Optimizing Filter-Based Feature Selection Method Flow for Intrusion Detection System

Murtaza Ahmed Siddiqi and Wooguil Pak*

Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea; murtazasiddiqi@ynu.ac.kr
* Correspondence: wooguilpak@yu.ac.kr; Tel.: +82-53-810-3092

Received: 6 November 2020; Accepted: 7 December 2020; Published: 10 December 2020

Abstract: In recent times, with the advancement in technology and revolution in digital information, networks generate massive amounts of data. Due to the massive and rapid transmission of data, keeping up with security requirements is becoming more challenging. Machine learning (ML)-based intrusion detection systems (IDSs) are considered as one of the most suitable solutions for big data security. Despite the progress in ML, unrelated features can drastically influence the performance of an IDS. Feature selection plays a significant role in improving ML-based IDSs. However, the recent growth of dimensionality in data poses quite a challenge for current feature selection and extraction methods. Due to high data dimensionality, feature selection methods suffer in terms of efficiency and effectiveness. In this paper, we are introducing a new process flow for filter-based feature selection with the help of a transformation technique. Generally, normalization or transformation is implemented before classification. In our proposed model, we implemented and evaluated the effects of normalization before feature selection. To present a clear analysis on the effects of power transformation, five different transformations were implemented and evaluated. Furthermore, we implemented and compared different feature selection methods with the proposed process flow. Results show that compared with existing process flow and feature selection methods, our proposed process flow for feature selection can locate a more relevant set of features with high efficiency and accuracy.

Keywords: intrusion detection; security; feature selection; machine learning; deep learning; sequential forward selection; sequential backward selection; correlation

1. Introduction

In recent years, the use of smart devices has grown at a rapid rate. This increase has led to a significant upsurge of network traffic. With such expansion of data traffic, network security becomes quite problematic [1]. Along with the growing number and kinds of threats, security methods are also improving and evolving. Among these advanced and improving security methods, intrusion detection systems (IDSs) are quite prominent. At present, IDSs are considered a critical part of network security against both external and internal threats [2]. IDSs monitor the network and detect malicious events or any activity that violates defined rules. Generally, IDSs are classified into two main categories that are misuse-based and anomaly-based. A misuse-based IDS detects an attack or malicious activity based on known signatures while an anomaly-based IDS detects known and unknown attacks [3]. Understandably, these approaches have some advantages and disadvantages. Misuse-based detection is very accurate in detecting known attacks but is unable to detect unknown attacks. On the other hand, anomaly-based detection is very good at detecting unknown attacks but has concerns with detecting the difference between an attack and evolving normal traffic [4]. Regardless of the type of IDS, massive data traffic adversely affects the efficiency of the IDS. The reason behind this degraded performance
is irrelevant and concerns redundant information within network traffic [5]. In the past few years, a number of different methods were proposed to improve the efficiency of IDSs over big data [6].

Currently, a majority of researchers are focused on ML-based solutions for IDSs. IDSs based on ML provide high accuracy and low false detection as compared to other methods. ML-based IDSs consider network security as a classification problem to distinguish between normal and malicious traffic. The majority of traditional ML techniques are shallow learning methods [7]. Compared to traditional ML methods deep learning-based IDSs can attain excellent results. Despite a number of advanced approaches to improve ML-based IDSs, there are still numerous crucial limitations [8].

Feature selection (FS) plays an integral part in building an efficient and effective ML-based IDS. As FS removes irrelevant features, it enables the ML-based IDS to train faster with low computation and high accuracy [9]. Due to the density of data in today’s communication networks, providing an efficient and effective method for FS is essential for ML-based IDSs. Hence, this paper focuses on providing a detailed experimental finding of a filter-based FS method with optimized process flow. The proposed process flow achieved high accuracy by using a deep neural network (DNN)-based IDS. The key contributions of the paper are:

1. Evaluating various power transformation and FS methods to identify the most efficient combination.
2. Identifying the preprocessing flow to achieve highest accuracy.
3. Comparing the proposed flow of FS with other FS methods.
4. Comparing resources utilized by the proposed approach and other approaches.
5. Proposing a flow for filter-based feature selection using power transformation to achieve higher accuracy and efficiency.

2. Related Work

Over the years, a number of methods were recommended to improve ML-based IDSs. Among those approaches are feature selection (FS), feature engineering and optimization techniques. FS methods are employed to counter the high dimensionality in large datasets. As mentioned earlier that not all data in a network is relevant and irrelevant data could affect the IDS’s performance. In order to avoid the curse of high dimensionality, FS works by selecting the most relevant features from the dataset. The selected features are a subset of the original features and are used to build improved IDS models. FS methods help an IDS to improve its performance as well as reducing training time for its model [10]. FS techniques can be categorized as filter, wrapping- and hybrid-based approaches. Figure 1 gives a clear idea on how these methods work [11].

Each of these methods has its own advantages and disadvantages. Filter-based methods select features based on their score computed by different statistical approaches. Wrapper-based FS methods select features by iteratively generating and calculating the predictive performance of the features.
subset [12]. Hybrid or embedded feature selection combines the merits of filter and wrapping methods. Table 1 covers the advantages and disadvantages of the three feature selection methods [13,14].

Table 1. Advantages and disadvantages of the feature selection methods.

| Feature Selection     | Pro                                              | Con                                          | Examples                           |
|-----------------------|--------------------------------------------------|----------------------------------------------|------------------------------------|
| Filter                | Computationally efficient. Fast processing.      | Low precision, could fail to find features   | Pearson Correlation, Chi-Square, Anova etc. |
|                       | Less prone to overfitting                        | Computationally expensive, slow process due to recursion. Overfitting issue | Forward selection, Backward elimination, Recursion feature elimination etc. |
| Wrapper               | High precision                                   | Learning dependent                           | LASSO, Elastic Net etc.            |
| Hybrid/Embedded       | Low extra cost                                   |                                              |                                    |

It has been observed that training a ML-based IDS on a dataset and then deploying it on a real time network may result in degraded performance [15]. This degraded perform is due to the high number of irrelevant features. Removing those irrelevant features efficiently and with fewer resources can play a vital role for high data networks. Unfortunately managing efficient and effective FS is quite challenging.

ML algorithms can perform better when feature distributions are nearly normal and similarly scaled [16]. When it comes to the dataset, it is not necessary that every dataset have a normal feature distribution. However, with the help of power transformation we may be able to achieve normal distribution for dataset features. This normalization can significantly improve the performance of a number of different ML algorithms [17]. Some of the well-known transformation methods include Yeo-Johnson, minmax, robust scalar, standard scalar and L2 normalization.

The Yeo-Johnson transformation belongs to the Box-Cox transformation family. Yeo-Johnson was proposed in 2000 and unlike Box-Cox, it was not restricted to operate only on positive values. Mathematical representation of Yeo-Johnson can be seen as Equation (1), where \( y' \) represent the values, \( \lambda \) can be any real number, and where \( \lambda = 1 \) gives the identity transformation [18]:

\[
y_i^{(\lambda)} = \begin{cases} 
\frac{(y_i+1)^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0, \ y \geq 0 \\
\log(y_i + 1) & \text{if } \lambda = 0, \ y \geq 0 \\
\frac{[-(y_i+1)^{\lambda-1}]^{\lambda}}{(2-\lambda)} & \text{if } \lambda \neq 2, \ y < 0 \\
-\log(-y_i + 1) & \text{if } \lambda = 2, \ y < 0 
\end{cases}
\]

Min–max is a very useful method for data normalizing, especially with neural network-based classifiers [19]. Mathematically, min–max can be represented as Equation (2). In Equation (2), \( 'x' \) represents a single feature/variable vector:

\[
x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}
\]

Robust scalar uses a similar approach to that of min–max. Nevertheless, it uses the interquartile range instead of the min–max. Robust scalar, scales the data according to the quintile range. Robust can be represented as Equation (3). In the equation \( 'x' \) represent the values while \( Q_1 = 25\text{th quantile} \) and \( Q_3 = 75\text{th quantile} \) [20]:

\[
x = \frac{x - Q_1(x)}{Q_3(x) - Q_1(x)}
\]

Standard scalar works by removing the mean and scales the data to unit variance. Mathematically it can be represented as Equation (4), where \( 's' \) is the standard deviation and \( '\mu' \) is the mean [21]:

\[
x_{\text{scaler}} = \frac{x - \mu}{s_{\text{stddiv}}}
\]
L2 normalization works by normalizing the dataset feature values in a way that in each row the sum of the squares (of each value) will always add up to one. L2 can be represented as Equation (5) [22]. In the equation, \( x' \) represent the values of features in the dataset:

\[
\| x \|_2 = (|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2)^{\frac{1}{2}}
\] (5)

In this era of big data, massive amounts of information are transmitted every second, making real-time intrusion detection very challenging. ML, which is a widely used approach for IDSs, does not perform well with high dimensional data [23]. To improve ML-based IDSs, feature selection plays a key role. Feature selection enhances performance by decreasing training time and reducing model complexity [24]. Over the last few years, a number of feature selection methods were suggested for ML-based techniques. A paper by Yulianto et al. [25] aimed to improve ML-based IDSs using principal component analysis (PCA) and ensemble feature selection (EFC)-based FS. In the paper, the authors used AdaBoost as a classifier. The dataset used for the experiment was CIC-IDS 2017. Based on the results, both feature selection methods were not able to achieve a high accuracy rate. One of the reasons for not achieving high accuracy could be the use of the AdaBoost classifier. As AdaBoost is extremely sensitive to noisy data [26], proper pre-processing is paramount to achieve high performance. Compared to the work by Yulianto et al. [25], we provide a much-improved data pre-processing approach with a DNN-based classifier to achieve much better results. The pre-processing approach used in our paper can also assist other researchers in understanding and applying improved pre-processing steps. A similar paper by A. Razan et al. [23] used PCA and auto-encoder (AE)-based feature selection methods with different classifiers. The focus of the paper was to identify a combination of feature selection and classification to achieve high accuracy. As per results, authors achieved a good accuracy with a PCA and random forest (RF)-based model. Unfortunately, the authors used only one dataset i.e., CIC-IDS 2017 for the experimentation. This paper provides a comprehensive analysis of a performance comparison of different FS methods and classifiers. However, using only one dataset to identify the best combination of FS and classifier is not suitable. In our paper, we used two datasets to validate the proposed process flow for feature selection.

In another paper, authors Yuyang et al. [27] proposed a novel framework for IDSs. The proposed framework was a combination of feature selection and ensemble classifiers. The authors highlighted that feature selection can play a vital role in effective intrusion detection for ML-based IDSs. Based on the results, the performance of the proposed framework was highly accurate. Despite the high accuracy, the authors showed concerns of detecting attacks over massive data networks. In future work, the authors highlighted that they plan to improve detection of uncommon attacks from massive network traffic. On the other hand, our proposed IDS utilizes a DNN-based classifier that is more suitable for massive data. Another recent paper [28] used a pigeon inspired optimizer (PIO)-based feature selection approach. The PIO method is among the newly developed bio-inspired swarm algorithms for optimizing problems. In this paper, authors used PIO-based feature selection for IDSs and achieved promising results. For the experimentation, the authors used three different datasets with a decision tree (DT)-based classifier. Authors in paper [29] introduced a novel approach for feature selection. The proposed feature selection approach was based on the intelligence water drop (IWD) algorithm. IWD is a bio-inspired method that utilizes a support vector machine (SVM) for building a classifier. In order to evaluate the proposed method, the dataset KDD CUP99 was used. Based on the experiment results, the proposed method achieved an accuracy of 93.12% and a false detection rate of 3.35%. The proposed approach in this paper was novel and as per the authors, they plan to improve the algorithm in the future. As the core focus of the paper was to propose an efficient and effective feature selection method, the authors highlighted that further improvement could be achieved by selecting a more suitable classifier. The authors, Kurniabudi et al. [30], presented an idea in which they used information gain (IG) for FS. They utilized IG to calculate the weights of the features in the CIC-IDS 2017 dataset. Based on the weights, features were divided into six groups. Then the authors
used traditional ML algorithms to perform classification. They next analyzed the impact of FS on IDS precision and its effect on training time of a ML model. The authors concluded that FS can play a vital role in improving ML-based IDSs and how different methods can be used to improve filter-based FS techniques. Their conclusion was based on the experimental results they performed on all six groups of features created by IG. Compared to the work by Kurniabudi et al. [30], we proposed a more elaborate and general idea on how we can use transformation to improve filter-based FS on any dataset. We experimented with two well-known datasets to prove that the proposed idea is more broadly applicable. We also used a DNN-based classifier for improved intrusion detection on large datasets.

3. Methodology and Method

The core idea of our work is to identify an efficient and effective process flow for filter-based feature selection. Traditionally, features are selected after performing basic pre-processing. In our proposed process flow, we experimented by using power transformation before filter-based feature selection, as normalizing data before applying a statistical-based FS can improve the probability of selecting relevant features. Based on our experimentations, we were able to identify a process flow that resulted in improving the traditional process flow for feature selection. In our proposed process flow, we experimented with applying transformation before feature selection. We can see the difference between normal flow and proposed flow in Figure 2a,b. The experiments were performed on an Intel Xeon Gold 32 core (64 threads) processor with 192GB RAM and RTX2080ti GPU. All the experiments were implemented using Python 3.6 as the programming language on Ubuntu. DNN was implemented using GPU-enabled TensorFlow 2.3.1 on the Keras framework. Both Python and TensorFlow are open source software, which are available online for free download. The datasets used for the experiment were CIC-IDS 2017 [31] and ISCX-IDS 2012 [32]. In most ML methods, transformation is applied before classification is performed [7,33,34].

![Image of Figure 2](image-url)  
**Figure 2.** Experimental flow: (a) traditional flow; (b) proposed flow.

In our proposed method, we initially start by performing basic data cleaning and then pre-processing on the raw dataset. Then we applied power transformation on the dataset. After transformation, we implemented the feature selection method. Once we acquired the relevant features based on the feature selection method, we divided the dataset into test and train subsets. In order to avoid any biasness, we applied the synthetic minority over sampling technique (SMOTE) [35] and we performed cleaning using edited nearest neighbors (ENN) [36] on the training dataset. For classification, we used a DNN-based classifier. We experimented with a number of well-known transformations to identify the most effective transformation in terms of performance. In the subsequent sections, we discuss a detailed methodical process that we followed for our experimentation. In order to give readers a clear idea, results of each section are presented at the end of the section. The classifier for all of the experimentations was based on NN.

3.1. Data Prepressing

In our experiments, we used two datasets: CIC-IDS 2017 and ISCX-IDS 2012. In this section, we will cover the pre-processing applied on the datasets. First, we applied basic data cleaning on both datasets. Basic data cleaning includes the removal of missing data samples, infinite data values
and duplicate data samples from the dataset. Then we removed negative time-based values from the datasets. In order to avoid model biasness, we reduced the normal traffic samples. In datasets CIC-IDS 2017 and ISCX-IDS 2012 classes “BENIGN” and “Normal” had the highest number of samples. We extracted 230124 BENIGN samples from the CIC-IDS 2017 dataset and discarded the remaining BENIGN samples. For the ISCX-IDS 2012 dataset we extracted 25,845 Normal samples and discarded the remaining samples. In CIC-IDS 2017, the dataset classes “Infiltration” and “Heartbleed” were dropped as they had only 36 and 11 samples, respectively. All three classes of web attacks in the CIC-IDS 2017 dataset were combined into a single class named “Web attack”. Details of both datasets after preprocessing can be seen in Table 2.

| Table 2. Details of the CIC-IDS 2017 and ISCX-IDS 2012 datasets after pre-processing. |
|-------------------------------|-------------------------------|
| **CIC-IDS 2017** | **ISCX-IDS 2012** |
| Total Features | 79 | 82 |
| Total Classes | 11 | 5 |
| Total Samples | 786,633 | 73,566 |
| Benign | 230,124 | BruteForceSSH | 10,029 |
| Bot | 1956 | DDoS | 25,845 |
| DDoS | 128,025 | HTTPDoS | 3928 |
| DoS GoldenEye | 10,293 | Infiltration | 7919 |
| DoS Hulk | 230,124 | Normal | 25,845 |
| DoS Slowhttptest | 5499 |  |
| DoS slowloris | 5796 |  |
| FTP-Patator | 7935 |  |
| PortScan | 158,804 |  |
| SSH-Patator | 5897 |  |
| Web Attack | 2180 |  |

3.2. Transformation

After the basic data processing, we applied the Yeo–Johnson power transformation. Before selecting Yeo-Johnson, we experimented with Min–Max, robust scaler, standard scaler and L2 normalizing transformation techniques. Experiments were conducted with the proposed flow and standard flow. Yeo-Johnson with the proposed flow achieved the highest accuracy. Based on experiment results, Yeo-Johnson outperformed the other transformations as shown in Table 3. Experiments in Table 3 are based on the proposed flow, which is shown in Figure 1. A DNN-based classifier was used for the experimentation. Details of the DNN will be covered in the coming section. Results shown in Table 3 were acquired after the complete processing and classification.

| Table 3. Test results of the CIC-IDS 2017 and ISCX-IDS 2012 datasets based on transformations. |
|-------------------------------|-------------------------------|
| **Dataset** | **ISCX-IDS 2012** |
| Transformation | Min–Max | Robust | Standard | L2 | Yeo–Johnson |
| Accuracy | 94.35% | 90.41% | 94.46% | 92.44% | 95.20% |
| F1 Score | 92.34% | 86.35% | 92.30% | 89.07% | 93.17% |
| Precision | 92.52% | 84.60% | 91.68% | 88.31% | 92.86% |
| Recall | 92.24% | 89.89% | 93.16% | 90.30% | 93.51% |

| **Dataset** | **CIC-IDS 2017** |
|-------------------------------|-------------------------------|
| Transformation | Min–Max | Robust | Standard | L2 | Yeo–Johnson |
| Accuracy | 98.58% | 91.81% | 98.89% | 98.08% | 99.73% |
| F1 Score | 94.52% | 80.14% | 93.77% | 90.06% | 98.05% |
| Precision | 91.26% | 85.21% | 90.15% | 85.15% | 96.68% |
| Recall | 99.17% | 85.91% | 99.29% | 98.06% | 99.58% |
After transformation, we removed zero variance [37] features from the datasets. Then we dropped the feature Destination Port in the CICIDS2017 dataset and “Dst Port” in the ISCXIDS2012 dataset. In the datasets, the destination port feature is a port address and logically it is not able to distinguish between normal and attack traffic.

3.3. Feature Selection

Based on our rigorous experimentation we concluded that using Yeo-Johnson and Pearson correlation gave us the greatest accuracy. Pearson correlation [38] is an assessment standard that computes the statistical correlation between two variables. The mathematical formula to compute Pearson correlation is given by Equation (6):

\[ r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \]

where:

- \( r_{xy} \): Pearson correlation coefficient value,
- \( x_i \): Individual sample points of each conditional attribute,
- \( y_i \): Individual sample points of the decision attribute,
- \( \bar{x} \): Average of all sample points of each conditional attribute,
- \( \bar{y} \): Average of all sample points of the decision attribute.

To compare the performance of the proposed flow we used two well-known wrapper-based feature selection techniques, sequential forward selection (SFS) and sequential backward selection (SBS) [39]. We used SFS and SBS with the DT algorithm and traditional flow (Figure 3b). As we know that despite high computation requirements, wrapping-based feature selection methods give superior performance [40,41]. We also performed Pearson correlation experiments based on traditional flow (Figure 3b) so that we can cross check if the normal flow can perform better than the proposed flow. Based on the DNN classifier, Table 4 shows the results acquired with different FS methods.

| Dataset | ISCX-IDS 2012 | CIC-IDS 2017 |
|---------|---------------|--------------|
| Feature Selection | SFS-DT | SBS-DT | Correlation | Proposed |
| Accuracy | 95.14% | 95.04% | 95.15% | 95.20% |
| F1 Score | 92.54% | 92.50% | 92.88% | 93.17% |
| Precision | 92.21% | 91.86% | 93.00% | 92.86% |
| Recall | 92.95% | 93.22% | 92.86% | 93.51% |
| Dataset | ISCX-IDS 2012 | CIC-IDS 2017 |
| Feature Selection | SFS-DT | SBS-DT | Correlation | Proposed |
| Accuracy | 99.62% | 99.58% | 99.59% | 99.73% |
| F1 Score | 97.91% | 97.66% | 95.65% | 98.05% |
| Precision | 96.48% | 96.04% | 93.29% | 96.68% |
| Recall | 99.50% | 99.54% | 99.32% | 99.58% |

Based on the results shown in Table 4, the proposed flow performs better than the normal flow, SBS-DT and SFS-DT methods. This highlights that the proposed idea to use transformation before filter-based feature selection can improve accuracy of the model. Figure 3a,b shows the process flow for the proposed and wrapper-based feature selection models.
Figure 3. (a) Proposed flow for feature selection, and (b) traditional flow for feature selection.

Table 5 shows the total number of features and number of features selected by each method. Table 5 also highlights that using the power transformation before feature selection can improve the efficiency of feature selection.

Table 5. Original number of features and features selected by each feature selection method.

| Dataset       | Original Features | Correlation | SBS-DT | SFS-DT | Proposed |
|---------------|-------------------|-------------|--------|--------|----------|
| CIC-IDS 2017  | 79                | 45          | 34     | 24     | 35       |
| ISCX-IDS 2012 | 82                | 42          | 31     | 33     | 37       |

Table 6 shows the comparison of time and memory consumption of the proposed model with SBS and SFS feature selection. The standard flow and proposed flow show the same memory and time consumption, as the resources were computed only for the process of feature selection. Similarly, the resources for SBS-DT and SFS-DT were computed during the process of feature selection. Results show that the proposed method utilizes minimum resources yet achieves better results than wrapping-based feature selection methods.

Table 6. Memory and time consumption comparison between the proposed and other feature selection methods.

| Feature Selection | Correlation | SBS-DT   | SFS-DT   | Proposed |
|-------------------|-------------|----------|----------|----------|
| ISCX-IDS 2012 Dataset |             |          |          |          |
| Memory (mb)        | 405.41      | 7052.74  | 4442.63  | 405.41   |
| Time (s)           | 0.75        | 539.90   | 150.72   | 0.75     |
| CIC-IDS 2017 Dataset |             |          |          |          |
| Memory (mb)        | 5081.01     | 51,486.12| 25,023.36| 5081.01  |
| Time (s)           | 6.46        | 6667.02  | 1,954.71 | 6.46     |

3.4. Classifier

In our proposed model, we used a DNN as our main classifier, as a DNN is an effective approach for high dimensionality problems [42]. The efficiency of DNNs on high dimension problems is the reason that most of the recent IDSs rely on DNN-based solutions [43,44]. The parameters for the DNN were same for all the experimentations. The DNN used had four dense layers with 120 nodes in each layer; parameter details can be seen in Table 7.

Table 7. DNN parameters for experiments.

| Activation Function | Epochs | Learning Rate | Batch Size | Seed | Layers |
|---------------------|--------|---------------|------------|------|--------|
| Elu                 | 100    | 0.0001        | 100        | 42   | 4      |
4. Performance Evaluation

This section covers detailed experimentation results based on the process flows in Figure 3. Results of the proposed method can be seen in Figure 4. Results show that the proposed approach performs better than the standard process flow, SBS-DT and SFS-DT feature selection techniques. Tables 8 and 9 illustrate the comparative detection of each class based on the proposed method, SBS-DT, SFS-DT and standard feature selection flow. Accuracy, precision, recall and the F1-score were computed using Equations (7)–(10). Figure 4 was generated by using MSWord Chart and the values for the figure were obtained by using Equations (7)–(10):

\[
\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}} \tag{7}
\]

\[
\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{8}
\]

\[
\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{9}
\]

\[
\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}
\]

Based on extensive experimentation it was found that using the Yeo-Johnson transformation before applying the Pearson correlation feature selection provided more relevant features for IDSs. Two well-known datasets were used for the experimentations. To validate the findings, we compared our results with two wrapper-based feature selection methods i.e., SFS and SBS and the Pearson correlation with the standard process flow. Based on the results, the highest accuracy was archived by the proposed process flow. The reason for comparing the proposed model with wrapper-based feature selection was to show that the proposed model not only outperforms them in terms of detection, but also consumes fewer resources compared to the wrapping methods (Table 6).

![CIC-IDS 2017 Dataset](image)

(a)

Figure 4. Cont.
Figure 4. Test results based on each feature selection method: (a) results of the CIC-IDS 2017 dataset, and (b) results of the ISCX-IDS 2012 dataset.

Table 8 displays the comparison matrix of class-based detection for the ISCX-IDS 2012 dataset. This dataset has five classes. SFS-DT achieved the best F1-score for the BruteforceSSH, DDoS and Normal classes, while the proposed flow achieved the highest F1-score for the HTTPDoS and Infiltration classes. Despite the fact that the proposed flow attained the highest F1-score, some classes had a better-detected rate by other feature selection methods.

Table 8. Class detection based on each feature selection method for the ISCX-IDS 2012 dataset.

|                | BruteforceSSH | DDoS  | HTTPDoS | Infiltration | Normal |
|----------------|---------------|-------|---------|--------------|--------|
| **Accuracy**   |               |       |         |              |        |
| Standard       | 99.80%        | 97.56%| 98.04%  | 98.20%       | 96.55% |
| SFS-DT         | 99.89%        | 97.70%| 97.73%  | 98.15%       | 96.59% |
| SBS-DT         | 99.87%        | 97.76%| 98.09%  | 98.32%       | 96.57% |
| Proposed       | 99.78%        | 97.51%| 98.09%  | 98.32%       | 96.57% |
| **Precision**  |               |       |         |              |        |
| Standard       | 99.20%        | 97.90%| 84.55%  | 88.93%       | 94.46% |
| SFS-DT         | 99.80%        | 98.09%| 77.25%  | 91.34%       | 94.60% |
| SBS-DT         | 99.30%        | 98.18%| 78.54%  | 88.10%       | 95.20% |
| Proposed       | 99.15%        | 97.50%| 82.67%  | 90.02%       | 95.01% |
| **Recall**     |               |       |         |              |        |
| Standard       | 99.30%        | 95.14%| 78.29%  | 95.61%       | 95.99% |
| SFS-DT         | 99.40%        | 95.29%| 84.55%  | 88.61%       | 96.95% |
| SBS-DT         | 99.75%        | 94.95%| 79.95%  | 96.18%       | 95.28% |
| Proposed       | 99.20%        | 95.39%| 82.25%  | 95.35%       | 95.41% |
| **F1-Score**   |               |       |         |              |        |
| Standard       | 99.25%        | 96.50%| 81.30%  | 92.15%       | 95.22% |
| SFS-DT         | 99.60%        | 96.67%| 80.73%  | 89.95%       | 95.76% |
| SBS-DT         | 99.52%        | 96.54%| 79.24%  | 91.97%       | 95.24% |
| Proposed       | 99.17%        | 96.43%| 82.46%  | 92.61%       | 95.21% |

For the CIC-IDS 2017 dataset, Table 9 shows the comparison of the class-based detection rate for the standard, SFS-DT, SBS-DT and proposed feature selection methods. In our experimentation, we used 11 classes. Based on the results, it can be seen that the proposed flow provided a higher F1-score on Benign, Bot, DDoS, FTP-Patator, Portscan and SSH-Patator classes, while the standard flow, SFS-DT and SFS-DT were able to detect other classes with a higher F1-score. The confusion matrix for the CIC-IDS 2017 and ISCX-IDS 2012 datasets can be seen in Figures 5 and 6, respectively.
|                | Benign | Bot    | DDoS   | DoS Gold | DoS Hulk | DoS Slowlhttp | DoS SlowLoris | FTP Patator | Port Scan | SSH Patator | Web Attack |
|----------------|--------|--------|--------|----------|----------|---------------|---------------|-------------|-----------|-------------|-------------|
| **Accuracy**   |        |        |        |          |          |                |                |             |           |             |             |
| Standard       | 99.64% | 99.82% | 99.99% | 99.97%   | 99.92%   | 99.98%        | 99.99%        | 99.99%     | 99.99%    | 99.99%      | 99.91%      |
| SFS-DT         | 99.69% | 99.96% | 99.97% | 99.96%   | 99.80%   | 99.98%        | 99.99%        | 99.97%     | 99.99%    | 99.98%      | 99.96%      |
| SBS-DT         | 99.64% | 99.94% | 99.99% | 99.96%   | 99.78%   | 99.98%        | 99.99%        | 99.97%     | 99.99%    | 99.98%      | 99.96%      |
| Proposed       | 99.79% | 99.96% | 99.99% | 99.97%   | 99.89%   | 99.97%        | 99.99%        | 99.97%     | 99.99%    | 99.99%      | 99.95%      |
| **Precision**  |        |        |        |          |          |                |                |             |           |             |             |
| Standard       | 99.92% | 56.67% | 99.95% | 98.27%   | 99.82%   | 98.18%        | 99.09%        | 99.57%     | 99.98%    | 98.83%      | 75.96%      |
| SFS-DT         | 99.80% | 86.16% | 99.93% | 97.81%   | 99.58%   | 97.57%        | 98.82%        | 97.25%     | 99.98%    | 96.90%      | 87.55%      |
| SBS-DT         | 99.90% | 80.40% | 99.97% | 97.63%   | 99.40%   | 97.48%        | 98.74%        | 97.08%     | 99.99%    | 97.29%      | 88.64%      |
| Proposed       | 99.97% | 86.78% | 99.98% | 97.72%   | 99.72%   | 97.64%        | 98.83%        | 97.60%     | 100%      | 98.59%      | 86.73%      |
| **Recall**     |        |        |        |          |          |                |                |             |           |             |             |
| Standard       | 98.86% | 98.89% | 99.98% | 99.57%   | 99.91%   | 98.63%        | 99.18%        | 99.88%     | 99.97%    | 99.49%      | 98.20%      |
| SFS-DT         | 99.12% | 100%   | 99.91% | 99.62%   | 99.74%   | 99.08%        | 99.09%        | 99.94%     | 99.96%    | 99.92%      | 98.20%      |
| SBS-DT         | 98.85% | 100%   | 99.96% | 99.81%   | 99.86%   | 99.08%        | 99.27%        | 99.94%     | 99.94%    | 99.83%      | 98.42%      |
| Proposed       | 99.30% | 100%   | 99.97% | 99.76%   | 99.92%   | 98.72%        | 99.27%        | 99.94%     | 99.96%    | 100%        | 98.65%      |
| **F1-Score**   |        |        |        |          |          |                |                |             |           |             |             |
| Standard       | 99.38% | 72.05% | 99.97% | 98.92%   | 99.87%   | 98.40%        | 99.14%        | 99.72%     | 99.97%    | 99.16%      | 85.66%      |
| SFS-DT         | 99.46% | 92.56% | 99.92% | 98.71%   | 99.66%   | 98.32%        | 98.96%        | 98.58%     | 99.97%    | 98.38%      | 92.57%      |
| SBS-DT         | 99.37% | 89.14% | 99.96% | 98.71%   | 99.63%   | 98.27%        | 99.00%        | 98.49%     | 99.96%    | 98.55%      | 93.28%      |
| Proposed       | 99.63% | 92.92% | 99.98% | 98.73%   | 99.82%   | 98.18%        | 99.05%        | 98.76%     | 99.98%    | 99.29%      | 92.31%      |
Figure 5. Confusion matrix of the CIC-IDS 2017 dataset: (a) proposed process flow, (b) standard process flow, (c) SFS-DT-based feature selection, and (d) SBS-DT-based feature selection.

Figure 6. Cont.
Figure 6. Confusion matrix of the ISCX-IDS 2012 dataset: (a) proposed process flow, (b) standard process flow, (c) SFS-DT-based feature selection, and (d) SBS-DT-based feature selection.

The receiver operating characteristics (ROC) curve for both the CIC-IDS 2017 and ISCX-IDS 2012 datasets can be seen in Figures 7 and 8, respectively. The order of labels in both the ROC figures are the same as for the confusion matrix i.e., class 0, 1 for CIC-IDS 2017 are Benign, BOT and onwards, while class 0, 1 for ISCX-IDS 2012 are BruteforceSSH, DDoS and onwards.

Figure 7. Receiver operating characteristics (ROC) of the CIC-IDS 2017 dataset: (a) proposed process flow, (b) standard process flow, (c) SFS-DT-based feature selection, and (d) SBS-DT-based feature selection.
Another useful measure to further highlight the effectiveness and efficiency of the proposed model is Cohan’s kappa coefficient [45]. The kappa coefficient score is a very useful measure when using ML for multi-class classification problems [46]. Kappa is calculated using Equation (11).

$$Kappa = \frac{p_0 - p_e}{1 - p_e}$$  \hspace{1cm} (11)

where $p_0$ is the overall accuracy of the model and $p_e$ represents the agreement between the model estimates and the actual class values as if happening by chance.

The results for the kappa score can be seen in Table 10. The kappa score is always less or equal to 1. As per Landis and Koch’s [47] kappa value scheme, values between 0.81–1 represent almost perfect classification.

| Process Flow | Normal | SBS-DT | SFS-DT | Proposed |
|--------------|--------|--------|--------|----------|
| ISCX-IDS 2012 Dataset | 0.9326 | 0.9313 | 0.9325 | 0.9335 |
| CIC-IDS 2017 Dataset  | 0.9947 | 0.9943 | 0.9955 | 0.9965 |

Figure 8. ROC of the ISCX-IDS 2012 dataset: (a) proposed process flow, (b) standard process flow, (c) SFS-DT-based feature selection, and (d) SBS-DT-based feature selection.
Achieving high precision with fewer resources is the key idea behind this article. In this study, the new process flow for filter-based FS can play an integral role in improving ML-based IDSs, as irrelevant features can greatly impair the performance of ML-based IDSs. The experiment results also illuminate improved detection due to the feature set selected by the proposed method. FS is a discrete optimization issue. Each dataset has its own different distribution of feature values. However, with the proposed approach of utilizing transformation it can improve the efficiency of ML- and statistics-based FS methods. The proposed approach is not only limited to traditional networks; it can also be effective in improving the Internet of Things (IoT). One of the primary constraints in implementing security for IoT devices is their lack of resources [48]. As Table 6 highlights, the proposed process flow in not resource-hungry, while it achieves a high accuracy rate.

5. Conclusions

The growing rate of intrusions in networks has seriously affected the security and privacy of users. ML can play an integral part in providing improved security measures. Optimizing ML-based IDS approaches is an ongoing research struggle. Feature selection is a process that provides a minimal and relevant subset of features. Feature selection is a critical component in today’s era of big data. As the data and its transition rate grows, feature selection becomes more significant. In this paper, an optimized process flow for filter-based feature selection is proposed. The proposed process flow aimed to reduce computational requirements, while achieving high accuracy. The proposed method is compared with the traditional process flow and two wrapper-based feature selection methods. Based on the defined metrics for evaluation, results illustrate that the proposed process flow achieved highest accuracy with efficiency as compared to the three mentioned feature selection methods. The memory profile assessment results highlight that the proposed process flow not only reduced execution time but also reduced the amount of memory used for feature selection. For better analysis, the confusion matrices for each class in the datasets were also calculated and compared. Despite the highest accuracy, the proposed flow was not able to achieve the highest detection scores for all the classes. In some classes SBS-DT, SFS-DT and standard flow were able to achieve the highest accuracy. Additionally, the experimentation highlights that filter-based feature selection can be further improved using statistical methods. In the future, we intend to further investigate and improve the proposed flow in order to achieve better class-based detection.

Author Contributions: M.A.S. and W.P. have written this paper and have done the research. Both authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2019R1F1A1062320) and the ITRC (Information Technology Research Center) support program (IITP-2020-2016-0-00313) supervised by the IITP (Institute for Information and Communications Technology Planning and Evaluation).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Samira, S.; Mohd, S.N.F.; Mohd, H.Z.; Taufik, A.M. An Efficient Anomaly Intrusion Detection Method With Feature Selection and Evolutionary Neural Network. *IEEE Access* 2020, 8, 70651–70663.
2. Thi-Thu-Huong, L.; Yongsu, K.; Howon, K. Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks. *Appl. Sci.* 2019, 9, 1392.
3. Ullah, I.; Mahmoud, Q.H. A Filter-based Feature Selection Model for Anomaly-based Intrusion Detection Systems. In Proceedings of the 2017 IEEE International Conference on Big Data (BIGDATA), Boston, MA, USA, 11–14 December 2017.
4. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. *IEEE Commun. Surv. Tutor.* 2019, 21, 686–728. [CrossRef]
5. Papamartzivanos, D.; Marmol, F.G.; Kambourakis, G. Introducing deep learning self-adaptive misuse network intrusion detection system. *IEEE Access* 2019, 7, 13546–13560. [CrossRef]
6. Hua, Y. An Efficient Traffic Classification Scheme Using Embedded Feature Selection and lightGBM. In Proceedings of the 2020 Information Communication Technologies Conference (ICTC), Nanjing, China, 29–31 May 2020.

7. Yu, Y.; Bian, N. An Intrusion Detection Method Using Few-Shot Learning. IEEE Access 2020, 8, 49730–49740. [CrossRef]

8. Sultana, N.; Chilamkurthy, N.; Peng, W.; Alhadjad, R. Survey on SDN based network intrusion detection system using machine learning approaches. Peer-to-Peer Netw. Appl. 2019, 12, 493–501. [CrossRef]

9. Amir, D. Feature Selection: Beyond Feature Importance? Medium. 22 September 2017. Available online: https://medium.com/fiverr-engineering/feature-selection-beyond-feature-importance-9b97e5a842f (accessed on 11 September 2020).

10. Balasaraswathi, V.R.; Sugumaran, M.; Hamid, Y. Feature selection techniques for intrusion detection using non-bio-inspired and bioinspired optimization algorithms. J. Commun. Inf. Netw. 2017, 2, 107–119. [CrossRef]

11. Wei, H. Feature Selection Methods with Code Examples. Analytics Vidhya, 21 August 2019. Available online: https://medium.com/analytics-vidhya/feature-selection-methods-with-code-examples-a78439477cd4 (accessed on 19 October 2020).

12. Wang, Z.; Xiao, X.; Rajasekar, S. Novel and Efficient Randomized Algorithms for Feature Selection. Big Data Min. Anal. 2020, 3, 208–224. [CrossRef]

13. Luhaniwal, V. Feature selection using Wrapper methods in Python. Towards Data Science, 4 October 2019. Available online: https://towardsdatascience.com/feature-selection-using-wrapper-methods-in-python-f0d352b346f (accessed on 19 October 2020).

14. Suto, J.; Oniga, S.; Sitar, P.P. Comparison of wrapper and filter feature selection algorithms on human activity recognition. In Proceedings of the 2016 6th International Conference on Computers Communications and Informatics (ICCCI), Oradea, Romania, 10–14 May 2016.

15. Sahu, A.; Mao, Z.; Davis, K.; Goulart, A.E. Data Processing and Model Selection for Machine Learning-based Network Intrusion Detection. In Proceedings of the 2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR), Stevenson, WA, USA, 14 May 2020.

16. Jackson, S. Feature Transformation, Understanding When to Scale and Standardize Data for Machine Learning. 30 April 2019. Available online: https://medium.com/@sjacks/feature-transformation-21282d1a3215 (accessed on 21 October 2020).

17. Brownlee, J. How to Use Power Transforms for Machine Learning. 18 May 2020. Available online: https://machinelearningmastery.com/power-transforms-with-scikit-learn/ (accessed on 21 October 2020).

18. Weisberg, S. Yeo-Johnson Power Transformations; University of Minnesota, Statistics Arc: Minnesota, MN, USA, 2001.

19. Loukas, S. Everything You Need to Know about Min-Max Normalization: A Python tutorial. Towards Data Sciences. 28 May 2020. Available online: https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79 (accessed on 19 October 2020).

20. Reddy, K.M. Data Preprocessing Using Python Sklearn. Medium, 4 September 2018. Available online: https://medium.com/@kesarimohan87/data-preprocessing-6c87d27156 (accessed on 19 October 2020).

21. Brownlee, J. How to Use StandardScaler and MinMaxScaler Transforms in Python. Machine Learning Mastery, 10 June 2020. Available online: https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/ (accessed on 20 October 2020).

22. Karim, R. Intuitions on L1 and L2 Regularisation. Towards Data Sciences, 27 December 2018. Available online: https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261#f810 (accessed on 20 October 2020).

23. Abdulhammed, R.; Musafer, H.; Alessa, A.; Faezipour, M.; Abuzneid, A. Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection. Electronics 2019, 8, 322. [CrossRef]

24. Gul, A.; Adali, E. A feature selection Algorithm for IDS. In Proceedings of the 2nd International Conference on Computer Science and Engineering (UBMK 17), Antalya, Turkey, 5–8 October 2017.

25. Yulianto, A.; Sukarno, P.; Suwastika, N.A. Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 dataset. In Proceedings of the 2nd International Conference on Data and Information Science, Bandung, Indonesia, 15–16 November 2018.

26. Kurama, V. A Guide to AdaBoost: Boosting To Save The Day. Paperspace Blog, 23 February 2020. Available online: https://blog.paperspace.com/adaboost-optimizer/#-[]:text=AdaBoost%20is%20an%20ensemble%20learning,turn%20them%20into%20strong%20ones (accessed on 25 October 2020).
27. Zhou, Y.; Cheng, G.; Jiang, S.; Daia, M. Building an Efficient Intrusion Detection System Based on Feature Selection and Ensemble Classifier. Comput. Netw. 2020, 174, 107247. [CrossRef]
28. Alazzam, H.; Sharieh, A.; Sabri, K.E. A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer. Expert Syst. Appl. 2020, 148, 113249. [CrossRef]
29. Acharya, N.; Singh, S. An IWD-based feature selection method for intrusion detection system. Soft Comput. 2018, 22, 4407–4416. [CrossRef]
30. Kurniabudi; Stiawan, D.; Darmawijoyo; Idris, M.B.; Bambi, A.M.; Budiarto, R. CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection. IEEE Access 2020, 8, 132911–132921. [CrossRef]
31. Brunswick, U. Intrusion Detection Evaluation Dataset (CIC-IDS2017). Canadian Institute for Cybersecurity, 7 July 2017. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 19 October 2020).
32. Brunswick, U. Intrusion detection evaluation dataset (ISCXIDS2012). Canadian Institute for Cybersecurity, 11 July 2010. Available online: https://www.unb.ca/cic/datasets/ids.html (accessed on 19 October 2020).
33. Safara, F.; Souri, A.; Serrizadeh, M. Improved intrusion detection method for communication networks using association rule mining and artificial neural networks. IET Commun. 2020, 14, 1192–1197. [CrossRef]
34. Toupas, P.; Chamou, D.; Giannoutakis, K.M.; Drosou, A.; Tsouvaras, D. An Intrusion Detection System for Multi-Class Classification based on Deep Neural Networks. In Proceedings of the 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019.
35. De Castro, E.P.; Toupas, P.; Chamou, D.; Giannoutakis, K.M.; Drosou, A.; Tzovaras, D. An Intrusion Detection System for Multi-Class Classification based on Deep Neural Networks. In Proceedings of the 10th International Work-Conference on Artificial Neural Networks, Intelligent Data Engineering and Automated Learning-IDEAL 2007; Springer: Berlin/Heidelberg, Germany, 2007.
36. Le, T.; Vo, M.T.; Vo, B.; Lee, M.Y.; Baik, S.W. A Hybrid Approach Using Oversampling Technique and Cost-Sensitive Learning for Bankruptcy Prediction. Appl. Mach. Learn. Methods Complex Econ. Financ. Netw. 2019, 2019, 8460934. [CrossRef]
37. Lesmeister, C. Zero and near-zero variance features. In Mastering Machine Learning with R, 3rd ed.; Packt Publishing: Birmingham, UK, 2019.
38. Sharma, S.R.; Parthasarathy, R.; Honnavalli, P.B. A Feature Selection Comparative Study for Web Phishing Datasets. In Proceedings of the 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECT), Bangalore, India, 2–4 July 2020.
39. Raschka, S. Sequential Feature Selector. 2020. Available online: http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/#properties (accessed on 5 November 2020).
40. Cabestany, J.; Sandoval, F.; Prieto, A.; Corchado, J.M. Bio-Inspired Systems: Computational and Ambient Intelligence. In Proceedings of the 10th International Work-Conference on Artificial Neural Networks, IWANN 2009, Salamanca, Spain, 10–12 June 2009.
41. Sánchez-Marñoñ, N.; Alonso-Betanzos, A.; Tombilla-Sanromán, M. Filter Methods for Feature Selection—A Comparative Study. In Intelligent Data Engineering and Automated Learning-IDEAL 2007; Springer: Berlin/Heidelberg, Germany, 2007.
42. Kruger, M. Neural Network Concepts. Missinglink. Available online: https://missinglink.ai/guides/neural-network-concepts/classification-neural-networks-neural-network-right-choice/ (accessed on 18 October 2020).
43. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. Appl. Sci. 2019, 9, 4396. [CrossRef]
44. Atefi, K.; Hashim, H.; Khodadadi, T. A Hybrid Anomaly Classification with Deep Learning (DL) and Binary Algorithms (BA) as Optimizer in the Intrusion Detection System (IDS). In Proceedings of the 2020 16th IEEE International Colloquium on Signal Processing & its Applications (CSPA 2020), Langkawi, Malaysia, 28–29 February 2020.
45. Cabitza, F.; Campagner, A.; Albano, D.; Aliprandi, A.; Bruno, A.; Chianca, V.; Corazza, A.; di Pietro, F.; Gambino, A.; Gitti, S.; et al. The Elephant in the Machine: Proposing a New Metric of Data Reliability and its Application to a Medical Case to Assess Classification Reliability. Appl. Sci. 2020, 10, 4014. [CrossRef]
46. Kampakis, S. Performance Measures: Cohan’s Kappa Statistic. The Data Scientist, 8 May 2016. Available online: https://thedatascientist.com/performance-measures-cohens-kappa-statistic/ (accessed on 29 November 2020).
47. Landis, J.; Koch, G. The measurement of observer agreement for categorical data. Biometrics 1977, 33, 159–174. [CrossRef] [PubMed]
Radanliev, P.; Roure, D.D.; Page, K.; Jason, R.C.N.; Montalvo, R.M.; Santos, O.; Maddox, L.; Burnap, P. Cyber risk at the edge: Current and future trends on cyber risk analytics and artificial intelligence in the industrial internet of things and industry 4.0 supply chains. *Cybersecurity* 2020, 3, 13. [CrossRef]

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).