Effects of Feature Selection and Normalization on Network Intrusion Detection

Mubarak Albarka Umar*[,1,2], Zhanfang Chen[2], Khaled Shuaib[1], and Yan Liu[3]
[1]College of Information and Technology, United Arab Emirates University, Al Ain, UAE
[2]School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, China
[3]Department of Computer Science, College of Engineering, Shantou University, Shantou, China
*Corresponding author
E-mail addresses: 700040629@uaeu.ac.ae (M.A. Umar), chenzhanfang@cust.edu.cn (Z. Chen), k.shuaib@uaeu.ac.ae (K. Shuaib), yanliu@stu.edu.cn (Y Liu)

Received: Date Month, Year; Accepted: Date Month, Year; Published: Date Month, Year

Abstract: The rapid rise of cyberattacks and the gradual failure of traditional defense systems and approaches led to the use of Machine Learning (ML) techniques 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 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 were 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 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 and five different ML algorithms. Decision tree wrapper-based and min-max methods 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. Whereas applying feature selection on the UNSW-NB15 dataset failed to reduce models' computational time, in the case of models built using the NSL-KDD dataset, it decreased their performance. The study also shows that, compared to the UNSW-NB15 dataset, the NSL-KDD dataset is less complex and unsuitable for building reliable modern-day IDS models. Furthermore, among the five used algorithms, Random Forest achieved the best performance on both datasets with an accuracy of 99.75% and 98.51% on NSL-KDD and UNSW-NB15 respectively.

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

1. INTRODUCTION

As the Internet, communication networks and computer systems play increasingly vital roles in performing our daily tasks, 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. Initially, the first line of defense (i.e. firewall), user authentication, and data encryption are used, but they were proven insufficient. Due to their limitations, Intrusion Detection Systems (IDSs) are nowadays being utilized to monitor any intrusions of computers and networks. IDSs use specific analytical techniques to detect attacks, identify their sources, and alert network administrators.

Anomaly and Signature based detection are the two major approaches used by IDSs. 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 less reliable, and thus, anomaly-based IDS is becoming an area of interest for cybersecurity researchers [1]. In the quest for developing more reliable and efficient IDSs, 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. 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 IDSs models and is influential on both 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 considered outdated (NSL-KDD) 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 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 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 below summarized 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	Primary Dataset	ML Algorithms	Total IDS Models Built
Program A	All Features	Yes	NSL-KDD and UNSW-NB15	All selected algorithms	10
Program B	Selected Features	Yes			10
Program C	Selected Features	No			10
Total number of IDS models implemented					30

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

1) Provides various in-depth comparisons on several aspects such as feature selection, normalization, datasets, and IDS models.
2) Proposes a hybrid approach towards IDS modeling using a classifier for feature selection alongside another classifier in implementing the IDS model to increase the accuracy and efficiency of IDS.
3) 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 aspects.
4) 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 are presented in the next section. Section 3 presents an explanation of some basic IDS concepts, Machine learning, and the two preprocessing techniques. The experimental procedures and tools used are presented in Section 4, while Section 5 provides the evaluation results along with discussions. Finally, conclusions are drawn, as well as limitations and directions for future research are suggested in Section 6 and 7 respectively.

2. LITERATURE REVIEW

2.1. Related Works

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. A summary of the related works is shown in Table 2.

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 measure 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 is 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 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 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 KDD99 and UNSW-NB15 datasets. DSAE feature selection was used to select 10 features in each dataset, which are 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 KDD99 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.

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 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.

2. 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 that can improve the accuracy and efficiency of especially classification algorithms [23] and has been shown to improve the accuracy of IDS models built with large datasets [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. Even so, [15] chooses to use it at the expense of normalization. Furthermore, most of the reviewed studies give very little emphasis on the impact of normalization on the performance of IDS models.

3. Feature selection

In recent years, due to the high dimensionality and size of IDS datasets, many researchers are using dimensionality reduction methods to reduce dataset dimension and select optimal subset features to represent 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 (explained in the next section of this work), only two of these methods were mainly applied in IDS modeling, and most of the reviewed literature used 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 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.

4. 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 of the reviewed work [15], [18] without validating their performances by comparing them with other algorithms. Unfortunately, tree-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].

5. Dataset Issues

Most of the reviewed literature made use of either KDDcup99 or its variant NSL-KDD which are among the widely used datasets in IDS academic research [6], [7]. However, despite that, they are currently 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. Some modern datasets were proposed by [27]–[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] made recommendations on using more than one dataset with at least one publicly available dataset to avoid overfitting of IDS model to one dataset and ensure the reproducibility 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 methods [8], is also used. Five among the 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 are developed and evaluated. In what follows next, the five selected ML algorithms are explained, and the feature selection concept is also introduced.

Table 2 - Summary of Related Works

Ref.	FS Method (No. of selected features)	ML Algorithm	Normalization type/method	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 Nuerotree	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 wrapper KDD99 (18/41) UNSW-NB (20/42)	C4.5, Random Forest, 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)

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 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 (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 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.

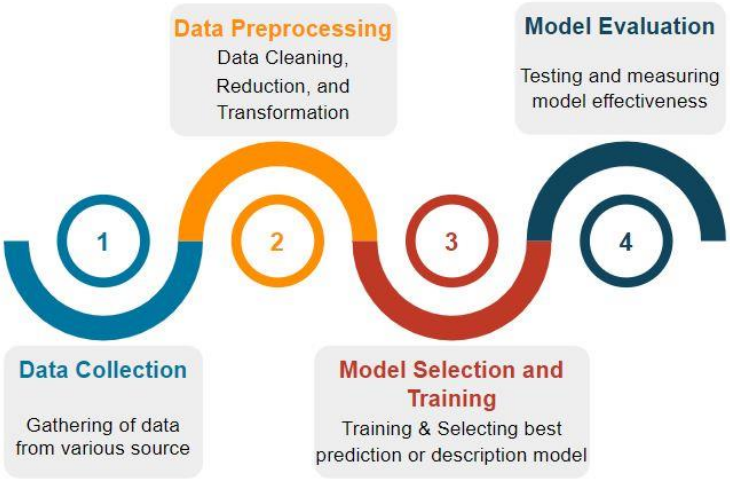

Figure 1 - Machine Learning Basic Steps

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 features 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 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 a classifier [37]. In this work, the wrapper method is used.

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:

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

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.

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 large than the number of samples [26]. In this work, Scikit-learn implementation of a support vector classifier based on LibSVM with the Radial Basis Function (RBF) as the kernel is utilized.

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 giving 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. *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].

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 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].

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 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 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, 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.

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 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 3 below

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%**

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+). Here also, 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 traffics in the training set of the NSL-KDD dataset.

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. 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).

1. Data Reduction

I. Data Filtration

As stated above, we used the UNSW-NB15 dataset (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 3. 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.

II. 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], and is employed in this work with a decision tree algorithm as the feature evaluator as shown in Figure 2.

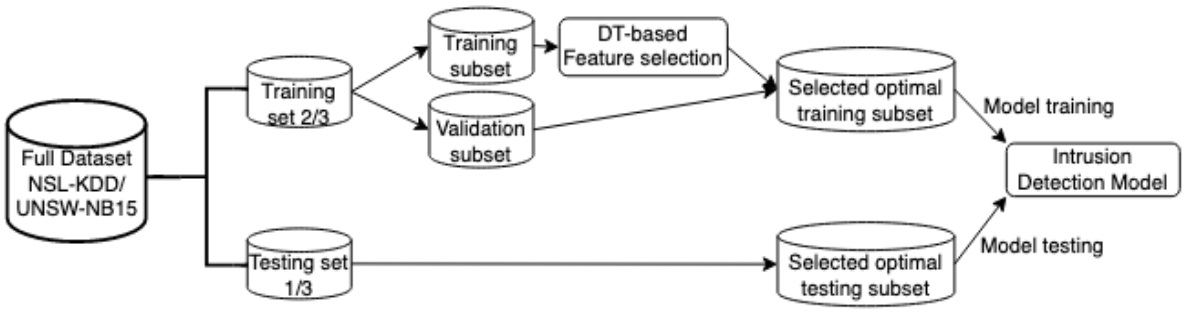

Figure 2 - DT Wrapper-Based FS

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 thus making 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 feature 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: 2 complete datasets and 2 feature selected versions of UNSW-NB15 and NSL-KDD, a description of the full datasets is available in [28] and [45] respectively, Table 5 shows the selected optimal features of the datasets.

Table 5 - 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

2. Data Transformation

I. 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 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.

II. 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 6. 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 one-hot encoded accordingly. Table 7 below 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.

Table 6 - 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 7. Only the final encoded features are used in training and 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]. In this work, the holdout method is used. Each of the datasets is divided into two sets, called the training set and the testing set. The building of the models constitutes 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 depicted the entire model training and testing process. A total of thirty (30) distinct IDS models are developed in three different programs using the 5 selected algorithms. The explanation of the selected algorithms is given in the previous section. 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. 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:

- Program A: models implemented with full, non-feature selected, normalized datasets.
- Program B: models implemented with feature-selected, normalized datasets.
- 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 anomaly 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. 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 (except for computational time) are based on the Confusion Matrix, we deemed it

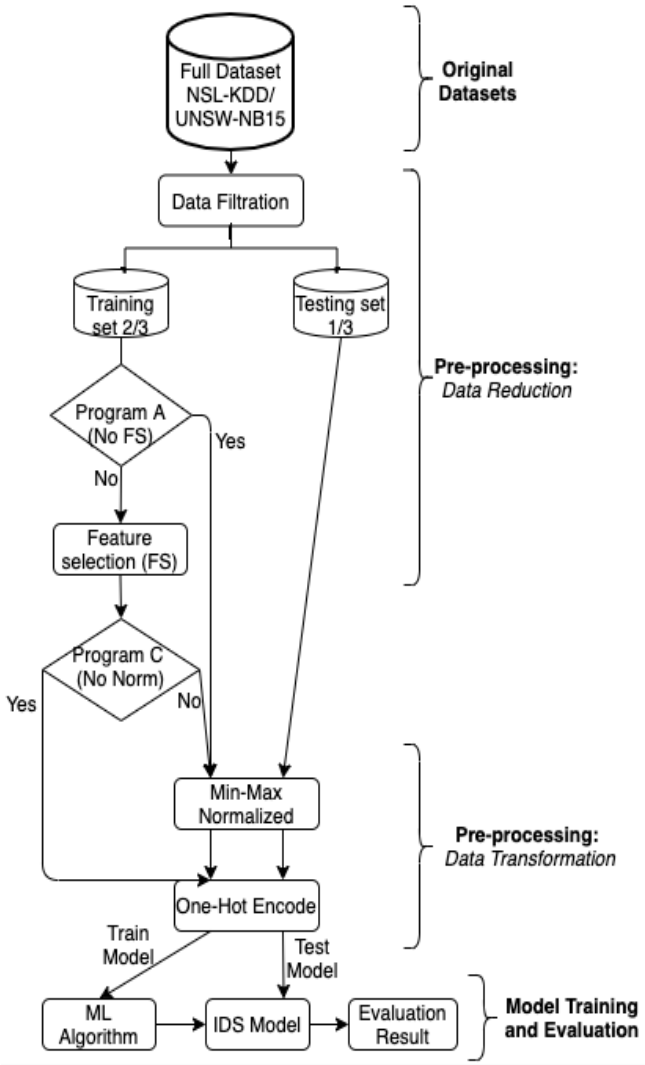

Figure 3 - Conceptual Framework of the IDS models

important to explain it.

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 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 incorrectly detected as an attack. Aka Type I error.
- True negative (TN): Number of normal correctly detected as normal.

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].

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

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.

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

4. 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.

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

5. Computational Time

In this work, the computational time is the entire time taken to train and evaluate a model exclusive of the time taken in feature selection operation. As 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, in addition to re-running the programs several times to verify the timing and ensure minimal interference.

5. RESULT AND DISCUSSION

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

- Comparison between the IDS models developed using full datasets and feature selected datasets to assess the impact of feature selection.
- Comparison between the IDS models developed using normalized and non-normalized feature selected datasets to assess the impact of normalization.
- 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 were implemented and executed in the same environment using the same programming language as shown in Table 9.

Table 9 - 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 10 below presents the evaluation results of models built using both NSL-KDD and UNSW-NB15 full features and feature selected data. Feature selection generally improve performances and reduce computational time [9], [10], [14]. The boldly written values indicate the expected improvement in performance or decrease in computational time after feature selection. 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 performed almost similarly albeit 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) improved using the 20 selected features with both KNN and RF achieving a remarkably higher performance across all the metrics, while ANN and SVM models achieved similar performance both on the selected features and 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. Remarkably, 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 were also observed from an overall perspective, Table 12 below summarizes the sum of computational time taken by the full and feature selected-based models across the two primary datasets.

Table 10 - Feature Selection Comparison

Models	**Eval. 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 11 below presents the evaluation results of models built using NSL-KDD and UNSW-NB15 normalized and non-normalized 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], were contrary. KNN's performance decreases whereas SVM's performance drastically improved. Given that the performance of SVM is not severely affected by lack of normalization in UNSW-NB15, the SVM's poor performances on non-normalized NSL-KDD raises a question about the reliability of the dataset. NB which is not a distance-based classifier also achieves two opposing results; with NSL-KDD, it performs surprisingly poor on non-normalized dataset and improved drastically on normalized NSL-KDD, however, with UNSW-NB15, it instead expectedly performed well on normalized dataset and improved its performance on non-normalized dataset with both performances in close range. RF also achieves two opposing performances across the two datasets; with NSL-KDD, it performed better without normalization and, on the contrary, it performed better with normalization on UNSW-NB15. Only ANN performed as expected, with its performances across the two datasets relatively better after normalization. Thus, as the performance of 4 classifiers on UNSW-NB15 has improved after normalization and similarly the performance of 3 of the 5 classifiers 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 is essentially as expected [40]. Both RF and NB do not necessarily need normalization [40], [54], thus their computational time with and without normalization are low and correspondingly close. 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 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 that of the models built with non-normalized data in both NSL-KDD and UNSW-NB15 as shown in Table 12, this is similar to what is observed on analyzing individual models.

Table 11 - Normalization Comparison

Models	**Eval. metrics**	**NSL-KDD**		**UNSW-NB15**	
		Normalized	Non-normalized	Normalized	Non-normalized
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 on 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 10 and Table 11 above, specifically focusing on models whose performances or computation time were 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.

1. Reliability Comparison

Since KNN typically uses Euclidian distance 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 11 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 10 above, the feature selection does not seem to improve the performance of KNN and RF (except 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 12 - 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 12, a clear decrease of models' computational time in the non-normalized datasets (Program C) can be observed after normalizing the datasets (in Program B), however, in comparison to full-featured datasets (Program A), two opposing computational time can be 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 uncertainty 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 highlight 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.

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.

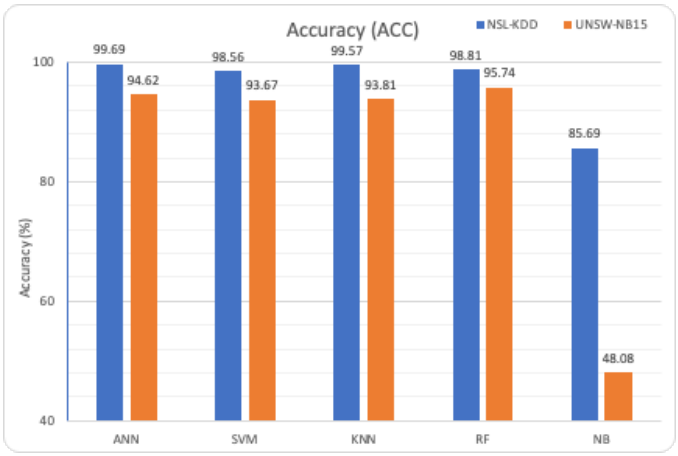

Figure 4 - Program A Models Accuracy

In Figure 4, all NSL-KDD models outperform their corresponding UNSW-NB15 implemented models. ANN and RF achieve the highest accuracy of 99.69% and 95.74% using NSL-KDD and UNSW-NB15 respectively, while both datasets achieved their worst accuracy with NB. Furthermore, it can be observed that, in the detection rate depicted in Figure 5, 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 lowest DR obtained for both datasets is using NB with its UNSW-NB15 model achieving the poorest DR of just 23.91%.

Figure 5 - Program A Models Detection Rate

Very low false alert rates are generally the target in IDS. The FAR depicted in Figure 6 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.

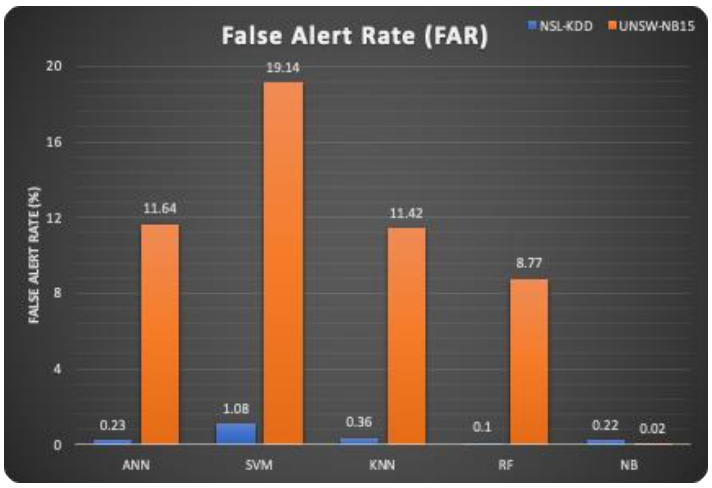

Figure 6 - Program A Models False Alert Rate

Overall, 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.

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 paper 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 from this work:

- 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 13 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.
- 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 take the lowest time, it however achieved the lowest performances on average, making it the worst IDS algorithm in this work, and, in comparison to many supervised ML algorithms [55], the poorest.
- 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.
- It is 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. As future work, it will be interesting to study the effect of various encoding methods in IDS modeling.

Table 13 - Preprocessing Effects Summary

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

LIMITATIONS AND FUTURE WORK

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 some limitations and open gaps for future research as follow:

- *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 would be interesting to use more datasets and more algorithms for broader insights.
- *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 changing of evaluation algorithm or searching strategy and soon. Similarly, the use of different normalization methods can also be explored.
- *Multi-classification*: This work primarily focused on the binary classification of normal and attack network traffics, however, IDS datasets typically contained 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 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.

ACKNOWLEDGMENT

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

REFERENCES

[1] T. Sowmya and E. A. Mary Anita, "A comprehensive review of AI based intrusion detection system," *Measurement: Sensors*, vol. 28, p. 100827, Aug. 2023, doi: 10.1016/J.MEASEN.2023.100827.

[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] K. Albulayhi, Q. A. Al-Haija, S. A. Alsuhibany, A. A. Jillepalli, M. Ashrafuzzaman, and F. T. Sheldon, "IoT Intrusion Detection Using Machine Learning with a Novel High Performing Feature Selection Method," *Applied Sciences 2022, Vol. 12, Page 5015*, vol. 12, no. 10, p. 5015, May 2022, doi: 10.3390/APP12105015.

[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] S. M. Kasongo and Y. Sun, "Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset," *J Big Data*, vol. 7, no. 1, pp. 1–20, Dec. 2020, doi: 10.1186/S40537-020-00379-6/TABLES/8.

[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. Kumar, S. Gupta, S. Arora, and S. Kumar, "A comparative simulation of normalization methods for machine learning-based intrusion detection systems using KDD Cup'99 dataset," *Journal of Intelligent and Fuzzy Systems*, vol. 42, no. 3, pp. 1749–1766, 2022, doi: 10.3233/JIFS-211191.

[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, Y. Breitbart, and C. Muoh, "Data discretization unification," *Knowl Inf Syst*, vol. 19, no. 1, pp. 1–29, May 2009, doi: 10.1007/S10115-008-0142-6/METRICS.
[23] M. Azizjon, A. Jumabek, and W. Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," *2020 International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2020*, pp. 218–224, Feb. 2020, doi: 10.1109/ICAIIC48513.2020.9064976.
[24] K. A. Taher, B. Mohammed Yasin Jisan, and M. M. Rahman, "Network intrusion detection using supervised machine learning technique with feature selection," *1st International Conference on Robotics, Electrical and Signal Processing Techniques, ICREST 2019*, pp. 643–646, Feb. 2019, doi: 10.1109/ICREST.2019.8644161.
[25] M. A. Umar, Z. Chen, and Y. Liu, "A Hybrid Intrusion Detection with Decision Tree for Feature Selection," *Information & Security: An International Journal*, 2021, doi: 10.11610/ISIJ.4901.
[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] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya, "An explainable deep learning-enabled intrusion detection framework in IoT networks," *Inf Sci (N Y)*, vol. 639, p. 119000, Aug. 2023, doi: 10.1016/J.INS.2023.119000.
[33] M. Alkasassbeh and S. Al-Haj Baddar, "Intrusion Detection Systems: A State-of-the-Art Taxonomy and Survey," *Arab J Sci Eng*, vol. 48, no. 8, pp. 10021–10064, Aug. 2022, doi: 10.1007/S13369-022-07412-1/METRICS.
[34] D. H. Lakshminarayana, J. Philips, and N. Tabrizi, "A survey of intrusion detection techniques," *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, pp. 1122–1129, Dec. 2019, doi: 10.1109/ICMLA.2019.00187.
[35] M. R. Ayyagari, N. Kesswani, M. Kumar, and K. Kumar, "Intrusion detection techniques in network environment: a systematic review," *Wireless Networks*, vol. 27, no. 2, pp. 1269–1285, Feb. 2021, doi: 10.1007/S11276-020-02529-3/METRICS.
[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] M. A. Umar, C. Zhanfang, and Y. Liu, "Network Intrusion Detection Using Wrapper-based Decision Tree for Feature Selection," *ACM International Conference Proceeding Series*, pp. 5–13, Jan. 2020, doi: 10.1145/3424311.3424330.
[38] J. Han, J. Pei, and H. Tong, *Data Mining – Concepts & Techniques*, 4th Edition. USA: Morgan Kaufmann Publishers, Elsevier Inc, 2022.
[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] B. P. Yadav, S. Ghate, A. Harshavardhan, G. Jhansi, K. S. Kumar, and E. Sudarshan, "Text categorization Performance examination Using Machine Learning Algorithms," *IOP Conf Ser Mater Sci Eng*, vol. 981, no. 2, p. 022044, Dec. 2020, doi: 10.1088/1757-899X/981/2/022044.
[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/.
[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.
[47] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques*, 4th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.
[48] M. Samadi Bonab, A. Ghaffari, F. Soleimanian Gharehchopogh, and P. Alemi, "A wrapper-based feature selection for improving performance of intrusion detection systems," *International Journal of Communication Systems*, vol. 33, no. 12, p. e4434, Aug. 2020, doi: 10.1002/DAC.4434.
[49] scikit-learn 0.22.2, 'Decision Trees — scikit-learn 0.22.2 documentation'. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html.
[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.

Authors' Profiles

Mubarak Albarka Umar received the BSc. (Honors) Degree in Software Engineering from the University of East London, Meng., Degree in Computer Applied Technology from the Changchun University of Science and Technology. He is currently a Ph.D. student at the United Arab Emirates University. He has published 9 articles in peer-review international journals and has 3 conference papers. His research interests include software engineering, network security, and artificial intelligence.

Zhanfang Chen, Ph.D., Professor of the School of Computer Science and Technology, Changchun University of Science and Technology, China. He's worked on over 20 projects and has published over 30 articles in international journals and numerous conference papers. His research interests include network engineering, software engineering, computer architecture, data mining, soft computing, and software engineering.

Khaleed Shuaib, Ph.D., received both his B.E., M.S., and Ph.D. from The City University of New York, US. He is currently a Professor at the College of Information Technology (CIT), United Arab Emirates University, UAE. He has over 100 refereed publications in journals and conferences and three U.S. patents. His research interests are in the areas of communication networks, network security, artificial intelligence, smart healthcare systems and smart grid.

Yan Liu, Ph.D., received her BSc., M.S., and Ph.D. from Harbin Institute of Technology, Shanghai Jiaotong University, and The Open University, UK respectively. She is currently an Associate Professor at the Department of Computer Science, College of Engineering, Shantou University, Shantou, China. She has published over 10 articles in international journals and several conference papers. Her research interests include supply chain management, software engineering, information systems (business informatics), and expert systems.