Effects of Feature Selection and Normalization on Network Intrusion Detection

Mubarak Albarka Umar[a,b,*], Zhanfang Chen[b], Khaled Shuaib[a], Yan Liu[c]

[a] *College of Information Technology, United Arab Emirates University, Al Ain, 15551, Abu Dhabi, United Arab Emirates*
[b] *School of CS and Technology, Changchun University of Science and Technology, Changchun, 130022, Jilin, China*
[c] *Department of Computer Science, College of Engineering, Shantou University, Shantou, 515063, Guangdong, China*

Abstract

The rapid rise of cyberattacks and the gradual failure of traditional defense systems and approaches led to using Machine Learning (ML) techniques to build more efficient and reliable Intrusion Detection Systems (IDSs). However, the advent of larger IDS datasets has negatively impacted the performance and computational complexity of ML-based IDSs. Many researchers used data preprocessing techniques such as feature selection and normalization to overcome such issues. While most of these researchers reported the success of these preprocessing techniques on a shallow level, very few studies have been performed on their effects on a wider scale. Furthermore, the performance of an IDS model is subject to not only the utilized preprocessing techniques but also the dataset and the ML algorithm used, which most of the existing studies give little emphasis on. Thus, this study provides an in-depth analysis of feature selection and normalization effects on various IDS models built using two IDS datasets namely, NSL-KDD and UNSW-NB15, and five different ML algorithms. The algorithms are support vector machine, k-nearest neighbor, random forest, naive bayes, and artificial neural network. For feature selection and normalization, the decision tree wrapper-based approach, which tends to give superior model performance, and min-max normalization methods were respectively used. A total of 30 unique IDS models were implemented using the full and feature-selected copy of the datasets. The models were evaluated using popular evaluation metrics in IDS modeling, intra- and inter-model comparisons were performed between models and with state-of-the-art works. Random forest achieved the best performance on both NSL-KDD and UNSW-NB15 datasets with prediction accuracies of 99.87% and 98.5%, as well as detection rates of 99.79% and 99.17% respectively, it also achieved an excellent performance in comparison with the recent works. The results show that both normalization and feature selection positively affect IDS modeling with normalization shown to be more important than feature selection in improving performance and computational time. The study also found that the UNSW-NB15 dataset is more complex and more suitable for building and evaluating modern-day IDS than NSL-KDD.

Keywords: Cybersecurity, Intrusion Detection System, Machine Learning, Feature Selection, Normalization, IDS Datasets, NSL-KDD and UNSW-NB15.

*Corresponding Author
Email addresses: 700040629@uaeu.ac.ae (Mubarak Albarka Umar), chenzhanfang@cust.edu.cn (Zhanfang Chen), k.shuaib@uaeu.ac.ae (Khaled Shuaib), yanliu@stu.edu.cn (Yan Liu)

1. Introduction

As the internet, networks, and computer systems play increasingly essential roles in our daily activities, they have become prime targets for cybercriminals. Consequently, it is imperative to explore optimal strategies to guarantee the security of our networks and systems. Initially, the first line of defense (i.e. firewall), user authentication, and data encryption are used, yet they have demonstrated inadequacies. Due to their limitations, Intrusion Detection Systems (IDSs) are now utilized to actively monitor computer and network intrusions. IDSs leverage distinctive analytical techniques to detect attacks, identify their sources, and promptly notify network administrators.

The two main approaches in IDSs are anomaly-based and signature-based detection. The signature-based detection has dominated IDS use in practice, however, the continual advent of new types of intrusion attacks, and the failure of the approach to detect those novel attacks made the signature-based IDS approach less reliable, and consequently, there is a growing interest among cybersecurity researchers in the anomaly-based IDS approach [1]. In the quest for developing more reliable and efficient IDS, ML techniques are generally used. The ML techniques require learning from experience, a dataset, in this case, to be able to correctly detect an intrusion attack. However, the advent of larger IDS datasets is increasing the computational complexity of developing ML-based IDS models, in addition to decreasing their performance. To overcome such issues, data preprocessing techniques such as feature selection and normalization are utilized by many researchers [2]. Feature selection is being widely used in selecting relevant features for building robust IDS models and is influential on both the efficiency and performance of IDS models [3]. Furthermore, the use of normalization in handling IDS dataset features with large value ranges has proven to be very influential on the implementation of IDS models, reducing learning time and improving IDS model performance [4, 5].

While most of the existing IDS studies aimed at utilizing either normalization or feature selection or both on IDS datasets using machine learning algorithms, very few in-depth studies were performed on the effects of those two preprocessing techniques. Furthermore, generalized conclusions on the positive effects of feature selection and/or normalization on the IDS models were typically reported in many shallow studies. However, this may not always be true because the performance of an IDS model is subject to not only feature selection and normalization but also the dataset and machine learning algorithms used, which most of the existing studies give little emphasis on. Thus, IDS models should be developed using various ML algorithms and more IDS datasets to enable fair comparison and a holistic understanding of the implication of feature selection and normalization in IDS modeling. Hence, this work aimed at addressing this gap.

An in-depth study of the effects of feature selection and normalization is performed in this work. Five of the most used ML algorithms in IDS modeling [6] are selected. Two datasets, one albeit considered outdated (NSL-KDD) yet often used by researchers, and the other considered current (UNSW-NB15) [7] are also selected. To determine the effects of feature selection, a feature-selected copy of each dataset is made using an optimal (wrapper-based) feature selection approach with a decision tree algorithm as the feature evaluator. To determine the effect of normalization, min-max normalization, one of the most common [8] and predominantly used normalization methods in IDS modeling [5, 9, 10] is used. Using the four final datasets (full and feature-selected copies) three different IDS programs are implemented, each program contains ten distinct IDS models with some of the models developed without applying normalization. Table 1 summarizes the three programs. The IDS models are evaluated using well-known and most-used evaluation metrics in IDS modeling.

Table 1: Programs Implemented

IDS Program	Dataset Features	Min-Max Normalized	Total IDS Models Built
Program A	All Features	Yes	10
Program B	Selected Features	Yes	10
Program C	Selected Features	No	10

In addition to the primary aim of this study, other important contributions are:

- Provides various in-depth comparisons on several aspects such as feature selection, normalization, datasets, and IDS models. A comparison with the state-of-the-art works is also made.
- Proposes a hybrid IDS modeling approach using a classifier for feature selection alongside another classifier in implementing the IDS model for better detection accuracy and efficiency.
- Dataset issue is one of the IDS challenges, though still oft-used, many consider the KDDcup99 dataset and its variant such as the NSL-KDD to be outdated and their usage a matter of concern [7], we contribute to the literature by verifying those claims; comparing it with a contemporary UNSW-NB15 dataset on effectiveness, reliability, and consistency aspects.
- The use of five of the most used ML algorithms in IDS modeling to implement many IDS models, and the use of many model evaluation metrics to assess and compare the performance of these models.

The rest of the paper is organized as follows: Section 2 presents a review of the existing works, their limitations, and ways of overcoming the limitations. Section 3 presents an explanation of some basic IDS concepts, Machine learning, and the preprocessing techniques used in this study. The experimental procedures and tools used are presented in Section 4, while Section 5 provides the evaluation results along with discussions. Section 6 concludes the study. Study limitations and directions for future research are identified in Section 7.

2. Literature Review

In this section, some of the recent and related works that made use of feature selection and normalization in modeling IDS using various approaches and ML methods are presented along with their limitations. A summary of the related works is shown in Table 2.

2.1. Related Works

Depren et al., [11] proposed a novel hybrid IDS model based on a self-organized map (SOM) for anomaly detection and a J48 tree for misuse detection on the KDDcup99 dataset. Some six basic features from 41 features were selected for modeling; however, no information about the used feature selection technique was provided. The attributes were normalized using the min-max technique and WEKA software was used for the modeling. The performance of the model was promising with a detection rate of 99.90%.

Wang et al., [4] modeled IDS on a normalized KDDCup99 dataset using three algorithms, namely, k-NN, PCA, and SVM. They used 34 numeric features, ignoring the remaining 7 nominal features of the dataset. Four different attribute normalization methods were employed and compared on the dataset for anomaly intrusion detection. The performances of the three models were evaluated based on detection rate, and false-positive rate; they found that Z-score (Statistical normalization) performs better on larger datasets than the rest of the normalization methods.

Somwang and Lilakiatsakun [12] proposed an anomaly-based IDS using a hybrid algorithm of supervised and unsupervised learning schemes on a non-zero normalized KDDcup99 dataset. The proposed technique integrates Principal Component Analysis (PCA) with Support Vector Machine (SVM). 10/41 features were selected using the PCA and the SVM was then used to model the IDS classifier. Hit, Miss, Detection rate and False positive rate were the performance measures used in evaluating the classifier. The experiment shows a detection rate of 97.4%, however, the authors suggested that more work needs to be done using various theories and techniques as one or two models can hardly provide a sufficient and reliable result.

Sivatha Sindhu et al., [13] propose a lightweight IDS for multi-class categorization using a wrapper-based genetic algorithm for feature selection and a hybrid of neural network and decision tree (neurotree) for actual classification. They used 16/41 features of NSL-KDD datasets and a min-max method to normalize the selected attributes. WEKA's evaluation measures were used to evaluate the performance of the models. Their proposed method achieved the highest detection rate of 98.38% in comparison to tree-based single classifiers.

Song et al., [14] proposed an IDS method consisting of a combination of feature selection, normalization, fuzzy C means clustering algorithm, and a C4.5 decision tree algorithm. They used the KDDcup99 dataset and selected 8/41 features using WEKA's CfsSubsetEval filter. Min-max normalization was used to convert the data to a range of between 0 and 1, then the fuzzy C means clustering algorithm was used to partition the training instances into clusters and for each cluster, a C4.5 algorithm was used for the detection of anomaly/normal instance on test data. The performance of the method was assessed using six measures and WEKA was used for comparison with a single C4.5 classifier, one with a feature selection algorithm and the other without. Their proposed method improves the performance results obtained by the C4.5 algorithm while using only 19.5% of the total number of features.

Thaseen and Kumar [15] evaluated the classification ability of six distinct tree-based classifiers on the NSL-KDD dataset. They used WEKA's CONS and CFS filters to select 15/41 features of the dataset, however, no normalization was done on the data (possibly because it has no impact on the performance of tree-based algorithms [16]). To evaluate the performance of the models, WEKA's evaluation measures were used and the RandomTree model holds the highest degree of accuracy and reduced false alarm rate.

Ghaffari Gotorlar et al., [17] proposed a harmony search-support vector machine (HS-SVM) method for intrusion detection on a KSL-KDD dataset. They used harmony search to select 21/41 best features and the numerical features were normalized using the min-max method whereas the nominal values were converted to numeric. LibSVM library was used for training the SVM model. Detection rate and test time were used to evaluate the model performance, and the results show that the proposed HS-SVM method overcomes the SVM drawback of being time-consuming during the testing phase.

Khammassi and Krichen [18] proposed the use of three distinct decision tree-based algorithms on a genetic algorithm-logistic regression wrapper selector (GALR-DT) in building IDS models. The three decision tree classifiers used are C4.5, Random Forest, and Naïve Bayes Tree. They applied a wrapper approach based on a genetic algorithm as a search strategy and logistic

regression as a learning algorithm to select the best subset of features on KDDcup99 and UNSW-NB15 datasets. 18/41 features were selected in KDDcup99, and 20/42 features were selected in UNSW-NB15 datasets by the GA-LR wrapper. Log-scaling and Min-max of the 0-1 range were applied to normalize the data. Dataset-wise performance of the models was compared using the detection rate, accuracy, and false alert rate. Their results show that UNSW-NB15 provides the lowest FAR of 6.39% and a good classification accuracy compared to KDDcup99 and thus, they conclude that the UNSW-NB15 dataset is more complex than the KDDcup99 dataset.

Setiawan et al., [19] proposed an IDS model using a combination of the feature selection method, normalization, and Support Vector Machine. WEKA's modified rank-based information gain filter was used to select 17/41 of the NSL-KDD dataset features and then the numerical features were log normalized. The model was evaluated using WEKA's evaluation measures and they achieved an overall accuracy of 99.8%.

Khan et al., [20] proposed a novel two-stage deep learning (TSDL) model, based on a stacked auto-encoder with a soft-max classifier, for efficient network intrusion detection. The model comprises two decision stages and is capable of learning and classifying useful feature representations in a semi-supervised approach. They evaluate the effectiveness of their methods using KDDcup99 and UNSW-NB15 datasets. DSAE feature selection was used to select 10 features in each dataset, which were then normalized using the min-max method. The most used IDS model evaluation metrics were used to assess the performance of their proposed model, achieving high recognition rates, up to 99.996%, and 89.134%, for the KDDcup99 and UNSW-NB15 datasets respectively.

2.2. Limitations of Related Works

After a thorough review of the related works, it's clear that each of the reviewed works suffers from one or more of the below listed five identified limitations. The limitation can be classified into three categories: preprocessing stage (transformation, and feature selection), modeling stage, and dataset issues.

1. Preprocessing: Transformation (encoding, discretization/normalization), Feature selection
2. Modeling Stage: Issues with classifier choice and fewer usage of multiple classifiers
3. Dataset Issues: Outdated datasets are used in most of the studies. No comparisons were made.

2.2.1. Data Encoding

Most ML algorithms cannot handle categorical features unless they are converted to numerical values. The categorical features can be nominal (no particular order) or ordinal (ordered). The performance of many algorithms varies based on how categorical features are encoded. For example, the `"Protocol_type"` feature of the NSL-KDD is a nominal feature with three values (UDP, TCP, and ICMP). By converting this attribute to a single numeric attribute using ordinal encoding, one is implicitly introducing an ordering over the nominal values, which is a bad representation of the data. This mistake can be seen in some of the reviewed literature [14, 17, 18, 20]. A better solution is to use binary encoding or, even better, one-hot (dummy) encoding that maps each category to a vector containing 1 and 0 denoting the presence or absence of the feature's value.

Table 2: Summary of Related Works

Ref.	FS Method (no. features)	Algorithm	Normalization	Dataset / ID approach	Evaluation Metrics
[11]	Not mentioned (6/41)	SOM/J.48, DSS (Weka)	Minmax (0-1)	KDD99/ Hybrid	Detection rate, False positive rate, and Missed rate
[4]	Not used, selected numeric features only (34/41)	PCA, k-NN, SVM	Z-score, Ordinal, Minmax, and Frequency	KDD99/ Anomaly	Accuracy, Detection rate, and False positive rate
[12]	PCA (10/41)	SVM	Non-zero	KDD99/ Anomaly	Detection rate, False positive rate, Mis, hit
[13]	GA (16/41)	Hybrid of Neurotree	Minmax	NSL-KDD/ Anomaly	TP Rate, FP Rate, Precision, Recall, F-Measure
[14]	Weka's Filters (8/41)	Fuzzy C/ C4.5 (Weka)	Minmax (0-1)	KDD99/ Anomaly	True positive rate, False positive rate, Precision, Recall, F-score
[15]	CFS & CON Filters (18/41)	Tree-based Classifiers	Not mentioned	NSL-KDD/ Anomaly	Accuracy, TP Rate, FP Rate, Precision, Recall, F-Measure
[17]	Harmony Search (20/41)	SVM (LibSvm)	Min-max (1-13)	NSL-KDD/ Anomaly	Detection rate, Test Time
[18]	GA-LR KDD99 (18/41) UNSW-NB (20/42)	Random Forest, C4.5, and Naïve Bayes Tree	Log-scaling, Minmax (0-1)	KDD99, UNSW-NB15 /Anomaly	Confusion Matrix, Accuracy, Detection rate, False alarm rate
[19]	IG Weka's Filter (17/41)	SVM	Log-norm	NSL-KDD	Accuracy, Sensitivity, Specificity, False, and True positive
[20]	DSAE (10/45)	Soft-max classifier	Min-max (0-1)	KDD99, UNSW-NB15 / Anomaly	Accuracy, precision, recall, F-measure, and false alarm rate (FAR)

2.2.2. Data Discretization and Normalization

While the discretization of numerical features is influential in data preprocessing [21], however, unlike normalization, it generally leads to a loss of information [22]. Normalization is an important data preprocessing step that can improve the accuracy and efficiency of classification algorithms, especially in the case of IDS models built with large datasets [4, 5, 23]. Although either or both can be applied, in the case of IDS where the data contains a wide range of traffic values, normalization is indispensable, and discretization alone should not be used as there is less need for value range. Even so, some studies choose to use discretization at the expense of normalization [15]. Furthermore, most reviewed studies give little emphasis on the impact of normalization on the performance of IDS models.

2.2.3. Feature Selection

Due to the high dimensionality and size of IDS datasets, many researchers use dimensionality reduction methods to reduce dataset dimension and select an optimal subset of features. This re-

duces computational time, and resource utilization, and increases the accuracy and performance of IDS models [24]. Three basic feature selection methods are explained in the next section. However, only two of these methods were mainly applied in IDS modeling, with most studies using the filter method, which generally ignores the effects of the selected feature subset on the performance of the IDS model [25]. Contrary to the wrapper method, which, though computationally expensive, produces better performance for the predefined classifier. Furthermore, some studies hand-pick certain features without using any feature selection methods, which may lead to removing influential features [4, 11].

2.2.4. Modeling Stage

To develop an accurate and good IDS Model, it is necessary to explore various algorithms and techniques, as using one or two algorithms may not offer reliable and good-performing IDS models [12]. However, most studies use only one or two algorithms. Furthermore, there is excessive usage of tree-based algorithms in some studies without validating their performances by comparing them with other algorithms [15, 18]. Unfortunately, tree-based algorithms, like ensemble trees such as a random forest, generally do not have the same level of predictive accuracy as some regression and classification algorithms [16].

2.2.5. Dataset Issues

Most reviewed literature made use of either KDDcup99 or its variant NSL-KDD, which are widely used datasets in IDS academic research [6, 7]. However, they are considered outdated and do not contain contemporary attacks [2, 26]. In the current environment of continually emerging new threats, building reliable and accurate IDS models requires using an up-to-date ID dataset. Some modern datasets were proposed by [27, 28, 29]. Ring et al. [30] also recommended a selected few datasets suitable for general network intrusion detection evaluation. Both the proposed and recommended datasets are publicly available and can be used for building better and more reliable IDS models. Furthermore, Ring et al. [30] recommended using more than one dataset with at least one publicly available dataset to avoid overfitting the IDS model to one dataset and ensure the reproducibility of the work and its generic evaluation. However, most studies used only one dataset.

In summary, this study addressed those limitations. Since most selected ML algorithms consider all features during training simultaneously, one-hot encoding, an approach suited for such ML algorithms [31], is used. Furthermore, Min-max, one of the most common normalization methods [8], is also used. Five widely used ML algorithms in IDS modeling [6] are selected, and as recommended, two datasets, one considered outdated (NSL-KDD) and the other considered current (UNSW-NB15) [7], are also selected for this study. The decision tree wrapper-based feature selection approach is used to select the best optimal subsets from the datasets. A total of thirty models were developed and evaluated. In the subsequent section, the five selected ML algorithms are explained, and the feature selection concept is also introduced.

3. Basic Theory and Related Knowledge

3.1. Intrusion Detection System (IDS)

An IDS is a software program or a device that monitors traffic passing across networks and through systems for malicious behavior, policy violations, and the presence of known threats, sending alarms when such things are encountered. IDS are security tools that, like other measures

such as firewalls, antivirus software, and access control schemes, are intended to strengthen the security of information and communication systems [32]. An IDS can be classified in two ways: based on data source/location and detection approach [33]. Based on the data source, Network Intrusion Detection Systems (NIDS) and Host Intrusion Detection Systems (HIDS) are the most well-known classifications. At the most basic level, NIDS looks at network traffic, while HIDS looks at actions and files on the host computers. Based on the detection approach, the most well-known types are Misuse-based (recognizing registered bad patterns) and Anomaly-based (detecting deviations from a model of "good" traffic, which often relies on machine learning) [34]; the former can only detect known attack types and the latter is prone to generate false positive alerts. Due to the complementary nature of these two approaches, a hybrid approach, combining both of these techniques, is often used [35]. The literature nowadays focuses on developing a wide variety of automated, fast, and efficient IDSs using expert-crafted rules, sophisticated statistical learning, and machine learning techniques [2, 32].

3.2. Machine Learning (ML)

Machine Learning (ML) algorithms are the most widely used techniques in designing IDSs [6]. The ML techniques are based on establishing an explicit or implicit model that enables classifying patterns in raw data. The use of ML techniques in IDSs can be with single, hybrid, or ensemble classifiers. The used classifier can be categorized into three operating modes: supervised, unsupervised, and semi-supervised. Generally, the supervised mode outperforms the remaining modes [3, 32]. Some of the ML algorithms used in IDSs include artificial neural networks, k-nearest neighbor, Naive Bayes, Genetic Algorithms, Support Vector Machines, Logistic Regression, and Decision Trees. Developing a ML model consists of four basic steps, namely, data collection, data preprocessing, model selection and training, and model evaluation [36]. The two important concepts in this work, feature selection, and normalization are among the many tasks performable in the data preprocessing step.

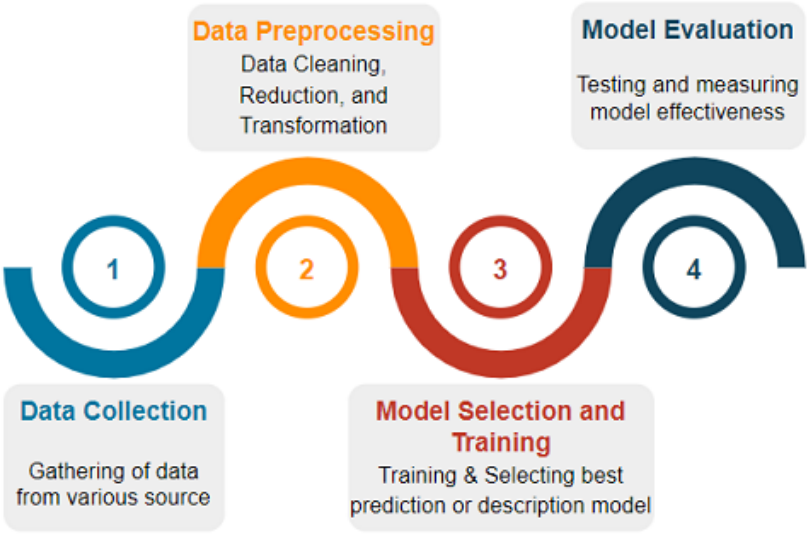

Figure 1: Machine Learning Basic Steps

3.2.1. Feature selection

This is a data reduction technique that involves selecting a subset of relevant features for building a model, without changing the dimensions of the features. Feature selection simplifies a model and improves classification accuracy and generalization while reducing overfitting chances and model training time. Feature subset selection requires a search strategy and direction to select a feature subset, an objective function to evaluate the selected features, a termination condition, and an evaluation of the result. There are three main feature selection approaches: (a) the filters that extract features from data without involving any learning algorithm, (b) the wrappers that use a learning algorithm to determine useful features, (c) the embedded techniques that combine the two mentioned approaches and a classifier [37]. In this work, the wrapper method is used.

3.2.2. Normalization

This is a data transformation technique that is used to transform wide-range numeric values in a dataset to a common scale without distorting differences in the range of the values. Normalizing data attempts to give all attributes an equal weight. Normalization helps speed up the model training stage and is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest-neighbor classification and clustering algorithms [8]. There are many normalization methods, some of the most used methods are min-max normalization, z-score normalization, and decimal scaling [38]. In this work, Min-max normalization is used, its general formula is as follows:

$$x_{\text{new}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

3.3. Selected ML Algorithms

A collection of the most used ML algorithms in IDS is provided in [6] and five among them are selected and used in this study. The algorithms are explained below.

3.3.1. Support Vector Machine (SVM)

A support vector machine is a supervised learning algorithm that uses hyperplane graphing to analyze new, unlabeled data. They are mostly utilized for classification problems but can also be used for regression modeling and outlier detection. SVMs are well known for their generalization capability and are mainly valuable when the number of features is larger than the number of samples [26]. In this work, the Scikit-learn implementation of a support vector classifier based on LibSVM with the Radial Basis Function (RBF) as the kernel is utilized.

3.3.2. Artificial Neural Network (ANN)

ANN is a computational model composed of interconnected artificial neurons capable of learning from their inputs to perform tasks without following any task-specific rules. ANNs aim to realize a very simplified model of the human brain [39]. There are three main ANN classes: Feedforward, Convolutional, and Recurrent neural networks (NNs). ANNs are used in IDS, mainly because of their flexibility and adaptability to environmental changes [32]. In this work, a Multi-layered perceptron (MLP), which is a widely used feedforward neural network, is used.

3.3.3. K-Nearest Neighbor (KNN)

The KNN algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It computes the approximate distances between different points on the input vectors and then assigns the unlabeled point to the class of its k-nearest neighbors. The assignment depends on the task K-NN is used for; in classification, the output is a class membership assigned to neighbors with the highest vote, whereas, in regression, the output is the property value for the object. This value is the average of the values of the k nearest neighbors [40].

3.3.4. Random Forests (RF)

Random forests are an ensemble learning method that operates by randomly creating and merging multiple decision trees at training time into a ”forest” and outputting the class result. For a classification task, the mode of classes is the result, whereas for regression, the result is the mean prediction of the individual trees. RF uses bagging ensembling methods to combine the decision tree's simplicity with the flexibility to increase accuracy and overcome the decision tree's habit of overfitting to its training set [41].

3.3.5. Naive Bayes (NB)

A Naive Bayes classifier is a form of probabilistic classifier inspired by the Bayes theorem with a simple assumption of independence among features. It aims to process, analyze, and categorize outcomes based on probabilities of their occurrence in training data. Naive Bayes classifiers require a small amount of training data to estimate the necessary parameters. The NB model is easy to build and particularly scalable to larger datasets since it takes linear time. NB is a popular baseline method for text categorization, and with appropriate pre-processing, it is competitive with more advanced methods, including support vector machines [42].

4. Methodology

This section explains how the experiment is conducted by following the four basic ML steps outlined above. It also provides the tools used in the experiment.

4.1. Experimental Tools

In the literature, several tools are used for implementing, evaluating, and comparing various IDS works. WEKA, general-purpose programming languages (such as Java, Python, etc.), and Matlab are the most used tools [6]. In this work, Excel, WEKA, and Python are used for data analysis and exploration, preprocessing, as well as for implementing and validating the IDS models respectively. Google Colaboratory with Python 3.10.12 version is used as the execution environment for Python and its libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, and so on. All the programs are implemented and executed in the same environment.

4.2. Dataset Acquisition

In this work, two datasets: the UNSW-NB15 dataset and, an old benchmark dataset, the NSL-KDD are used to evaluate and compare the models.

4.2.1. UNSW-NB15 dataset

The UNSW-NB15 dataset is a new IDS dataset created at the Australian Center for Cyber Security (ACCS) in 2015. About 2.5 million samples or 100GB of raw data were captured in modern network traffic including normal and attack behaviors and are simulated using the IXIA Perfect Storm tool and a tcpdump tool. 49 features were created using the Argus tool, the Bro-IDS tool, and 12 developed algorithms. The created features can be categorized into five groups: flow features, basic features, content features, time features, and additional generated features. The dataset has nine different modern attack types, five more attack types than NSL-KDD, the attacks are Backdoor, DoS, Generic, Reconnaissance, Analysis, Fuzzers, Exploit, Shellcode, and Worms [28]. The UNSW-NB15 is considered a new benchmark dataset that can be used for IDS evaluation by the NIDS research community [43] and is recommended by [30]. For easy use and work reproducibility, the UNSW-NB15 comes along with predefined splits of a training set (175,341 samples) and a testing set (82,332 samples) [44], however, the publicly available training and testing set both contain only 44 features: 42 attributes and 2 classes. Only the training set (UNSW NB15 training set) is used for both training and testing in this work. Since our primary focus is binary classification, the broad distribution of total attacks (anomaly) and normal traffic samples of the training set used is shown in Table 3.

Table 3: UNSW-NB15 Distribution Sample

Category	**Sample Size**	**Distribution**
Total Attacks	119,341	68.06%
Normal	56,000	31.94%
Overall Samples	175,341	100%

4.2.2. NSL-KDD dataset

The KDDcup99 dataset is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining in 1999. The dataset contains more than 5 million training samples and more than 2 million testing samples. It also has a huge number of redundant samples, and imbalance classes [45]. The NSL-KDD is an optimized version of the KDDcup99 dataset [46], it removes redundant records and provides reasonable and diversified samples in training and testing sets. Like the KDDcup99, the NSL-KDD dataset also has 41 features, with 3 categorical features and 38 numeric features. The dataset has four different attack types: Denial of Service (DoS), Probe, User to Root (U2R), and Root to Local (R2L) attacks. The NSL-KDD dataset is considered to be outdated [2]. The NSL-KDD is also arranged into a training set of 125,973 samples (KDDTrain+) and a testing set of 22,544 samples (KDDTest+). Only the training set (KDDTrain+) is used for both training and testing in this work. Table 4 summarizes the sample's distribution of all attacks (anomaly) and normal traffic in the training set of the NSL-KDD dataset.

4.3. Data Preprocessing

In this study, two major preprocessing steps are used, namely, data reduction (filtration and feature selection) and data transformation (data normalization and encoding).

Table 4: NSL-KDD Distribution Sample

Category	Sample Size	Distribution
Total Attacks	58,630	46.54%
Normal	67,343	53.46%
Overall Samples	125,973	100%

4.3.1. Data Reduction

Data Filtration: Irrelevant data was removed to reduce computational time and prepare the data for feature selection. The UNSW-NB15 dataset comes with 42 attributes, 2 class attributes, and an additional id attribute, the id is removed. Since we are interested in binary classification, the class attribute attack_cat indicating the categories of attacks and normal state is also removed before feature selection. No issues were found with the NSL-KDD, thus no filtration was done on the dataset. Both the UNSW-NB15 and the NSL-KDD datasets are divided into train and test sets of unique samples in a proportion of 2/3 (66.7%) and 1/3 (33.3%) respectively as shown in Figure 3. To avoid developing overfitted models that might perform poorly when given out-of-sample data, WEKA's unsupervised instance Resample filter is used to ensure balanced splitting of the data.

Feature Selection: In feature selection, avoiding information leakage and subsequent building of misleading models is very important [47], thus only the training set is used for feature selection, while the testing set is solely used for performance assessment to ensure getting a reliable model capable of detecting abnormal intrusion attack in a real-world scenario. The wrapper-based approach, though computationally expensive, tends to give superior model performance [48] and is employed in this work with a decision tree algorithm as the feature evaluator as shown in Figure 2.

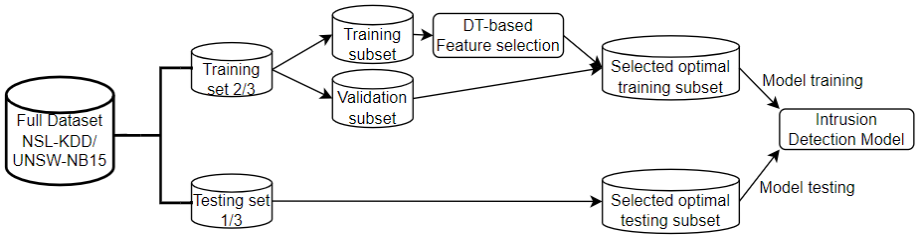

Figure 2: DT Wrapper-Based FS

Scikit-learn implementation of feature selection was initially considered but because their current decision tree implementation does not support categorical features [49] and encoding the categorical features will result in removing some of the then values, now features, of encoded features thereby leading to partial feature selection and losing count on the actual number of selected features, thus the WEKA's implementation is used instead [50]. The J48, a Java implementation of Quinlan's C4.5 [51] decision tree algorithm [47] is used as the feature evaluator. BestFirst Forward search strategy is used in feature search with 5 consecutive non-improving nodes as the search-stopping criteria, and accuracy as the evaluation measure. After performing the feature selection, twenty (20) and nineteen (19) features were the best optimal features for UNSW-NB15 and NSL-KDD respectively. The WEKA's supervised attribute Remove filter was used to collect the feature subsets. Thus, two more datasets are derived bringing our total datasets

to four: the 2 full datasets and 2 feature selected versions of UNSW-NB15 and NSL-KDD, a description of the full datasets is available in [28] and [46] respectively, Table 5 shows the selected optimal features of the datasets.

Table 5: Selected Optimal Features

No.	UNSW-NB15 Feature	No.	NSL-KDD Feature
2	proto	1	duration
3	service	3	service
4	state	4	flag
5	spkts	5	src_bytes
7	sbytes	6	dst_bytes
8	dbytes	11	num_failed_logins
11	dttl	14	root_shell
14	sloss	17	num_file_creations
15	dloss	23	count
17	dinpkt	24	srv_count
18	sjit	25	serror_rate
27	smean	26	srv_serror_rate
31	ct_srv_src	27	rerror_rate
32	ct_state_ttl	32	dst_host_count
33	ct_dst_ltm	34	dst_host_same_srv_rate
34	ct_src_dport_ltm	35	dst_host_diff_srv_rate
36	ct_dst_src_ltm	38	dst_host_serror_rate
39	ct_flw_http_mthd	39	dst_host_srv_serror_rate
40	ct_src_ltm	40	dst_host_rerror_rate
41	ct_srv_dst		

4.3.2. Data Transformation

Data Normalization: The full and formed datasets consist of two types of features: numeric and nominal. To avoid classifier bias towards numeric features with large value ranges, normalization is performed on all the numeric features across the four datasets. Min-max normalization is applied to normalize all the numeric features within a range of 0 to 1 using equation (1) above. The normalization process is performed after feature selection in order not to affect the feature selection process.

Data Encoding: All the categorical (nominal) features across the datasets are one-hot encoded. In the NSL-KDD dataset, three features (protocol_type, service, and flag) are nominal and are one-hot encoded. Table 6 shows an example of how the protocol_type feature is encoded from 6(a) to 6(b). This procedure maps the 41-dimensional features into 122-dimensional features: 38 continuous and 84 with encoded binary values of the 3 categorical features (protocol_type, service, and flag). The encoding is also performed on the NSL-KDD feature-selected version that has only two (service and flag) nominal features. Similarly, both the UNSW-NB15 and its feature-selected version have three nominal features (proto, service, and state) and were all one-hot encoded accordingly.

Table 7 provides a summary of the four datasets' dimensions before and after encoding. Because one-hot encoding increases the dataset dimension, to avoid losing some nominal features' values encoded in the feature selection process, the encoding is performed after the feature selection and normalization processes. Only the final encoded features are used in training and

Table 6: One-Hot Encoding Example

(a)

Protocol_type
UDP
TCP
ICMP

(b)

UDP	TCP	ICMP
1	0	0
0	1	0
0	0	1

evaluating the models.

Table 7: Final Datasets Dimensions

One-Hot Encoding	UNSW-NB15 dataset features		NSL-KDD dataset features	
	All	Feature selected	All	Feature selected
Before Encoding	42	20	41	20
After Encoding	194	172	122	98

4.4. Model Selection and Training

Model selection is the process of choosing one among many candidate models for a predictive problem using various methods [41]. Each of the datasets is divided into two sets, called the training set and the testing set. For a statistically optimal model, we used the 10-fold GridSearchCV method on the training dataset to build several models and select the optimal model. The building of the models consists of two stages: training and testing stage. During the training stage, the algorithms are trained using the training set to build the models, then in the testing stage, the testing set is used to assess the performance of the built IDS models. Figure 3 depicts the entire model training and testing process. A total of thirty (30) distinct IDS models are developed using the 5 selected algorithms. To measure the impact of normalization and feature selection as well as the effectiveness of IDS datasets, some evaluation metrics are used to evaluate and compare the models. The evaluation metrics and the results of the evaluations are provided in section 4.5 and 5 respectively. A summary of the three different IDS model implementations performed is as follows:

1. **Program A:** models implemented with full, non-feature selected, normalized datasets.
2. **Program B:** models implemented with feature-selected, normalized datasets.
3. **Program C:** models implemented with feature-selected, non-normalized datasets.

4.5. Model Evaluation Metrics

This is a criterion by which the performance of a model can be assessed. The performance of an IDS model can be measured based on its ability to correctly classify network traffic as anomalous or normal. Most of the existing IDS works used either some or all of the following three metrics: classification accuracy, detection rate (DR), and false alarm rate (FAR) [26]. Similarly, in this work, the same metrics are adopted in addition to computational time. The confusion matrix and the metrics are explained below. The confusion matrix in itself is not a performance measure per se, but because all the evaluation metrics used in this study (except for computational time) are based on the Confusion Matrix, we deem it important to explain it.

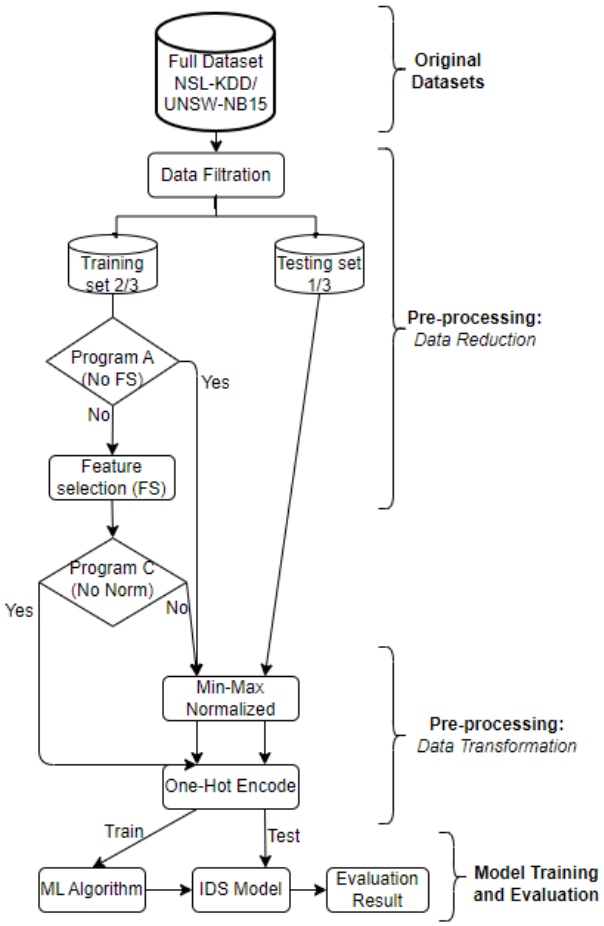

Figure 3: IDS Modeling Conceptual Framework

4.6. Confusion Matrix

The confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of a model. It is used for classification problems where the output can be of two or more types of classes [52]. No confusion matrix is included in this work due to the number of implemented models.

Table 8: Confusion Matrix

		Predicted Class	
		Anomaly	Normal
Actual Class	Anomaly	TP (Good: Correct detection)	FN (Bad: Incorrect prediction)
	Normal	FP (Bad: Incorrect detection)	TN (Good: Correct prediction)

Basic Confusion Matrix terminologies:

- **True positive (TP):** Number of attacks correctly detected as an attack.

- **False negative (FN):** Number of attacks incorrectly detected as normal. Aka Type II error.
- **False positive (FP):** Number of normal instances incorrectly detected as an attack. Aka Type I error.
- **True negative (TN):** Number of normal instances correctly detected as normal.

4.7. Accuracy (ACC)

Accuracy is the amount of correctly classified instances of the total instances, defined as the ratio of the number of correct predictions to the total number of predictions. It is suitable to use on a dataset with symmetric target classes and equal class importance [52]. Both training (ACC_{Tr}) and testing (ACC_{Pr}) accuracies are reported in this study.

$$Accuracy\ (ACC) = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

4.8. Detection Rate (DR)

Aka Recall, Sensitivity, Hit rate, or True positive rate (TPR), it is the measure of correctly identified positive (anomaly) instances from all the actual positive instances, defined as the ratio of correct positive predictions to the total number of positive predictions. Or more simply, how sensitive the classifier is for detecting positive instances. The higher its value the better [52].

$$Detection\ Rate\ (DR) = \frac{TP}{TP + FN} \tag{3}$$

4.9. False Alert Rate (FAR)

Aka Fall-out or False positive rate (FPR) is the measure of incorrectly classified negative (normal) instances as positive (anomaly) from all the actual negative instances or defined as the proportion of negative prediction that is mistakenly considered as positive (anomaly) for all negative predictions. The lower its value the better [52].

$$False\ Alert\ Rate\ (FAR) = \frac{FP}{FP + TN} \tag{4}$$

4.10. Computational Time

In this work, the computational time is the entire time taken to train, evaluate, and test a model excluding the time taken in the feature selection process. Note that training time is derived from using 10-fold cross-validation, which may result in substantial computational time for many algorithms.

5. Result and Discussion

This chapter presents the results obtained as well as a discussion of the results. The following comparisons are made:

1. Comparison between the IDS models developed using full datasets and feature-selected datasets to assess the impact of feature selection.
2. Comparison between the IDS models developed using normalized and non-normalized feature-selected datasets to assess the impact of normalization.
3. Comparison to measure the effectiveness, reliability, and complexity of the IDS datasets.
4. Comparison of the best-performing IDS model with state-of-the-art works to assess the significance of the result.

5.1. Feature Selection

Table 9 presents the results of models built using both NSL-KDD and UNSW-NB15 full features and feature-selected data. Feature selection generally improves performances on new unseen data and reduces computational complexity [9, 10, 14]. With NSL-KDD, it can be seen that in comparison to the performance of models built using full features, the models built using the 19 selected features achieved similar performances albeit slightly lower (this is consistent with [18]). Whereas with UNSW-NB15, the performance accuracy of three of the models (RF, KNN, and NB) improved on both new unseen data and using the 20 selected features with both KNN and RF achieving a remarkably higher performance across all the metrics (except time). ANN and SVM models achieved similar performance both on the selected features and full features. It can thus be seen that feature selection improves the performances of models implemented with UNSW-NB15 more than those implemented with NSL-KDD; and in both datasets, RF benefitted the most from the feature selection. In both datasets, the NB models achieved the best and lowest computational time whereas SVM achieved the opposite.

Table 9: Feature Selection Comparison

Models	Metrics	NSL-KDD Features		UNSW-NB15 Features	
		Full	FS	Full	FS
ANN	ACCTr	99.59	98.84	94.44	94.27
	ACCPr	99.65	98.95	94.49	94.48
	DR	99.61	98.72	97.06	97.93
	FAR	0.31	0.85	11.03	12.88
	Time	15.82m	22.27m	59.33m	47.26m
SVM	ACCTr	98.53	98.01	93.63	93.51
	ACCPr	98.56	98.06	93.67	93.56
	DR	98.13	97.18	96.63	96.54
	FAR	1.08	1.17	15.14	15.19
	Time	60.83m	86.39m	136.9m	276.1m
KNN	ACCTr	99.52	99.11	93.80	94.82
	ACCPr	99.57	99.07	93.82	95.83
	DR	99.49	99.12	96.26	97.35
	FAR	0.36	0.98	11.42	7.41
	Time	14.29m	13.75m	37.64m	41.49m
RF	ACCTr	99.79	99.71	95.86	96.1
	ACCPr	99.81	99.75	95.78	98.49
	DR	99.71	99.73	97.87	99.16
	FAR	0.1	0.23	8.72	2.93
	Time	1.5m	1.94m	3.37m	3.8m
NB	ACCTr	85.68	84.59	48.14	48.13
	ACCPr	85.69	84.81	48.08	48.11
	DR	69.5	67.61	23.91	23.76
	FAR	0.22	0.22	0.02	0.01
	Time	10.44s	11.25s	23.4s	23.57s

5.2. Normalization

Table 10 presents the results of models built using NSL-KDD and UNSW-NB15 normalized and non-normalized feature-selected data. Normalization typically improves performance and

decreases computation time [4, 5]. In the case of NSL-KDD, the performance of three of the five models (ANN, SVM, and NB) were all improved by normalization with the performances of the distance-related classifier, SVM, benefiting the most, this is as expected [40, 53]. However, normalization does not have much effect on both KNN and RF models built using NSL-KDD. NB, which is not a distance-based classifier also achieves two opposing results; with NSL-KDD, its performance was hugely improved by normalization, however, with UNSW-NB15, the normalization negatively affects its performance. Normalization does not have huge effects on RF, as it performed well achieving similar performances across the two datasets. The performance of ANN across the two datasets is relatively better after normalization. Overall, normalization has improved the performance of 4 classifiers on UNSW-NB15 as well as the performance of 3 of the 5 classifiers on NSL-KDD. Thus it can be inferred that normalization, although its importance in IDS tasks is often ignored [4], does certainly improve model performances in IDS. These findings are consistent with Wang et al., [4], who compare four different normalizations for anomaly intrusion detection using SVM, PCA, and KNN. Both RF and NB do not necessarily need normalization [40, 54], hence their computational time with and without normalization are low and correspondingly close. Similarly, in both normalized and non-normalized versions of each dataset, the NB models achieved the best and lowest computational time whereas SVM achieved the opposite. Overall, the sum of computational time taken by the normalization-based models is considerably lower than that of the models built with non-normalized data in both NSL-KDD and UNSW-NB15 as shown in Table 11. Thus, normalization also reduces computational time in addition to improving performance.

5.3. Dataset Comparison

To evaluate the reliability and complexity of the datasets, we consider two perspectives. Firstly, a more general close observation of the models' performances in Table 9 and 10, specifically focusing on models whose performances or computation time were rather surprising or did not show similar effects after performing related actions to the similar models on corresponding datasets. Secondly, although the feature selection process on both datasets is the same, more features were selected in UNSW-NB15 (20) than in NSL-KDD (19), so for fair and transparent comparisons, we consider the performances of the models implemented using full and normalized features (Program A) since the same normalization is performed on both full datasets.

5.3.1. Reliability Comparison

Since KNN typically uses distance measures to find k nearest points from any given point, using the normalized features should generally enable all features to be of equal importance thereby improving its performance [40]. However, while normalization does improve KNN performance with UNSW-NB15, the reverse is seemingly the case with NSL-KDD across all the metrics. Moreover, while the poor performances of SVM and NB on NSL-KDD in Table 10 can easily be attributed to a lack of normalization, however, a closer look at how SVM performed without normalization with UNSW-NB15 implies that the poor performances have more to do with the dataset itself. As seen in Table 9, the feature selection did not significantly improve the performance of KNN and RF with NSL-KDD, however, with UNSW-NB15, the performances of both KNN and RF across all the metrics were improved.

As shown in Table 11, a clear decrease in models' computational time in the non-normalized datasets (Program C) can be observed after normalizing the datasets (in Program B), however, the overall computational time taken by models built using full normalized features (Program A) is

Table 10: Normalization Comparison

Models	Metrics	NSL-KDD Features		UNSW-NB15 Features	
		Normalized	Non-normalized	Normalized	Non-normalized
ANN	ACCTr	98.84	95.62	94.27	92.8
	ACCPr	98.95	95.55	94.48	93.45
	DR	98.72	96.71	97.93	97.99
	FAR	0.85	5.45	12.88	16.23
	Time	22.27m	3.92m	47.26m	39.7m
SVM	ACCTr	98.01	53.3	93.51	74.43
	ACCPr	98.06	53.8	93.56	75.59
	DR	97.18	0.31	96.54	92.88
	FAR	1.17	0.08	19.19	62.31
	Time	86.39m	256.3m	276.1m	749.8m
KNN	ACCTr	99.11	99.38	94.82	93.58
	ACCPr	99.07	99.42	95.83	94.89
	DR	99.12	99.47	97.35	97.07
	FAR	0.98	0.63	7.41	9.74
	Time	13.75m	11.77m	41.49m	35.57m
RF	ACCTr	99.71	99.86	96.1	96.07
	ACCPr	99.75	99.87	98.49	98.5
	DR	99.73	99.79	99.16	99.17
	FAR	0.23	0.05	2.93	2.94
	Time	1.94m	1.37m	3.8m	2.82m
NB	ACCTr	84.59	53.51	48.13	52.34
	ACCPr	84.81	53.41	48.11	52.3
	DR	67.61	1.71	23.76	32.25
	FAR	0.22	1.57	0.01	4.97
	Time	11.25s	10.12s	23.57s	20.08s

Table 11: Computational Time Summary

	A (Norm)	**B (FS + Norm)**	**C (FS)**	Total
NSL-KDD	92.61m	124.54m	273.53m	490.68m
UNSW-NB15	237.63m	369.04m	828.22m	1434.89m
Total	330.24m	493.58m	1101.75m	

somewhat lower this shows that, on average, feature selection has less impact than normalization on computational complexity. Overall, the performance of models built using both datasets is somewhat consistent and is reliably achieved via a 10-fold CV.

5.3.2. Complexity Comparison

As explained above, the performances of Program A models (implemented using full and normalized features) are used for the comparison. Figure 4, Figure 5, and Figure 6 provide a summary of the comparisons on Accuracy, Detection rate, and False alert rate respectively.

In Figure 4, all NSL-KDD models outperform their corresponding UNSW-NB15 implemented models. In both datasets, RF achieved the highest prediction accuracy of 99.81% and 95.78% for NSL-KDD and UNSW-NB15 respectively, while NB achieved the worst prediction accuracy in both datasets. Furthermore, in the detection rate shown in Figure 5, which stands for

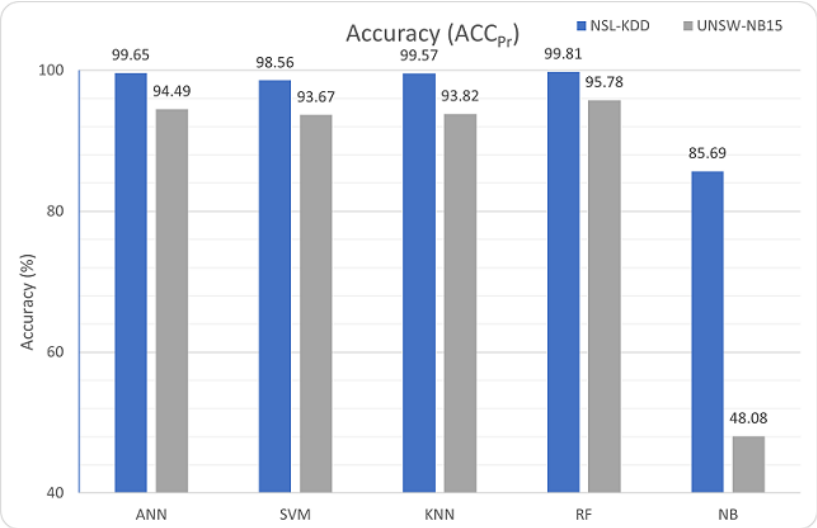

Figure 4: Program A Models Accuracy

the accuracy rate for the attack classes, the NSL-KDD models were generally able to detect more attacks than the UNSW-NB15 models. RF again achieves the overall highest DR of 99.71% with NSL-KDD and 97.87% with UNSW-NB15 Once more, NB achieved the lowest DR for both datasets with its UNSW-NB15 model achieving the poorest DR of just 23.76%.

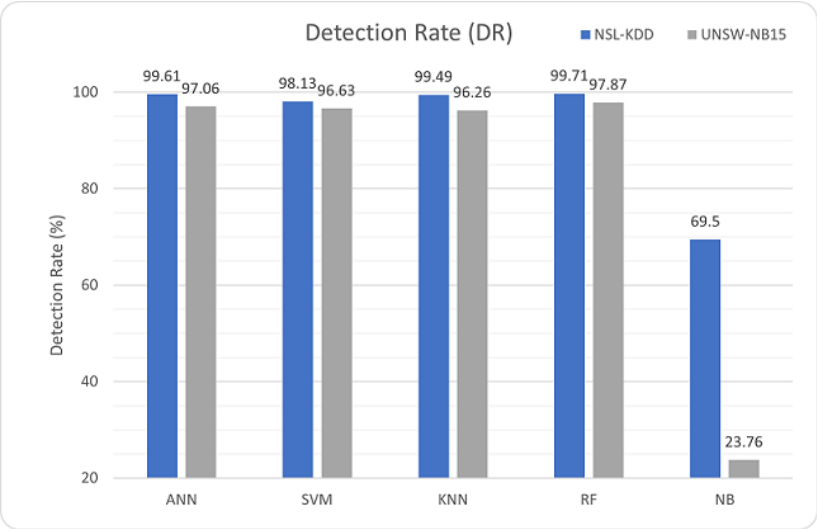

Figure 5: Program A Models Detection Rate

A high detection rate and very low false alert rates are generally the targets in IDS. The FAR depicted in Figure 6 shows notable alert rate differences between the NSL-KDD and UNSW-NB15 models. All the UNSW-NB15 models, except NB which has the lowest FAR of 0.02%,

have higher percentages of false alerts than their corresponding NSL-KDD models. The highest FAR of 15.14% is achieved by SVM on UNSW-NB15. The summation of the FAR percentage for all the NSL-KDD models is 2.07, averaging 0.414% per model, whereas for the UNSW-NB15 models is 46.33 percentage, averaging 9.266% FAR per model, this is 22.38 times less than the average of NSL-KDD models.

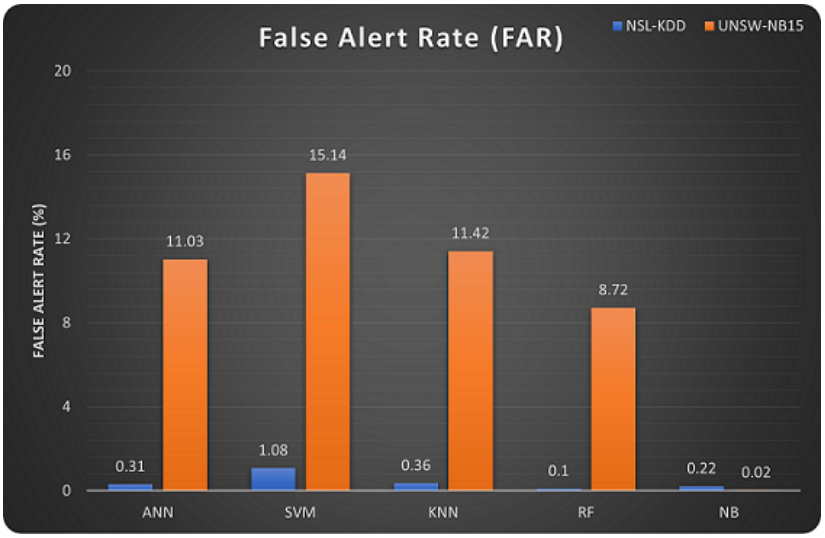

Figure 6: Program A Models False Alert Rate

Overall, the prediction accuracy, DR, and FAR of models built with NSL-KDD appear to be higher than those built using UNSW-NB15, a typical understanding is that NSL-KDD is indeed better; however, it can be observed that UNSW-NB15, unlike NSL-KDD which have few and outdated attack families, contains different modern low footprint attack families which exhibit similar behavior to normal network traffic, this will essentially make it difficult for many classifiers to accurately detect its attack patterns. This reflects, precisely, the contemporary real-world network traffic scenarios, thus UNSW-NB15 can be considered more complex and reliable for evaluating modern-day IDSs than NSL-KDD. The current real-world environment is much more challenging than the ones depicted by the outdated NSL-KDD dataset.

5.4. Comparison with Similar Works

To assess the effectiveness of the models, we compare the best-performing models in each dataset with the recent studies utilizing similar datasets and feature selection in the modeling process. The comparison was performed based on the major evaluation metrics as depicted in Table 12.

From Table 12, our methods achieve the best prediction accuracies (ACC) of 99.87% and 98.5% with NSL-KDD as well as detection rates (DR) of 99.79% and 99.17% with the UNSW-NB15 dataset. The best false alert rate (FAR) is achieved by [57] for NSL-KDD and [20] for UNSW-NB15 using ANN and soft-max classifier respectively. Their low FAR results may be driven by their use of neural network-based classifiers which are proving to be good in IDS classification tasks recently [59]. With the lowest ACC of 85.56% and second to-worst FAR

Table 12: State-of-the-art Comparisons

Ref	**Dataset**	**FS**	**Algorithm**	**ACC**	**DR**	**FAR**
[19]	NSL-KDD	Modified-RIGFS	SVM	99.6	99.7	0.566
[55]	NSL-KDD	IG-Filters	Voting classifier	86.67	86.7	0.12
[56]	NSL-KDD	NSGAII-ANN	RF	99.4	99.4	6
[57]	NSL-KDD	MCF	ANN	98.81	97.25	0.02
[58]	NSL-KDD	PIO	DT	88.3	86.6	8.8
This work	NSL-KDD	DT-based	RF	99.87	99.79	0.05
[20]	UNSW-NB15	DSAE	Soft-max classifier	89.13	-	0.75
[37]	UNSW-NB15	DT-based	RF	86.41	97.95	27.73
[56]	UNSW-NB15	NSGAII-ANN	RF	94.8	94.8	6
[43]	USWN-NB15	-	DT	85.56	-	15.78
This work	UNSW-NB15	DT-based	RF	98.5	99.17	2.94

(15.78%) achieved by [43], which is the only work not utilizing feature selection, this comparison further highlights the importance of feature selection in addition to demonstrating the effectiveness of RF in IDS modeling. We achieved second best FAR in each of the datasets, however, it is important to note that in IDS, not detecting an attack can be more harmful than misclassifying normal traffic [60], thus DR can be more important than FAR and any other metrics and hence, our method can be considered as quite good and very effective for real-world scenarios.

6. Conclusions

Network and computer systems are continually facing increasing attacks while existing protective mechanisms are failing, thus more effort should be exerted towards analyzing and improving such protective methods, as well as in developing more sophisticated ones to secure networks and systems and address current challenging environments. This study analyzes the effects of feature selection and normalization techniques on NSL-KDD and UNSW-NB15 IDS datasets using five machine learning algorithms. Accuracy, Detection rate, False alert rate, and Computational time were used to evaluate and compare the models. The following conclusions can be deduced:

- Both normalization and feature selection are indispensable in building effective and efficient IDS models faster, as they both improve the performance and computational time of models built with the two datasets. Normalization is shown to be more important than feature selection, especially when using algorithms such as SVM [53] for better accuracy.

- Generally, RF models achieved remarkably higher performances across all datasets, hence RF is the most robust among the oft-used ML algorithms in IDS. SVM requires normalization and takes higher computational time than the other algorithms. NB models take the lowest time but achieve the lowest performances on average, making it the least effective algorithm for modeling IDS in this work and the poorest in comparison to many supervised ML algorithms [61].

- Compared to NSL-KDD, UNSW-NB15, which contained more modern low-footprint attack families, is found to be more complex and suitable for building and evaluating modern-

day IDS than NSL-KDD, which contained fewer outdated attack families. Thus, we recommend using UNSW-NB15 for building reliable IDS models.

- It is interesting to note that most of the reviewed works using methods other than one-hot encoding generally achieve lower performance results compared to our work. Thus, although its effect is not quite clear, the use of the One-hot encoding method certainly influenced the classifiers' performances. In future work, it will be interesting to study the effect of various encoding methods in IDS modeling.

7. Limitations and Future Work

While the overall results have shown great promises and distinctive conclusions, particularly concerning the effects of the preprocessing techniques and the UNSW-NB15 and NSL-KDD datasets, there exist some limitations and open gaps for future research as follows:

- Dataset and Algorithms: Among the many available IDS dataset benchmarks and machine learning algorithms, only two datasets are used in this study with five algorithms. It would be interesting to use more datasets and more algorithms for broader insights.

- Alternative feature selection and normalization: There are three feature selection methods; the Wrapper method is considered in this study. However, other methods can be explored in future studies. The same wrapper method can also be used in a different approach, such as changing the evaluation algorithm or the search strategy. Similarly, the use of different normalization methods can also be investigated.

- Multi-classification: This work primarily focused on the binary classification of normal and attack network traffic; however, IDS datasets typically contain diverse attack types. Thus, multi-classification work can be done to enable further analysis of the effect of preprocessing techniques on distinctive classes of attacks.

- Reduction of FAR: A good IDS should have high detection rates and very low false alert rates. While the models built with the UNSW-NB15 datasets generally achieved higher Detection Rates (DR), a higher number of False Alerts (FAR) is also observable. Although this highlights the complexity of the dataset, further work can be performed to reduce it.

Declaration of competing interest

The authors declare that there are no conflicts of interest.

Acknowledgement

The authors wish to thank numerous reviewers whose comments and feedback have contributed to and improved this work.

Notes

A preliminary version of this paper appeared as a preprint [62]. In addition to some slight corrections, the paper is significantly improved.

References

[1] T. Sowmya, E. Mary Anita, A comprehensive review of AI based intrusion detection system, Measurement: Sensors 28 (2023) 100827. doi:10.1016/j.measen.2023.100827.
URL https://linkinghub.elsevier.com/retrieve/pii/S2665917423001630

[2] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, Q. G. Chen, A Survey of Intrusion Detection Systems Leveraging Host Data, ACM Computing Surveys 52 (6) (2019) 128:1–128:35. doi:10.1145/3344382.
URL https://dl.acm.org/doi/10.1145/3344382

[3] K. Albulayhi, Q. Abu Al-Haija, S. A. Alsuhibany, A. A. Jillepalli, M. Ashrafuzzaman, F. T. Sheldon, IoT Intrusion Detection Using Machine Learning with a Novel High Performing Feature Selection Method, Applied Sciences 12 (10) (2022) 5015. doi:10.3390/app12105015.
URL https://www.mdpi.com/2076-3417/12/10/5015

[4] W. Wang, X. Zhang, S. Gombault, S. J. Knapskog, Attribute Normalization in Network Intrusion Detection, in: 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, 2009, pp. 448–453, iSSN: 2375-527X. doi:10.1109/I-SPAN.2009.49.
URL https://ieeexplore.ieee.org/document/5381578

[5] S. M. Kasongo, Y. Sun, Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset, Journal of Big Data 7 (1) (2020) 105. doi:10.1186/s40537-020-00379-6.
URL https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00379-6

[6] A. Özgür, H. Erdem, A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015, Tech. Rep. e1954v1, PeerJ Preprints (Apr. 2016). doi:10.7287/peerj.preprints.1954v1.
URL https://peerj.com/preprints/1954

[7] K. Siddique, Z. Akhtar, F. Aslam Khan, Y. Kim, KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research, Computer 52 (2) (2019) 41–51. doi:10.1109/MC.2018.2888764.
URL https://ieeexplore.ieee.org/document/8672520

[8] S. Kumar, S. Gupta, S. Arora, A comparative simulation of normalization methods for machine learning-based intrusion detection systems using KDD Cup'99 dataset, Journal of Intelligent & Fuzzy Systems 42 (3) (2022) 1749–1766. doi:10.3233/JIFS-211191.
URL https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/JIFS-211191

[9] Y. Zhou, G. Cheng, S. Jiang, M. Dai, Building an efficient intrusion detection system based on feature selection and ensemble classifier, Computer Networks 174 (2020) 107247. doi:10.1016/j.comnet.2020.107247.
URL https://linkinghub.elsevier.com/retrieve/pii/S1389128619314203

[10] S.-H. Kang, K. J. Kim, A feature selection approach to find optimal feature subsets for the network intrusion detection system, Cluster Computing 19 (1) (2016) 325–333. doi:10.1007/s10586-015-0527-8.
URL http://link.springer.com/10.1007/s10586-015-0527-8

[11] O. Depren, M. Topallar, E. Anarim, M. K. Ciliz, An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks, Expert Systems with Applications 29 (4) (2005) 713–722. doi:10.1016/j.eswa.2005.05.002.
URL https://linkinghub.elsevier.com/retrieve/pii/S0957417405000989

[12] P. Somwang, W. Lilakiatsakun, Computer network security based on Support Vector Machine approach, in: 2011 11th International Conference on Control, Automation and Systems, 2011, pp. 155–160, iSSN: 2093-7121.
URL https://ieeexplore.ieee.org/document/6106397

[13] S. S. Sivatha Sindhu, S. Geetha, A. Kannan, Decision tree based light weight intrusion detection using a wrapper approach, Expert Systems with Applications 39 (1) (2012) 129–141. doi:10.1016/j.eswa.2011.06.013.
URL https://linkinghub.elsevier.com/retrieve/pii/S0957417411009080

[14] J. Song, Z. Zhu, P. Scully, C. Price, Selecting features for anomaly intrusion detection: A novel method using fuzzy c means and decision tree classification, in: G. Wang, I. Ray, D. Feng, M. Rajarajan (Eds.), Cyberspace Safety and Security, Springer International Publishing, Cham, 2013, pp. 299–307.

[15] S. Thaseen, C. A. Kumar, An analysis of supervised tree based classifiers for intrusion detection system, in: 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, IEEE, 2013, pp. 294–299. doi:10.1109/ICPRIME.2013.6496489.
URL http://ieeexplore.ieee.org/document/6496489/

[16] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning: with Applications in R, Springer Texts in Statistics, Springer US, New York, NY, 2021. doi:10.1007/978-1-0716-1418-1.
URL https://link.springer.com/10.1007/978-1-0716-1418-1

[17] H. G. Gotorlar, M. P. Aghababa, J. Bagerzadeh, M. S. Osalu, Improving intrusion detection using a novel normalization method along with the use of harmony search algorithm for feature selection, in: 2015 7th Conference on Information and Knowledge Technology (IKT), 2015, pp. 1–6. doi:10.1109/IKT.2015.7288796.
URL https://ieeexplore.ieee.org/document/7288796/

[18] C. Khammassi, S. Krichen, A GA-LR wrapper approach for feature selection in network intrusion detection, Computers & Security 70 (2017) 255–277. doi:10.1016/j.cose.2017.06.005.
URL https://www.sciencedirect.com/science/article/pii/S0167404817301244

[19] B. Setiawan, S. Djanali, T. Ahmad, Increasing Accuracy and Completeness of Intrusion Detection Model Using Fusion of Normalization, Feature Selection Method and Support Vector Machine, International Journal of Intelligent Engineering and Systems 12 (4) (2019) 378–389. doi:10.22266/ijies2019.0831.35.
URL http://www.inass.org/2019/2019083135.pdf

[20] F. A. Khan, A. Gumaei, A. Derhab, A. Hussain, A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection, IEEE Access 7 (2019) 30373–30385. doi:10.1109/ACCESS.2019.2899721.
URL https://ieeexplore.ieee.org/document/8643036

[21] S. Ramírez-Gallego, S. García, H. Mouriño-Talín, D. Martínez-Rego, V. Bolón-Canedo, A. Alonso-Betanzos, J. M. Benítez, F. Herrera, Data discretization: taxonomy and big data challenge, WIREs Data Mining and Knowledge Discovery 6 (1) (2016) 5–21. doi:10.1002/widm.1173.
URL https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1173

[22] R. Jin, Y. Breitbart, C. Muoh, Data Discretization Unification, in: Seventh IEEE International Conference on Data Mining (ICDM 2007), 2007, pp. 183–192, iSSN: 2374-8486. doi:10.1109/ICDM.2007.35.
URL https://ieeexplore.ieee.org/document/4470242/

[23] M. Azizjon, A. Jumabek, W. Kim, 1D CNN based network intrusion detection with normalization on imbalanced data, in: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2020, pp. 218–224. doi:10.1109/ICAIIC48513.2020.9064976.
URL https://ieeexplore.ieee.org/document/9064976

[24] K. A. Taher, B. Mohammed Yasin Jisan, M. M. Rahman, Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection, in: 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), IEEE, Dhaka, Bangladesh, 2019, pp. 643–646. doi:10.1109/ICREST.2019.8644161.
URL https://ieeexplore.ieee.org/document/8644161/

[25] M. A. Umar, Z. Chen, Y. Liu, A Hybrid Intrusion Detection with Decision Tree for Feature Selection, Information & Security: An International Journal (2021). doi:10.11610/isij.4901.
URL https://isij.eu/article/hybrid-intrusion-detection-decision-tree-feature-selection

[26] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, Survey of intrusion detection systems: techniques, datasets and challenges, Cybersecurity 2 (1) (2019) 20. doi:10.1186/s42400-019-0038-7.
URL https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7

[27] W. Haider, J. Hu, J. Slay, B. Turnbull, Y. Xie, Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling, Journal of Network and Computer Applications 87 (2017) 185–192. doi:10.1016/j.jnca.2017.03.018.
URL https://linkinghub.elsevier.com/retrieve/pii/S1084804517301273

[28] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: 2015 Military Communications and Information Systems Conference (MilCIS), IEEE, Canberra, Australia, 2015, pp. 1–6. doi:10.1109/MilCIS.2015.7348942.
URL http://ieeexplore.ieee.org/document/7348942/

[29] G. Creech, J. Hu, Generation of a new IDS test dataset: Time to retire the KDD collection, in: 2013 IEEE Wireless Communications and Networking Conference (WCNC), 2013, pp. 4487–4492, iSSN: 1558-2612. doi:10.1109/WCNC.2013.6555301.
URL https://ieeexplore.ieee.org/document/6555301

[30] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho, A survey of network-based intrusion detection data sets, Computers & Security 86 (2019) 147–167. doi:10.1016/j.cose.2019.06.005.
URL https://www.sciencedirect.com/science/article/pii/S016740481930118X

[31] P. Cerda, G. Varoquaux, B. Kégl, Similarity encoding for learning with dirty categorical variables, Machine Learning 107 (8) (2018) 1477–1494. doi:10.1007/s10994-018-5724-2.
URL https://doi.org/10.1007/s10994-018-5724-2

[32] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, A. Y. Zomaya, An explainable deep learning-enabled intrusion detection framework in IoT networks, Information Sciences 639 (2023) 119000. doi:10.1016/j.ins.2023.119000.
URL https://linkinghub.elsevier.com/retrieve/pii/S0020025523005856

[33] M. Alkasassbeh, S. Al-Haj Baddar, Intrusion Detection Systems: A State-of-the-Art Taxonomy and Survey, Arabian Journal for Science and Engineering 48 (8) (2023) 10021–10064. doi:10.1007/s13369-022-07412-1.
URL https://doi.org/10.1007/s13369-022-07412-1

[34] D. H. Lakshminarayana, J. Philips, N. Tabrizi, A Survey of Intrusion Detection Techniques, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), IEEE, Boca Raton, FL, USA, 2019,

pp. 1122–1129. doi:10.1109/ICMLA.2019.00187.
URL https://ieeexplore.ieee.org/document/8999075/

[35] M. R. Ayyagari, N. Kesswani, M. Kumar, K. Kumar, Intrusion detection techniques in network environment: a systematic review, Wireless Networks 27 (2) (2021) 1269–1285. doi:10.1007/s11276-020-02529-3.
URL https://link.springer.com/10.1007/s11276-020-02529-3

[36] A. L. Buczak, E. Guven, A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, IEEE Communications Surveys & Tutorials 18 (2) (2016) 1153–1176. doi:10.1109/COMST.2015.2494502.
URL https://ieeexplore.ieee.org/document/7307098

[37] M. A. Umar, C. Zhanfang, Y. Liu, Network Intrusion Detection Using Wrapper-based Decision Tree for Feature Selection, in: Proceedings of the 2020 International Conference on Internet Computing for Science and Engineering, ACM, Male Maldives, 2020, pp. 5–13. doi:10.1145/3424311.3424330.
URL https://dl.acm.org/doi/10.1145/3424311.3424330

[38] J. Han, J. Pei, H. Tong, Data mining: concepts and techniques, fourth edition Edition, The Morgan Kaufmann series in data management systems, Morgan Kaufmann is an imprint of Elsevier, Cambridge, MA, United States, 2023, oCLC: on1346308160.

[39] M. T. Hagan, H. B. Demuth, M. H. Beale, O. De Jésus, Neural network design, 2nd Edition, Martin T. Hagan, s.L, 2014.

[40] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (1) (2008) 1–37. doi:10.1007/s10115-007-0114-2.
URL https://doi.org/10.1007/s10115-007-0114-2

[41] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer, New York, NY, 2009. doi:10.1007/978-0-387-84858-7.
URL http://link.springer.com/10.1007/978-0-387-84858-7

[42] B. P. Yadav, S. Ghate, A. Harshavardhan, G. Jhansi, K. S. Kumar, E. Sudarshan, Text categorization Performance examination Using Machine Learning Algorithms, IOP Conference Series: Materials Science and Engineering 981 (2) (2020) 022044. doi:10.1088/1757-899X/981/2/022044.
URL https://iopscience.iop.org/article/10.1088/1757-899X/981/2/022044

[43] N. Moustafa, J. Slay, The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, Information Security Journal: A Global Perspective 25 (1-3) (2016) 18–31. doi:10.1080/19393555.2015.1125974.
URL http://www.tandfonline.com/doi/full/10.1080/19393555.2015.1125974

[44] N. Moustafa, The UNSW-NB15 data set description (2015).
URL https://research.unsw.edu.au/projects/unsw-nb15-dataset

[45] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6, iSSN: 2329-6275. doi:10.1109/CISDA.2009.5356528.
URL https://ieeexplore.ieee.org/document/5356528

[46] C. Institute, NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB (2009).
URL https://www.unb.ca/cic/datasets/nsl.html

[47] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal (Eds.), Data mining: practical machine learning tools and techniques, fourth edition Edition, Morgan Kaufmann, Amsterdam, 2016.

[48] M. Samadi Bonab, A. Ghaffari, F. Soleimanian Gharehchopogh, P. Alemi, A wrapper-based feature selection for improving performance of intrusion detection systems, International Journal of Communication Systems 33 (12) (2020) e4434. doi:10.1002/dac.4434.
URL https://onlinelibrary.wiley.com/doi/10.1002/dac.4434

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
URL https://scikit-learn.org/stable/modules/tree.html

[50] G. Holmes, A. Donkin, I. Witten, WEKA: a machine learning workbench, in: Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference, IEEE, Brisbane, Qld., Australia, 1994, pp. 357–361. doi:10.1109/ANZIIS.1994.396988.
URL http://ieeexplore.ieee.org/document/396988/

[51] S. L. Salzberg, C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993, Machine Learning 16 (3) (1994) 235–240. doi:10.1007/BF00993309.
URL https://doi.org/10.1007/BF00993309

[52] G. Shobha, S. Rangaswamy, Machine Learning, in: Handbook of Statistics, Vol. 38, Elsevier, 2018, pp. 197–228.

doi:10.1016/bs.host.2018.07.004.
URL https://linkinghub.elsevier.com/retrieve/pii/S0169716118300191

[53] C.-W. Hsu, C.-C. Chang, C.-J. Lin, A Practical Guide to Support Vector Classication (2016).
URL https://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf

[54] E. Lewinson, Python for finance cookbook: over 50 recipes for applying modern Python libraries to finance data analysis, Packt, Birmingham, UK, 2020, oCLC: 1139921653.

[55] M. Abdullah, A. Balamash, A. Alshannaq, S. Almabdy, Enhanced Intrusion Detection System using Feature Selection Method and Ensemble Learning Algorithms, International Journal of Computer Science and Information Security (IJCSIS) 16 (02) (2018) 48–55.
URL https://sites.google.com/site/ijcsis/

[56] A. Golrang, A. M. Golrang, S. Yildirim Yayilgan, O. Elezaj, A Novel Hybrid IDS Based on Modified NSGAII-ANN and Random Forest, Electronics 9 (4) (2020) 577. doi:10.3390/electronics9040577.
URL https://www.mdpi.com/2079-9292/9/4/577

[57] S. Sarvari, N. F. Mohd Sani, Z. Mohd Hanapi, M. T. Abdullah, An Efficient Anomaly Intrusion Detection Method With Feature Selection and Evolutionary Neural Network, IEEE Access 8 (2020) 70651–70663. doi:10.1109/ACCESS.2020.2986217.
URL https://ieeexplore.ieee.org/document/9058689/

[58] H. Alazzam, A. Sharieh, K. E. Sabri, A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer, Expert Systems with Applications 148 (2020) 113249. doi:10.1016/j.eswa.2020.113249.
URL https://linkinghub.elsevier.com/retrieve/pii/S0957417420300749

[59] R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran, S. Akarsh, Application of Deep Learning Architectures for Cyber Security, in: A. E. Hassanien, M. Elhoseny (Eds.), Cybersecurity and Secure Information Systems, Springer International Publishing, Cham, 2019, pp. 125–160. doi:10.1007/978-3-030-16837-7_7.

[60] A. Kaushik, H. Al-Raweshidy, A novel intrusion detection system for internet of things devices and data, Wireless Networks 30 (1) (2024) 285–294. doi:10.1007/s11276-023-03435-0.
URL https://link.springer.com/10.1007/s11276-023-03435-0

[61] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: Proceedings of the 23rd International Conference on Machine Learning, ICML '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 161–168. doi:10.1145/1143844.1143865.
URL https://doi.org/10.1145/1143844.1143865

[62] M. A. Umar, C. Zhanfang, Effects of Feature Selection and Normalization on Network Intrusion Detection (Jun. 2020). doi:10.36227/techrxiv.12480425.v2.
URL https://www.techrxiv.org/doi/full/10.36227/techrxiv.12480425.v2