Effects of Feature Selection and Normalization on Network Intrusion Detection

Mubarak Albarka Umar [1*], Chen Zhanfang [2]

[1,2]School of Computer Science and Technology,
Changchun University of Science and Technology,
7186 Weixing Road, Jilin, China.
Corresponding Author: [*1]2018300037@mails.cust.edu.cn, [2]chenzhanfang@cust.edu.cn

Abstract

The rapid rise of cyberattacks and the gradual failing of traditional defense systems and approaches led to the use of Machine Learning (ML) techniques aiming to build more efficient and reliable Intrusion Detection Systems (IDSs). However, the advent of larger IDS datasets brought about negative impacts on the performance and computational time of ML-based IDSs. To overcome such issues, many researchers utilized data preprocessing techniques such as feature selection and normalization. While most of these researchers reported the success of these preprocessing techniques on a shallow level, very few studies are performed on their effects on a wider scale. Furthermore, the performance of an IDS model is subject to not only the preprocessing techniques used but also the dataset and the ML algorithm used, which most of the existing studies on preprocessing techniques give little emphasis on. Thus, this study provides an in-depth analysis of the effects of feature selection and normalization on various IDS models built using four separate IDS datasets and five different ML algorithms. Wrapper-based decision tree and min-max are used in feature selection and normalization respectively. The models are evaluated and compared using popular evaluation metrics in IDS. The study found normalization to be more important than feature selection in improving performance and computational time of models on both datasets, while feature selection on UNSW-NB15 failed to reduce models computational time, and in the case of models built using NSL-KDD, it decreases their performance. The study also reveals that, compared to the UNSW-NB15 dataset, the NSL-KDD dataset is less complex and unsuitable for building reliable modern-day IDS models. Furthermore, the best performance on both datasets is achieved by Random Forest with accuracy of 99.75% and 98.51% on NSL-KDD and UNSW-NB15 respectively.

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

1 Introduction

As the internet and computer systems play increasingly vital roles in modern society, they have become the targets of cybercriminals. Therefore, we need to find the best ways possible to ensure the safety of our network and systems. User authentication, data encryption, and firewall were initially used, but they are proved to be insufficient; due to their limitations, Intrusion Detection Systems (IDSs) are nowadays been utilized to monitor

intrusion on a computer and network security. The IDSs use specific analytical techniques to detect attacks, identify their sources, and alert network administrators.

The signature-based detection approach has dominated IDSs 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 unreliable and thus, anomaly-based IDS become an area of interest for cybersecurity researchers. In quest of developing more reliable and efficient IDSs, ML techniques are generally used. The ML techniques need to learn from experience, a dataset in this case, to be able to correctly detect network intrusion. However, the advent of larger IDS datasets has increased the computational time needed to develop an IDSs model, as well as decreased their performance [1]. To overcome such issues, data preprocessing techniques such as feature selection and normalization are utilized by many researchers. Feature selection has been widely used in selecting relevant features for building robust IDSs models and has been influential on both efficiency and performance of IDS models [2], [3]. Furthermore, the use of normalization in handling IDS dataset features with large value range has proven to be very influential on the implementation of IDS models, reduce 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 a few IDS datasets using a few machine learning algorithms, very few in-depth studies are performed on the effects of those two preprocessing techniques. Furthermore, a generalized conclusion on the positive effect of feature selection and/or normalization on the IDS models are normally 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 dataset and machine learning algorithm used, which most of the existing studies give little emphasis on. Thus, IDS models must be developed using various ML algorithms and many IDS datasets to enable fair comparison and more holistic understandings on the implication of feature selection and normalization in IDS modeling. Hence, this work aimed at addressing this gap.

An in-depth analysis of the effects of feature selection and normalization is performed in this study. Five of the widely used ML algorithms in IDS modeling [6] are selected, two datasets, one considered outdated (NSL-KDD), and the other considered current (UNSW-NB15) [7], are also selected. To enable comparison and analysis of the effects of feature selection, a feature selected version of each dataset was made using a 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 the predominantly used normalization method in IDS modeling [5], [9], [10], is used, Using the four final datasets (full and feature selected version of each) three different IDS programs were implemented, each program contains ten distinct IDS models, with some of the models developed without applying normalization. Table 1.1 below summarized the three programs. The IDS models are evaluated using well-known and most used evaluation metrics in IDS modeling.

Table 1.1 – Programs Implemented

IDS Program	Dataset Features	Min-Max Normalized	Primary Dataset	ML Algorithms	Total IDS models
Program A	Full Datasets	Yes			10

Program B	Feature Selected	Yes	Both NSL-KDD and UNSW-NB15	All selected algorithms	10
Program C	Feature Selected	No			10
Total number of IDS models implemented					30

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

I. This study provides various in-depth comparisons on many aspects such as feature selection, normalization, datasets, and IDS models.
II. We propose a hybrid approach towards IDS modeling using a classifier for feature selection alongside another classifier in implementing IDS model to increase accuracy and efficiency of IDS
III. Dataset issue is one of the IDS challenges, many consider the oft-used KDD99 and its variant such as the NSL-KDD dataset 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 facets.
IV. 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: A review of literature, their limitations, and way of overcoming the limitations are presented in Section 2, Section 3 presents an explanation about basic IDS concepts, Machine learning, and the two preprocessing techniques. The experimental procedures observed are presented in Section 4, while in Section 5, we provide the evaluation results and the stated comparative analysis is performed. Finally, we conclude the paper and give suggestions for further research in Section 6.

2 Literature review

2.1 Related Works

In this section, some of the recent and related work combining the use of feature selection and normalization in modeling IDS using various approaches and ML methods are presented along with their limitation. A summary of the related works is shown in Table 2.1

Depren et al., [11] proposed a novel hybrid IDS model based on a self-organized map (SOM) for anomaly detection and J48 tree for misuse detection on the KDDcup99 dataset. A number of six basic features from 41 features were selected, however without explanation of the used feature selection technique, for modeling. The attributes were normalized using the min-max technique and WEKA software was used for the modeling work. The performance of the model was promising with a detection rate of % 99.90.

Wang et al., [4] modeled IDS using three methods: k-NN, PCA, and SVM on a normalized KDDCup99 dataset. 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 are evaluated using detection rate, and false positive rate and they found 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 classify. Hit, Miss, Detection rate and false positive rate were the performance measure used in evaluating the classifier. The experiment shows a detection rate of 97.4%, however, as they hinted, 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] proposes 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 their, and compared to tree-based single classifiers their proposed methods achieved the highest detection rate of 98.38%.

Song et al., [14] proposed an IDS method consisting of a combination of feature selection, normalization, fuzzy C means clustering algorithm, and 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 fuzzy C means clustering method is used to partition the training instances into clusters and for each cluster, a C4.5 algorithm was used for 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 C4.5 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 time-consuming during testing.

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 normalized 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 with 6.39% and a good classification accuracy compared to KDDcup99 and thus, they conclude that the UNSW-NB15 dataset is more complex than the KDD99 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 NSL-KDD dataset features and 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 mode. They evaluate the effectiveness of their methods using KDD99 and UNSW-NB15 datasets. DSAE feature selection is used to select 10 features in each dataset, which are normalized using the min-max method. 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 KDD99 and UNSW-NB15 datasets respectively.

Table 2.1 - Summary of Related Works

Paper 'Year	FS Method (No. of selected features)	ML Algorithm	Normalization type/method	Dataset (ID approach)	Evaluation Metrics
[11] '05	Not mentioned (6/41)	SOM/ J.48, DSS (Weka)	Minmax (0-1)	KDD99 / Hybrid	Detection rate, False positive rate, and Missed rate
[4] '09	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] '11	PCA (10/41)	SVM	Non-zero	KDD99 / Anomaly	Detection rate, False positive rate, Mis, hit
[13] '12	GA (16/41)	Hybrid of Nuerotree	Minmax	NSL-KDD/ Anomaly	TP Rate, FP Rate, Precision, Recall, F- Measure
[14] '13	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] '13	CFS & CON Filters (18/41)	Tree-based Classifiers	Not mentioned	NSL-KDD/ Anomaly	Accuracy, TP Rate, FP Rate, Precision, Recall, F- Measure
[17] '15	Harmony Search (20/41)	SVM (LibSvm)	Min-max (1-13)	NSL -KDD / Anomaly	Detection rate, Test Time
[18] '17	GA-LR wrapper KDD99 (18/41) UNSW-NB (20/42)	C4.5, Random Forest, and Naïve Bayes Tree (C++)	Log-scaling, Minmax (0-1)	KDD99, UNSW-NB15 / Anomaly	Confusion Matrix, Accuracy, Detection rate, False alarm rate
[19] '19	IG Weka's Filter (17/41)	SVM	Log-norm	NSL-KDD	Accuracy, Sensitivity, Specificity, False, and True positive.
[20] '19	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 Limitations of Related Works

After a thorough review of the related works, it's clear that each of the reviewed work 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.

I. Preprocessing: Transformation (encoding, discretization/normalization), Feature selection
II. Modeling Stage: Issues with classifier choice and less use of multiple classifiers
III. Dataset Issues: Outdated datasets are used in most of the studies. No comparisons made.

I. Data Encoding

Most of the Machine learning algorithms cannot handle categorical features unless they are converted to numerical values. The categorical features can be nominal (no particular order) or ordinal (ordered). Many algorithm's performances vary 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, because it does not make sense to say TCP should be in between UDP and ICMP, and this may be misinterpreted by the algorithm and can have an unwanted effect on the IDS model. This mistake can be seen in some of the reviewed literature [14], [17], [18], [20]. A better solution to this is to use binary encoding or yet better, one-hot (dummy) encoding, that map each category to a vector that contains 1 and 0 denoting the presence or absence of the features' value.

II. Data Discretization and Normalization

While the discretization of numerical features is influential in data preprocessing [21], however, unlike normalization, it generally also leads to a loss of information [22]. Normalization is an important data preprocessing step which can improve the accuracy and efficiency of especially classification algorithms [23] and have shown to improve the accuracy of IDS model built with large dataset [4], [5], 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 ranging the values, nonetheless, [15] chooses to use it at the cost of normalization. Furthermore, most of the reviewed studies give very little emphasis on the impact of normalization on the performance of the IDS models.

III. Feature selection

In recent years, due to the high dimensionality and size of IDS datasets, many researchers are using dimensionality reduction methods to reduce the dataset dimension and select optimal subset features to represents the entire dataset [24] thereby reducing computational time, resource utilization, as well as increase accuracy and performance of IDS models. There are three basic feature selection methods, only two of these methods were mainly applied in IDS modeling, and most of the reviewed literature used filters that ignore the effects of the selected feature subset on the performance of the IDS model [25]. Contrary to wrappers which, though computationally expensive, produce better performance for the predefined classifier. Furthermore, some of the reviewed work [4], [11] hand-picked certain features without using any of the feature selection methods which may lead to removing influential features.

IV. Modeling Stage

To develop an accurate and good IDS Model, there is a need of exploring various algorithms and techniques [12] because using one or two algorithms can hardly offer reliable and good performing IDS models; however, most of the reviewed work uses one or two algorithms. Furthermore, there was too much usage of tree-based algorithms in some reviewed work [15], [18] without validating their performances by comparing with other algorithms. Unfortunately, trees-based algorithms (bar ensemble trees such as a random forest) generally do not have the same level of predictive accuracy as some regression and classification algorithms [16].

V. Dataset Issues

Most of the reviewed literature made use of either KDDcup99 or its variant NSL-KDD which are among the widely used in IDS academic research [6], [7]. However, despite that, they are considered to be outdated and not containing 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. A number of modern datasets were proposed [27]–[29], Ring et al., [30] also recommended some 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] also made recommendations on using more than one dataset with at least one publicly available dataset to avoid overfitting of IDS model to one dataset, ensure reproducing of the work, and its generic evaluation. However, most of the reviewed literature used only one dataset.

In summary, this study addressed those limitations. Since most of the selected ML algorithms in this work consider all features during training simultaneously, One-hot encoding, an approach suited for such ML algorithms [31], is used. Furthermore, Minmax, one of the most common normalization method [8], is also used. Five among the widely used ML algorithms in IDS modeling [6] were 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 are developed and evaluated. In what follows next, the five selected ML algorithms are explained, 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 hardware device that monitors traffic passing across networks and through systems to check for suspicious behavior, policy violations, and 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 classification. At the most basic level, NIDS looks at network traffic, while HIDS looks at actions and files on the host computers. By 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 alarms. Due to the complementary nature of these two approaches, a hybrid approach, combining both of these techniques, is often used [35]. The literature nowadays is focusing 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

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, supervised mode outperforms the remaining modes [3], [32]. Some of the ML algorithms used in IDSs include Artificial Neural Network, k-Nearest Neighbor, Naive Bayes, Genetic Algorithm, Support Vector Machine, Logistic Regression, and Decision Trees. Developing any Machine learning model consist of four basic steps, namely, data collection, data preprocessing, model selection and training, and model evaluation [36]. The two important concepts of this work, feature selection, and normalization are among the many tasks performable in the data preprocessing step.

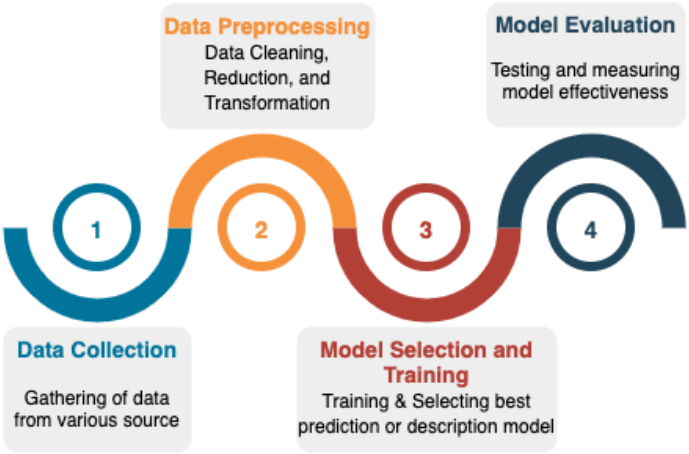

Figure 3.1 - Machine Learning Basic Steps

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 dimension of the features. Feature selection reduces model training time, simplifies a model, and improves generalization while reducing the chances of overfitting. Besides, it improves classification accuracy. Feature subset selection requires a search strategy and direction to select features subset, an objective function to evaluate the selected features, a termination condition, and an evaluation of result. There are three main feature selection approaches: (a) the filters which 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 the classifier establishing [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 [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 follow:

$$x_{new} = \frac{x - min(x)}{max(x) - min(x)} \quad (3.1)$$

3.3 Selected Machine Learning 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 employed 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 large than the number of samples [26]. In this work, Scikit-learn implementation of 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 given any task-specific rules. ANNs aims 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: classification, the output is a class membership assigned to neighbors with the highest vote, whereas with 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 decorrelated 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 decision tree's simplicity with the flexibility to increase accuracy and overcome the decision tree's habit of overfitting to their training set [41].

3.3.5 Naive Bayes (NB)

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 outcome based on probabilities of its occurrence in training data. They require a small amount of training data to estimate the necessary parameters. 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 described how the experiment is conducted by following the four basic machine learning steps. It also provides the tools used in experimenting.

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, implementing, and validating the IDS models. Jupyter Notebook is used as the execution environment for Python and its libraries.

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 as a new benchmark dataset that can be used for IDSs 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. And 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 4.1.

Table 4.1 - 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 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 [46] is an optimized version of the KDDcup99 dataset, which removes redundant records and provide 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 125973 samples (KDDTrain+) and a testing set of 22544 samples (KDDTest+). Here also, only the training set (KDDTrain+) is used for both training and testing in this

work. Table 4.2 summarizes the sample's distribution of all attacks (anomaly) and normal traffics in the training set of the NSL-KDD dataset.

Table 4.2 - 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 Data Preprocessing

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

4.3.1 Data Reduction

4.3.1.1 Data filtration

As stated above, we used the UNSW-NB15 dataset (UNSW NB15 training-set.csv), and the NSL-KDD dataset (KDDTrain+.arff). 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, so 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 with a proportion of 67% (2/3) and 33% (1/3) respectively as shown in Figure 4.2 below. WEKA's unsupervised instance *Resample* filter is used to ensure balance splitting and to avoid developing overfitted models that might perform poorly when given out-of-sample data.

4.3.1.2 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. The wrapper-based approach, though computationally expensive, tends to give superior model performance [48], is employed in this work with a decision tree algorithm as the feature evaluator as shown in Figure 4.1.

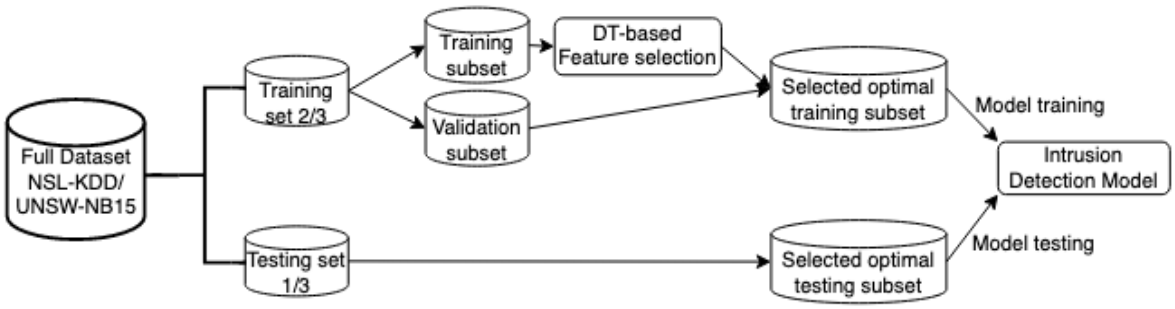

Figure 4.1 - DT Wrapper-Based FS

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 thus making partial feature selection and losing count on the actual number of selected features, 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 feature for UNSW-NB15 and NSL-KDD respectively and WEKA's supervised attribute *Remove* filter is used to collect the features subsets. Thus, two more datasets are derived bringing our total datasets to four: 2 complete datasets and 2 feature selected versions of UNSW-NB15 and NSL-KDD, description of the full datasets is available in [28] and [45] respectively, Table 4.3 below shows the selected optimal features of the datasets.

Table 4.3 - Selected Optimal Features

UNSW-NB15		NSL-KDD	
No.	Feature name	No.	Feature name
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		

(*) – indicates categorical features

4.3.2 Data Transformation

4.3.2.1 Data Normalization

The full and formed datasets consist of features of two types: 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 normalized all the numeric features within a range of 0 to 1 using equation (3.1) above. The normalization process is performed after feature selection in order not to affect the selection process.

4.3.2.2 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, an example of the *protocol_type* encoding is shown in Table 4.4 below. 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 same encoding is 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 has three nominal features (*proto*, *service*, and *state*) and one-hot encoded accordingly. Table 4.5 provides a summary of the four datasets dimensions before and after encoding. Because one-hot encoding increase dataset dimension, so to avoid losing some nominal features' values encoded in the feature selection process, the encoding is performed after the feature selection and normalization processes.

Table 4.4 - One-Hot Encoding Example

Protocol_type
UDP
TCP
ICMP

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

After encoding the features, the dimensions of the four final datasets increased as shown in Table 4.5. Only the final encoded features are used in training and evaluating the models.

Table 4.5 - Final Datasets Dimensions

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

4.4 Model Selection and Training

Model selection is the estimation of different model's performance using a cross-validation or hold-out approach to choose the best one [41]. However, model selection in this work is the selected ML algorithms. An explanation of the algorithms was given in the previous section of this work. Default Scikit-learn implementation of these algorithms is used in developing the models with SVM's probability set to true as the only changed parameter.

Three different programs are implemented and using the algorithms, a total of thirty (30) distinct IDS models are developed using the four datasets. The model developing constitute of two stages: training stage and testing stage. During the training stage, the algorithms are trained using the training dataset, then in the testing stage, the test dataset is used to assess the performance and reliability of the built IDS models. Figure 4.2 below depicted the entire model training and testing process. 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 result of the evaluations are provided in the next section of this work. A summary of the three different IDS model implementations performed is as follows:

I. Program A: models implemented with full, non-feature selected, normalized datasets.
II. Program B: models implemented with feature selected, normalized datasets.
III. Program C: models implemented with feature selected, unnormalized datasets.

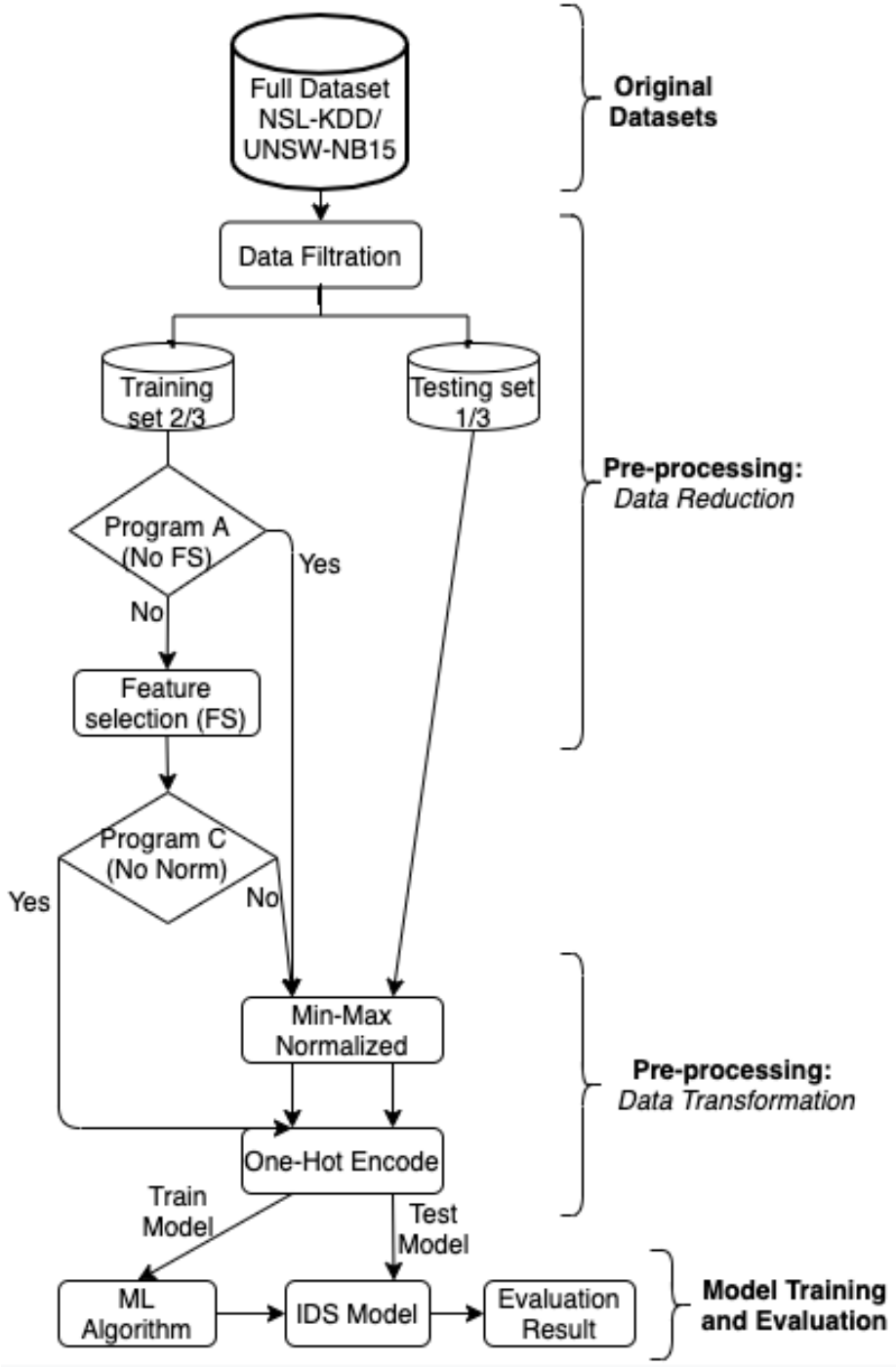

Figure 4.2 - Conceptual Framework of the IDS models

4.5 Model Evaluation Metrics

A model evaluation metric is a criterium by which the performance of a model can be measured. The performance of an IDS model can be evaluated based on its ability to classify network traffic into the correct type. Most of the IDS works used three of those metrics, namely; classification accuracy, detection rate (DR), and false alarm rate (FAR) [26]. Similarly, in this work, these metrics are adopted in addition to computational time. 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 work (bar time) are based on the Confusion Matrix and the numbers inside it, we see it important to explain it.

4.5.1 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. No confusion matrix is included in this work due to the number of implemented models.

Table 4.6 - 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 incorrectly detected as an attack. Aka Type I error.

True negative (TN): Number of normal correctly detected as normal.

4.5.2 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].

$$Accuracy(ACC) = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

4.5.3 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

$$Detection\ Rate(DR) = \frac{TP}{TP + FN} \tag{4.2}$$

4.5.4 False Alert Rate (FAR)

Aka fall-out or False positive rate (FPR), is the measure of the incorrectly classified negative (normal) instance as an anomaly from all the actual negative instances, or defined as the proportion of negative prediction this is mistakenly considered as positive (anomaly) for all negative predictions. The lower its value the better.

$$False\ Alert\ Rate(FAR) = \frac{FP}{FP + TN} \tag{4.3}$$

4.5.5 Computational Time

The computational time is the entire time taken to train and evaluate a model. However, the time reported in this work does not include time taken by feature selection operation. Because the timing depends on factors beyond our control (such as CPU task switching, etc.), we try to avoid running heavy tasks whilst executing the programs, prevent the computer from sleeping and also re-run the programs a number of times to verify the timing and ensure minimal interference.

5 Result and Discussion

This chapter presents the platform on which the experiment is performed, the result obtained, and the interpretation of the results. The following comparisons are made:

I. Comparison between the IDS models developed using full datasets and feature selected datasets to assess the impact of feature selection.
II. Comparison between the IDS models developed using normalized and unnormalized feature selected datasets to assess the impact of normalization.
III. Comparison to measure the effectiveness, reliability, and complexity of the IDS datasets.

5.1 Experimental Platform

To avoid interference from the experimental platform, all the programs are implemented and executed in the same environment using the same programming language as shown in Table 5.1.

Table 5.1 - Experimental Platform

Name	Details
Computer	MacBook Pro
OS	macOS Catalina version 10.15.3
CPU	2.5 GHz Dual-Core Intel Core i5 processor
RAM	8GB 1600 MHz DDR3
Storage Disk	480GB SSD
Execution platform	Jupyter Notebook
Experimental Tools	Excel, WEKA, Python

5.2 Comparisons on Feature Selection

Table 5.2 below presents the evaluation results of models built using both NSL-KDD and UNSW-NB15 full features and feature selected data. Feature selection generally improves performances and reduce computational time [9], [10], [14]. The boldly written values indicate the anticipated improvement in performances or decrease in computational time after feature selection. It can be seen that, in the case of NSL-KDD, with 19 selected features the performances of all the models are, although almost the same as the full features, slightly lower (this is consistent with [18]) with only RF achieving better performance, in terms of accuracy (99.75%) and detection rate (99.72%), while maintaining the second-lowest false alert rate of 0.22%. Unlike with UNSW-NB15, the performance accuracy of three of the models (RF, KNN, and NB) is better using only 20 selected features with both KNN and RF achieving a remarkably higher performance across all the metrics, while ANN and SVM performances with the 20 selected features are almost the same with full features. It can thus be deduced 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. The computational time of all models built with the selected features of NSL-KDD, except ANN, reduces. Conversely, the exact contrary is observed with models built on selected features of UNSW-NB15. Although high computational complexity is often observed on large datasets [32], it is not clear why higher computational time is observed on UNSW-NB15 models even after feature selection. In both datasets, the NB models achieved the best and lowest computational time whereas SVM achieved the opposite. Strangely, feature selection increases both computational time and performance on UNSW-NB15 models, whereas the exact contrary effect is observed on NSL-KDD models. Similar effects are observed from an overall perspective, Table 5.4 below summarizes the sum of computational time taken by the full and feature selected-based models across the two primary datasets.

Table 5.2 - Feature Selection Comparison

Models	Evaluation metrics	NSL-KDD		UNSW-NB15	
		Full Features	Features Selected	Full Features	Features Selected
ANN	ACC	99.69	98.98	94.62	94.32
	DR	99.6	99.09	97.54	**98.48**
	FAR	0.23	1.11	11.64	14.56
	Time	1.57m	2.05m	5.81m	**5.43m**
SVM	ACC	98.56	98.06	93.67	93.56
	DR	98.13	97.18	99.63	99.54
	FAR	1.08	1.17	19.14	19.19
	Time	16.21m	**15.36m**	86.89m	170.61m
KNN	ACC	99.57	99.13	93.81	**95.8**
	DR	99.49	99.24	96.24	**97.28**
	FAR	0.36	0.97	11.42	**7.36**
	Time	9.32m	**5.52m**	8.38m	12.46m
RF	ACC	98.81	**99.75**	95.74	**98.51**
	DR	99.71	**99.72**	97.84	**99.17**
	FAR	0.1	0.22	8.77	**2.89**
	Time	0.25m	**0.22m**	0.54m	0.56m
NB	ACC	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	1.31s	**1.07s**	2.89s	4.8s

5.3 Comparisons on Normalization

Table 5.3 below presents the evaluation results of models built using NSL-KDD and UNSW-NB15 normalized and unnormalized feature selected data. Normalization typically improves performance and decrease computation time [4], [5], the boldly written values indicate the increase in performances or decrease in computational time on models built without normalization, and the italic written values indicate surprising results. It can be seen that in the case of NSL-KDD, the performances of distance-related classifiers, SVM and KNN both expected to improve after normalization [40], [53], are contrary. KNN's performance decreases whereas SVM's performance drastically improved, however, SVM's poor performances on unnormalized NSL-KDD raises a question about the reliability of the dataset, especially given that the performance of SVM is not severely affected by lack of normalization in UNSW-NB15. NB which is not a distance-based classifier also achieves two opposing results; with NSL-KDD, it performs surprisingly poor on the unnormalized dataset and improved drastically on normalized NSL-KDD, however, with UNSW-NB15, it instead expectedly performs well on the normalized dataset and performs better on the unnormalized dataset with both performances in close range. RF also achieves two opposing performances across the two datasets; with NSL-KDD, it performs better without normalization and, on the contrary, it performs better with normalization on UNSW-NB15. Only ANN performs as anticipated, with its performances both relatively better after normalization across the two datasets. Thus, since with UNSW-NB15, the performance of 4 classifiers have improved after normalization and similarly, performances of 3 of the 5 classifiers have also improved after normalization with NSL-KDD, it can be deduced 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. In the case of computational time, most of the NSL-KDD and UNSW-NB15 models correspondingly observed a similar pattern of behavior except KNN which, unlike the rest, seen an increase in computational time after normalization. This, essentially, is as expected [40]. The

computational time of RF and NB is also quite acceptable giving that they both do not necessarily need normalization [40], [54]. Similarly, in both datasets, the NB models achieved the best and lowest computational time whereas SVM achieved the opposite. Therefore, it can be deduced that the normalization does also reduce computational time in addition to improving performance. From an overall perspective, the sum of computational time taken by the normalization-based models is considerably lower than those built with unnormalized data in both NSL-KDD and UNSW-NB15 as shown in Table 5.4 below, this is similar to what was observed on analyzing individual models.

Table 5.3 - Normalization Comparison

Models	**Evaluation metrics**	**NSL-KDD**		**UNSW-NB15**	
		Normalized	Unnormalized	Normalized	Unnormalized
ANN	ACC	98.98	96.0	94.32	93.38
	DR	99.09	94.6	98.48	96.93
	FAR	1.11	2.78	14.56	**14.18**
	Time	2.05m	3.59m	5.43m	5.93m
SVM	ACC	98.06	*53.5*	93.56	75.49
	DR	97.18	*0.11*	99.54	97.88
	FAR	1.17	***0.03***	19.19	72.23
	Time	15.36m	165.65m	170.61m	484.48m
KNN	ACC	99.13	***99.42***	95.8	94.91
	DR	99.24	***99.47***	97.28	97.11
	FAR	0.97	***0.63***	7.36	9.78
	Time	5.52m	**0.76m**	12.46m	**6.31m**
RF	ACC	99.75	**99.88**	98.51	98.49
	DR	99.72	**99.8**	99.17	99.17
	FAR	0.22	**0.04**	2.89	2.95
	Time	0.22m	**0.2m**	0.56m	0.6m
NB	ACC	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	1.07s	**0.02s**	4.8s	5.36s

5.4 Comparisons of Datasets

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 5.2 and Table 5.3 above, specifically focusing on those models whose performances or computation time was rather surprising or does not show similar effects after performing related actions to the similar models on corresponding datasets. And 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.

5.4.1 Reliability Comparison

Since KNN typically uses Euclidian distance to find k nearest points from any given point, using 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 with NSL-KDD in Table 5.3 can largely be attributed to lack of normalization, however, a closer look at how they both performed without normalization with UNSW-NB15 implies that the poor performances have more to do with the dataset itself. As seen in Table 5.2, the feature selection doesn't seem to improve the performance of KNN

and RF (bar on RF's accuracy and detection rate) with NSL-KDD, however, with UNSW-NB15, the performances of both KNN and RF across all the metrics improved.

Table 5.4 - Computational Time Summary

	Program A (Norm)	Program B (FS + Norm)	Program C (FS)	Total
NSL-KDD	27.37m	23.17m	170.22m	220.76m
UNSW-NB15	101.67m	**189.14m**	497.41m	788.22m
Total	129.04m	**212.31m**	667.63m	

As shown in Table 5.4, a clear decrease of models' computational time in the unnormalized datasets (Program C) can be observed after normalizing the datasets (in Program B), however, in comparison to full featured datasets (in Program A), two opposing computational times were seen on the selected features (in Program B), with the computational time of NSL-KDD models decreasing and those of UNSW-NB15 increasing. The unexpected increase in models computational time on the selected features of UNSW-NB15 would have been normal had the increased occurred on NSL-KDD instead, but given the reported reliability of UNSW-NB15 in literature [30], this is quite strange and although it is obscure whether this is an indication of its complexity or not, this however, leaves query on UNSW-NB15 dataset. Nonetheless, this highlights the opposing nature of the two datasets. Furthermore, the constant deviations from the anticipated performance of models built using NSL-KDD highlights an inconsistent and unreliable behavior in the NSL-KDD. Thus, the reaction of UNSW-NB15 on feature selection and normalization operations in comparison to NSL-KDD summarizes the very distinct differences between the two and shows how one is more reliable, conformant, and consistent over the other.

5.4.2 Complexity Comparison

Accuracy, Detection rate, and False alert rate metrics are used in this comparison. As explained above, the performances of Program A models (implemented using full and normalized features) are used for the comparison. Figure 5.1, Figure 5.2 and Figure 5.3 provide a summary of the comparisons on Accuracy, Detection rate, and False alert rate respectively.

In Figure 5.1, all NSL-KDD models outperform their corresponding UNSW-NB15 implemented models. ANN and RF achieve the highest accuracy of 99.69% and 95.74% over NSL-KDD and UNSW-NB15 respectively, while both datasets achieved their worst accuracy on NB. Furthermore, it can be observed that, in the detection rate depicted in Figure 5.2, which stands for the accuracy rate for the attack classes, the NSL-KDD models were largely able to detect more attacks than the UNSW-NB15 models except with SVM which achieves DR of 99.63%. RF again achieves the overall highest DR of 99.71% with NSL-KDD and once more, the low DR obtained for both datasets is on NB with its UNSW-NB15 model achieving the poorest DR of just 23%.

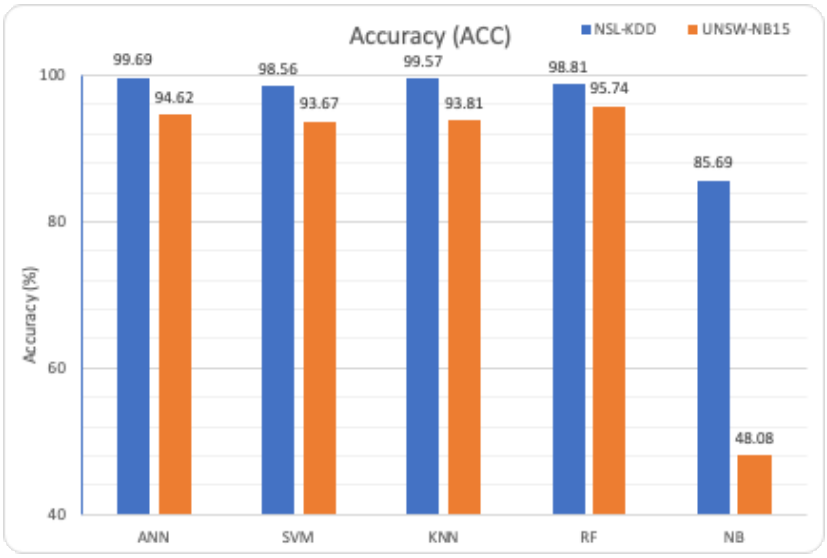

Figure 5.1 - Program A Models Accuracy

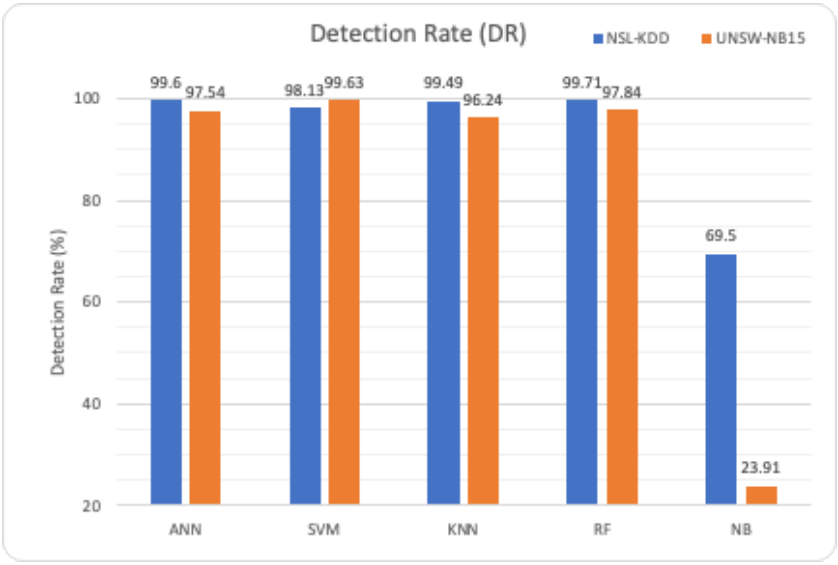

Figure 5.2 - Program A Models Detection Rate

Very low false alert rates are generally the target in IDS. The FAR depicted in Figure 5.3 shows a 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 alert than their corresponding NSL-KDD models. The highest FAR of 19.14% is achieved by SVM on UNSW-NB15. The summation of the FAR percentage for all the NSL-KDD models is 1.99, averaging 0.398% per model which is 25.62 times less than the average on UNSW-NB15 models. The UNSW-NB15 models reported a total of 50.99 FAR percentage, averaging 10.198% FAR per model.

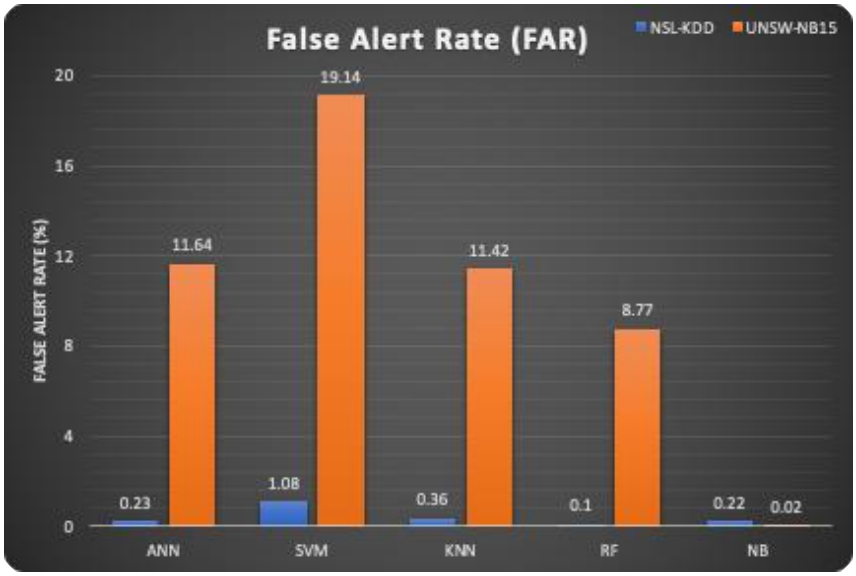

Figure 5.3 - Program A Models False Alert Rate

Overall, the results of the accuracy, DR, and the FAR of models built with NSL-KDD appears to be higher than those built using UNSW-NB15, a typical interpretation will be 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 it's attack patterns. And 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.

6 Conclusion

Network and computer systems are continually facing attacks like never before and our existing protective mechanisms are failing, thus more effort should be exerted towards analyzing and improving them, as well as in developing more sophisticated protection methods to secure our systems and meet up with the current challenging environments. This paper analyzes the effects of feature selection and normalization techniques on NSL-KDD and UNSW-NB15 IDS datasets using five machine learning algorithms to implement thirty IDS models. Accuracy, Detection rate, False alert rate, and Computational time are used to evaluate and compare the models. The following conclusions can be deduced from this work:

I. Normalization improves the performance and computational time of both datasets. Whereas feature selection improves the performance of models on UNSW-NB15 only, in the case of models built using NSL-KDD, it reduces computational time only. Thus, Normalization is found to be more important than feature selection. We hence, strongly recommend the use of normalization, especially when using algorithms such as SVM [53], to increase accuracy. Table 6.1 below summarized the performance-wise and computational time-wise effects of feature selection and normalization on NSL-KDD and UNSW-NB15 datasets. A tick indicates where either of the criteria is improved by the given preprocessing technique.

II. Generally, RF models achieved remarkably higher performances across all the datasets, making the random forest algorithm the most robust among the oft-used ML algorithms in IDS. The SVM requires normalization and takes higher computational time than the other algorithms. The NB models takes the lowest time, it however achieved the lowest performances on average also, making it the worst IDS algorithm in this work, and, in comparison to many supervised ML algorithms, the poorest [55].

III. Compared to NSL-KDD, the UNSW-NB15 which contained more modern low footprint attack families is found to be more complex and reliable for evaluating modern-day IDSs than NSL-KDD which contained fewer and outdated attack families. Thus, based on our findings, NSL-KDD is not suitable for the IDS evaluation benchmark and hence, we recommend using UNSW-NB15 for building reliable IDS models.

IV. It's interesting to also note that, most of the reviewed works using other than one-hot encoding method, 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 has influenced the classifiers' performances. It will be interesting to study the effect of various encoding methods in IDS modeling

Table 6.1 - Preprocessing Effects Summary

	NSL-KDD		UNSW-NB15	
Criteria	Feature Selection	Normalization	Feature Selection	Normalization
Performances	×	✓	✓	✓
Computation Time	✓	✓	×	✓

While the overall results have shown great promises and distinctive conclusions particularly concerning the preprocessing techniques influence and the UNSW-NB15 and NSL-KDD datasets, there exist more open gaps for future research. Some of them are as follow:

I. *Dataset and Algorithms*: Among the many available IDS dataset benchmarks and machine learning algorithms, this work only made use of two datasets, and a maximum of five algorithms. It will be interesting to use more datasets and more algorithms for broader insights.

II. *Alternative feature selection and normalization*: There are three feature selection methods, Wrapper method is considered in this study, the other two methods can be considered in future studies. The same wrapper method can also be used in a different approach such as change of evaluation algorithm, or searching strategy and soon. Similarly, the use of different normalization methods can also be explored.

III. *Multi-classification*: This work primarily focused on binary classification of normal and attack network traffics, however, both the used datasets contained varied attack types, thus multi-classification work can be done to enable further analysis of the effect of preprocessing techniques on distinctive classes of attacks.

IV. *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 DR, however, a higher number of FAR is also eminent, and although this highlights the complexity of the dataset, further work can be performed to reduce it.

Acknowledgement

We would like to thank Dr. Yan Liu for her advice and encouragement at the initial stage of this work.

Reference

[1] H. Hindy *et al.*, 'A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets', *arxiv.org*, vol. 1, no. 1, pp. 1–35, 2018.

[2] T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, Q. Chen, and R. A. Bridges, 'A survey of intrusion detection systems leveraging host data', *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–40, 2018.

[3] H. M. Koupaie, S. Ibrahim, and J. Hosseinkhani, 'Outlier detection in stream data by machine learning and feature selection methods', *Int. J. Adv. Comput. Sci. Inf. Technol.*, vol. 2, no. 3, pp. 17–24, 2013.

[4] W. Wang, X. Zhang, S. Gombault, and S. J. Knapskog, 'Attribute normalization in network intrusion detection', in *I-SPAN 2009 - The 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, 2009, pp. 448–453.

[5] W. Li and Z. Liu, 'A method of SVM with normalization in intrusion detection', *Procedia Environ. Sci.*, vol. 11, pp. 256–262, 2011.

[6] A. Özgür and H. Erdem, 'A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015', *PeerJ*, vol. 4, pp. 0–21, 2016.

[7] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, 'KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research', *IEEE Comput. Graph. Appl. Bridg.*, vol. 52, no. 2, pp. 41–51, 2019.

[8] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, 'Data preprocessing for supervised leaning', *Int. J. Comput. Sci.*, vol. 1, no. 1, pp. 111–117, 2006.

[9] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, 'An Efficient Intrusion Detection System Based on Feature Selection and Ensemble Classifier', *J. Latex Cl. Files*, vol. 14, no. 8, pp. 1–12, 2015.

[10] S. H. Kang and K. J. Kim, 'A feature selection approach to find optimal feature subsets for the network intrusion detection system', *Cluster Comput.*, vol. 19, no. 1, pp. 325–333, 2016.

[11] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, 'An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks', *Expert Syst. Appl.*, vol. 29, no. 4, pp. 713–722, 2005.

[12] P. Somwang and W. Lilakiatsakun, 'Computer network security based on Support Vector Machine approach', in *International Conference on Control, Automation and Systems*, 2011, pp. 155–160.

[13] S. S. Sivatha Sindhu, S. Geetha, and A. Kannan, 'Decision tree based light weight intrusion detection using a wrapper approach', *Expert Syst. Appl.*, vol. 39, no. 1, pp. 129–141, 2012.

[14] J. Song, Z. Zhu, P. Scully, and C. Price, 'Selecting features for anomaly intrusion detection: A novel method using fuzzy C means and decision tree classification', *Cybersp. Saf. Secur. CSS 2013. Lect. Notes Comput. Sci.*, vol. 8300 LNCS, pp. 299–307, 2013.

[15] S. Thaseen and C. A. Kumar, 'An analysis of supervised tree based classifiers for intrusion detection system', in *Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, PRIME 2013*, 2013, pp. 294–299.

[16] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York: Springer, 2013.

[17] H. Ghaffari Gotorlar, J. Bagerzadeh, M. Pourmahmood Aghababa, and M. Samadi Osalu, 'Improving Intrusion Detection Using a Novel Normalization Method along with the Use of Harmony Search Algorithm for Feature Selection', in *International Conference on Information and Knowledge Technology*, 2015, pp. 1–6.

[18] C. Khammassi and S. Krichen, 'A GA-LR wrapper approach for feature selection in network intrusion detection', *Comput. Secur.*, vol. 70, pp. 255–277, 2017.

[19] B. Setiawan, S. Djanali, and T. Ahmad, 'Increasing accuracy and completeness of intrusion detection model using fusion of normalization, feature selection method and support vector machine', *Int. J. Intell. Eng. Syst.*, vol. 12, no. 4, pp. 378–389, 2019.

[20] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, 'A Two-Stage Deep Learning Model for Efficient Network Intrusion Detection', *IEEE Access*, vol. 7, pp. 30373–30385, 2019.

[21] S. Ramírez-Gallego *et al.*, 'Data Discretization: Taxonomy and Big Data Challenge', *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 6, no. 1, pp. 5–21, 2016.

[22] R. Jin and Y. Breitbart, ‘Data discretization unification’, in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007, pp. 183–192.
[23] L. Al Shalabi and Z. Shaaban, ‘Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix’, *Proc. Int. Conf. Dependability Comput. Syst. DepCoS-RELCOMEX 2006*, pp. 207–214, 2006.
[24] Akashdeep, I. Manzoor, and N. Kumar, ‘A feature reduced intrusion detection system using ANN classifier’, *Expert Syst. Appl.*, vol. 88, pp. 249–257, 2017.
[25] R. Kohavi and G. H. John, ‘Wrappers for feature subset selection’, *Artif. Intell.*, vol. 97, pp. 273–324, 1997.
[26] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, ‘Survey of intrusion detection systems: techniques, datasets and challenges’, *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.
[27] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, ‘Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling’, *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, 2017.
[28] N. Moustafa and 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 2015 - Proceedings*, 2015.
[29] G. Creech and J. Hu, ‘Generation of a new IDS test dataset: Time to retire the KDD collection’, in *IEEE Wireless Communications and Networking Conference, WCNC*, 2013, pp. 4487–4492.
[30] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, ‘A survey of network-based intrusion detection data sets’, *Comput. Secur.*, vol. 86, pp. 147–167, 2019.
[31] P. Cerda, G. Varoquaux, and B. Kégl, ‘Similarity encoding for learning with dirty categorical variables’, *Mach. Learn.*, vol. 107, no. 8–10, pp. 1477–1494, 2018.
[32] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, ‘Anomaly-based network intrusion detection: Techniques, systems and challenges’, *Comput. Secur.*, vol. 28, no. 1–2, pp. 18–28, 2009.
[33] S. Axelsson, ‘Intrusion Detection Systems: A Survey and Taxonomy’, 2000.
[34] Z. Inayat, A. Gani, N. B. Anuar, M. K. Khan, and S. Anwar, ‘Intrusion response systems: Foundations, design, and challenges’, *J. Netw. Comput. Appl.*, vol. 62, pp. 53–74, 2016.
[35] T. Verwoerd and R. Hunt, ‘Intrusion detection techniques and approaches’, *Comput. Commun.*, vol. 25, no. 15, pp. 1356–1365, 2002.
[36] A. L. Buczak and E. Guven, ‘A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection’, *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
[37] J. Tang, S. Alelyani, and H. Liu, ‘Feature selection for classification: A review’, in *Data Classification: Algorithms and Applications*, 2014, pp. 37–64.
[38] J. Han, M. Kamber, and J. Pei, *Data Mining – Concepts & Techniques*, 3rd Editio. USA: Morgan Kaufmann Publishers, Elsevier Inc, 2012.
[39] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design*, 2nd Ed. 2014.
[40] X. Wu *et al.*, ‘Top 10 algorithms in data mining’, *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.
[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.
[42] J. D. M. Rennie, L. Shih, J. Teevan, and D. Karger, ‘Tackling the Poor Assumptions of Naive Bayes Text Classifiers’, *Proceedings, Twent. Int. Conf. Mach. Learn.*, vol. 2, no. 1973, pp. 616–623, 2003.
[43] N. Moustafa and 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’, *Inf. Secur. J.*, vol. 25, no. 1–3, pp. 18–31, 2016.
[44] Unsw.adfa.au, ‘The UNSW-NB15 data set description’, 2015. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/. [Accessed: 24-Mar-2020].
[45] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, ‘A Detailed Analysis of the KDD CUP 99 Data Set’, in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)*, 2015, pp. 1–6.
[46] unb.ca, ‘NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB’. [Online].

Available: https://www.unb.ca/cic/datasets/nsl.html. [Accessed: 24-Mar-2020].
[47] I. H. Witten, E. Frank, and H. Mark A., *Data Mining Practical Machine Learning Tools and Techniques Third Edition*, 3rd Editio., vol. 1, no. 5. USA: Morgan Kaufmann Publishers, Elsevier Inc, 2011.
[48] H. Liu and L. Yu, 'Toward Integrating Feature Selection Algorithms for Classification and Clustering', *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, 2005.
[49] . scikit-learn 0.22.2, 'Decision Trees — scikit-learn 0.22.2 documentation'. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html. [Accessed: 25-Mar-2020].
[50] G. Holmes, A. Donkin, and I. H. Witten, 'WEKA: A Machine Learning Workbench'.
[51] by J. Ross Quinlan, M. Kaufmann Publishers, and S. L. Salzberg, 'Programs for Machine Learning', 1994.
[52] G. Shobha and S. Rangaswamy, 'Machine Learning', in *Handbook of Statistics*, 1st ed., vol. 38, Elsevier B.V., 2018, pp. 197–228.
[53] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, 'A Practical Guide to Support Vector Classification', *Theory, Culture and Society*. pp. 1–16, 2016.
[54] E. Lewinson, *Python for finance cookbook*. Birmingham: Packt Publishing, Limited, 2020.
[55] R. Caruana and A. Niculescu-Mizil, 'An empirical comparison of supervised learning algorithms', in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 161–168.