Sensing and Instrumentation in 5G and IoT Era (ISSI), Shanghai, China, 2022, pp. 160-164, doi: 10.1109/ISSI55442.2022.9963155.
22. T. -C. Vuong, H. Tran, M. X. Trang, V. -D. Ngo and T. V. Luong, "A Comparison of Feature Selection and Feature Extraction in Network Intrusion Detection Systems," 2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Chiang Mai, Thailand, 2022, pp. 1798-1804, doi: 10.23919/APSIPAASC55919.2022.9979923.
23. Raman, S. K. Jha and A. Arora, "An Enhanced Intrusion Detection System Using Combinational Feature Ranking and Machine Learning Algorithms," 2022 2nd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2022, pp. 1-8, doi: 10.1109/CONIT55038.2022.9847815.
24. P. H. Prastyo, I. Ardiyanto and R. Hidayat, "A Review of Feature Selection Techniques in Sentiment Analysis Using Filter, Wrapper, or Hybrid Methods," 2020 6th International Conference on Science and Technology (ICST), Yogyakarta, Indonesia, 2020, pp. 1-6, doi: 10.1109/ICST50505.2020.9732885.

 doi:10.20944/preprints202403.0211.v1

25. T. Bubolz, M. Grellert, B. Zatt, and G. Correa, "Coding Tree Early Termination for Fast HEVC Transrating Based on Random Forests," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 2019, pp. 1802-1806, doi: 10.1109/ICASSP.2019.8683833.
26. M. Bakro et al., "An Improved Design for a Cloud Intrusion Detection System Using Hybrid Features Selection Approach With ML Classifier," in IEEE Access, vol. 11, pp. 64228-64247, 2023, doi: 10.1109/ACCE