symmetry

MDPI

Article

Feature Selection Methods Simultaneously Improve the Detection Accuracy and Model Building Time of Machine Learning Classifiers

Saleh Alabdulwahab * and BongKyo Moon

Department of Computer Science & Engineering, Dongguk University, 30 Pildong-ro 1-gil Jung-gu, Seoul 04620, Korea; bkmoon@dgu.edu

* Correspondence: saleh@dongguk.edu

Received: 31 July 2020; Accepted: 25 August 2020; Published: 27 August 2020

Abstract: The detection accuracy and model building time of machine learning (ML) classifiers are vital aspects for an intrusion detection system (IDS) to predict attacks in real life. Recently, researchers have introduced feature selection methods to increase the detection accuracy and minimize the model building time of a limited number of ML classifiers. Therefore, identifying more ML classifiers with very high detection accuracy and the lowest possible model building time is necessary. In this study, the authors tested six supervised classifiers on a full NSL-KDD training dataset (a benchmark record for Internet traffic) using 10-fold cross-validation in the Weka tool with and without feature selection/reduction methods. The authors aimed to identify more options to outperform and secure classifiers with the highest detection accuracy and lowest model building time. The results show that the feature selection/reduction methods, including the wrapper method in combination with the discretize filter, the filter method in combination with the discretize filter, and the discretize filter, can significantly decrease model building time without compromising detection accuracy. The suggested ML algorithms and feature selection/reduction methods are automated pattern recognition approaches to detect network attacks, which are within the scope of the Symmetry journal.

Keywords: IDS; ML classifiers; information security; network; Weka; NSL-KDD

1. Introduction

Cyberattacks have significantly increased with the rapid advancement of the Internet, massive data electronic transmission, and the growing number of users. These fast changes and challenges require a powerful mechanism to maintain stable and secure networks. One of these mechanisms is an intrusion detection system (IDS), which monitors the processes prevailing in networks and evaluates them for signs of any possible deviations that violate security policies [1]. Intrusion detection has two approaches: signature based and anomaly based. The signature-based approach detects attacks by looking for exact and specific patterns (i.e., byte sequences or known malicious packets in network traffic). By contrast, researchers use the anomaly-based approach to detect unknown attacks and identify any unacceptable deviations from normal network traffic. Unlike the signature-based approach, the anomaly-based approach shows many false alarms when dealing with large, high-dimensional data. It relies on its knowledge of normal behaviors and any deviation from normal patterns, and it has gained popularity as an effective approach against new attacks.

An IDS uses machine learning (ML) algorithms to deal with huge volumes of high-dimensional data in order to accurately identify intrusions [2,3], as well as to generate a model of reliable activities and compare a new behavior against this model. Although there are many ML algorithms for intrusion detection purposes, further investigation is needed to evaluate their efficacy and accuracy (i.e., a low

false positive rate (FPR) for a huge volume of data) in intrusion detection. Researchers are aware that some ML algorithms are superior to others in intrusion detection accuracy, but these need more training time to build models on big datasets. Researchers have documented that model building time is a vital aspect of predicting intrusion in real life [4]. Some integrated ML algorithms with IDSs are offline and cannot be deployed in real time because of their high model building time. A delay in IDSs can compromise networks for a period of time before raising any alarm, making fast classification essential to guarantee the quick detection of attacks and ensure that IDSs have a fast data stream monitoring capability. As time plays a crucial role in detecting attacks in a high-speed network, a practical real-time IDS is needed to detect intrusions.

Consequently, researchers need to conduct further studies on the performance of ML classifier algorithms with feature selection/reduction approaches in order to lower the model building time without compromising intrusion detection accuracy. This study was designed to identify more ML classifiers with the highest detection accuracy and lowest possible model building time for the development of an efficient network IDS.

2. Research Motivation

Although model building time is vital for predicting intrusion in real life, previous studies have mainly focused on achieving classifiers' high detection accuracy with limited consideration of model building time. Researchers need to further investigate ML classifiers that meet both criteria.

Evaluating the performance of more ML classifiers to identify those with both high detection accuracy and the lowest possible model building time is feasible using feature selection/reduction methods. Researchers have already documented that feature selection/reduction methods significantly reduce the model building time without compromising the detection accuracy of a very small number of classifiers, which can help a number of ML classifiers perform well. Presenting more options for the best classifiers with both high detection accuracy and the lowest model building time can address the needs in real-life situations and help build an efficient network IDS.

3. Related Work

The rapid growth of online social networks and electronic applications and the expansion of electronic information resources have generated huge and accumulative volumes of data, forming large datasets with many domains. Big data research, management, supervision, data storage/access, information processing, and analysis are all challenging because of highly unstructured and heterogeneous data and sources. The huge dimension of social networks, spatiotemporal influence, and user interactions are among the several challenges in uncovering behavioral mechanisms. Other challenges of big data include the extraction of essential information, privacy and security protection, and the prediction of web content to understand the interests, sites, and search history of users and accurately predict users' behaviors. Big data and ML use IDS to address these issues and difficulties [5].

Moreover, new technologies, including 4G/5G networks, have significant prospective application in unmanned aerial vehicles (UAVs) equipped with sensors, cameras, and GPS receivers in delivering internet of things (IoT) service, generating useful heterogeneous big data. However, there are many challenges to be resolved before UAVs and their accumulative heterogeneous big data can be effectively and safely used. An advanced framework supporting multi-level and multi-domain defense mechanisms in protecting UAVs from spoofing, false information, signal jamming, physical attacks, hijacking, abusing, and firmware hacks/sabotage is required using ML algorithm solutions that are built in IDSs [6].

IDS is one of the key assets to protect IT infrastructure against any potential threats and to enhance network security in most organizations. As a result, many researchers have worked intensively to develop an intelligent IDS and further improve network security. Jia et al. [7] developed a new deep neural network (NDNN) model with four hidden layers to capture and classify the intrusion features of

the NSL-KDD and KDD99 training datasets. They claimed that the NDNN-based model improved the performance of the IDS and its accuracy rate to as high as 99.9%. Li et al. [8] presented a new intelligent IDS by applying ensemble and unsupervised ML techniques to address the security challenges in software-defined 5G networks. Dey et al. [9] suggested a multi-layered IDS for mobile clouds involving heterogeneous customer networks. They proposed a system with two steps, namely multi-layer traffic screening and decision-based virtual machine (VM) selection, and concluded that using the system is a highly effective method to detect intrusions. Leite and Girardi [10] proposed an IDS that can adapt itself to its environment and recognize new intrusions not previously specified in a system design. They integrated case-based reasoning, reactive behavior, and learning to acquire information from past solutions and support the evolution of case-based reasoning and reactive behavior in improving IDS performance. Hajisalem and Babaie [11] developed an intelligent classification and regression tree (CART) classifier for IDS by optimizing speed and accuracy. They combined the artificial bee colony and artificial fish swarm algorithms to select effective if-then rules for the CART classifier, ran this classifier on the NSL-KDD and UNSW-NB datasets, and concluded that it can achieve a 99% detection rate and 0.01% false positive rate FPR. Li et al. [12] proposed a framework based on regularized semi-nonnegative matrix tri-factorization, which mapped a signed network from a high-dimensional space to a low-dimensional one. They also presented graph regularization to distribute pairs of nodes, which were connected with negative links, into different communities for improving detection accuracy. The authors claimed that the application of their framework in both synthetic and real-world datasets confirmed the effectiveness of the suggested method. Moreover, Li et al. [13] studied the bit error rate performance of bi-quadrature physical-layer network coding (BQ-PNC) in asymmetric two-way relay channels and found that BQ-PNC can significantly improve bit error rate (BER) performance either in symmetric or asymmetric circumstances.

Most real-world network traffic data are not available because of privacy and security issues, but several public datasets are available for IDS performance assessment. Some of these datasets suffer from a lack of adequate traffic types and up-to-date low-footprint attack styles, so researchers use an older benchmark dataset, NSL-KDD, to compare the performance of ML classifiers in a fair and reasonable manner. For example, Revathi and Malathi [14] used the NSL-KDD dataset to evaluate the performance and accuracy of five ML classifiers with or without feature reduction using the Weka tool. They reported that the NSL-KDD dataset is ideal for comparing different intrusion detection models and that the random forest method has the highest accuracy among J48, support vector machine (SVM), CART, and Naive Bayes. Using the Weka tool also, Dhanabal and Shantharajah [15] used 20% of the NSL-KDD dataset to measure the effectiveness of four ML classifiers (CFS, J48, SVM, and naïve bayes (NB)) in detecting anomalies in a network traffic pattern. They concluded that the detection accuracy of J48 outperformed that of SVM and NB. Chand et al. [16] evaluated the SVM classifier's performance when integrated with other classifiers, such as BayesNet, AdaBoost, logistic, IBK, J48, random forest, JRip, OneR, and simple CART, on the NSL-KDD dataset using the Weka tool. They concluded that a multi-classifier algorithm had better performance than a single-classifier algorithm. They showed that the stacking of SVM and random forest gave the best performance, with a detection accuracy rate of 97.50%. However, they did not report the time taken to build the model. Ikram and Cherukuri [17] proposed an ID model using chi-square feature selection and multi-class SVM. The main idea behind this model was to form a multi-class SVM to reduce the training and testing time and increase the classification accuracy of network attacks. They tested this model on the NSL-KDD and gurekddcup datasets and showed that the proposed method performed better; it had higher accuracy, faster convergence speed, and better generalization.

Choudhury and Bhowal [18] studied and compared the performance of nine classifiers (BayesNet, logistic, IBK, J48, PART, JRip, random tree, random forest, and REPTree) over the training and testing datasets of NSL-KDD using 10-fold cross-validation in the Weka tool. They concluded that random forest and BayesNet were suitable for the proper detection of network intrusion. Belavagi and Muniyal [3] analyzed four supervised ML classifiers, namely SVM, random forest (RF), LR, and NB,

for intrusion detection over the NSL-KDD dataset. They used accuracy, true positive rate (TPR), and FPR as the parameters to measure the performance of the classifiers, as well as an Intel Core (TM) i5-3230M CPU @ 2.60 GHz, 4 GB RAM to execute the experimental procedure. They found that the RF classifier outperformed the other tested classifiers in identifying whether data traffic was normal or was an attack. Next, Biswas [19] compared the performance of five classifiers (k-NN, DT, NN, SVM, and NB) using 5-fold cross-validation on the NSL-KDD dataset and concluded that the k-NN classifier showed better performance than the other classifiers. Wang et al. [20] suggested an effective IDS based on SVM with augmented features. They applied logarithm marginal density ratio transformation to form new and better transformed features that can improve the detection performance of the SVM-based detection model. They used the NSL-KDD dataset to evaluate the suggested method and concluded that it achieves better performance than other existing methods in terms of accuracy, detection rate, false alarm rate, and training speed for the SVM model. Yin et al. [21] explored how to model an IDS based on a deep learning approach using recurrent RNN-IDS on the NSL-KDD dataset. They compared this approach with J48, artificial neural network, random forest, SVM, and other ML classifiers proposed by previous researchers on the benchmark dataset. They reported that RNN-IDS had superior accuracy compared with traditional ML classifiers.

Malhotra and Sharma [4] evaluated 10 ML algorithms on the NSL-KDD dataset with and without the feature selection/reduction technique using the Weka tool. They reported that random forest, bagging, PART, and J48 were the best-ranked classifiers without using the feature selection/reduction technique, but they consumed much time in building the model. The authors subjected these four classifiers to further evaluation using feature selection/reduction methods to reduce the model building time while achieving high intrusion detection accuracy. They concluded that the feature selection/reduction methods significantly reduced the model building time without compromising the detection accuracy of the tested classifiers. They also believed that feature selection/reduction helps a number of ML classifiers perform well. Thus, researchers can assume that the results and feature selection/reduction methods of this study are promising for the investigation of more ML classifiers to determine the best classifier with the highest detection accuracy and lowest model building time, which can satisfy the real-life need to build an efficient network IDS. Abdullah et al. [22] presented an IDS framework with feature selection within the NSL-KDD dataset; the framework was based on dividing the input dataset into various subsets and combining them using the information gain filter in the Weka tool. The authors showed that the feature selection methods used improved detection accuracy and decreased complexity. They also demonstrated that the highest intrusion detection accuracy was obtained when using the random forest and PART classifiers under combination methods of the product probability rule. Setiawan et al. [23] proposed a combination of the feature selection method, normalization, and SVM using Weka's modified rank-based information gain filter to select 17/41 NSL-KDD dataset features. They achieved an overall detection accuracy rate of 99.8%. Zhou et al. [24] proposed an IDS framework based on the feature selection and ensemble learning techniques. They used a hybrid approach by combining CFS with the BA algorithm to select the optimal subset based on the correlation between features. Then, they formed an ensemble that combines the C4.5, RF, and ForestPA classifiers and applied these on the NSL-KDD, AWID, and CIC-IDS2017 datasets. They demonstrated that the proposed CFS-BA-ensemble approach showed better performance than the other related approaches. Notably, these previous studies emphasized detection accuracy but not the time taken to build the model.

Furthermore, Mahfouz, Venugopal, and Shiva [25] evaluated and compared the performance of six supervised ML classifiers, namely NB, logistic, multilayer perceptron (MLP), Sequential Minimal Optimization (SMO), IBK, and J48, with the full NSL-KDD using Weka software. They investigated this in the context of intrusion detection along various dimensions, mainly feature selection, sensitivity to hyper-parameter tuning, and class imbalance problems. They used accuracy, TPR, FPR, precision, recall, F-measure, and receiver operating characteristic (ROC) area as the parameters to evaluate the performance of the tested classifiers. They also used a PC installed with Intel(R) CORE(TM) i5-6600K

CPU @ 3.50 GHz, 3.50 GHz, 8 GB RAM running a 64-bit Windows 10 OS, x64-based processor to carry out the experiment protocol. They concluded that J48 and IBK are the two best classifiers in terms of accuracy detection, but IBK was much better when applying feature selection techniques.

Previous studies, except that by Malhotra and Sharma [4], clearly focused on achieving the high detection accuracy of classifiers without considering the model building time, which is a crucial aspect for predicting intrusion in real-life situations. Therefore, evaluating the performance of more ML classifiers is needed to determine the most appropriate classifiers with both high detection accuracy and the lowest possible model building time using the NSL-KDD dataset with feature selection/reduction methods in the Weka tool. In particular, researchers have already documented that feature selection/reduction methods significantly reduce the model building time without compromising the detection accuracy of some classifiers and that they could also help a number of ML classifiers perform well. Presenting more options for the best classifiers with both high detection accuracy and the lowest possible model building time can satisfy real-life needs and help build an efficient network IDS. These classifiers can enhance and accelerate IDS function to accurately deal with the huge dimension of social networks and large data flow.

ML classifiers with both the highest detection accuracy and lowest model building time in IDSs can be used to make broadband wireless heterogeneous networks more efficient and secure. Broadband wireless heterogeneous networks, including communication systems and multi-media services and applications, need a fast real-life abnormal tracing system to deliver high service quality. This can be achieved by ML classifiers with the highest detection accuracy and lowest model building time [26].

4. Reserch Scope and Method

Research Scope

The purpose of this experiment was to assess the performance of six supervised classifiers (REPTree, SMO, LogitBoost, BayesNet, RBF, and NBTree) in detecting intrusion at the lowest model building time possible with high detection accuracy in two phases. To do so, the authors used the Weka ML package and the NSL-KDD training dataset, which were downloaded to an LG PC with a configuration of 2.20 GHz Intel(R) Core (TM) i7-8750H and 16 GB of RAM running on Microsoft Windows 10. In the first phase, the authors evaluated the performance of each classifier on a full NSL-KDD training dataset using 10-fold cross-validation in the Weka tool. The aim of this phase was to identify additional options to outperform classifiers with the lowest model building time and high detection accuracy and to use this as a baseline for evaluating the efficiency of different feature selection/reduction methods used in the second phase. The authors utilized the Weka tool because it is well known and widely used among researchers for developing ML algorithms and their applications to real-world data mining problems. It has a collection of ML algorithms and supports various standard data mining tasks, particularly data pre-processing, clustering, classification, regression, visualization, and feature selection. It has a cross-validation mode and global minimum adjustment to lessen any possible overfitting.

In the second phase, the authors evaluated the performance of each classifier on a full NSL-KDD training dataset using 10-fold cross-validation with different feature selection/reduction techniques, including the wrapper and filter methods in the Weka tool. The authors also used a discretize filter classifier individually and in combination with the wrapper and filter methods to evaluate the performance of the tested classifiers. The aim of this phase was to assess the effect of different feature selection/reduction methods and the discretize filter on the performance of the tested classifiers in terms of detection accuracy and model building time; this would allow the authors to determine the most appropriate classifiers for securing a network with high detection accuracy at the lowest possible time. During the two phases, the authors used model building time, detection accuracy, and correctly and incorrectly classified instances as performance measures to identify the classifiers with both the highest detection accuracy and lowest model building time simultaneously.

5. Research Method

In this section, the authors describe the NSL-KDD dataset, Weka tools, ML classifiers, feature selection/reduction approaches, and the performance measures they used in this study.

NSL-KDD Dataset: This is a refined version of the well-known KDDcup99 dataset [27] that is widely used for building an IDS. This dataset contains 125,973 instances with 41 features and an assigned label classifying each record as either normal or an attack [Table 1]. Many researchers have used this dataset to conduct different types of analyses and to apply various methods and tools for developing effective IDSs [28]. The NSL-KDD dataset has a subset named KDDTrain+-20Percent dataset, which has 25,192 instances and represents 20% of the entire train dataset. Researchers can group it into four main classes, namely denial of service (DoS), Probe, U2R, and R2L.

Table 1. The 41 features of the NSL-KDD dataset.

Number	Function	Number	Function
1	Duration	22	is_guest_login
2	protocol_type	23	Count
3	Service	24	srv_count
4	Flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	Land	28	srv_rerror_rate
8	worng_fragment	29	same_srv_rate
9	Urgent	30	diff_srv_rate
10	Hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login		

Weka Package: This is an open-source package that contains a number of ML classifiers used to perform different types of data mining tasks. Written in JAVA [29], the Weka package consists of tools for data pre-processing, regression, classification, clustering, association rules, and visualization. Researchers can evaluate the performance of available classifiers in Weka directly on the NSL-KDD dataset. To easily compare the classifiers in this study, the authors used some default features provided by Weka for ML classifiers, feature selection/reduction techniques, and the discretize filter. Weka has four applications: Explorer, Experimenter, Knowledge Flow, and Simple Command Line Interface.

In this study, data were discretized using Weka, and the CfsSubsetEval wrapper method was used with a BestFirst search and 10-fold cross-validation. The InfoGainAttributeEval filter method was also used with Ranker search and 10-fold cross-validation.

Machine Learning Algorithms: Researchers have proposed many ML classifiers to monitor and analyze network traffic for various anomalies. Among the several classifiers available in Weka, six well-known supervised classifiers that were compatible with the NSL-KDD training dataset were evaluated by the authors. They ran each classifier only once on the NSL-KDD training dataset, and the outcome performance was analyzed. To minimize any possible overfitting problem, a cross-validation mode and global minima adjustment were used.

The supervised classifiers used were as follows:

Reduced Error Pruning Tree (REPTree): This is a rapid decision tree learner based on C4.5. It is an algorithm used as a representative method in an attempt to clarify the problems of decision tree learning. REPTree builds a decision/regression tree based on information gain or by reducing variance [30]. It sorts values for numeric attributes once, deals with missing values by splitting them into fractional instances using C4.5, and creates multiple trees in various iterations and selects the best from all trees created. REPTree has been presented to decrease tree structure complexity without reducing classification accuracy. The basic of the pruning of this classifier is used REP with back overfitting. It forms a decision/regression tree by splitting and pruning the regression tree based on the highest information gain ratio value.

Sequential Minimal Optimization (SMO): This is widely used for training SVMs and was formulated by J. Platt [31]. SMO is one way to solve a quadratic programming (QP) issue that arises during SVM training. SMO divides the large QP problem into a series of very tiny sub-problems. These small sub-problems are solved analytically, preventing the use of time-consuming numerical QP optimization as an inner loop. It is the fastest for linear SVMs and sparse datasets and can be more than 1000 times faster than the chunking algorithm. The amount of memory needed for SMO is linear in the training dataset size, allowing SMO to handle very large training sets. It scales somewhere between linear and quadratic in the training set size for several test problems.

LogitBoost: This is a popular boosting variant formulated by Friedman, Hastie, and Tibshirani [32]. Researchers can apply it to either binary or multi-class classification, and they considered it an additive tree regression by minimizing logistic loss. LogitBoost has two essential setting factors, invariant property and the density of Hessian matrices, and it is seen as a convex optimization with the following formula: $f = \sum_{t} \alpha_t h_t$. Compared with other AdaBoost classifiers, it is appropriate for handling noisy and outlier data. It uses a binomial that changes the loss function linearly. It is one of the popular boosting variants and has three main components: the loss, the function model, and the optimization algorithm.

BayesNet: This is a broadly used method that works on the basic Bayes theorem; it constructs a Bayesian network [33] by calculating the conditional probability on every node. It is related to the family of the probabilistic graphical model. BayesNet learns in two stages: learning of the network structure followed by the probability table. It is a powerful instrument for data representation and inference under uncertainty conditions. It is a probabilistic graphical algorithm that is used to represent a set of random variables and their conditional dependencies with the assistance of a directed acyclic graph. This graph algorithm is applied to represent knowledge about an uncertain domain.

Radial Basis Function (RBF): This is an artificial neural network formulated by Broomhead and Lowe [34]. RBF uses radial basis functions for activation to change along the distance from a location. For functional approximation, it uses time-series prediction, classification, and system control. A multi-layer feedforward neural network, RBF is used to classify data in a non-linear mode and compare input data with training data. The production of the RBF neural network is weighted linear superposition of all basis functions. The frequently used basis function in the RBF model is the Gaussian basis function.

NBTree: This is a hybrid of naive Bayes and decision trees; it uses both classifiers and has the advantage of univariant splits at each node in the decision tree and the leaves in the naive Bayes algorithm. NBTree is useful when there are many attributes in the classification that are likely to be relevant [35]. It is a highly scalable method for big data and is described as a decision tree with both nodes and branches. NBTree categorizes an example to a leaf and allocates a class label by applying a naïve Bayes on that leaf. It represents the learned information in the form of a tree built recursively. NBTree is a popular baseline method for text categorization and with appropriate pre-processing. It significantly improves upon the performance of its constituents by inducing highly perfect classifiers.

Feature Selection/Reduction Approaches: Professionals understand that feature selection/reduction is essential to use data mining tools effectively. Feature selection/reduction has been an active area of research and development for many years in the field of ML and data mining [4]. This method selects

a subset of original features according to certain criteria by removing irrelevant or redundant features from the dataset to improve mining performance, such as prediction accuracy and the reduction of model building time [4,36]. There are mainly two methods for feature selection/reduction, namely the wrapper method and the filter method. Researchers can use these methods with or without the discretize filter.

The wrapper model needs one predetermined mining algorithm and uses its performance as the assessment criterion. It searches for features better suited to the mining algorithm in order to improve mining performance, but it is most likely to be more computationally expensive than the filter model. In this study, the authors used the procedure presented in the work of Malhotra and Sharma [4] to identify a subset of features that has the best performance with the classification algorithm using an attribute evaluator and a search method (CfsSubsetEval + BestFirst).

Next, the filter method depends on the general structures of data to evaluate and select feature subsets without involving any mining algorithm. In this study, the authors assigned ranks to all attributes in the dataset by using an attribute evaluator and a ranker method (InfoGainAttributeEval + Ranker), as described in the work of Malhotra and Sharma [4]. The attribute ranked first had the highest priority. The authors omitted features with a lower rank at a time to assess the accuracy of the classifier at that point in time, as well as omitted features one after the other until the global minimum was reached. After the global minima, the dataset started overfitting and generating additional instances that were incorrectly classified.

Malhotra and Sharma [4] described discretization as a process of altering numeric attributes into nominal attributes by dividing each attribute's numeric values into a number of intervals. Consequently, discretization assists in improving accuracy and reducing the learning complexity of the classifier with which it is used. To attain this benefit, the authors discretized the dataset by applying an unsupervised discretize filter to the attributes.

Performance Measures: The authors used performance measurements, particularly accuracy, time taken to build the model, and correctly and incorrectly classified instances, to evaluate and compare the performance of the tested classifiers. The descriptions of these parameters are as follows:

The confusion matrix is a visualization tool that works as a basis for calculating all other parameters. It includes four values: true positive (TP), false negative (FN), false positive (FP), and true negative (TN). They are described as follows:

True Positive (TP): Normal instances that are identified correctly
False Negative (FN): Abnormal instances incorrectly identified as normal
False Positive (FP): Anomalous instances incorrectly recognized as normal
True Negative (TN): Anomalous instances identified correctly as an attack

Accuracy is one of the essential measures for describing the performance of any algorithm. It is the degree to which an algorithm can properly predict positive and negative instances, and it can be determined by the following formula: Accuracy **=** TP + TN/TP + FN + FP + TN.

Correctly Classified Instances: Number of instances that were correctly identified
Incorrectly Classified Instances: Number of instances that were incorrectly identified

Time Taken to Build the Model: Time taken by a classifier to generate a model. Researchers usually measure it in seconds (s). The lesser time taken to build the model, the better the classifier.

Sensitivity is another basic measurement to assess the performance of any algorithm. It is known as a true positive rate that is determined correctly and can be calculated using the following formula: Sensitivity = TP/TP + FN.

Specificity is a measure of the proportion of an actual negative rate recognized correctly by a learning algorithm and can be measured using the following formula: *Specificity = TN/FP + TN.*

Precision is one of the primary performance indicators. It indicates the fact of being accurate and correct. It gives the impression of correctly predicted instances. It is calculated as *TP/TP + FP*.

The **F-measure** is the weighted average of precision and sensitivity and can be calculated by 2TP/2TP + FP + FN. It indicates the accuracy of a test by numerating the balance between precision and recall. It is the harmonic mean of sensitivity and precision.

The **Receiver Operating Characteristics (ROC)** is a graphical means of evaluating ML classifiers and visualizing the relationship between the TP and FP rates of IDSs. Essentially, it describes the performance of ML classifiers. It is used to effectively compare ML classifiers in terms of accuracy. ML classifiers having more area under the curve have high performance.

The **TP rate** indicates the possibility of an ML classifier predicting positive instances as correct and normal. Also called sensitivity, the TP rate is the probability that an actual positive will test positive. A high TP is preferable. The TP rate can be calculated using the following formula: $TP\ rate = TP/TP + FN$.

The **FP rate**, also called the false alarm rate, implies the possibility of an ML algorithm forecasting a normal instance as an attack. A consistent increase in the FP rate might mislead the network manager to deliberately ignore alerts from the network system. A low FP is therefore desirable. The FP rate can be measured using the following formula: $FP\ rate = FP/FP + TN$.

6. Results and Discussion

In this section, the authors present the results of the first and second phases in two parts, namely classifier comparison and performance evaluation using feature selection/reduction methods, respectively.

6.1. Classifier Comparison

Phase 1 of this study shows that the NBTree classifier had the highest detection accuracy of 99.87%, and the REPTree classifier had the shortest model building time of 3.59 s. On the other hand, the LogitBoost classifier had the lowest detection accuracy of 97.10%, and the SMO classifier had the longest model building time of 1137.71 s when evaluated on the full NSL-KDD training dataset using 10-fold cross-validation in the Weka tool, as described in Table 2. Thus, it is clear that both the REPTree and NBTree classifiers had the best performance among the RBF, BayesNet, SMO, and LogitBoost classifiers even without the need for feature selection/reduction methods or discretization. Although NBTree outperformed the other tested classifiers in all measured parameters except in the model building time (213.18 s), its real-life efficiency might be affected by its very long model building time. By contrast, the REPTree classifier had the shortest model building time (3.59 s) among the tested classifiers; it had a very slightly compromised detection accuracy rate, but it had significantly high performance in terms of the TP rate, FP rate, ROC area, specificity %, and sensitivity %. Therefore, the authors suggest the use of the REPTree classifier as an appropriate option for intrusion detection, as it has a very high detection accuracy and short model building time.

Table 2. Performance of the six classifiers using 10-fold cross-validation.

Parameters \ Classifier	SMO	REPTree	NBTree	RBF	LogitBoost	BayesNet
Correctly Classified Instances	122,704 97.40%	125,766 99.83%	125,819 99.87%	123,394 97.95%	122,329 97.10%	122,409 97.17%
Incorrectly Classified Instances	3269 2.595%	207 0.164%	154 0.122%	2579 2.0473%	3644 2.892%	3564 2.82%
TP Rate	0.974	0.998	0.999	0.980	0.971	0.972
FP Rate	0.028	0.002	0.001	0.022	0.030	0.032
Precision	0.974	0.998	0.999	0.980	0.971	0.972
F-measure	0.974	0.998	0.999	0.980	0.971	0.972
ROC Area	0.973	0.999	1.000	0.987	0.996	0.997
Specificity (%)	96.0	99.8	99.81	96.8	96.2	94.6
Sensitivity (%)	98.5	99.8	99.9	98.9	97.8	99.3
Model Building Time (second)	1137.71 s	3.59 s	213.18 s	81.01 s	18.3 s	4.69 s

For further investigation, the tested classifiers were ranked from 1 to 6 based on their individual performance (Table 3). Rank 1 was considered the best, whereas rank 6 was considered the worst. Subsequently, the classifiers with lower numerical ranks were preferred over those with higher numerical ranks. Table 3 shows that the REPTree and NBTree classifiers outperformed the other tested classifiers, whereas the LogitBoost classifier performed the worst. Moreover, REPTree was the only classifier with the shortest model building time and very slightly compromised detection accuracy, TP rate, FP rate, precision, F-measure, ROC area, specificity, and sensitivity. Previous studies ranked the random forest classifier the best, as it outperformed nine other classifiers, but it consumed a huge amount of time (191.06 s) to build a model [4]. This indicates that REPTree outperforms random forest in terms of model building time.

Table 3. Ranks of the tested classifiers based on their reported performance in Table 2.

Classifier / Parameters	SMO	REPTree	NBTree	RBF	LogitBoost	BayesNet
Correctly Classified Instances	4	2	1	3	6	5
Incorrectly Classified Instances	4	2	1	3	6	5
TP Rate	4	2	1	3	6	5
FP Rate	4	2	1	3	6	5
Precision	4	2	1	3	6	5
F-measure	4	2	1	3	6	5
ROC Area	6	2	1	5	4	3
Specificity (%)	5	2	1	3	4	6
Sensitivity (%)	5	2	1	4	6	3
Model Building Time	6	1	5	4	3	2

6.2. Performance Evaluation Using the Feature Selection/Reduction Methods

In this section, the authors assessed the tested classifiers using different feature selection and reduction methods with or without discretization to determine whether their performance could be improved further. They used two combinations of feature selection/reduction methods, namely the wrapper method (CfsSubsetEval + BestFirst) and the filter method (InfoGainAttributeEval + Ranker), to improve the performance of six tested classifiers. Table 4 describes the details of the features selected by each combination in the second phase.

Table 4. Feature selection/reduction using the wrapper method (CfsSubsetEval + BestFirst) and the filter method (InfoGainAttributeEval + Ranker) on the full NSL-KDD training dataset in the second phase of this study.

Attribute Evaluator, Search Method, and Ranker	Features Selected	Method Used
CfsSubsetEval + BestFirst	4, 5, 6, 12, 26, 30	Wrapper method
InfoGainAttributeEval + Ranker	5, 3, 6, 4, 30, 29, 33, 34, 35, 38, 12, 3,9, 25, 23, 26, 37, 32, 36, 31, 24, 41, 2, 27, 40, 28, 1, 10, 8, 13, 16, 19, 22, 17, 15, 14, 18, 11, 7, 21, 20, 9	Filter method

The authors implemented the wrapper-based feature selection method on the full NSL-KDD training dataset to select the best subset of features. The method selected the optimal subset of 4, 5, 6, 12, 26, and 30 features from the full NSL-KDD training dataset (Table 4). Next, the authors ranked the datasets and omitted low-ranked features one by one until overfitting occurred. If a feature/attribute was further removed, the model started overfitting, and the percentage of correctly classified instances was reduced. Consequently, in the filter method, the authors removed features one after the other until global minima were reached. In the full NSL-KDD training dataset, the global minima were achieved when the authors were left with the top 10 features (5, 3, 6, 4, 30, 29, 33, 34, 35, and 38) for

detecting intrusion attacks (Table 5). The authors evaluated the performance of the ranked classifiers after eliminating all redundant features, as suggested by the wrapper and filter methods, and presented the results in Tables 5–11. The authors also used the discretize filter classifier individually and in combination with the wrapper and filter methods to evaluate the performance of the tested classifiers.

Table 5. List of the highest-ranked features by the filter method for the tested ML classifiers.

Number of Global Minima of the Top 10 Features	Feature Name and Description
5	Src_bytes: Bytes transferred in a single connection from the source to the destination
3	Service: Service used by the destination network
6	Dst_bytes: Bytes transferred in a single connection from the source to the destination
4	Flag: Connection status, normal or error
30	Diff_srv_rate: Rate of connections among the connections collected in count (23) that were to different services
29	Same_srv_rate: Rate of connections among the connections collected in count (23) that were to the same service
33	Dst_host_srv_count: Number of connections with a similar port number
34	Dst_host_same _srv_rate: Connection rate among the connections collected in dst_host_count (32) that were to the same service
35	Dst_host_diff_ srv_rate: Connection rate among the connections collected in dst_host_count (32) that were to different services
38	Dst_host_serro r_rate: Connection rate among the connections collected in dst_host_count (32) that have activated the flag (4) s0, s1, s2, or s3

Table 6. Performance of the REPTree classifier on the NSL-KDD training dataset using different feature selection/reduction methods.

Parameters \ Methods	General Method	Discretize Filter Classifier	Wrapper Method	Wrapper Method + Discretize Filter	Filter Method	Filter Method + Discretize Filter
Correctly Classified Instances	125,766	125,665	125,274	125,382	125,706	125,637
	99.83%	99.75%	99.44%	99.53%	99.78%	99.73%
Incorrectly Classified Instances	207	308	699	591	267	336
	0.164%	0.244%	0.554%	0.469%	0.212%	0.266%
TP Rate	0.998	0.998	0.994	0.995	0.998	0.997
FP Rate	0.002	0.002	0.006	0.005	0.002	0.003
Precision	0.998	0.998	0.994	0.995	0.998	0.997
F-measure	0.998	0.998	0.994	0.995	0.998	0.997
ROC Area	0.999	0.999	0.999	0.999	0.999	0.999
Specificity (%)	99.8	99.7	99.7	99.2	99.7	99.7
Sensitivity (%)	99.8	99.7	99.7	99.7	99.8	99.7
Model Building Time (second)	3.59 s	1.93 s	5.76 s	1.62 s	4.65 s	0.6 s

Table 7. Performance of the LogitBoost classifier on the NSL-KDD training dataset using different feature selection/reduction methods.

Parameters \ Methods	General Method	Discretize Filter Classifier	Wrapper Method	Wrapper Method + Discretize Filter	Filter Method	Filter Method + Discretize Filter
Correctly Classified Instances	122,329	121,639	118,612	119,910	121,770	122,447
	97.10%	96.55%	94.15%	95.18%	96.66%	97.20%
Incorrectly Classified Instances	3644	4334	7361	6063	4203	3526
	2.892%	3.440%	5.843%	4.812%	3.336%	2.799%
TP Rate	0.971	0.966	0.942	0.952	0.967	0.972
FP Rate	0.030	0.034	0.058	0.049	0.034	0.027
Precision	0.971	0.966	0.942	0.952	0.967	0.972
F-measure	0.971	0.966	0.942	0.952	0.967	0.972
ROC Area	0.996	0.992	0.990	0.990	0.995	0.992
Specificity (%)	96.2	97.1	94.1	94.5	96.0	97.7
Sensitivity (%)	97.8	96.0	94.1	95.7	97.1	96.7
Model Building Time (second)	18.3 s	2.65 s	9.96 s	5.1 s	17.48 s	0.98 s

Table 8. Performance of the Sequential Minimal Optimization (SMO) classifier on the NSL-KDD training dataset using different feature selection/reduction methods.

Parameters \ Methods	General Method	Discretize Filter Classifier	Wrapper Method	Wrapper Method + Discretize Filter	Filter Method	Filter Method + Discretize Filter
Correctly Classified Instances	122,704	125,777	112,240	125,311	121,651	125,578
	97.40%	99.84%	89.09%	99.47%	96.56%	99.68%
Incorrectly Classified Instances	3269	196	13,733	662	4322	395
	2.595%	0.155%	10.901%	0.525%	3.430%	0.313%
TP Rate	0.974	0.998	0.891	0.995	0.966	0.997
FP Rate	0.028	0.002	0.117	0.006	0.035	0.003
Precision	0.974	0.998	0.895	0.995	0.966	0.997
F-measure	0.974	0.998	0.890	0.995	0.966	0.997
ROC Area	0.973	0.998	0.887	0.995	0.965	0.997
Specificity (%)	96.0	99.7	82.6	99.2	96.1	99.6
Sensitivity (%)	98.5	99.8	94.6	99.6	96.9	99.7
Model Building Time (second)	1137.71 s	321.52 s	514.7 s	372.68 s	1011.69 s	269.69 s

Table 9. Performance of the Radial Basis Function (RBF) classifier on the NSL-KDD training dataset using different feature selection/reduction methods.

Parameters \ Methods	General Method	Discretize Filter Classifier	Wrapper Method	Wrapper Method + Discretize Filter	Filter Method	Filter Method + Discretize Filter
Correctly Classified Instances	123,394	125,672	114,151	125,322	122,368	125,595
	97.95%	99.76%	90.61%	99.48%	97.13%	99.69%
Incorrectly Classified Instances	2579	301	11,822	651	3605	378
	2.047%	0.238%	9.384%	0.516%	2.861%	0.300%
TP Rate	0.980	0.998	0.906	0.995	0.971	0.997
FP Rate	0.022	0.003	0.104	0.006	0.030	0.003
Precision	0.980	0.998	0.913	0.995	0.972	0.997
F-measure	0.980	0.998	0.905	0.995	0.971	0.997
ROC Area	0.987	1.000	0.955	0.999	0.984	0.999
Specificity (%)	96.8	99.6	82.8	99.1	95.8	99.6
Sensitivity (%)	98.9	99.8	97.3	99.7	98.2	99.7
Model Building Time(second)	81.91 s	163.07 s	45.91 s	106.39 s	52.98 s	127.68 s

Table 10. Performance of the BayesNet classifier on the NSL-KDD training dataset using different feature selection/reduction methods.

Parameters \ Methods	General Method	Discretize Filter Classifier	Wrapper Method	Wrapper Method + Discretize Filter	Filter Method	Filter Method + Discretize Filter
Correctly Classified Instances	122,409	122,422	121,270	121,274	119,540	120,367
	97.17%	97.18%	96.26%	96.26%	94.89%	95.54%
Incorrectly Classified Instances	3564	3551	4703	4699	6433	5606
	2.82%	2.818%	3.733%	3.730%	5.106%	4.450%
TP Rate	0.972	0.972	0.963	0.963	0.949	0.955
FP Rate	0.032	0.031	0.036	0.036	0.057	0.049
Precision	0.972	0.973	0.963	0.963	0.951	0.957
F-measure	0.972	0.972	0.963	0.963	0.949	0.955
ROC Area	0.997	0.998	0.995	0.997	0.997	0.997
Specificity (%)	94.6	94.6	97.6	97.6	90.4	92.0
Sensitivity (%)	99.3	99.3	95.0	95.0	98.7	98.6
Model Building Time(second)	4.69 s	0.41 s	5.64 s	1.68 s	4.95 s	0.29 s

The filter methods showed that the tested ML classifiers were left with the same top 10 features in the same order (Table 5). This implies that all classifiers were run on the same dataset and conditions simultaneously, allowing a fair comparison of their efficacies and performances. Table 5 shows the global minimum, name, and description of each of the top 10 features ranked with the filter method.

Table 6 shows the evaluation of REPTree classifier performance based on different methods. The authors observed that the detection accuracy of REPTree was initially 99.83% with a 3.59 s model building time. However, after the different combinations of feature selection/reduction techniques were applied, the model building time was reduced significantly by the discretize filter classifier (1.93 s), the wrapper method in combination with the discretize filter (1.62 s), and the filter method

in combination with the discretize filter (0.6 s). These improvements occurred without significantly compromising the detection accuracy range (99.44–99.78%), suggesting the use of the REPTree classifier with either the discretize filter classifier or the filter method in combination with the discretize filter as an efficient tool in an IDS. Researchers can consider this an additional and feasible classifier option with high detection accuracy and low model building time. Figures 1 and 2 are the diagrams of the detection accuracy and model building time of the REPTree classifier with different feature selection/reduction methods.

Table 7 shows the evaluation of the LogitBoost classifier using different feature selection/reduction methods with or without the discretize filter. As shown in the table, feature selection/reduction methods, the discretize filter, and the combination of the methods significantly reduced the model building time from 18.3 s at the initial point to the lowest limit of 0.98 s. However, this compromised detection accuracy, except for the filter method in combination with the discretize filter, which showed slightly higher detection accuracy (97.20%) than the initial one (97.10%) with lower model building time (0.98 s). Therefore, the authors recommend the use of the LogitBoost classifier with the filter method in combination with the discretize filter for better performance. Figures 3 and 4 illustrate the performance of the LogitBoost classifier in terms of accuracy and model building time when used with different feature selection/reduction methods.

Table 11. Performance of the NBTree classifier on the NSL-KDD training dataset using different feature selection/reduction methods.

Parameters \ Methods	General Method	Discretize Filter Classifier	Wrapper Method	Wrapper Method + Discretize Filter	Filter Method	Filter Method + Discretize Filter
Correctly Classified Instances	125,819	125,787	125,300	125,414	125,764	125,634
	99.87%	99.85%	99.46%	99.55%	99.83%	99.73%
Incorrectly Classified Instances	154	186	673	559	209	339
	0.122%	0.147%	0.534%	0.443%	0.165%	0.269%
TP Rate	0.999	0.999	0.995	0.996	0.998	0.997
FP Rate	0.001	0.002	0.006	0.005	0.002	0.003
Precision	0.999	0.999	0.995	0.996	0.998	0.997
F-measure	0.999	0.999	0.995	0.996	0.998	0.997
ROC Area	1.000	1.000	0.999	1.000	1.000	1.000
Specificity (%)	99.8	99.8	99.0	99.3	99.7	99.6
Sensitivity (%)	99.9	99.8	99.7	99.7	99.9	99.7
Model Building Time(second)	213.18 s	70.95 s	14.23 s	8.7 s	23.94 s	13.6 s

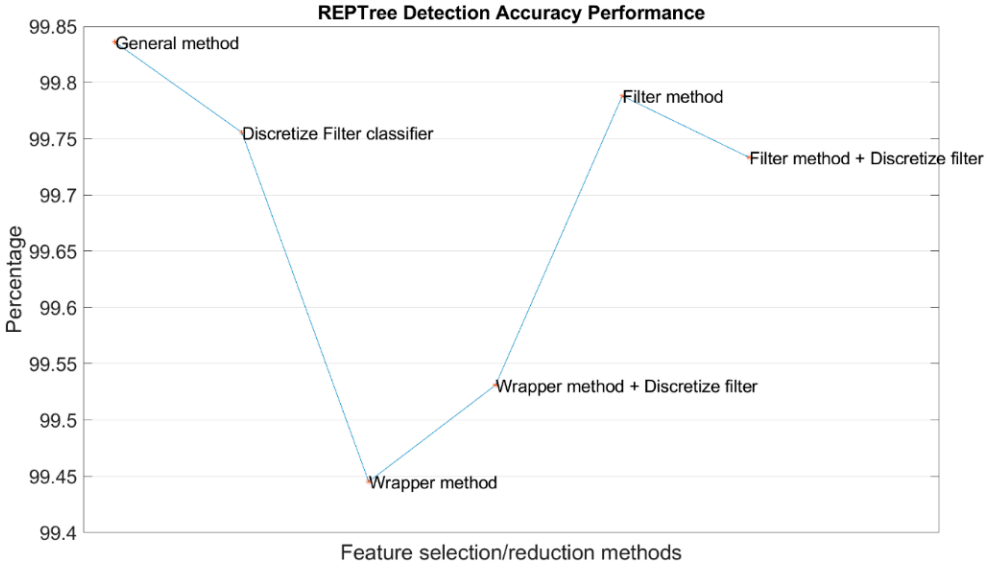

Figure 1. REPTree detection accuracy performance.

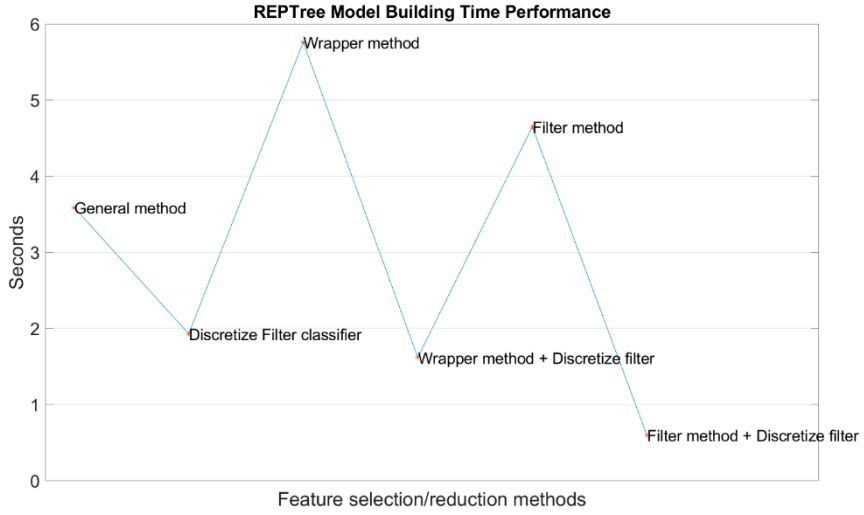

Figure 2. REPTree model building time performance.

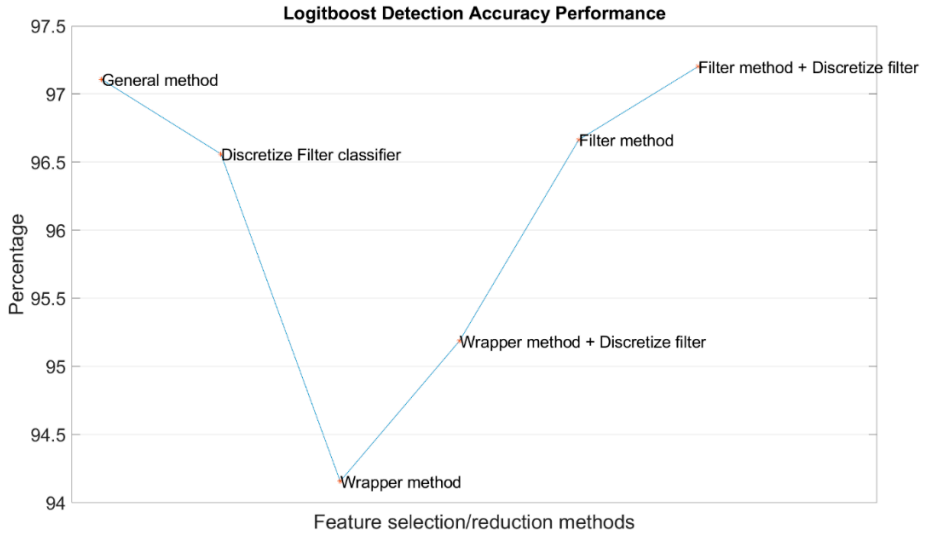

Figure 3. LogitBoost detection accuracy performance.

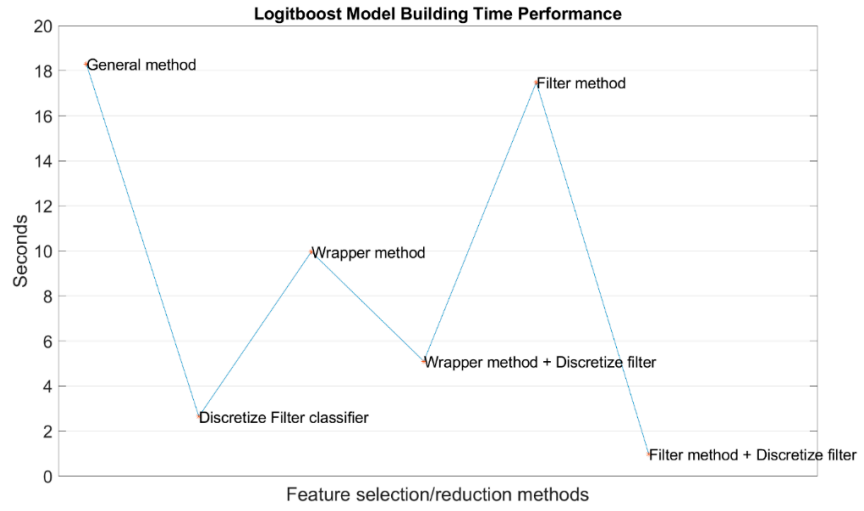

Figure 4. LogitBoost model building time performance.

Table 8 presents the evaluation of the SMO classifier's performance based on different feature selection/reduction methods with or without the discretize filter. The feature selection/reduction methods, the discretize filter, and the combination of these methods significantly reduced the model building time from 1137.71 s at the initial reading to its lowest reading at 269.69 s. Furthermore, the discretize filter classifier, the wrapper method in combination with the discretize filter, and the filter method in combination with the discretize filter significantly improved the performance of the SMO classifier from 97.40% detection accuracy and 1137.71 s model building time to 99.84% and 321.52 s, 99.47% and 372.68 s, and 99.68% and 269.69 s, respectively. These results suggest using the SMO classifier as an effective option for high-level detection accuracy but with an unreasonable model building time. Figures 5 and 6 present the performance of the SMO classifiers in terms of accuracy and model building time when used with different feature selection/reduction techniques.

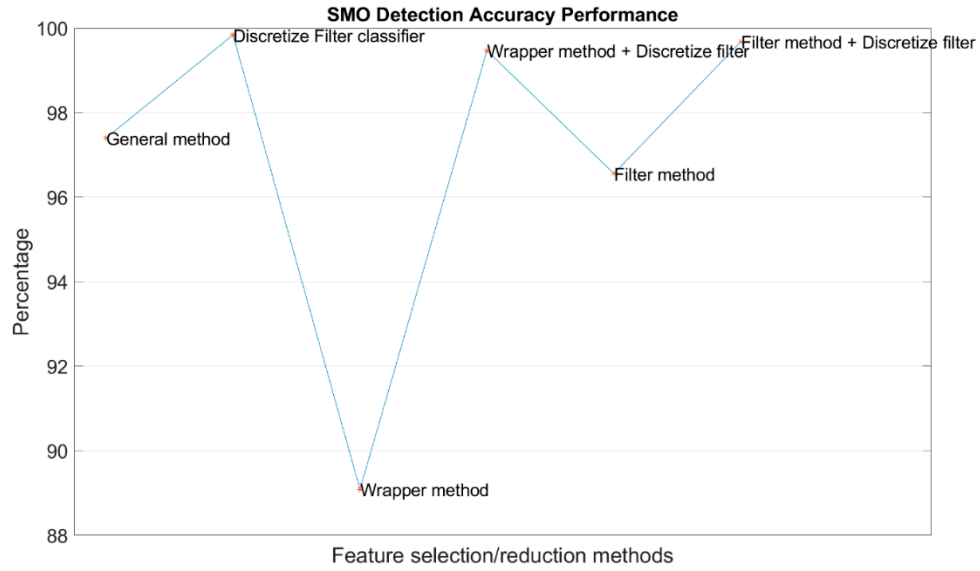

Figure 5. SMO detection accuracy performance.

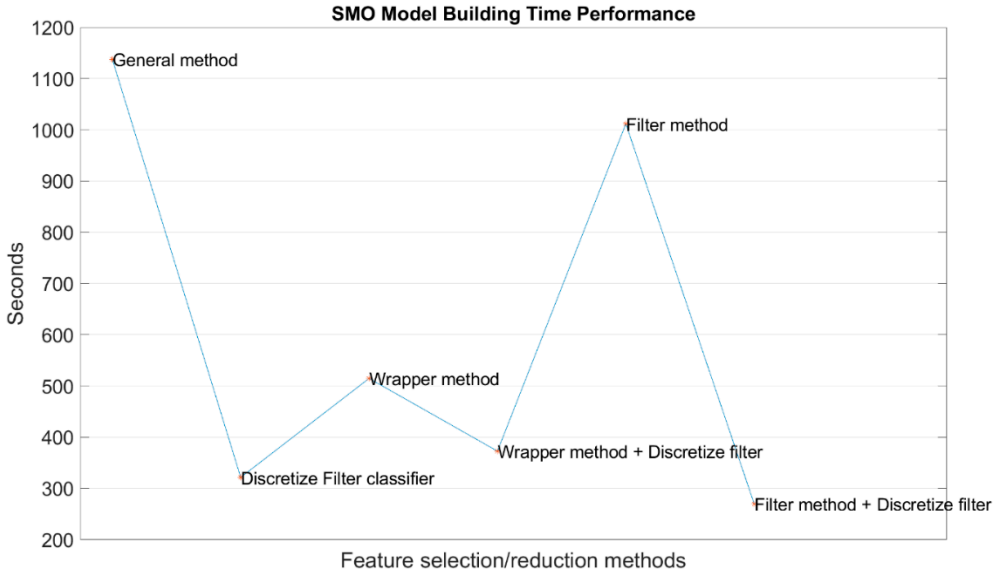

Figure 6. SMO model building time performance.

Table 9 demonstrates the performance of the RBF classifier based on various feature selection/reduction methods. It is clear that the discretize filter, the wrapper method in combination with the discretize filter, and the filter method in combination with the discretize filter increased

the model building time from 81.91 s at the initial reading to 163.07 s at the maximum. However, they enhanced the detection accuracy from 97.95% at the initial reading to 99.76% at the maximum. The discretize filter classifier improved the detection accuracy to 99.76% but compromised the model building time (163.07 s). Additionally, the wrapper method in combination with the discretize filter also enhanced detection accuracy to 99.48% but compromised the model building time (106.39 s). The filter method in combination with the discretize filter significantly improved the accuracy to 99.69% but compromised the model building time (127.68 s). Moreover, the wrapper method and the filter method significantly decreased the model building time but compromised the detection accuracy. Although the RBF classifier showed an increase in the model building time and maintained its high detection accuracy, it might still not be suitable for detecting intrusion in real-life situations. Figures 7 and 8 present the performance of the RBF classifier in terms of accuracy and model building time when used with different feature selection/reduction techniques.

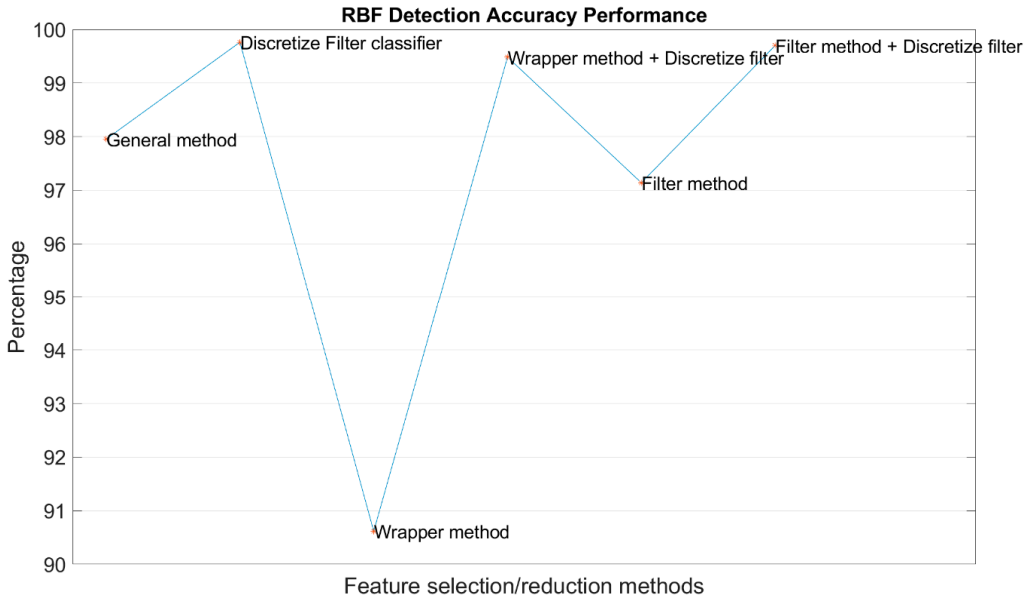

Figure 7. RBF detection accuracy performance.

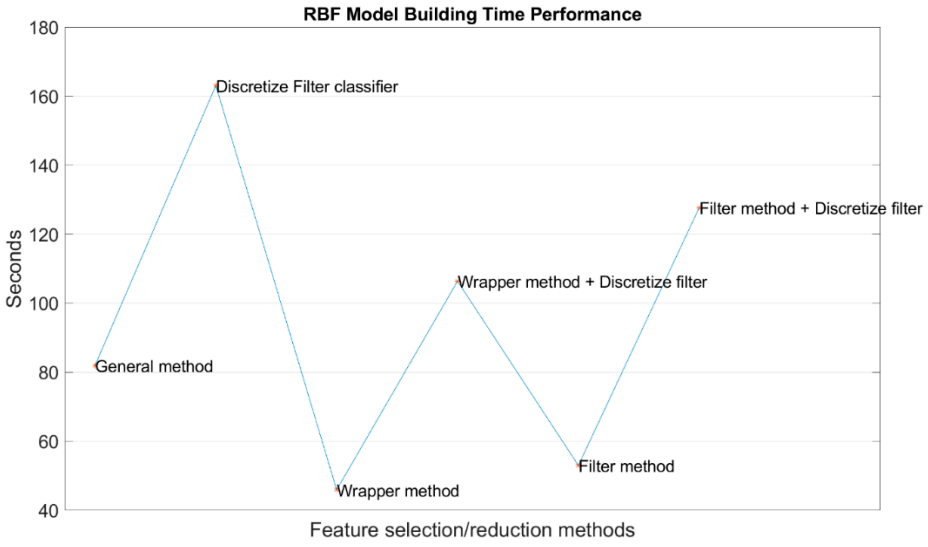

Figure 8. RBF model building time performance.

Table 10 shows that the performance of the BayesNet classifier was positively and negatively influenced by feature selection/reduction. The discretize filter significantly reduced the model building

time of the BayesNet classifier from 4.69 s at the initial reading to 0.41 s without compromising detection accuracy (97.18%). Furthermore, both the wrapper method in combination with the discretize filter and the filter method in combination with the discretize filter enhanced the model building time (1.68 s and 0.29 s, respectively) but compromised the detection accuracy (96.26% and 95.54%, respectively). Both the wrapper and filter methods also negatively affected the performance of the BayesNet classifier, decreased the detection accuracy, and increased the model building time. Thus, the authors suggest that using the BayesNet classifier with the discretize filter is appropriate to achieve low model building time with practical detection accuracy. Figures 9 and 10 show the performance of the BayesNet classifier in terms of accuracy and model building time when used with different feature selection/reduction methods.

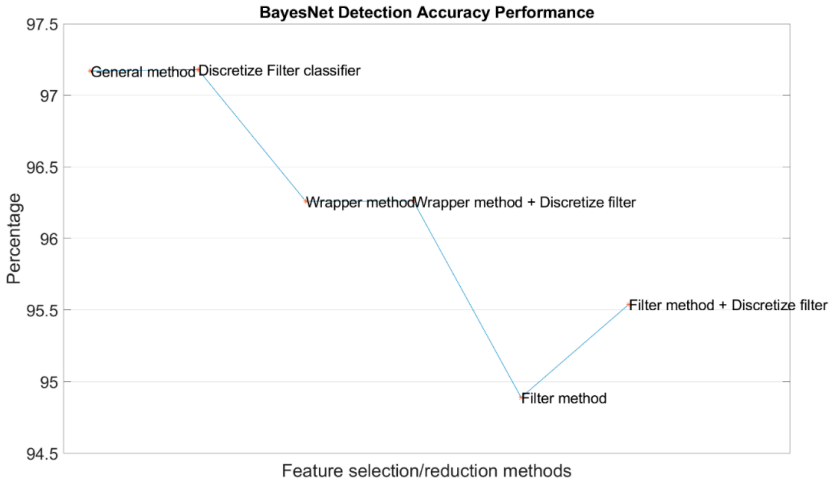

Figure 9. BayesNet detection accuracy performance.

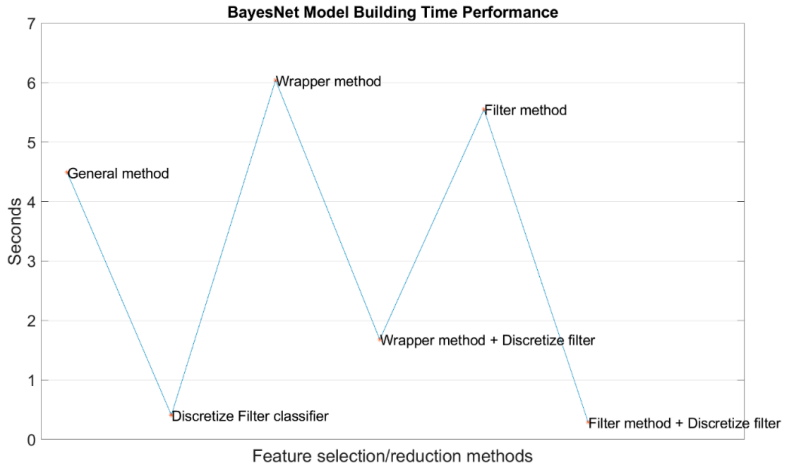

Figure 10. BayesNet model building time performance.

Table 11 presents the evaluation of the NBTree classifier performance based on different feature selection/reduction methods with or without the discretize filter. The feature selection/reduction methods, the discretize filter, and the combination of these methods significantly decreased the model building time from 213.18 s at the initial reading to a low reading at 8.7 s while slightly compromising the detection accuracy. The initial detection accuracy was 99.87% compared with the lowest detection accuracy at 99.46%, which was achieved by the wrapper method, and the highest detection accuracy was 99.85%, which was achieved by the discretize filter classifier. These results suggest using the NBTree classifier for a high level of detection accuracy with a reasonable model building time. Figures 11 and 12 present the performance of the NBTree classifier in terms of accuracy and model building time when used with different feature selection/reduction techniques.

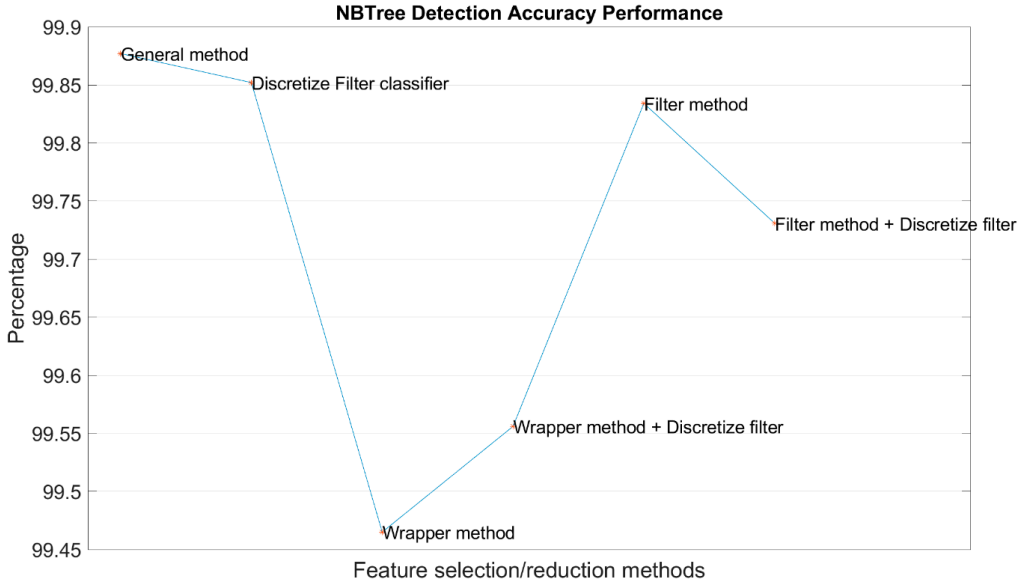

Figure 11. NBTree detection accuracy performance.

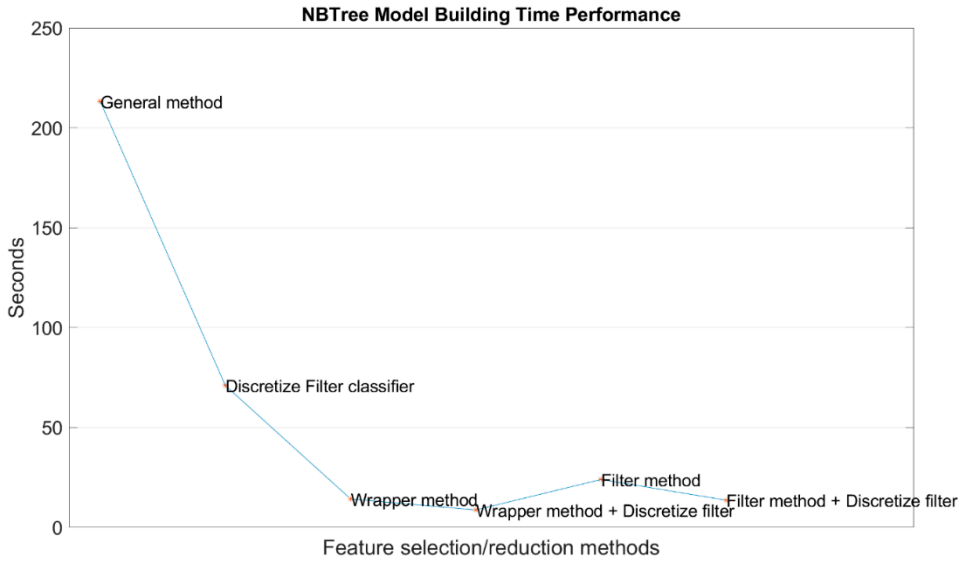

Figure 12. NBTree model building time performance.

From the above results, the authors confirm that feature selection/reduction methods affect the detection accuracy and model building time of classifiers when run on the full NSL-KDD training dataset. Most likely, the model building time of the tested classifiers was clearly further reduced by the use of the discretize filter, the wrapper method, the filter method, the wrapper method in combination with the discretize filter, and the filter method in combination with the discretize filter. Malhotra and Sharma [4] also reported this finding when they tested different classifiers. Researchers can further improve or slightly compromise detection accuracy with the discretize filter, the wrapper method in combination with the discretize filter, and the filter method in combination with the discretize filter (see Tables 6–11). Malhotra and Sharma [4] obtained similar results but with different tested classifiers.

The best classifier with lesser model building time and high detection accuracy was the REPTree classifier using the filter method in combination with the discretize filter (see Table 4). Its model building time was 0.6 s, and its detection accuracy was 99.73%. This implies that REPTree outperforms the J48, random forest, bagging, and PART classifiers that Malhotra and Sharma [4] reported. The results of this study support Malhotra and Sharma's [4] assumption that feature selection and reduction methods help a number of classifiers perform well.

7. Conclusions

In this study, the authors examined the performance of six supervised classifiers on the NSL-KDD training dataset using the Weka tool. The results demonstrate that the researchers can significantly reduce the model building time without compromising detection accuracy by using the discretize filter, the wrapper method in combination with the discretize filter, and the filter method in combination with the discretize filter. The REPTree classifier outperformed the other classifiers with the lowest model building time (0.6 s) and high detection accuracy (99.73%). This classifier used the filter method along with the discretize filter. In future work, the authors will suggest the evaluation of classifier performance on 20% of the NSL-KDD dataset grouped into classes of attacks for a more precise detection accuracy with lower time.

Author Contributions: S.A. conceived of the research concept, designed the experimental setting, performed the experiments, and wrote the manuscript. B.M. supervised, revised, and coordinated the overall research with revision and improvement. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This paper is part of a thesis of the first author at Dongguk University, Seoul, Korea.

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

References

1. Ghosh, P.; Biswas, S.; Shakti, S.; Phadikar, S. An improved intrusion detection system to preserve security in cloud environment. *Int. J. Inf. Secur. Priv.* **2020**, *14*, 67–80. [CrossRef]
2. Kunal, R.; Dua, M. Attribute selection and ensemble classifier based novel approach to intrusion detection system. *Procedia Comput. Sci.* **2020**, *167*, 2191–2199. [CrossRef]
3. Belavagi, M.; Muniyal, B. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Comput. Sci.* **2016**, *89*, 117–123. [CrossRef]
4. Malhotra, P.; Sharma, P. Intrusion detection using machine learning and feature selection. *Int. J. Comput. Netw. Inf. Secur.* **2019**, *4*, 43–52.
5. Thai, M.; Wu, W.; Xiong, H. *Big Data in Complex and Social Networks*; CRC Press, Kindle: Boca Raton, FL, USA, 2017.
6. Vamvakas, P.; Tsiropoulou, E.E.; Papavassiliou, S. Exploiting prospect theory and risk-awareness to protect UAV-assisted network operation. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 1–20. [CrossRef]
7. Jia, Y.; Wang, M.; Wang, Y. Network intrusion detection algorithm based on deep neural network. *IET Inf. Secur.* **2019**, *13*, 48–53. [CrossRef]
8. Li, J.; Zhao, Z.; Li, R. Machine learning-based IDS for software-defined 5G network. *IET Netw.* **2018**, *7*, 53–60. [CrossRef]
9. Dey, S.; Ye, Q.; Sampalli, S. A machine learning based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks. *Inf. Fusion* **2019**, *49*, 205–215. [CrossRef]
10. Leite, A.; Girardi, R. A hybrid and learning agent architecture for network intrusion detection. *J. Syst. Softw.* **2017**, *130*, 59–80. [CrossRef]
11. Hajisalem, V.; Babaie, S. A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Comput. Netw.* **2018**, *136*, 37–50. [CrossRef]
12. Li, Z.; Chen, J.; Fu, Y.; Hu, G.; Pan, Z.; Zhang, L. Community Detection Based on Regularized Semi-Nonnegative Matrix Tri-Factorization in Signed Networks. *Mob. Netw. Appl.* **2017**, *23*, 71–79. [CrossRef]
13. Li, B.; Ding, X.; Yang, H.; Liu, G.; Peng, X. Physical-Layer Network Coding Scheme over Asymmetric Rayleigh Fading Two-Way Relay Channels. *Mob. Netw. Appl.* **2017**, *23*, 80–85. [CrossRef]
14. Revathi, S.; Malathi, A. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *Int. J. Eng. Res. Technol.* **2013**, *2*, 1848–1853.
15. Dhanabal, L.; Shantharajah, S. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2015**, *4*, 446–452.

16. Chand, N.; Mishra, P.; Krishna, C.R.; Pilli, E.; Govil, M. A Comparative Analysis of SVM and its Stacking with other Classification Algorithm for Intrusion Detection. In Proceedings of the IEEE International Conference on Advances in Computing, Communication, & Automation, Dehradun, India, 8–9 September 2016. [CrossRef]
17. Ikram, S.; Cherukuri, A. Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *J. Comput. Inf. Technol.* **2016**, *24*, 133–148. [CrossRef]
18. Choudhury, S.; Bhowal, A. Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection. In Proceedings of the International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Avadi, Chennai, India, 6–8 May 2015. [CrossRef]
19. Biswas, S. Intrusion detection using machine learning: A comparison study. *Int. J. Pure Appl. Math.* **2018**, *118*, 101–114.
20. Wang, H.; Gu, J.; Wang, S. An effective intrusion detection framework based on SVM with feature augmentation. *Knowl.-Based Syst.* **2017**, *136*, 130–139. [CrossRef]
21. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
22. Abdullah, M.; Balamash, A.; Alshannaq, A.; Almabdy, S. Enhanced intrusion detection system using feature selection method and ensemble learning algorithms. *Int. J. Comput. Sci. Inf. Secur.* **2018**, *16*, 48–55.
23. Setiawan, B.; Djanali, S.; Ahmad, T.; Nopember, I.T.S. Increasing Accuracy and Completeness of Intrusion Detection Model Using Fusion of Normalization, Feature Selection Method and Support Vector Machine. *Int. J. Intell. Eng. Syst.* **2019**, *12*, 378–389. [CrossRef]
24. Zhou, Y.; Cheng, G.; Jiang, S.; Dai, M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Comput. Netw.* **2020**, *174*, 107247. [CrossRef]
25. Mahfouz, A.; Venugopal, D.; Shiva, S. Comparative analysis of ML classifiers for network intrusion detection. In Proceedings of the Fourth International Congress on Information and Communication Technology, London, UK, 3 January 2020. [CrossRef]
26. Singhal, C.; De, S. *Resource Allocation in Next-Generation Broadband Wireless Access Networks*; IGI Global: Delhi, India, 2017.
27. NSL-KDD Dataset for Network-Based Intrusion Detection Systems. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessed on 14 July 2020).
28. Ingre, B.; Yadav, A. Performance analysis of NSL-KDD dataset using ANN. In Proceedings of the IEEE International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96.
29. Weka Machine Learning Project. Available online: http://www.cs.waikato.ac.nz/~{}ml/weka/index.html (accessed on 27 July 2020).
30. Quinlan, J. Simplifying decision trees. *Int. J. Man-Mach. Stud.* **1987**, *27*, 221–223. [CrossRef]
31. Platt, J. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*; Technical Report; MSR-TR-98-14; Microsoft Research: Redmond, WA, USA, 1998.
32. Friedman, J.; Hastie, T.; Tibshirani, R. Additive logistic regression: A statistical view of boosting. *Ann. Stat.* **2000**, *28*, 337–374. [CrossRef]
33. John, G.H.; Langley, P. Estimating continuous distributions in Bayesian classifiers. In Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, Montreal, QU, Canada, 18–20 August 1995; Morgan Kaufmann Publisher: San Mateo, CA, USA, 1995; pp. 338–345.
34. Broomhead, D.S.; Lowe, D. Radial basis functions, multi-variable functional interpolation and adaptive networks. *Complex Syst.* **1988**, *2*, 321–355.
35. Kumar, S.; Naveen, D.C. A survey on improving classification performance using data preprocessing and machine learning methods on NSL-KDD data. *Int. J. Eng. Comput. Sci.* **2016**, *5*, 16156–16161.
36. Almomani, O. A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms. *Symmetry* **2020**, *12*, 1046. [CrossRef]