IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.8, Issue.7, July 2020 E-ISSN: 2347-2693

A Review of Big Data in Network Intrusion Detection System: Challenges, Approaches, Datasets, and Tools

Reem Alshamy[1*], Mossa Ghurab[2]

[1,2]Dept. of Computer Science, Faculty of Computer and Information Technology (FCIT), Sana'a University, Sana'a, Yemen

**Corresponding Author: r.alshamy@su.edu.ye, Tel.: +967-770609449*

DOI: https://doi.org/10.26438/ijcse/v8i7.6275 | Available online at: www.ijcseonline.org

Received: 07/July/2020, Accepted: 19/July/2020, Published: 31/July/2020

Abstract— Intrusion Detection System (IDS) is a promised research field in the cybersecurity due to the rapid development of the Internet. Many IDS employ classification algorithms for classifying network traffic, and these classification algorithms failed to achieve accurate attack detection due to the huge amount of data. However, by applying dimensional reduction, data can be efficiently reduced and achieve accurate attack detection. The main work in this paper is to provide a comprehensive review of the IDS types and methods used to detect attack, advantages and disadvantages of each type. Furthermore, the authors focus on the Network Intrusion Detection System (NIDS) type and introduce the ten characteristics of Big Data and the challenges of Big Data in NIDS. Furthermore, we analyze different approaches used in NIDS based on machine learning algorithms, for each approach we study the performance of classifiers (Binary or Multi classification) under eight datasets and dimensional reduction techniques. A comparison of some machine learning algorithms and the five tools used for analyzing Big Data are presented. Discussions came from our analysis of current research. Finally, we will finish this paper by representing conclusions and describe future work.

Keywords— Big Data, Network Intrusion Detection System, Classification, Big Data Techniques

I. INTRODUCTION

Recently, the number of Internet users has grown, this has led to the created a large number of data and the emergence of various types of attacks, this large amount of data is called Big Data [1]. Providing the protection and privacy for Big Data is one of the most challenges facing developers of security management systems, especially with the widespread use of the internet networks and the rapid growth of data generated from multi sources, this creates more space for intruders to launch attacks malicious [2, 3].

Intrusion detection indicates the act of disclosing actions that attempt to compromise the confidentiality, integrity, or availability of a resource [4]. Intrusion Detection System (IDS) is the most fundamental considerations of cybersecurity that can detect intrusion before and/or after an attack. The first to use term IDS is James Anderson in the late 1970s and early 1980s [5]. The IDS can be defined as an intrusion detection process which is to find events violation of security policies in computer networks, it is usually located within the network to monitor all internal traffics [6].

Over the years, IDS has been enhanced using various approaches such as machine learning, statistical, bio-inspired, fuzzy, Markov, and a lot more [7]. The automatic IDS is a type of Artificial Intelligence that allows computers to learn and the ability to detect intrusion. Until now, researchers have developed different IDS with the ability to detect attacks in many available environments. Machine learning is a vast field, it has a broad range of applications including medical diagnosis, natural language processing, speech recognition, pattern detection, search engines, game playing, and a lot more. It is a set of algorithms that learn through experience, which is classified as supervised, unsupervised, and reinforcement learning depending on the presence or absence of a labeled dataset [7, 8].

In supervised learning, the algorithm is trained with labeled datasets and determines a function to assign instances to classes, and the trained algorithm can predict a similar unlabeled dataset. In unsupervised learning, the algorithm is trained with an unlabeled dataset, and it works through the principle of finding the hidden design of the data by clustering or grouping similar data [9]. In reinforcement, learning concentrates on software agents that need to take action in an environment that maximizes the cumulative reward. This paper concentrates on two types of machine learning techniques (supervised and unsupervised) that are used by researchers in this field to detect attacks in the network.

A. Big Data

Big Data refers to a large amount of complex data that traditional techniques are insufficient for management.

There are various clarification of Big Data via Vs models. Big Data is usually defined in terms of 3Vs, a designation originally developed by Gartner Doug Laney [10] in 2001: Volume, Velocity, and Variety. Volume indicates the quantity of data; can be a Big Data defiance. Velocity indicates to the high speed data processing, which can be an issue with Big Data. Variety indicates the complexity of the data and this also a Big Data defiance when the data contains difficult problems such as data from heterogeneous origins or data having different data structures [11, 12].

Zikopoulous defines Big Data in terms of 5Vs [13] that add Veracity and Value to existing 3Vs of Volume, Velocity, and Variety. Veracity accounts for the data correctness and can include data quality problems such as missing values or noise which also refers to as Big Veracity. Value for Big Data indicates to the sense that if particular data does not provide an important value, which is not relevant for Big Data analysis. Big Data is also defined in terms of 10Vs. The five characteristics: Validity, Variability, Viscosity, Viability, and Volatility are added to 5Vs [14, 15]. Although these 10Vs are the characteristics of Big Data, they are known as the 10 big challenges for Big Data as well. Figure 1 shows the 10Vs characteristics of Big Data.

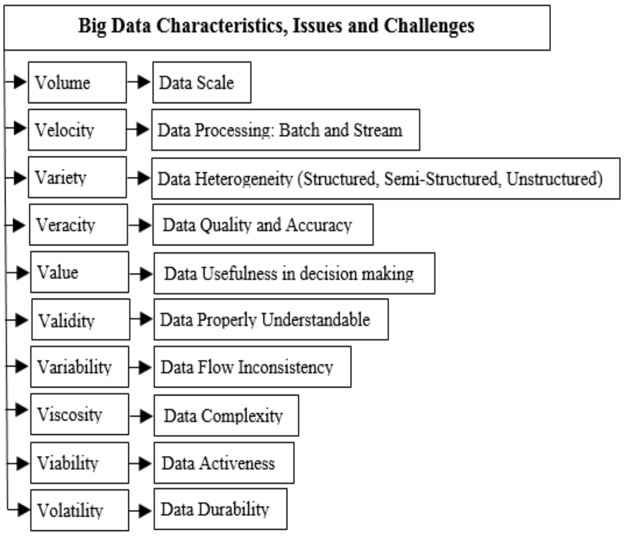

Figure 1. Big Data characteristics in terms of 10Vs [16].

B. IDS types

There are many types of IDS, which can be divided depending on the placement and method used in analyzing the events [5, 17]. Depending on the placement, the IDS divided into three types: network-based IDS (NIDS), host-based IDS (HIDS), and hybrid or mixed IDS (MIDS).

On the other hand, IDS can be divided based on the detection method into three types methods: A signature-based IDS (also known as a misuse-based IDS), anomaly-based IDS, and hybrid-based IDS [18, 19]. Description of IDS types based on the placement and the detection methods are displayed in Table 1.

Table 1. IDS types based on the placement and the detection method.

Classification Aspect	IDS Type	Description
Monitoring environment (placement)	HIDS	It monitors activities and events of the host or device on the network (such as logs, system calls, file system modifications, and incoming and outgoing packets to and from the host) by analysing any change that occurs within hosts to discover unauthorized behaviours [20, 21]. It can detect intrusions by comparing a predefined pattern with the logs of the operating system [19].
	NIDS	It checks communications in a network to observe intrusions [21]. It attempts to identify unauthorized, illegal, and anomalous activities based solely on network traffic [18].
	MIDS	It combines Host-based (HIDS) and Network-based (NIDS) in a network for more efficient and effective detection of cyber-attacks.
Detection method	Signature-based	Refers to detecting network attacks by searching for specific data patterns, this term originates from anti-virus applications, which refer to these detected patterns as signatures.
	Anomaly-based	Primarily introduced to detect unknown attacks or zero-day attacks, this is partly due to the rapid development of malware. Machine learning techniques are trained to create a model and then compare the new behavior to this model [22].
	Hybrid	The combination of a signature-based IDS with an anomaly-based IDS

This paper also displays the Advantages and Disadvantages of IDS types to provide a general overview of the features and disadvantages of each type. The Advantages and Disadvantages of IDS types are exhibited in Table 2.

Table 2. Advantages and Disadvantages of IDS types.

Classification Aspect	IDS Type	Advantages	Disadvantages
Monitoring environment (placement)	HIDS	It analyze encrypted data and communication activities. It does not require additional hardware [23].	HIDS breakdown if the operating system crashes by the attack. HIDS tends to be resource-intensive.
	NIDS	The operating	It does not indicate

		environment is independent, so NIDS will not affect host performance [5].	whether the attack was successful or no. Encrypted traffic cannot be analysed. Internal attacks are difficult to detect by NIDS in this case [19].
	MIDS	More flexible and efficient. Hybrid or mixed IDSs (MIDS) takes advantage of the strengths of the combined types.	High overhead load on the monitored system based on embedded methodologies.
Detection method	Signature-based	The simplest and effective way to detect known attacks [22]. More signatures work well versus a fixed behavioural pattern only.	This approach is not good for finding unknown attacks [17]. Increasing the amount of zero-day attacks [21]. Need to update signatures [4].
	Anomaly-based	Effective to detect new attack. Less depending on operating system.	A high false-alarm rate (FAR), it may consider the unknown legitimate activity as malicious activity. Difficulty of defining rules.
	Hybrid	It takes advantages of both methods.	High resource consuming.

The result of the comparison between IDS types can help developers for easy understanding of IDS types and researchers to develop appropriate types. This paper will concentrate on the use of anomaly-based IDS method in a Network Intrusion Detection System.

C. Network Intrusion Detection System

Nowadays society increasingly dependents on the use of computers in various areas such as security, finance, and many aspects of daily life. On the other hand, threats and attacks on the network are increasingly. The cyber-security Research Area looks at the ability to act proactively to mitigate or prevent attacks. The NIDS is placed at a strategic point in the network where it monitors all the traffic; it responsible for analyzing traffic to detect potential attacks on the network. As a solution to detect new attacks, machine learning techniques are used.

Mostly, NIDS follows one of the two major detection mechanisms: Anomaly-based network intrusion detection and Signature-based network intrusion detection. Moreover, many researchers have offered hybrid methods; each method detection has weaknesses and strengths. Anomaly-based IDS detection method is major in detecting network level attacks. It is better than Signature-based IDS in detecting new attacks. The machine learning model is trained to distinguish between normal and abnormal activity [4, 6].

D. Big Data in Network Intrusion Detection System

In 1994, a study by Frank [24] for Intrusion Detection focusing on data reduction and classification found: “a user typically generates between 3 – 35 Megabytes of data in eight hours and it can take several hours to analyze a single hour’s worth of data.” They further suggested that filtering, clustering and feature selection on the data are important if real-time detection is desired”, which can improve detection accuracy. This example indicates that Big Data challenges in Intrusion Detection appeared long before the term "Big Data" was introduced. Big Data techniques can alleviate the challenges and costs that Big Data imposes on Intrusion Detection [25].

Due to the complexity of network data, Big Data techniques are very important for analyzing network patterns and finding out what has happened in the network. Moreover, network data faces big problems with high dimensionality [11, 26]. NIDS should be dealing with problems such as large traffic volumes and high dimensionality [27].

E. Challenges in Network Intrusion Detection System

Although many techniques have been developed, NIDS is still facing many issues that need to be addressed. Some of these issues are:

1) Huge amount of data

The NIDS must have low computational complexity for training as well as for testing (to be able to learn the behaviour of new attacks) [28].
To solve this problem, there are two strategies used by researchers:

a) Active learning strategies can be used to identify relevant input samples for training instead of using the full training dataset [27].
b) Big Data platforms can be used to solve this problem [2, 29].

2) High false alarms

A false alarm rate of anomaly-based IDS is a crucial concern [30]. Most NIDS has a high false positive rate (FPR) that can be catastrophic on the network. If the classifier generates falser positives, an attacker can easily exploit network vulnerabilities. In the case of false negatives, an alarm is raised even if the packet is normal. This leads to waste time and effort for the network administrators [29].

To solve this problem, probabilistic data mining and machine learning techniques must be used [31].

3) Imbalance Data

The dataset is imbalanced if the classification class is not distributed evenly [31].

This problem can be solved by implementing a weighted extreme learning machine (ELM) to improve performance, another method to solve imbalanced data, the class balance of the training dataset is adjusted through resampling before learning [29].

The rest of this paper is organized as follows, Section I contained the introduction of Big Data, Intrusion Detection System types, NIDS, Big Data in NIDS, and challenges in NID, Section II offers the related work, Section III analyzes related work, Section IV provides the Big Data tools and techniques, section V shows discussions and recommendations, Section VI concludes this work with future work.

II. RELATED WORK

Many approaches have been offered to solve the problem of improving the efficiency of NIDS using machine learning techniques. However, very bordered research available on Big Data. Therefore, many researchers intend to use tools and techniques for Big Data to analyzing and storing data in NIDS, which can reduce training and computation time.

This paper summarizes some studies that used Big Data or traditional techniques to solve the classification problems in NIDS using machine learning algorithms.

Table 3 ordered by publication year from newest to oldest summarizes the related work. Then datasets are used to evaluate the performance of NIDS order from oldest to recent. Furthermore, Binary or Multi classification problems, algorithms used for classification and feature selection, performance metrics are also shown.

Table 3. Summary of Related work.

Study	Year	Classifier Algorithm	Feature selection	Dataset used	Classification problem	Tools	Performance metrics
[32]	2020	Support Vector Machine (SVM)	Feature selection used	KDD99	Multi	NS-3 simulation	Accuracy: 99
[33]	2020	DEGSA-HKELM	kernel principal component analysis (KPCA)	KDD99	Multi	Not available	Accuracy: 99.00 Training time: 13.204581 s Testing time: 0.012569
				UNSW-NB 15			Accuracy: 89.01 Training time: 43.306235 Testing time: 2.567050
[34]	2020	SVM	Intelligent Water Drop (IWD)	KDD99	Multi	Not available	Accuracy: 95.2 Detection rate: 95.1 Precision: 95.3
[35]	2020	SVM-based neighbor classification	Elman neural network	KDD99	Binary	Not available	Detection rate: 87.3 False alarm rate: 87.3
[36]	2020	Kernel Extreme Learning Machine (KELM)	Genetic Algorithms (GA)	KDD99	Multi	Not available	Detection rate: 97.88
				NSL-KDD			Detection rate: 94.01
[37]	2020	MeanShift	All feature used	KDD99	Multi	Python	Accuracy: 81.2 Detection rate: 79.1
[38]	2020	Weighted k-Nearest Neighbour WK-NN	Hyperbolic tangent function	Kyoto 2006+	Binary	Not available	Accuracy: (99.5%)
[39]	2020	RBF SVM	Information Gain Ratio	NSL-KDD	Binary	Not available	Accuracy: 96.24 Computation time: 4.90
[40]	2020	C5	Information Gain	UNSW-NB 15	Multi	IOT environment	Accuracy: 89.86 Detection rate: 99.32 False alarm rate: 0.72
[41]	2020	Fast kNN (FkNN)	variance function	CICIDS 2017	Binary	Java	Accuracy: 99.8 Precision: 99.91 Recall: 99.93 Computational time: 1,784
[42]	2020	AdaBoost	All features	CSE-CIC-IDS2018	Multi	Python	Accuracy: 95.49
[43]	2019	Random forest	All feature	KDD99	Multi	Weka	Accuracy:93.775 True positive rate TPR):93.8 False positive rate (FPR):0.1 Precision:99.1 ROC area:99.6
[44]	2019	Network Anomaly Detection Algorithm	Relief-F	KDD99	Binary	MATLAB	Detection rate: 94.66 Accuracy: 97.02

		(NADA)					False alarm rate: 00.47 F-score: 83.31 MCC: 82.42
				Kyoto 2006+			Detection rate: 90.10 Accuracy: 98.22 False alarm rate: 01.13 F-score: 91.24 MCC: 90.26
[45]	2019	SVM	Principal component neural network (PCNN)	KDD99	Multi	Not available	Correct Rate: 97.42 False Alarm Rate: 1.48 Average Recognition Time (ms): 0.38
[46]	2019	k-means and Random forest	Correlation coefficient	KDD99	Binary	Not available	Accuracy: 99.97 True Positives 1.000 False Positives 0.000 F-Measure 0.999 Training time: 235.52s Predict time: 4.29e-5
[47]	2019	SVM	kernel functions	NSL-KDD	Multi	MATLAB	Accuracy: 98.7 Error: 1.3
[48]	2019	Artificial Neural Network (ANN)	Correlation based	NSL-KDD	Binary	Weka	Detection Rate: 94.02
[49]	2019	SVM	Hyper Clique—Improved Binary Gravitational Search Algorithm (HC-IBGSA)	NSL-KDD	Binary	Python	Accuracy 98.85 Detection rate 98.72 False alarm rate 1.27
				UNSW-NB 15			Accuracy 94.11 Detection rate 98.47 False alarm rate 2.18
[50]	2019	distributed online averaged one dependence estimator (DOAODE)	Averaged One Dependence Estimator (AODE)	UNSW-NB 15	Multi	Not available	Accuracy: 83 Training time: less than 10 seconds.
[51]	2019	K-Nearest Neighbors (KNN)	Feature selection	CICIDS 2017	Multi	Not available	Precision: 99.53 Recall: 99.55 F1-Score: 99.50 Accuracy: 99.55
[52]	2019	Artificial Neural Networks.	All features	CSE-CIC-IDS2018	Binary	Anaconda	Training Accuracy: 0.99 Testing Accuracy: 0.99
[53]	2018	SVMwithSGD	Chi-Square	KDD99	Binary	Apache spark	AUROC: 99.55 AUPR: 96.24 Training time: 10.79 s Predict time: 1.21 s
[54]	2018	Decision Tree|	All feature	KDD99	Multi	Anaconda Fog computing	Calculation time: 0.655
[55]	2018	Logistic regression	All feature	KDD99	Multi	Apache spark	Accuracy: 99.1 Precision: 98.9 Recall: 99.5 F-Measure: 99.2 Prediction time (h): 0.089
[56]	2018	k-Means	All feature	KDD99	Binary	Apache Spark	Time computation.
[57]	2018	Kmeans++	PCA	KDD99	Binary	Anaconda	Time complexity
[58]	2018	Random forest	Information Gain (IG) (filter method)	NSL-KDD	Binary	MATLAB	FP: 0.001 TP: 0.993 Accuracy: 99.33 Precision: 0.993
[59]	2018	SVM	Multi-Linear Dimensionality Reduction (ML-DR)	NSL-KDD	Multi	MATLAB	Accuracy: 98.44 False Alarm Rate: 0.112
[60]	2018	k- nearest neighbor (k-NN)	Information Gain Ratio (IGR)	NSL-KDD	Binary	Weka	Accuracy: 99.07
[25]	2018	Random forest	All feature	UNSW-NB 15	Binary	Apache spark	Accuracy: 97.49 Sensitivity: 93.53 Specificity: 97.75 Training Time: 5.69

							Prediction Time: 0.08
[61]	2018	Random Tree	Linear Discriminant Analysis (LDA) (filter method)	UNSW-NB 15	Binary	Apache spark	Accuracy: 93.56 FPR: 0.025 Precision: 0.861 Recall: 0.865 ROC Area: 0.974 Training Time: 2.55
[62]	2018	Restricted Boltzmann Machine (RBM) Contrastive Divergence (CD)	Feature extraction	ISCX 2012	Binary	Weka	Accuracy: 88.6 True positive rate: 88.4 True negative rate: 88.8
		Persistent Contrastive Divergence (PCD)					Accuracy: 89 True positive rate: 84.2 True negative rate: 93.8
[63]	2018	K-Nearest Neighbors (*KNN*)	Feature selection	CIDDS-001	Multi	Weka	Average accuracy: 99
[64]	2018	Random forest	decision tree	CICIDS 2017	Multi	Apache Spark	Precision: 96.4 Recall: 96.9 F1: 96.6 Build time (s): 0.03 Detect Time (s): 0.01
[65]	2017	Multi-Class SVM	Information Gain Feature Selection (IGFS)	KDD99	Multi	Not available	Accuracy:90.59 Calculation time: 52.25
[66]	2017	SVM	PCA	KDD99	Multi	Apache Spark	Accuracy: 92.48 Computation time.
[67]	2017	SVM	All feature	KDD99	Multi	Apache Storm	Accuracy: 98.03 Detection rate: 92. 60
[68]	2017	Random forest	Decision tree	NSL-KDD	Multi	Python	Accuracy: 94
[69]	2017	SVM	chi-square	NSL-KDD	Multi	MATLAB	Accuracy: 98 FAR: 0.13

The related work is summarized in Table 3. Machine learning algorithms were used to design anomaly-based IDS to detect intrusion in network, both supervised and unsupervised learning methods were used. The next sections will analyze approaches that have been used by researchers in the related work.

III. ANALYSIS OF RELATED WORK

Many researchers have suggested different approaches and techniques to improve the NIDS efficiency since the late 1980s. They suggested various approaches and techniques that summarized in the related work. Figure 2 shows the general methodology that the researchers follow her in the related work to detect intrusions.

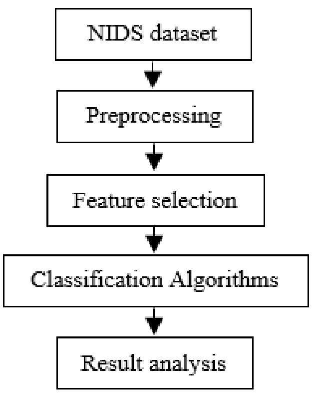

Figure 2. General methodology for NIDS.

Most researchers have followed the same steps to improve NIDS efficiency. These steps are:

1. Determine the dataset: NIDS dataset plays a vital role in validating any NIDS approach by allowing researchers to evaluate the ability of the proposed model in detecting intrusion behavior.
2. Preprocessing: the dataset must be processed to be compatible with machine learning techniques.
3. Feature selection/extraction: the best group of features can be chosen from all features using different techniques to reduce time consumption and improve the accuracy of NIDS.
4. Classification algorithms: are used to determine whether it is a normal behavior intruder or anomalies.
5. Finally, performance metrics are used to estimate results. The following subsections will be focused on providing an overview of the datasets that researchers used in the related work, how datasets are processed, techniques they used to reduce the dimensionalities, classification problems in NIDS, and performance measures that researchers used in evaluating the proposed model.

A. NIDS dataset

Several datasets are publicly available to assess the proposed models. Datasets are used to analyze network packets in commercial products that are not readily available due to privacy issues [70]. However, publicly datasets are available such as KDD99, the NSL-KDD, Kyoto 2006+, UNSW-NB15, and ADFA-LD [71, 72]. In the related work, researchers used eight datasets namely: KDD99, NSL-KDD, KYOTO 2006+, ISCX2012, UNSW-NB 15, CICIDS2017, CIDDS-001, and CSE-CIC-IDS2018, which publicly available.

Machine learning approaches were applied to improving NIDS in Table 3. We can notice the most studies used three datasets from Table 3, including the KDD99 dataset, NSL-KDD dataset, and UNSW-NB 15 dataset. However, other datasets can be used for improving NIDS. Table 4 presents these datasets, comparing them based on the content and different parameters, ordered by publication year from oldest to recent.

Table 4. Comparison of benchmark datasets for NIDS.

Dataset	Year	Modern attacks	Duration of data collected	Number of features	Number of attack	Publicly available
KDD	1999	No	7 weeks	41	4	Yes
NSL-KDD	2009	No	16 h and 15 h	41	4	Yes
KYOTO 2006+	2009	No	3 years	24	3	Yes
ISCX2012	2012	Yes	7 days	14	7	Yes
UNSW-NB 15	2015	Yes	16 h and 15 h	49	9	Yes
CIDDS-001	2017	Yes	4 weeks	14	5	Yes
CICIDS-2017	2017	Yes	5 days	86	14	Yes
CSE-CIC-IDS2018	2018	Yes	10 days	80	7	Yes

B. Preprocessing

Preprocessing is crucial for the dataset in NIDS to enhance the machine learning algorithm for the classification of the patterns. This dataset is composed of large data, redundant and different types of data that present crucial challenges to data modeling and knowledge discovery, this produces the results to be overfitting. Overfitting causes the model to perform well on the training set, but not as well on the test data. These data characteristics made it necessary to preprocess the data before using it for building the NIDS model [73, 74]. The preprocessing steps are as follows:

1) Transformation

Input data for the model may contain different types of values (binary, numeric, symbolic). The NIDS datasets contain numeric and some non-numeric features. Non-numeric features need to be converted as numeric features because the training input and testing input should be numeric [75, 76].

2) Standardization

In machine learning, the standardization of datasets are very significant for algorithms that use Euclidean distance. If they are not standardized, there is a potential that features that have values in a larger range may have been given greater importance. Since not every feature may be represented in the same measurement range, features of different sizes will have a negative impact on machine learning algorithms. This can be avoided by standardization through converting data to the same range [75, 77].

C. Dimensionality Reduction

There are varied techniques to perform dimensional reduction on high dimensional data, many different feature selection/extraction methods that are widely used [77, 78]. All of these methods aim to remove redundant and irrelevant features, so that the classification of new instances can be more accurate and the complexity time is reduced if the number of features of the dataset is reduced [79]. As the dimension increases, the computational cost also increases, usually exponentially. There are two techniques that are often used:

1) Feature Extraction

Feature extraction creates new features as combinations of others to reduce the dimensionality of the selected features from the original features through some functional mapping to reduce the cost of feature measurement, increase classifier efficiency, and improve classification accuracy [80].

There are varied ways for feature extraction to reduce data dimensionality that has been widely used such as Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA) [81, 82].

2) Feature Selection

Feature selection methods are widely used, as a dimensionality reduction technique aims for selecting a small subset of related features from the original features by removing redundant, irrelevant, or noisy features [83].
The main difference between feature selection and extraction methods is that feature selection method is used to achieve the subset of the most related features without repeating them, Feature extraction methods are used to decrease dimensionality by combining existing features [84].

The advantages of feature selection in machine learning are [77, 85]:

1. Reduce the dimensionality of feature space.
2. Speed up a learning algorithm.
3. Improve the predictive accuracy of a classification algorithm.
4. Improve the comprehensibility of the learning results.
5. Performance improvement, to gain in predictive accuracy.
6. Removes the redundant, irrelevant or noisy data.
7. Improving the data quality.

There are two methods for feature selection and reduction are:

a) Filter Method

In this method, instead of taking chosen features, the ranks of all features in the dataset are assigned by using features evaluator and a ranker method [86].

Generally, filter methods perform feature selection before classification and clustering tasks and usually fall into a two-step strategy. First step, all features are ranked approving to certain criteria. Next step, the subset of the picked features can be the final subset which is used as the input to the classifiers. Features that have a lower rank are dropped one at a time to assess the accuracy of the classifier at that point of time [78, 87].

Many filter methods have been used such as relief, F-statistic, mRMR, and information gain. Figure 3 shows the filter feature selection method.

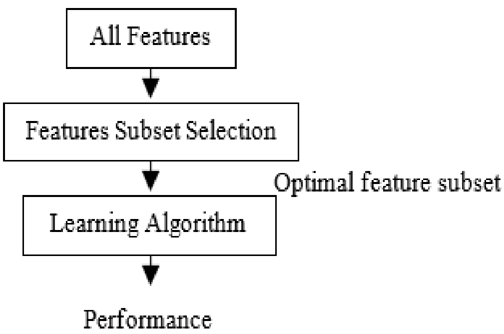

Figure 6. Filter Feature Selection method [87].

b) Wrapper Method

In this method, feasible subsets are created with the help of a subset evaluator. Varied classifiers are produced using a classification algorithm and features of every subset to find out which subset of features performs the best with the classification algorithm [88]. Figure 4 shows the wrapper feature selection method.

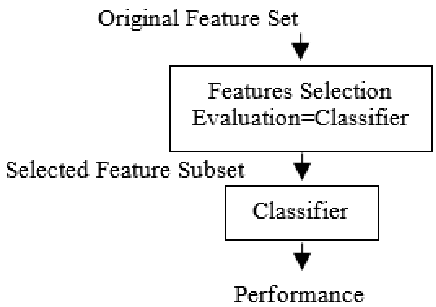

Figure 4: Wrapper Feature Selection Method [87].

The two methods for feature selection technologies have been greatly used by researchers; each method has pros and cons. Table 6 summarizes the difference between filter and wrapper methods for selected features.

Table 5. Difference between Filter and Wrapper methods.

Filter	Wrapper
The significance of features are measured by their association with a dependent variable.	The best subset of the features is measured by a really training model on it.
Faster.	Very expensive mathematically.
Statistical ways are applied to evaluate a subset of features.	Used cross validation.
It might fail to find the best subset of features.	It can always supply the best subset of features.

D. Classification

Classification is a supervised learning technique [89]. In machine learning, you encounter the problem of classification in various fields, such as temperature to mark a low, medium, or high temperature or medicine to mark a disease of a patient. Binary and Multi classifications are major problems in NIDS. In the following subsections, the difference between Binary and Multi classification problem in NIDS will be highlighted.

1) Binary Classification

Intrusion generally can be considered as a binary classification problem, classify a sample dataset as normal or attack [90]. Supervised machine learning classification techniques goal to build a learning model from a labeled training dataset to be able to classify new instances with unknown labels [91]. Several successful techniques have been suggested to solve the problem in the binary classification case.

In Table 3 we can observe that the Support Vector Machine algorithm is widely used in intrusion detection research and machine learning techniques to classify intrusion as normal or abnormal.

2) Multi Classification

Intrusion can also be considered as Multi classification problems, classify a sample dataset as normal, or a specific attack [90]. Machine learning algorithms have been suggested to solve Binary classification problem and some algorithms have been extended to solve the Multi classification problems [92].

There are common ways to solve Multi classification problems. The first way includes methods that can be extended from the Binary state. The second includes ways to convert a Multi classification problem into several Binary classification problems [65]. The third way describes by hierarchical classification methods. On the other hand, many researchers have been used the hybrid method. The popular methods for solving Multi classification problem are:

a) Extensible method

The problem of Multi classification can be solved by naturally expanding the binary classification technique of some algorithms. These include Neural Networks, Decision Trees, K-Nearest Neighbourhood, Random Forest, and Naive Bayes [93].

b) Decomposing into binary classification

The decomposing into a binary classification is the most common method used in Multi classification. It is to decompose the problem into multiple two-class classification problems and then solve those using efficient binary classifiers [93, 94]. The popular methods for solving Multi classification problem by decomposing into a binary classification are:

1. One-versus-all (OVA)

The OVA is the simplest approach to reduce the classification problem among K classes into K binary problems, each problem distinguishes a given class from the other K−1 classes.

2. All-versus-all (AVA)

In the AVA approach, the Binary classifier is prepared to distinguish between each pair of classes while eliminating the rest of the classes. It compares each class to each other.

3. Error-Correcting Output-Coding (ECOC)

The Error Correcting Output Coding (ECOC) approach is to apply Binary (two-class) classifiers to solve the Multi classification problems. It works by converting the K class classification problem into a large number of two-class classification problems. ECOC approach gives a unique code word to a class instead of assigning each class a label.

c) Hierarchical Classification

The classes in Hierarchical Classification are ordered into a tree. The tree is generated so as the classes at each parent node are divided into several clusters, one for each child node, it continues until the leaf nodes contain only a single class. At each node of the tree, a simple classifier, usually, a Binary classifier discriminates between the different child class clusters.

Furthermore, there are many studies combination of machine learning algorithms for intrusion systems to classify attacks [89, 95].

There are many machine learning algorithms have been used in NIDS researches. A comparison between machine learning techniques used in NIDS by researchers to solve the Binary or Multi classification problem also introduced. Table 6 displays the advantages and disadvantages of some algorithms were used in NIDS [28, 95, 96].

Table 6. General Comparison of Machine Learning Algorithms used in NIDS.

Algorithms	Advantage	Disadvantage
Support Vector Machine (SVM)	High training rate and accuracy. Ability to deal with high-dimensional data.	Limited to binary classifiers.
Logistic regression (LR)	High accuracy.	Limited to binary classifiers. High the training time.
Decision Tree (DT)	Ability to deal with huge data sets. High accuracy.	It is computationally intensive to build.
Bayesian Network (BN)	Simple and computationally efficient. High accuracy.	If prior knowledge is incorrect, it is possible not to contain any good classifiers. Difficult to implement and the cost is also high.
Genetic Algorithm (GA)	Can derive best classification rules and select optimal parameters.	Can be over-fitted. Constant optimization response times are not assured.
Hybrid methods	Higher attack detection rate.	High false negative rate.
Neural Networks (NN)	Do not need expert knowledge and can find novel or unknown intrusions.	Possible to over-fit during training. Not suitable for real-time detection.
Random Forest (RF)	It is improving accuracy, reducing variance and avoiding over-fitting.	Random Forest comes with an increase in computational cost.
K Nearest Neighbor Artificial	Simple in implementation. Uses local information. Very easily to parallel implementations	It can be noted that the KNN requires significantly more time during the training and testing process. Large storage requirements.

System performance can be determined depending on various metrics such as false alarm rate, detection rate, accuracy, recall, F-measure, and time taken to build the model. Performance metrics are utilized to evaluate and compare different classifiers performance. Confusion Matrix shows the relationship between well-classified records and misclassified records [45].

Table 7 shows the general confusion matrix uses in the evaluation. The terminology in the confusion matrix can be explained as follows:

- True Positive (TP): Number of records correctly detected as a normal class.
- False Positive (FP): Number of records not correctly detected as a normal class.
- False Negative (FN): Number of records not correctly detected as attack class.
- True Negative (TN): Number of records correctly detected as attack class.

Table 7. Confusion Matrix.

Confusion Matrix		Predicted value	
		Normal	Attack
Actual value	Normal	TP	FN
	Attack	FP	TN

Most performance metrics are built on the confusion matrix that is used to evaluate the performance. The values in the confusion matrix demonstrate the performance of the prediction algorithm. A detailed about the varied performance metrics for the evaluation of NIDS are shown in Table 8.

Table 8. Performance measures used to evaluate NIDS.

Measure	Description
Accuracy	The correctly classified records over all the rows of the data set. $\mathbf{Accuray} = \frac{\mathbf{TP+TN}}{\mathbf{TP+TN+FN+FP}}$
Precision/detection rate /positive prediction value	Proportion of correct labels that were classified over all labels. $\mathbf{p} = \frac{\mathbf{TP}}{\mathbf{TP+FP}}$
Recall	Proportion of correct labels that were classified correctly over all positive labels. $\mathbf{R} = \frac{\mathbf{TP}}{\mathbf{TP+FN}}$
F-measure	Harmonic average of Precision and Recall. $\mathbf{FM} = 2\frac{\mathbf{P*R}}{\mathbf{P+R}}$
False alarm rate	False positive rate (FPR) also known as false alarm rate (FAR), refers to the proportion that normal data is falsely detected as attack. $\mathbf{FAR} = \frac{\mathbf{FP}}{\mathbf{FP+TN}}$

IV. TOOLS AND TECHNIQUES

Traditional machine learning tools have been become insufficient due to the data that increases rapidly. Choosing machine learning tools for Big Data can be a difficult task due to the plenty of options [97]. Available Big Data tools have pros and cons, and many of them have overlapping uses. This section shows an overview of the tools that are used to analyzing Big Data using machine learning techniques including MapReduce, Spark, Flink, Storm, and H2O with a comparison of the engines that implement them.

A. Hadoop ecosystem

Many people see the terms Hadoop and MapReduce interchangeable but this is not entirely accurate. Hadoop ecosystem was introduced as an open-source implementation in 2007 for the MapReduce Processing linked with a distributed file system [98]. It has been developed into a vast network of projects related to every step of the Big Data workflow, including data collection, storage, and processing, and much more.

The Hadoop project itself currently consists of four units: Hadoop distributed file system (HDFS), MapReduce Data processing engine, YARN ("Yet Another Resource Negotiator"), and Common a set of common utilities needed by the other Hadoop modules [99]. To fully understand Hadoop platform, one should look at the project itself and the ecosystem that surrounds it.

B. Data processing engines

The MapReduce idea paved the way for Hadoop which played an important role in entering the era of Big Data [100, 101]. In recent years, MapReduce has begun to fall out of favour. Especially, in the machine learning community, because of its lack of speed, high overhead costs, and the reality that many machine learning tasks do not fit readily into the MapReduce paradigm.

Over the past years, many projects have been introduced that attempt to solve underlying problems inherent in MapReduce. In following subsection will show some of the most tools used to analyse Big Data.

1) MapReduce

MapReduce approach to machine learning performs batch learning, in which the training dataset is read in its entirety to build a learning model. The shortage of efficiency in speed and computational resources is the largest problem in batch model [97]. MapReduce has been hugely successful in implementing large-scale data intensive applications on commodity clusters [102].

2) Spark

Spark initially developed at the University of California, Berkeley and now a high-level Apache project is based on MapReduce. It supports iterative computing and improves speed and resource problems by utilizing in-memory computation [98]. The major abstractions used in this project are called Flexible Distributed Data Sets (RDD), which store data in-memory and provide fault tolerance without replication [103].

3) Storm

It was initially conceived to overcome deficiencies of other processors in collecting and analysing social media streams, it is used to process data in real-time. Development on Storm started at BackType, a social media analytics company, and continued at Twitter [104]. The machine learning community attaches growing importance to real-time processing. As a result, storm dependence is increasing in search environments.

4) Flink

Flink developed at the Technical University of Berlin under the name Stratosphere. It provides the ability to handle batches and stream processing, allowing Lambda Architecture to be implemented. It has its own runtime, instead of being built on top of MapReduce [96].

5) H2O

H2O is an open source framework that provides a parallel engine for processing, analytics, math, and machine learning libraries along with data preprocessing and evaluation tools. It also provides a web user interface, which makes learning tasks easier for analysts and statisticians who may not have strong programming backgrounds. For those who want to modify implementations, it offers support for Java, R, Python, and Scala [97].

There are many significant consideration for evaluation of these tools such as fault-tolerance methods, efficiency, scalability, interface language, and usability are summarized in Table 9.

Table 9. Data processing engines for Hadoop [97].

Engine	Execution model	Supported language	Associated ML tools	In memory processing	Low latency	Fault tolerance	Enterprise support
MapReduce	Batch	Java	Mahout	×	×	✓	×
Spark	Batch, Streaming	Java, Python, R, Scala	MLlib, Mahout, H_2O	✓	✓	✓	✓
Fink	Batch, Streaming	Java, Scala	Flink-ML, SAMOA	✓	✓	✓	×
Storm	Streaming	Any	SAMOA	✓	✓	✓	×
H2O	Batch	Java, Python, R, Scala	H2O, Mahout, MLlib	✓	✓	✓	✓

V. DISSCUSION AND RECOMMENDATION

Many researches attempt to find an effective model for NIDS, using machine learning techniques. In this paper, the authors introduce an overview of Big Data in NIDS. Furthermore, it offers a general review of IDS types, advantages, and disadvantages for each type. The different approaches that have been used to improving NIDS efficiency using machine learning algorithms and publicly available NIDS datasets are introduced to help researchers to open new issues and keep research time to solve problems in NIDS.

In section I, the authors discussed different IDS types and introduced an overview of it. Table 1 provided a summary of IDS types. Table 2 summarized the advantages and disadvantages for IDS types based on the environment and methods that can be used to detect attacks. Moreover, Figure 1 summarized the ten Big Data characteristics. It also introduced challenges in NIDS.

Although there are many studies to enhance the efficiency of NIDS, still many issues and challenges exist. In section II, several researchers attempt to find effective models for NIDS. Table 3 summarized the related work ordered by publication year from newest to oldest and the datasets that have been used from oldest to recent. Related work in section II focused on studies from 2017 to 2020 that were summarized in Table 3. Figure 5 displays the percentage of covered papers in the related work over the publication year.

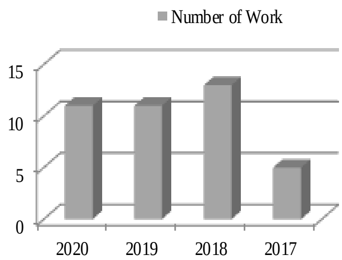

Figure 5. Related work over publication year.

The authors analysed different methodologies under eight datasets that have been used by the researcher in the related work offered in section III and the comparison between datasets was displayed in Table 4 order by publication year from oldest to recent. Preprocessing steps are important for machine learning, two methods were showed for preprocessing datasets.

The feature selection/extraction techniques for dimensionality reduction were displayed in this section. Table 5 summarized the difference between filter and wrapper methods for selected features. Furthermore, the authors focused on the Binary and Multi classification problems in NIDS and introduced many methods to solve Multi classification problems. Many machine learning algorithms used in NIDS have advantages and disadvantages. The advantages and disadvantages of some machine learning algorithms that have been used in NIDS were displayed in Table 6. In addition, the performance measures were presented in this section

Big Data tools were presented in section IV, the comparison between data processing engines for Hadoop were displayed in Table 9. Furthermore, this section provides discussions and recommendations from our analysis of the different studies which were offered in the related work.
Many approaches and techniques were used by researchers to improving the NIDS efficiency, as noted in Table 3, and some of these approaches have merits and demerits. From Table 3 the authors make the following observations and recommendations about improving the efficiency of NIDS:

- Anomaly-based IDS detection methods are prime in detecting network-level attacks, known and unknown attacks in networks.
- Machine learning techniques are useful in NIDS, but they have limitations in dealing with Big Data on the network.
- Although there are several benchmark NIDS datasets publicly available, many of them contain old-fashioned, incomplete, inflexible, and irreproducible intrusion. On the other hand, these datasets are outdated and insufficient to reflect actual network attack scenarios.
- Three important factors for development NIDS are preprocessing, features reduction, and algorithms used for classifier.
- Using effective features in designing classifiers not only reduces the dataset but also improves the performances of classifier.
- The comprehensive review shows that the false alarm rate and the detection rate of the classifier depend on the type of dataset is used, and the accuracy of the system also depends on the algorithms is used for feature selection, learning, and classification.
- Classification techniques are based on supervised and unsupervised learning.
- MapReduce, Storm, Flink, H2O, Fog, and Spark Streaming are primary open source platforms for distributed stream processing.
- Apache Spark widely uses as a Big Data processing tool because of its ability to rapidly analyse network traffic

data. It is confirmed by researchers that the Apache Spark is a fairly suitable tool for machine learning algorithms.

- When the researcher uses an old dataset to evaluate the performance of the proposed model for improving the NIDS, the authors in this paper recommend using more than one dataset to evaluate the proposed model because the old dataset not reflect modern attacks and the intruders are developing their attacks constantly.

VI. CONCLUSION AND FUTURE SCOPE

This paper provided an overall review of IDS types and introduced comparative between them, as well as introduced advantages and disadvantages of each type. Moreover, the authors introduced the challenges in NIDS. This paper analysed different approaches that have been used in NIDS based on machine learning algorithms. For each model, we studied the performance in two categories of classification (Binary or Multiclass) under eight datasets and dimensionality reduction techniques that have been used. Moreover, this paper offered a comparison of some machine learning algorithms. In addition, the authors introduced an overview of MapReduce, Spark, Flink, Storm, and H2O tools used to analyse Big Data. The outcome of this review will help in understanding the challenges of Big Data in NIDS. This paper also recommended to use up-to-date datasets and Big Data techniques. In future work, several Multi classification techniques may be studied to get more accurate classifiers on the Big Data environment.

REFERENCES

[1] D. Gaurav, J. K. P. S. Yadav, R. K. Kaliyar, and A. Goyal, "An Outline on Big Data and Big Data Analytics," pp. **74-79**.

[2] R. Devakunchari, "Analysis on big data over the years," International Journal of Scientific and Research Publications, vol. **4,** pp. **1-7, 2014.**

[3] A. Ju, Y. Guo, Z. Ye, T. Li, and J. Ma, "HeteMSD: A Big Data Analytics Framework for Targeted Cyber-Attacks Detection Using Heterogeneous Multisource Data," Security and Communication Networks, vol. **2019**, **2019.**

[4] L. Wang and R. Jones, "Big data analytics for network intrusion detection: A survey," International Journal of Networks and Communications, vol. **7**, pp. **24-31**, **2017.**

[5] S. M. Othman, N. T. Alsohybe, F. M. Ba-Alwi, and A. T. Zahary, "Survey on Intrusion Detection System Types," International Journal of Cyber-Security and Digital Forensics, vol. **7**, pp. **444-46**3, **2018.**

[6] K. Kim, M. E. Aminanto, and H. C. Tanuwidjaja, Network Intrusion Detection Using Deep Learning: A Feature Learning Approach: Springer, **2018.**

[7] S. Gulghane, V. Shingate, S. Bondgulwar, G. Awari, and P. Sagar, "A Survey on Intrusion Detection System Using Machine Learning Algorithms," pp. **670-675.**

[8] Y. Hamid, M. Sugumaran, and L. Journaux, "Machine learning techniques for intrusion detection: a comparative analysis," pp. **1-6.**

[9] P. Dehariya, "An Artificial Immune System and Neural Network to Improve the Detection Rate in Intrusion Detection System," International Journal of Scientific Research in Network Security and Communication, vol. **4**, pp. **1-4**, **2016.**

[10] E. Guerra, J. de Lara, A. Malizia, and P. Díaz, "Supporting user-oriented analysis for multi-view domain-specific visual languages," Information and Software Technology, vol. **51**, pp. **769-784**, **2009.**

[11] P. Adluru, S. S. Datla, and X. Zhang, "Hadoop eco system for big data security and privacy," pp. **1-6.**

[12] M. Kaur and A. M. Aslam, "Big Data Analytics on IOT: Challenges, Open Research Issues and Tools," International Journal of Scientific Research in Computer Science and Engineering, vol. **6**, pp. **81-85**, **2018.**

[13] P. Zikopoulos, D. deRoos, K. Parasuraman, T. Deutsch, D. Corrigan, J. Giles, et al., "Harness the Power of Big Data—The IBM Big Data Platform. 2011," www-01. ibm. com/software/data/bigdata (letzter Zugriff am 31.03. 2018), **2011.**

[14] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," Journal of Big Data, vol. **2**, pp. **3-3**, **2015.**

[15] Z. Sun, "10 Bigs: Big data and its ten big characteristics," PNG UoT BAIS, vol. **3**, pp. **1-1**0, **2018.**

[16] N. Khan, M. Alsaqer, H. Shah, G. Badsha, A. A. Abbasi, and S. Salehian, "The 10 Vs, issues and challenges of big data," pp. **52-56, 2018**.

[17] C. Zouhair, N. Abghour, K. Moussaid, A. El Omri, and M. Rida, "A Review of Intrusion Detection Systems in Cloud Computing," ed: IGI Global, , pp. **253-283, 2018.**

[18] K. Siddique, Z. Akhtar, M. A. Khan, Y.-H. Jung, and Y. Kim, "Developing an Intrusion Detection Framework for High-Speed Big Data Networks: A Comprehensive Approach," KSII Transactions on Internet & Information Systems, vol. **12, 2018.**

[19] F. A. B. H. Ali and Y. Y. Len, "Development of host based intrusion detection system for log files," pp. **281-285.**

[20] A. K. Saxena, S. Sinha, and P. Shukla, "General study of intrusion detection system and survey of agent based intrusion detection system," pp. **421-471.**

[21] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," ACM Computing Surveys (CSUR), vol. 51, pp. **1-36**, **2018.**

[22] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," Journal of Network and Computer Applications, vol. 36, pp. 16-24, 2013.

[23] L. Wang, "Big Data in intrusion detection systems and intrusion prevention systems," J Comput Netw, vol. **4**, pp. **48-55**, **2017.**

[24] J. Frank, "Artificial intelligence and intrusion detection: Current and future directions," pp. **1-12.**

[25] M. Belouch, S. El Hadaj, and M. Idhammad, "Performance evaluation of intrusion detection based on machine learning using Apache Spark," Procedia Computer Science, vol. **127**, pp. **1-6**, **2018.**

[26] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," pp. **86-93.**

[27] R. Chapaneri and S. Shah, "A comprehensive survey of machine learning-based network intrusion detection," ed: Springer, **2019**, pp. **345-356.**

[28] R. Patel, A. Thakkar, and A. Ganatra, "A survey and comparative analysis of data mining techniques for network intrusion detection systems," International Journal of Soft Computing and Engineering (IJSCE), vol. **2**, pp. **260-265**, **2012.**

[29] S. Suthaharan, "A single-domain, representation-learning model for big data classification of network intrusion," pp. **296-310.**

[30] P. Singh, S. Krishnamoorthy, A. Nayyar, A. K. Luhach, and A. Kaur, "Soft-computing-based false alarm reduction for hierarchical data of intrusion detection system," International Journal of Distributed Sensor Networks, vol. **15**, **2019.**

[31] K. K. Wankhade and K. C. Jondhale, "An ensemble clustering method for intrusion detection," International Journal of Intelligent Engineering Informatics, vol. **7**, pp. **112-140**, **2019**.

[32] M. U. Farooq, H. Xiaoli, and S. A. Rauf, "Big Data Security Analysis in Network Intrusion Detection System," International Journal of Computer Applications, vol. **975**, pp. **8887-8887**, **2020.**

[33] L. Lv, W. Wang, Z. Zhang, and X. Liu, "A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine," Knowledge-Based Systems, pp. **105648-105648**, **2020.**

[34] N. Hariyale, M. S. Rathore, R. Prasad, and P. Saurabh, "A Hybrid Approach for Intrusion Detection System," ed: Springer, 2020, pp. **391-403.**

[35] W. Fang, X. Tan, and D. Wilbur, "Application of intrusion detection technology in network safety based on machine learning," Safety Science, vol. 124, pp. **104604-104604, 2020.**

[36] J. Ghasemi, J. Esmaily, and R. Moradinezhad, "Intrusion detection system using an optimized kernel extreme learning machine and efficient features," Sādhanā, vol. **45**, pp. **1-9**, **2020.**

[37] A. Kumar, W. Glisson, and H. Cho, "Network Attack Detection using an Unsupervised Machine Learning Algorithm."

[38] D. Protić and M. Stanković, "Detection of Anomalies in the Computer Network Behaviour," European Journal of Engineering and Formal Sciences, vol. **4**, pp. **7-13**, **2020.**

[39] S. Krishnaveni, P. Vigneshwar, S. Kishore, B. Jothi, and S. Sivamohan, "Anomaly-Based Intrusion Detection System Using Support Vector Machine," ed: Springer, **2020**, pp. **723-731.**

[40] V. Kumar, A. K. Das, and D. Sinha, "Statistical analysis of the UNSW-NB15 dataset for intrusion detection," ed: Springer, **2020,** pp. **279-294.**

[41] K. V. Krishna, K. Swathi, and B. B. Rao, "A Novel Framework for NIDS through Fast kNN Classifier on CICIDS2017 Dataset," **2020.**

[42] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset," IEEE Access, vol. **8**, pp. **32150-32162**, **2020.**

[43] I. Obeidat, N. Hamadneh, M. Alkasassbeh, M. Almseidin, and M. AlZubi, "Intensive pre-processing of kdd cup 99 for network intrusion classification using machine learning techniques," **2019.**

[44] D. A. Kumar and S. R. Venugopalan, "A design of a parallel network anomaly detection algorithm based on classification," International Journal of Information Technology, pp. **1-14**, **2019.**

[45] K. Ye, "Key feature recognition algorithm of network intrusion signal based on neural network and support vector machine," Symmetry, vol. **11**, pp. **380-380**, **2019.**

[46] N. Kaja, A. Shaout, and D. Ma, "An intelligent intrusion detection system," Applied Intelligence, vol. **49**, pp. **3235-3247, 2019.**

[47] B. S. Bhati and C. S. Rai, "Analysis of Support Vector Machine-based Intrusion Detection Techniques," Arabian Journal for Science and Engineering, pp. **1-13**, **2019.**

[48] K. A. Taher, B. M. Y. Jisan, and M. M. Rahman, "Network intrusion detection using supervised machine learning technique with feature selection," pp. **643-646.**

[49] M. R. G. Raman, N. Somu, S. Jagarapu, T. Manghnani, T. Selvam, K. Krithivasan, et al., "An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm," Artificial Intelligence Review, pp. **1-32**, **2019.**

[50] M. Nawir, A. Amir, N. Yaakob, A. R. Badlishah, A. M. Safar, M. N. M. Warip, et al., "Distributed Online Averaged One Dependence Estimator (DOAODE) Algorithm for Multi-class Classification of Network Anomaly Detection System," pp. **12015-12015.**

[51] M. Alrowaily, F. Alenezi, and Z. Lu, "Effectiveness of machine learning based intrusion detection systems," pp. **277-288.**

[52] V. Kanimozhi and T. P. Jacob, "Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," pp. **33-36.**

[53] S. M. Othman, F. M. Ba-Alwi, N. T. Alsohybe, and A. Y. Al-Hashida, "Intrusion detection model using machine learning algorithm on Big Data environment," Journal of Big Data, vol. **5**, pp. **34-3**4, **2018.**

[54] K. Peng, V. Leung, L. Zheng, S. Wang, C. Huang, and T. Lin, "Intrusion detection system based on decision tree over big data in fog environment," Wireless Communications and Mobile Computing, vol. **2018**, **2018.**

[55] E. M. Kurt and Y. Becerikli, "Network Intrusion Detection on Apache Spark with Machine Learning Algorithms," pp. **130-141.**

[56] F. Karataş and S. A. Korkmaz, "Big Data: controlling fraud by using machine learning libraries on Spark," International Journal of Applied Mathematics Electronics and Computers, vol. **6**, pp. **1-5**, **2018.**

[57] K. Peng, V. C. M. Leung, and Q. Huang, "Clustering approach based on mini batch kmeans for intrusion detection system over big data," IEEE Access, vol. **6**, pp. **11897-11906**, **2018**.

[58] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," Journal of Computational Science, vol. **25**, pp. **152-160**, **2018.**

[59] S. K. Biswas, "Intrusion detection using machine learning: A comparison study," International Journal of Pure and Applied Mathematics, vol. **118**, pp. **101-114**, **2018.**

[60] B. N. Kumar, M. S. V. S. B. Raju, and B. V. Vardhan, "Enhancing the performance of an intrusion detection system through multi-linear dimensionality reduction and Multi-class SVM," International Journal of Intelligent Engineering and Systems, vol. **11**, pp. **181-19**2, **2018.**

[61] P. Dahiya and D. K. Srivastava, "Network intrusion detection in big dataset using Spark," Procedia Computer Science, vol. **132**, pp. **253-262**, **2018.**

[62] T. Aldwairi, D. Perera, and M. A. Novotny, "An evaluation of the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection," Computer Networks, vol. **144**, pp. **111-119, 2018.**

[63] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning," Procedia Computer Science, vol. **125**, pp. **709-716**, **2018.**

[64] H. Zhang, S. Dai, Y. Li, and W. Zhang, "Real-time Distributed-Random-Forest-Based Network Intrusion Detection System Using Apache Spark," pp. **1-7.**

[65] J. Maharani and Z. Rustam, "The Application of Multi-Class Support Vector Machines on Intrusion Detection System with the Feature Selection using Information Gain."

[66] H. Wang, Y. Xiao, and Y. Long, "Research of intrusion detection algorithm based on parallel SVM on spark," pp. **153-156.**

[67] M. A. Manzoor and Y. Morgan, "Network intrusion detection system using apache storm," Probe, vol. **4107**, pp. **4166-4166**, **2017**.

[68] M. C. Belavagi and B. Muniyal, "Multi Class Machine Learning Algorithms for Intrusion Detection-A Performance Study," pp. **170-178.**

[69] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," Journal of King Saud University-Computer and Information Sciences, vol. **29**, pp. **462-472**, **2017.**

[70] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set IEEE Symp,"

Comput. Intell. Secur. Def. Appl. CISDA 2009, no. Cisda, pp. **1-6**, **2009**.
[71] M. K. Siddiqui and S. Naahid, "Analysis of KDD CUP 99 dataset using clustering based data mining," International Journal of Database Theory and Application, vol. **6**, pp. **23-34**, **2013.**
[72] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," Computers & Security, **2019.**
[73] P. Kar, S. Banerjee, K. C. Mondal, G. Mahapatra, and S. Chattopadhyay, "A Hybrid Intrusion Detection System for Hierarchical Filtration of Anomalies," ed: Springer, **2019**, pp. **417-426.**
[74] C. Azad, A. K. Mehta, and V. K. Jha, "Evolutionary Decision Tree-Based Intrusion Detection System," pp. **271-282.**
[75] T. Ahmad and M. N. Aziz, "Data Preprocessing and Feature Selection for Machine Learning Intrusion Detection Systems," ICIC Express Letter, vol. **13**, pp. **93-101**, **2019.**
[76] J.-h. Woo, J.-Y. Song, and Y.-J. Choi, "Performance Enhancement of Deep Neural Network Using Feature Selection and Preprocessing for Intrusion Detection," pp. **415-417.**
[77] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," pp**. 372-378.**
[78] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," Advances in bioinformatics, vol. **2015, 2015.**
[79] A. Subasi, Practical Guide for Biomedical Signals Analysis Using Machine Learning Techniques: Academic Press, **2019.**
[80] H. Motoda and H. Liu, "Feature selection, extraction and construction," Communication of IICM (Institute of Information and Computing Machinery, Taiwan) Vol, vol. **5**, pp**. 2-2, 2002.**
[81] A. A. Aburomman and M. B. I. Reaz, "Ensemble of binary SVM classifiers based on PCA and LDA feature extraction for intrusion detection," pp. **636-640.**
[82] G. Karatas, O. Demir, and O. K. Sahingoz, "Deep learning in intrusion detection systems," pp. **113-116.**
[83] J. Miao and L. Niu, "A survey on feature selection," Procedia Computer Science, vol. **91**, pp. **919-92**6, **2016.**
[84] M. Ziaye, S. Khalid, and Y. Mehmood, "Survey of Feature Selection/Extraction Methods used in Biomedical Imaging," International Journal of Computer Science and Information Security (IJCSIS), vol. **16**, **2018.**
[85] L. Ladha and T. Deepa, "Feature selection methods and algorithms," International journal on computer science and engineering, vol. **3**, pp. **1787-1797, 2011.**
[86] P. Kumbhar and M. Mali, "A survey on feature selection techniques and classification algorithms for efficient text classification," International Journal of Science and Research, vol. **5**, pp. **9-9**, **2016.**
[87] B. Sahu, S. Dehuri, and A. Jagadev, "A Study on the Relevance of Feature Selection Methods in Microarray Data," The Open Bioinformatics Journal, vol. **11**, **2018.**
[88] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Machine learning, vol. **46**, pp. **389-422**, **2002.**
[89] M. Abdulrazaq and A. Salih, "Combination of multi classification algorithms for intrusion detection system," Int. J. Sci. Eng. Res., vol. **6,** pp. **1364-1371**, **2015.**
[90] D. S. Kim and J. S. Park, "Network-based intrusion detection with support vector machines," pp. **747-756.**
[91] M. Praveena and V. Jaiganesh, "A literature review on supervised machine learning algorithms and boosting process," International Journal of Computer Applications, vol. **169**, pp. **32-35**, **2017.**
[92] M. Aly, "Survey on multiclass classification methods," Neural Netw, vol. **19**, pp. **1-9, 2005.**
[93] M. Topczewska, "Multiclass classification strategy based on dipoles," Zeszyty Naukowe Politechniki Białostockiej. Informatyka, pp. **79-90**, **2011.**
[94] S. A. Mulay, P. R. Devale, and G. V. Garje, "Intrusion detection system using support vector machine and decision tree," International Journal of Computer Applications, vol. 3, pp. **40-4**3, **2010.**
[95] I. A. Solomon, A. Jatain, and S. B. Bajaj, "Neural Network Based Intrusion Detection: State of the Art," Available at SSRN 3356505, **2019.**
[96] S. Ewen, S. Schelter, K. Tzoumas, D. Warneke, and V. Markl, "Iterative parallel data processing with stratosphere: an inside look," pp. **1053-1056.**
[97] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, "A survey of open source tools for machine learning with big data in the Hadoop ecosystem," Journal of Big Data, vol. **2**, pp. **24-24, 2015.**
[98] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," **2004.**
[99] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, et al., "Apache hadoop yarn: Yet another resource negotiator," pp. **1-16.**
[100] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. **51**, pp. **107-113, 2008.**
[101] A. K. Gupta and S. Gupta, "Security issues in big data with cloud computing," Int J Sci Res Comput Sci Eng, vol. **5**, pp. **27-32, 2017.**
[102] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," HotCloud, vol. **10,** pp. **95-95, 2010.**
[103] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, et al., "Fast and interactive analytics over Hadoop data with Spark," Usenix Login, vol. **37**, pp. **45-51, 2012.**
[104] N. Marz, "History of Apache Storm and lessons learned," Thoughts from the Red Planet, vol. **10**, **2014.**

Authors Profile

Reem Alshamy, presently working as a lecturer in the Faculty of Computer and Information Technology at Sana'a University. She held a B.S. in Information System from the Faculty of Science, Sana'a University, Republic of Yemen, and a preliminary master degree from Faculty of Computer and Information Technology, Sana'a University, Republic of Yemen. In addition, she concentrates on researches and conferences. She published a paper in a conference about energy efficiency in the data center using a green cloud simulator and it published as a small book in 2017. She has 9 years of teaching experience and more than a year of Research Experience.

Dr. Mossa Mosleh Ghurab, presently working as the Vice-dean Student Affairs, Faculty of Computer and Information Technology, Sana'a University. In addition, he is working as an Assistant Professor in Faculty of Computer and Information Technology at Sana'a University, Republic of Yemen. He held a Ph.D. in Computer Science and Technology from Zhejiang University, China. He completed his master's degree in Computer Application Technology from Central South University, China, and earned his B.S. in Computer Science and Technology from Central South University, China.