ITC 4/52
Information Technology and Control
Vol. 52 / No. 4 / 2023
pp. 898-914
DOI 10.5755/j01.itc.52.4.32042

A Scalable and Stacked Ensemble Approach to Improve Intrusion Detection in Clouds

Received 2022/08/09 | Accepted after revision 2023/10/18

HOW TO CITE: Ghazi, M. R., Raghava, N. S. (2023). A Scalable and Stacked Ensemble Approach to Improve Intrusion Detection in Clouds. *Information Technology and Control*, 52(4), 898-914. https://doi.org/10.5755/j01.itc.52.4.32042

A Scalable and Stacked Ensemble Approach to Improve Intrusion Detection in Clouds

Mohd. Rehan Ghazi, N. S. Raghava
Department of Electronics and Communication Engineering, Delhi Technological University, Main Bawana Road, Shahbad Daulatpur, Delhi -110042, India, e-mails: er.rehan.aras@gmail.com, nsraghava@gmail.com

Corresponding author: nsraghava@gmail.com

The availability of automated data collection techniques and the growth in the amount of data collected from cloud network traffic and cloud resource activities has transformed into a big data challenge, compelling the engagement of big data tools to handle, manage, and interpret it. A single classification method may fail to execute successfully for the amount of acquired data. Despite being more complex and consuming more computational resources, the research shows that stacking-based ensemble Machine Learning (ML) methodologies perform better in data classification approaches than single classifiers. This research proposes Intrusion Detection Systems (IDS), both based on the ensemble of ML algorithms built on the Stacked Generalization Approach (SGA) and big data technology. The suggested approaches are tested and assessed on NSL-KDD and UNSW-NB15 datasets, utilizing a Gain Ration (GR) based Feature Selection (FS) approach, J48, OneR, Support Vector Machine (SVM), Random Forest (RF), Multi- layer Perceptron (MLP) and Extreme Gradient Boosting (XGBoost) classifiers and Apache Spark, a prominent big data processing platform. The first technique involves storing data on HDFS, while the second involves selecting the most suitable subset of base classifiers for stacking. A thorough performance investigation reveals that our proposed model outperforms other current IDS models either in terms of accuracy or FPR or other performance metrics, in discovering intrusions for the Cloud.

KEYWORDS: Cloud Security, Big Data, Machine learning, Intrusion detection system, Apache Spark, Stacking.

1. Introduction

The use of the cloud is expanding as cloud computing advances and evolves from distributed computing, and numerous clients use the same cloud resources utilizing conceptual isolation techniques. Recent technological breakthroughs have resulted in an exponential rise of Cloud Computing, IoT, and WSNs applications in a variety of fields, including smart cities, smart transportation, e-health, e-commerce, and

smart grids [51]. As per NIST, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [33]."

Statistics show that security vulnerabilities in the virtual network layer of cloud computing have increased dramatically in recent years [42]. Various threats such as denial of service (DoS) [58], unauthorized access [58], zero-day attack [4], data leak [54], malware attack [32], social engineering or phishing [21], have increased as a result of the growing reliance on digitalization. According to AV-TEST [8], companies and institutions experienced 182.90 million malware attacks in 2013, which increased to 1312.64 million by 2021, and the figure is continuously rising. Organizations and people have suffered significant harm and financial losses as a result of these cyber-attacks. Cyber attackers will continue to attack healthcare providers and vaccine manufacturers, according to Juniper Research [27]. As a result, it is necessary to develop a strong cybersecurity mechanism capable of detecting various cyber-attacks quickly and ensuring the security of relevant systems [48]. According to Juniper Research, the amount of data breaches will triple in the next five years, and the yearly cost of these breaches would exceed $5 trillion USD globally by 2024 [27]. Overall, malware data has evolved into a big data challenge [23], demanding the use of big data technologies [22] capable of successfully handling a tremendous flow of malicious data.

Cloud security refers to a set of technologies and procedures that safeguard Cloud resources such as virtual machines (VMs), storage, networks, programmes, and data against attack, damage, or illegal access. However, as cyber threats get more complex, conventional strategies fail to adequately address the problem. In the context of computing, cloud security has seen massive technological and operational advances in recent years, with ML, a key component of "Artificial Intelligence," playing a significant role. To intercept cyber-attacks, several components (such as a firewall) and cryptographic procedures are installed, and an IDS is utilized to block the external attack and safeguard both business networks and Cloud resources [61]. IDS' objective is to highlight various types of anomalous activity in the cloud's network traffic and then use available protection mechanisms to prevent it.

Intrusion detection is a significant decision-making challenge that may be solved through the use of classification techniques [17]. Significant work has been performed on the use of ML techniques to improve the performance of intrusion security mechanisms [52, 53]. In the subject of intrusion detection, numerous ML methods such as fuzzy logic, neural networks, support vector machines, Naive Bayes, K nearest neighbor, and decision trees have been used [2]. However, due to the usage of single classifiers, these systems are restricted in their ability to identify and prevent serious threats. Individual algorithm performance can be improved when a combination or ensemble technique is used.

As shown in the research, stacking or stacked generalization is favorable since the approach is based on merging predictions from several individual classifiers, which may significantly increase generalization [57, 67, 68]. The benefit of stacking for protein categorization was stated in [12], and desirable performance was achieved.

Cloud Service Providers (CSPs) with sophisticated cloud resources prefer to collect massive amounts of data regularly to update the model. As a result, a stacked ensemble of ML algorithms orientated models can provide additional benefits in IDS due to their excellent detection performance. Based on its applicability, we, therefore, provide an ensemble of ML algorithms employing SGA for improving IDS performance in this study. We employ FS in addition to a stacked ensemble of ML algorithms to attain high accuracy by selecting highly sensitive features for model construction. We initially rank the feature in descending order according to the value computed by the GR attribute evaluation method and select the features using the threshold method.

Stacking or SGA is an approach for combining the forecasts of two or more models. Depending on the learning technique, stacking can decrease bias or variance error. It employs a set of homogeneous or heterogeneous base classifiers, whose outputs are utilized to train a meta-classifier that makes the final prediction. The meta-classifier improves generalization by correcting any errors generated by the base classifiers. Six classifiers are included in the proposed methodology. Each base classifier is assigned a rank

using a statistical approach. Following that, the rankings are employed to choose the best classifiers for stacking. The proposed methodology is built using Apache Spark. The following are the accomplishments of this paper:

1. Measure the significance of features i.e., FS in ML-oriented IDS to reduce feature dimensions.
2. Develop an ensemble using SGA to detect intrusion in Clouds, which considers FS based on a score for ranking and then builds an ensemble SGA model around those features.
3. Scaling the input feature, fine-tuning model hyper-parameter, and decreased model over-fitting cross-validation.
4. Apache Spark, which can easily manage large intrusion samples, is presented as a scalable approach. By adding extra nodes to an Apache Spark cluster, data processing scalability may be enhanced.
5. The procedure for determining the rankings of the base classifiers to be selected for stacking.

On a recent dataset, NSL-KDD [60] and UNSW-NB15 [35], experiments are carried out to evaluate the performance of suggested ensemble techniques and its comparison is carried out.

The rest of this article is written as such. The second section addresses relevant research. Section 3 delves into the proposed methodologies and procedures. Section 4 presents the experimental data as well as the environmental setup, and Section 5 concludes the study with recommendations for further research.

2. Related Work

An ensemble is a collection of ML algorithms that were used to identify intrusions using a dataset. Based on the combination methods utilized, several algorithms are integrated. According to multiple studies, ensemble approaches generate more accurate findings than a single model [34]. Using a combo of algorithms and FS methodologies, researchers have developed many methods for detecting intrusions. The IDS research that employed hybridization and ensembles ML approaches is highlighted in this section.

Panigrahi and Patra [39] introduced a hybrid IDS model based on fuzzy and rough set theory to recognize normal and abnormal activity in network data. There are two steps to their model. It first used rank and search based FS techniques to find the most significant features, and then used the five classifiers to classify the NSL-KDD dataset. Performance study shows that out of all five classifiers, the "fuzzy rough nearest neighbor" classification approach produces better results.

The hybrid model proposed in [20] appeared to be more accurate than separate models. The dataset used is from "The University of New Mexico" and comprised both normal and anomalous mail program records. A combination of SVM and K-nearest neighbor (KNN) was used to create an IDS in [1]. The weights obtained by particle swarm optimization (PSO) and the model outperformed the best base expert classifier accuracy by 0.756 percent. Because of its performance degradation with increasing data size, the SVM is not a good option for analysis of large amounts of data. In order to increase the detection accuracy, hybrid IDSs were discussed. However, they have the disadvantage of requiring a lot of processing resources.

Mukkamala et al. [36] advocated an intelligent IDS based on ML and ensemble classifiers, which detected intrusions with a 91 percent success rate. Their findings showed that when a two-level decision was made, ensemble approaches performed well in classification. As a result, numerous other researchers have developed intrusion detection systems based on ensemble methods. The contribution of this method to the detection of zero-day attacks is minimal and to create permanent signatures with all conceivable changes and non-intrusive actions to reduce FPR. Panigrahi and Patra [40] published Bayes net classifier-based IDS, where they employed search (K2, Tabu, and Hill Climbing) and Tree Augmented Naive Bayesian (TAN) techniques. They also used FS methods based on entropy and statistics for the NSL-KDD dataset.

Shrivas and Dewangan [56] suggested employing ensemble learning to present an efficient FS and classification for an intrusion detection system, in which an ensemble classifier with FS and boosting algorithm was utilized to provide the best intrusion detection rates using the kdd99 and NSL-KDD datasets. Rajagopal et al. [45] suggested a stacking ensemble-based IDS in which the base classifiers were Linear Regres-

sion (LR), RF, and KNN, and the meta-classifier was SVM. To find key features, they employed an entropy-based feature selection strategy. UNSW-NB15 and UGR'16 heterogeneous datasets were used in the experiments. When the dataset is in IoT systems, the SVM classifier algorithm cannot process it effectively since the systems have a high rate of traffic, which makes SVM inappropriate.

Wankhade and Chandrasekaran [62] proposed a novel hybrid classifier-based IDS that uses the Self Organized Ant Colony Optimization (ACO) approach and SVM. Salo et al. [50] proposed a hybrid IDS in which they coupled two feature selection approaches (information gain and principle component analysis (PCA) with an ensemble method that included SVM, KNN, and ANN. Tama and Rhee [59] proposed a novel Intrusion Detection System based on the Gradient Boosted Method (GBM), which they compared to four other classifiers: random forest, deep neural network, support vector machine, and regression tree. The results suggest that gradient-boost performs better than other approaches.

A novel intrusion detection model was suggested by Ahmad [3] to calculate common patterns and choose features, they used an ensemble feature selection method based on PSO and correlation-based FS methods. They built ensemble classifiers using FS. The proposed method showed some promise in terms of the number of features chosen and the false alarm rate, but it still needs improvement in terms of accuracy and detection rate. Additionally, when proposing this strategy, the execution time necessary to identify every attack is overlooked. Prusti [43] suggested an effective intrusion detection system based on ensemble learning. They employed three distinct ensemble approaches to enhance accuracy and lower the false-positive rate, including bagging, boosting, and stacking. Rashid et al. [47] used a feature selection approach to examine alternative machine learning algorithms for detecting cyber-attacks from IoT-based smart city applications. "UNSW-NB15 and CIC-IDS2017", two recent incursion datasets were used in the study. The results reveal that the stacking ensemble outperforms other classifiers, with UNSW-NB15 and CICIDS2017 datasets.

A new FS technique for IDS based on a pigeon-inspired optimizer (PIO) is depicted in [5]. Using the described PIO FS approach, the number of features in the utilized datasets was minimized which attains high accuracy. Rashid et al. [46] deal with a tree-based stacking model with a selectKbest FS strategy to identify intrusion and developed a stacking ensemble method on UNSW-NB15 datasets. Performance assessment demonstrates that the proposed model exceeds previous recent efforts in terms of accuracy and false alarm rate. SelectKBest approach utilisation, however, is less adaptable to new malicious network traffic over time.

Zhou et al. [69] offer a novel intrusion detection system based on FS and ensemble learning approaches to deal with high-dimensional and imbalanced network data. At initial point, CFS-BA approach employed for FS based on feature correlation. The classification model is then generated using an ensemble classifier using C4.5, RF, and ForestPA. The experimental findings for the NSL-KDD dataset are promising.

The following are some of the findings of the studies reviewed: (1) A single machine learning model or parameter optimization does not generate the best results, thus combining numerous learning models to build a powerful model is a superior strategy, and (2) For IDS, researchers have yet to study big data technologies and methods combined with ensemble methods. We employed an FS methodology and employed a ranking approach to rank the base classifiers for the optimal set of classifiers to be used for stacking in innovative ensemble learning methods to improve the efficiency of IDS, in this research. At the lower level, the proposed solution uses Apache Spark for fast dataset processing and produces multiple different base models. The findings of the experiments at this stage are used to produce rankings for the base models, which are then used to pick a set of appropriate base classifiers for stacking.

3. Implementation Strategy

This section delves into the methods for detecting intrusions, in depth. Apache Spark is used to evaluate the dataset, which is stored in a distributed storage system. Following the FS technique, we built models using six different classifiers as base classifiers. We tested them on true positives, true negative rate, accuracy, precision, and F-measure. The results are utilized to create techniques for computing the ranks for base classifiers

and selecting the best pool of classifiers for stacking. These stages are outlined in the sections below.

3.1. Datasets for IDS

Datasets comprise many entries, including various properties and information. These features and information may be utilized to create cloud security oriented IDS [55]. NSL-KDD and UNSW-NB15, two publicly available intrusion datasets were used in this work with two detection categories: normal and abnormal.

The KDDcup99 dataset was used to produce the NSL-KDD dataset, which is a prominent intrusion dataset. NSL-KDD's "KDDTrain+" and "KDDTest+" are used as a training set and testing set, respectively. The testing set has 22544 records from various attack and normal categories, whereas the training set contains 125973 records, as shown in Table 1. The UNSW-NB15 intrusion dataset comprises contemporary attacks and is frequently utilized. The "Australian Centre for Cyber Security" (ACCS) created this dataset in 2015, using the IXIA PerfectStrom programme to produce raw network packets. There are about 2.5 million records in this collection. UNSW-NB15 training-set and UNSW-NB15 testing-set, two partitions from overall dataset, are set up as a training set and testing set, respectively. The testing set has 82,332 records from various attack and normal categories, whereas the training set contains 175,341 records as shown in Table 1.

Table 1
NSLKDD and UNSW-NB15 datasets distribution

	NSL-KDD		UNSW-NB15	
	KDD Train+	KDD Test+	Training Set	Testing Set
Attack	58630	12833	119,321	45332
Normal	67343	9711	56,000	37000
Total	125973	22544	175341	8233

3.2. Dataset Pre-processing

Feature normalization and encoding depending on the intrusion dataset's features are part of data pre-processing.

A Feature normalization: The range of features is normalized by feature scaling, which guarantees that distinct features have different values. Furthermore, training high-dimensional datasets require high computational power. Data is frequently scaled using methods like Z-score standardization, decimal scaling, Max normalization, and Min–Max scaling to address these difficulties [30]. The approach to utilize is frequently determined by the application. We have incorporated Min–Max scaling (Equation (1)).

$$\text{MinMax scaling X}: \; X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{1}$$

where X_{min} and X_{max} are the minimum and maximum values of feature X, respectively. The standardization calculation occurs as specified in Algorithm 1 for a dataset with an input vector (feature space) represented by $\boldsymbol{U\,(x_1,\ldots\ldots,x_n)}$, 1<n<N, where N is the total number of instances (features) in the space.

Algorithm 1 for Min-Max scaling

Input: $U\,(x_1,\ldots\ldots,x_n)$, where 1<n<N

Output: $U_{normalised}(x_1^{norm},\ldots,x_n^{norm})$

for i from 1 to k **do**

if (x_i a non-numeric input value) **then**

Step 1: encode using "label encoding"

Step 2: Compute Min-Max scaling:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

end if

Step 1: Compute Min-Max scaling: $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$

end for

B Feature Encoding: For efficient model training, all categorical features will be encoded into vectors. There are several methods for converting categorical data into vectors. 'Label Encoding,' 'One Hot Encoding,' and 'scikit-learn feature mapping' are the most often utilized approaches. We adopted the first approach since the number of feature dimensions in the later techniques significantly rises [41]. It took a straightforward approach to convert feature values to numeric numbers, for example, the values of instances like "icmp, http, tcp" in the dataset, will turn into vectors 0,1,2.

3.3. Feature Selection (FS)

Gain Ratio (GR) [28] is a variant of Information Gain (IG) that tackles IG's bias toward qualities with a larger range of values. When selecting a gain ratio attribute, the number and size of branches are considered. By accounting for the inherent information of a split, the IG is fixed (i.e., how much info is needed to identify which branch a given instance belongs to), and the entropy of instance distribution into branches is the intrinsic information. It is calculated as follows for a given attribute $\boldsymbol{a}$ and an attribute value of $\boldsymbol{b}$:

$$Gain\ Ratio(b,a) = \frac{gain(b,a)}{intrinsic\ info(a)} \quad (2)$$

where,

$$intrinsic\ Info(a) = -\sum \frac{|S_i|}{|S|} * log_2 \frac{|S_i|}{|S|}. \quad (3)$$

The number of possible values for attribute $\boldsymbol{a}$ is |S|, and the number of actual values for attribute $\boldsymbol{a}$ is $|S_i|$. Based on a user-defined threshold, a subset of features is picked, in this instance, **GR > 0.1**, with the assumption that higher-valued features may contain more important information.

3.4. IDS using Base Classifiers

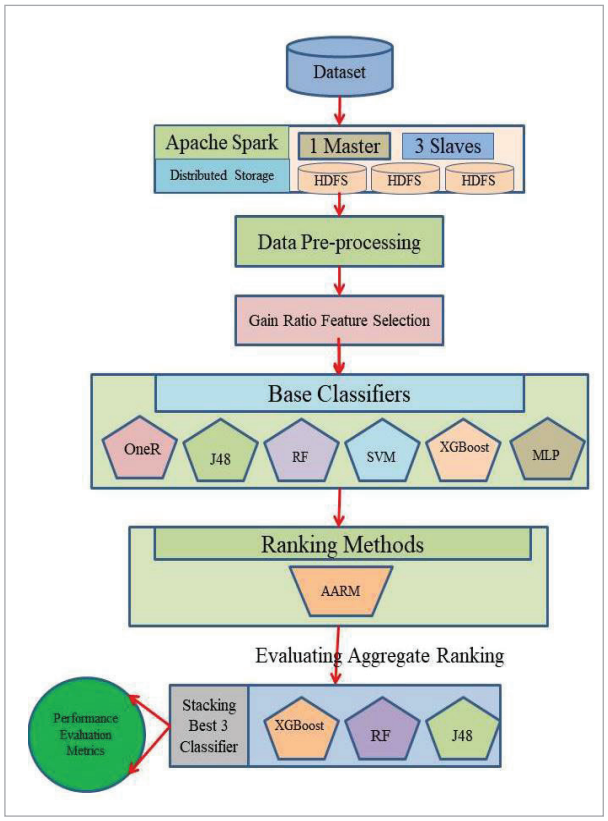
Ensemble approaches are a type of machine learning methodology in which numerous base classifiers are combined to generate a single best prediction model [24, 63]. The final model will overcome each learner's flaws, yielding a strong model that will improve prediction results. Ensemble learning is more successful when the classifiers are heterogeneous, which may be done by combining multiple types of classifiers that employ different methodologies to classify the incoming data. As a result, six heterogeneous classifiers, namely J48, RF, XGBoost, OneR, MLP, and SVM, were employed as base classifiers.

The J48 tree classification algorithm is the most widely utilized. Quinlan [44] was the one who came up with it. To reduce classification errors, the J48 approach uses an improved tree pruning methodology. Several studies investigated the influence of applying the J48 algorithm to enhance IDS accuracy [7, 31]. L.Breiman proposed an RF classifier [10]. It's an ensemble learning classifier based on trees. It was created by combining the forecasts of numerous trees that had each been trained independently. RF classifier has a substantial favorable impact on IDS accuracy, according to several prior research [15, 38]. RF is a highly successful classification strategy across a range of real-world applications,' Fernandez-Delgado et al. write in [16]. Other studies have found that random forest produces good results [6, 9, 14]. Existing research supports all of the proposed classifiers, particularly because their findings are simply interpretable and their training is robust against outliers. All of the classifiers employed are gaining popularity in a variety of fields, including cyber security and cloud security, due to their high performance. The XGBoost idea is based on gradient-boosted trees with supervised learning as the primary approach and was pioneered by Chen and Guestrin [11]. It is Sparse Aware, meaning it can handle missing values, enables parallel tree construction, and has the unique ability to execute boosting on data that has already been added to the trained model [49].

OneR [26] is a rule-based model-based algorithm. It creates a one-level decision tree in the form of a series of rules, each of which tests a single feature. OneR is a basic, low-cost approach that frequently produces useful rules for describing data structure. MLP refers to feed-forward neural networks trained using the back-propagation technique [25]. Because they are supervised networks, they must be taught the desired response. They can approximate any input-output map with one or two hidden layers. The binary classifier category includes SVM. It's a popular method for categorizing two groups. The "Radial Basis Function" (RBF) kernel is employed in this experiment. This kernel function is a suitable choice because it has fewer adjustable parameters and performs well in nonlinear forecasting.

There are two reasons for utilizing stacking ensemble: first, it can improve performance by lowering the variance component of prediction errors, and second, it can improve robustness by lowering prediction dispersion [13]. Furthermore, with increasingly powerful computing resources available at reduced hardware costs, improved model performance surpasses the computational cost of model construction. J48, RF, XGBoost, OneR, MLP, and SVM are six classifiers. Figure 1 shows the proposed framework.

Figure 1
The proposed ensemble method's overall framework



We used the min-max feature scaling strategy to minimize model complexity by scaling all input features into a range between 0 and 1 [65]. We employed cross-validation as a precautionary measure against overfitting by dividing the original training dataset into several small train-test parts. These splits were then used to fine-tune the model. The data was partitioned into 10 subgroups called folds using 10-fold cross-validation. The algorithm was then trained on 9 folds in a row, with the last fold acting as a test set. Finally, the algorithm's parameters were fine-tuned utilizing the scikit-learn RandomizedSearchCV() class, which uses the randomized search approach for parameter optimization.

3.5. Ranking Classifiers

The proposed methods for constructing an ensemble classifier are based on rank algorithms [64] for picking the best pool of classifiers for stacking. According to the literature, most classifiers produce different responses for distinct classes. They do not, for example, classify intrusion and benign classes equally accurately. This information was utilized to calculate the scores of the base classifiers that are used to create the ranking. The Average Accuracy based Ranking Method (AARM) is employed for ranking the classifiers. The ranking is obtained using this technique by considering the average of class prediction accuracies. A higher rank will be given to the base classifier that has greater average accuracy. Assuming the usage of s classifiers, represented by $x_1,x_2,.....x_s$ to divide the data into two categories: intrusion, u, and benign, v. Let A_{u,x_i} and A_{v,x_i} represent the malware and benign class accuracies for a classifier xi, respectively. Equation (4), as shown below, is used to calculate the average accuracy A_{avgx_i} of each classifier:

$$A_{avgx_i} = \frac{p_u * A_{u,x_i} + p_v * A_{v,x_i}}{p_u + p_v} \forall i \in \{1,2,3,\ldots\ldots,s\}, \quad (4)$$

where, P_u and P_v represent the numbers of intrusion and benign files, respectively. Assume $C = \{A_{avgx_1}, A_{avgx_2}, A_{avgx_3}, \ldots\ldots, A_{avgx_s}\}$ is the collection of all classifiers' average accuracies. The rank is then computed using the $R_{desc}()$ function, which assigns a rank to each classifier based on its average accuracy as shown in Equation (5) (i.e. larger the value of A_{avgx_i}, higher is the rank).

$$RANK(x_i, A_{avgx_i}) = R_{desc}(C) \forall i \in \{1,2,3..,s\}. \quad (5)$$

Instead of using all of the base classifiers, the top three are chosen as the best set, using the rank approach, for stacking.

4. Experiment

The experimental setup, evaluation settings, and results of the proposed solution are described in this section. Table 2 shows the hyper-parameter values of the classifiers selected by randomised search.

4.1. Environmental Setup

The proposed method uses Hadoop Distributed File System (HDFS), Apache Spark, and Python programming language. The tests are run on a multi-node cluster based on Apache Spark, with one master node and

Table 2
Classifier hyper-parameters

Classifier	Parameters
J48	"Confidence factor" = 0.7, "num_folds" = 5,"minimum_number_of_instance_per_leaf"=2
RF	"max_depth"= 90, "min_samples_leaf"= 6,"max_features"= 4 "n_estimators"= 200, "min_samples_split"= 7
SVM	C=10, cache_size=200, coef0=0.0, kernel="rbf", degree=3, max_iter=-1, gamma=0.0001, tol = 001, verbose=False.
XGBoost	learning rate=0.2, n_estimators=150, max depth=3, random state=1
OneR	minBucketSize = 8
MLP	LearningRate = 0.3, Momentum= 0.2, numberOfEpochs = 300

three slave nodes. 2.40 GHz Intel Xeon CPU, 32 GB RAM, and 512 GB SSD power at the master node. Intel Core i5 CPU of 3.1 GHz, 8 GB RAM, and 500 GB HDD are used in the slave nodes. Ubuntu 14.04.6 is the operating system in a multi-node cluster. The following is a quick explanation of the software system components:

Apache Spark: A resilient distributed dataset (RDD) [66] of Spark, is a collection of fault-tolerant units that can execute in parallel. Spark is known for its ability to process large datasets in memory which is a set of fault-tolerant units that may run in parallel. Spark is well recognized for its ability to handle massive datasets in memory.

HDFS: HDFS is a widely used solution for large-scale distributed data storage [19]. High fault tolerance, high throughput, and easy portability across diverse systems are just a few of the advantages of HDFS [18].

Python: Python is favored as a programming language because of the availability of a large ecosystem of scientific libraries, PySpark, a Python API for Apache Spark, and Jupyter Notebook, which offers the programming environment, are the additional development tools. The Pandas library, Numpy, Spark MLlib, and Sklearn were used to implement the models that were tested.

4.2. Performance Assessment Matrices

How well a classification system performs, is shown by the confusion matrix. Table 3 below represents the confusion matrix.

Table 3
Confusion Matrix

		Actual	
		Benign	**Malware**
Predicted	**Benign**	TP	FP
	Malware	FN	TN

The below mentioned metrics in Table 4 are widely used to assess models. The following are the performance metrics: True Positive (TP) represents obfuscated malware specimen that is accurately detected as malware. True Negative (TN) represents a normal specimen that is accurately classified as normal. False Positive (FP) means a normal specimen is misidentified as malware. False Negative (FN) represents obfuscated malware specimen that is wrongly labelled as normal.

Table 4
Metrics for performance evaluation

Metrics	Formula
Sensitivity	TP/(TP+FN)
Specificity	TN/(FP+TN)
Precision	TP/(TP+FP)
FPR	FP/(FP+TN)
FNR	FN/(FN+TP)
Accuracy	(TP+TN)/(FP +TP+FN+TN)
F1-Score	2TP/(2TP+FP+FN)

4.3. Impact of FS Approach

The dataset features do not all contribute equally to its effective design. In this experiment, we used the Gain Ratio approach to identify the best features, with a threshold value of 0.1 for selecting the top rank features, delivering us the most important features based on their assessed value. Tables 5 and 7 provide the selected features of the NSL-KDD and UNSW-NB15 datasets. The range of feature values is set between 0 to1. A feature with a higher score has a more significant impact on differentiating between normal and abnormal activity, so the value of features greater than or equal to 0.1, is selected and rejected the remaining features as shown in Tables 6 and 8.

Table 5
Selected Features from NSL-KDD Dataset

Feature Name	Feature No.	Value ≥ 0.1
"logged_in"	12	0.418
"srv_serror_rate"	26	0.3739
"flag"	4	0.3399
"serror_rate"	25	0.3327
"dst_host_srv_serror_rate"	39	0.332
"diff_srv_rate"	30	0.2673
"dst_host_serror_rate"	38	0.2648
"dst_bytes"	6	0.2585
"src_bytes"	5	0.2313
"same_srv_rate"	29	0.2243
"service"	3	0.1728
"dst_host_srv_diff_host_rate"	37	0.1452
"wrong_fragment"	8	0.134
"dst_host_srv_count"	33	0.1327
"dst_host_same_srv_rate"	34	0.1283
"dst_host_diff_srv_rate"	35	0.1246
"srv_diff_host_rate"	31	0.1143

Table 6
Rejected Features from NSL-KDD Dataset

Feature Name	Feature No.	Value < 0.1
"dst_host_srv_rerror_rate"	41	0.097
"count"	23	0.0953
"dst_host_count"	32	0.0899
"srv_rerror_rate"	28	0.0821
"num_root"	16	0.0798
"rerror_rate"	27	0.0786
"dst_host_same_src_port_rate"	36	0.0775
"num_access_files"	19	0.0743
"protocol_type"	2	0.0725
"num_compromised"	13	0.072
"su_attempted"	15	0.0671
"hot"	10	0.058
"duration"	1	0.0554
"is_host_login"	21	0.0491
"dst_host_rerror_rate"	40	0.0449
"num_file_creations"	17	0.0394
"num_failed_logins"	11	0.0297
"srv_count"	24	0.0252
"root_shell"	14	0.022
"num_shells"	18	0.0218
"is_guest_login"	22	0.0152
"land"	7	0.014
"num_outbound_cmds"	20	0
"urgent"	9	0

Table 7
Selected Features from UNSW-NB15 Dataset

Feature Name	Feature No.	Value ≥ 0.1
"sttl"	11	0.40367
"dttl"	12	0.32594
"ct_state_ttl"	33	0.30407
"is_sm_ips_ports"	43	0.22527
"xState"	5	0.17808
"ackdat"	27	0.16677
"tcprtt"	25	0.16217
"synack"	26	0.15394
"id"	1	0.14481
"dinpkt"	18	0.13868
"dload"	14	0.13382
"dbytes"	9	0.13126
"dpkts"	7	0.12333
"rate"	10	0.11923
"sbytes"	8	0.11524
"dmean"	29	0.11521
"dur"	2	0.11234
"ct_dst_sport_ltm"	36	0.10629

Table 8
Rejected Features from UNSW-NB15 Dataset

Feature Name	Feature No.	Value < 0.1
"response_body_len"	31	0.0951
"smean"	28	0.09274
"djit"	20	0.08907
"sjit"	19	0.08702
"sload"	13	0.08522
"sinpkt"	17	0.08504
"swin"	21	0.08203
"spkts"	6	0.08037
"dloss"	16	0.07891
"stcpb"	22	0.07517
"dtcpb"	23	0.07512
"dwin"	24	0.0751
"xProt"	3	0.07344
"sloss"	15	0.06997
"ct_src_dport_ltm"	35	0.05152
"ct_dst_ltm"	34	0.04926
"ct_srv_dst"	42	0.03889
"ct_dst_src_ltm"	37	0.03766
"ct_src_ltm"	41	0.03742
"ct_srv_src"	32	0.0348
"xServ"	4	0.03074
"trans_depth"	30	0.00173
"is_ftp_login"	38	0.00165
"ct_ftp_cmd"	39	0.00165
"ct_flw_http_mthd"	40	0.00119

We investigate only 17 of the 41 features in the NSL-KDD dataset based on their value, and only 18 of the 43 features in the UNSW-NB15 dataset based on their ranking. By reducing the number of features, the feature significance score may assist us in designing a simpler model. The model's detection expertise is also improved by removing redundant features.

4.4. Model Performance

Tables 9-10 illustrate the detection results. Table 9 indicates that the sensitivity is 0.9967, specificity is 0.9983, precision is 0.9977, FPR is 0.0017, FNR is 0.0033, Accuracy is 0.9976 and F1 score is 0.9972, XGBoost performance metrics for the NSL-KDD dataset. Table 10 indicates that the sensitivity is 0.9665, specificity is 0.9416, precision is 0.9722, FPR is 0.0584, FNR is 0.0335, Accuracy is 0.9585 and F1 score is 0.9693, XGBoost performance metric for the UNSW-NB15 dataset.

The top three classifiers, XGBoost, RF, and J48, are picked for stacking based on their rankings. XGBoost has been used as a level-2 meta-classifier in stacking. The final model will overcome each learner's flaws, yielding a strong model that will improve prediction results. The SGA is a general architecture made up of two types of classifiers: base and meta-classifiers. The training dataset is used to train the base (initial) classifiers, while a new dataset is created for the meta-classifier. This new dataset is then used to train the meta-classifier. Finally, the test dataset is predicted using the trained meta-classifier. We provide a model based on a stacked ensemble of ML algorithms, with XGBoost serving as a meta-classifier.

Table 9
Results of the base classifiers for the NSL-KDD dataset

Classifier	Sensitivity	Specificity	Precision	FPR	FNR	Accuracy	F1-Score	Rank
J48	0.9906	0.9950	0.9934	0.0050	0.0094	0.9931	0.9920	**3**
RF	0.9916	0.9947	0.9930	0.0053	0.0084	0.9933	0.9923	**2**
SVM	0.9891	0.9867	0.9823	0.0133	0.0109	0.9871	0.9857	5
OneR	0.9796	0.9790	0.9721	0.0210	0.0204	0.9792	0.9758	6
MLP	0.9905	0.9881	0.9842	0.0119	0.0095	0.9891	0.9873	4
XGBoost	0.9967	0.9983	0.9977	0.0017	0.0033	0.9976	0.9972	**1**

Table 10
Results of the base classifiers for the UNSW-NB15 dataset

Classifier	Sensitivity	Specificity	Precision	FPR	FNR	Accuracy	F1-Score	Rank
J48	0.9516	0.9171	0.9369	0.3829	0.0084	0.9459	0.9604	**3**
RF	0.9581	0.9238	0.9638	0.0762	0.0419	0.9471	0.9609	**2**
SVM	0.9443	0.8947	0.9499	0.1053	0.0557	0.9284	0.9471	5
OneR	0.9444	0.8921	0.9486	0.1079	0.0556	0.9275	0.9465	6
MLP	0.9498	0.9063	0.9554	0.0937	0.0502	0.9358	0.9526	4
XGBoost	0.9665	0.9416	0.9722	0.0584	0.0335	0.9585	0.9693	**1**

Table 11
Results of stacking for the NSL-KDD dataset

Classifier	Sensitivity	Specificity	Precision	FPR	FNR	Accuracy	F1-Score
Stacking (All Classifiers)	0.9971	0.9985	0.9980	0.0015	0.0029	0.9979	0.9976
Stacking (XGBoost, RF, J48)	0.9974	0.9988	0.9985	0.0012	0.0015	0.9982	0.9979

Table 12
Results of stacking for the UNSW-NB15 dataset

Classifier	Sensitivity	Specificity	Precision	FPR	FNR	Accuracy	F1-Score
Stacking (All Classifiers)	0.9668	0.9425	0.9726	0.0575	0.0332	0.9590	0.9697
Stacking (XGBoost, RF, J48)	0.9672	0.9433	0.9730	0.0567	0.0328	0.9596	0.9701

Figure 2
Performance Comparison for NSL-KDD

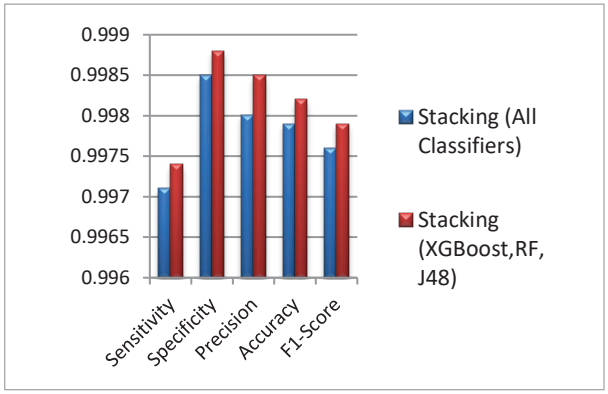


Figure 3
FPR and FNR Comparison for NSL-KDD

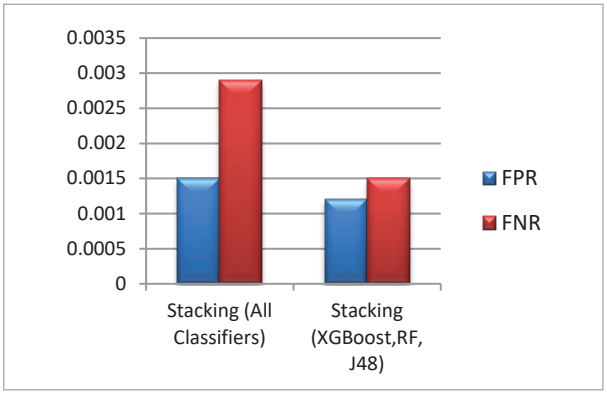


Figure 4
Performance Comparison for UNSW-NB15

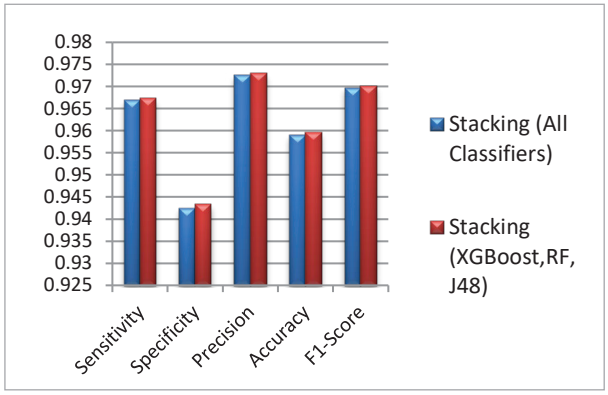
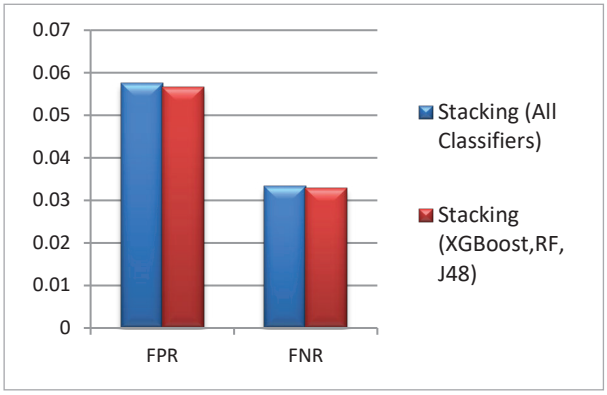


Figure 5
FPR and FNR Comparison for UNSW-NB15



Tables 11-12 and Figures 2-5 depict that compared to stacking all of the specified classifiers, stacking with the top three base classifiers gives better performance. As a result, instead of considering all base classifiers, it is preferable to select the best collection of classifiers because it minimizes computing time. As a result, the presented ensemble technique schemes may be employed to increase generalization performance in detecting unknown intrusions.

4.5. Performance Comparison with Current Methodologies

Using the NSLKDD and UNSW-NB15 datasets, Table 13 compares the performance of our stacking model to current attempts in intrusion detection for cloud security. In terms of accuracy and FPR, our SGA oriented ensemble model exceeds several current approaches. We present several explanations

Table 13
Performance comparison with current methodologies

Author	Year	Methodologies	Dataset	FS Approach	FS	Acc.	FPR
Panigrah and Patra [39]	2016	"Fuzzy-Rough Ownership NN Classification"	NSL-KDD	Greedy Step-wise	11	99.6145	0.309
Panigrahi and Patra [40]	2019	JRip	NSL-KDD	Genetic Search	16	99.82	0.1396
Salo et al. [50]	2019	"Ensemble classifier based on SVM, Instance-based learning algorithms (IBK), and multilayer perceptron (MLP)"	NSL-KDD	Information Gain (IG) + Principle Component Analysis (PCA)	7	99.011	0.01
Tama and Rhee[59]	2017	Gradient boosting machine (GBM)	NSL-KDD	NO	NA	91.82	4.19
Rashid et al. [46]	2022	Stacking-DT, RF, XGBoost	NSL-KDD	SelectKbest	20	99.90	0.0009
Proposed		**Stacking- XGBoost, RF & J48**	**NSL-KDD**	**GR**	**17**	**99.82**	**0.0012**
Rajagopal et al. [45]	2020	Stacking (RF, LR, kNN, SVM)	UNSW-NB15	Information Gain and Hashing	11	94	5.2
Alazzam et al. [5]	2020	Decision Tree (DT)	UNSW-NB15	Sigmoid -pigeon inspired optimizer (PIO)	14	91.3	0.052
				Cosine-PIO	5	91.7	0.034
			NSL-KDD	Sigmoid - PIO	18	86.9	0.064
				Cosine-PIO	5	88.3	0.088
Rashid et al. [46]	2022	Stacking-DT, RF, XGBoost	UNSW-NB15	SelectKbest	20	94.00	0.06
Proposed		**Stacking- XGBoost, RF & J48**	**UNSW-NB15**	**GR**	**18**	**95.96**	**0.0567**

for why our suggested model outperforms current models. Our model first selects relevant features before building a model. This phase, according to some, reduces variance and over-fitting. Moreover, the proposed stacking ensemble method employs heterogeneous classifiers, which overcomes the disadvantages of homogeneous classifiers.

As a result, the proposed method is more effective in detecting a previously unknown input. To minimize overfitting, we used hyper-parameter adjustment. Table 14 shows the model construction time for the given dataset. J48 takes the least amount of time among the classifiers, whereas stacking takes the most time to create models for both datasets.

The stacking method needs extra processing time since it combines several base classifiers, each taking time to build. Table 14 shows how much time it takes for classifiers. We discovered that J48 and RF are the fastest classifiers in this environment. Even if the complexity of the stacking model has risen, and as a result, the time requirements have increased, the fact that it beats conventional IDS, as noted in the previous result, is a significant consideration. The cost of missing an intrusion in such a system can be quite expensive. As an outcome, the cost of a little extra time, which is still secs for the datasets examined and therefore potentially well scalable in comparison to earlier approaches, is justified. As a result, the sug-

Table 14
Model building and testing time

Methodologies	NSL-KDD		UNSW-NB15	
	Building Time (s)	Testing Time (s)	Building Time (s)	Testing Time (s)
J48	**0.567**	**0.233**	**1.00**	**0.541**
RF	0.684	0.342	1.51	0.583
SVM	1.42	0.81	2.31	1.54
XGBoost	1.52	0.97	2.56	1.79
MLP	2.58	1.25	3.54	1.02
OneR	1.52	0.85	3.98	0.58
Stacking (All)	9.24	5.85	14.52	8.52
Stacking (Top 3)	7.10	4.05	10.54	6.58

gested model has significant practical usefulness. As a result, rather than examining all base classifiers, it is better to choose the optimal set of classifiers because it minimizes computing time.

Indeed, intrusion detection has been a topic of discussion, with the task being named one of the top data mining applications in enterprises [29]. In industries such as CSPs and the banking sector, Education, etc., the ability to identify unexpected network behavior quickly is critical to the long-term viability of services. Attacks that go unnoticed in these areas can be expensive, and identifying them manually might be challenging [37]. The focus of such systems, which frequently employ large computing resources for automatically identifying intrusions, is on accurate intrusion detection. In comparison to existing algorithms, the suggested stacking strategy shows a lot of potential in this regard, as the findings reveal.

5. Conclusion

Based on big data technologies and ensemble learning, this research provided a scalable system for intrusion detection. The ensemble classifier is built using an approach based on determining rankings of the base classifiers. As a result, it can be inferred that the offered strategies can improve generalization performance when identifying new intrusions. The cloud environment's effective intrusion detection system is created and built using feature selection and SGA-oriented ensemble method classification. We used a feature selection technique based on GR to reduce the dimensionality of the network data and discover the most important features. Then, using J48, RF, OneR, MLP, XGBoost, and SVM, we introduced an SGA ensemble approach. By merging multiple classifiers, ensemble classifiers build a robust classifier capable of recognizing network intrusions and improving forecast accuracy. In which we used a majority SGA ensemble approach to distinguish between attack data and legitimate data in network traffic using the top three classifiers i.e., XGBoost, RF and J48. The findings of the experiments suggest that stacking with the top three base classifiers yields greater accuracy than stacking with all base classifiers, which might assist save computation time.

To assess the efficacy, we utilize the widely available NSLKDD and UNSW-NB15 datasets. The proposed model beats previous recent research in terms of accuracy, having attained 99.82 percent and 95.96 percent accuracy for the NSL-KDD and UNSW-NB15 datasets, respectively. Aside from that, the proposed method might be supplied as a cloud-based intrusion detection service. To handle data on local and remote clusters, a hybrid technique may be employed. We'll investigate several hybrid approaches in the future to surge prediction accuracy and identify different kinds of intrusions.

Acknowledgement

First Author would like to thank DST for INSPIRE Fellowship.

References

1. Aburomman, A. A., Reaz, M. B. I. A Novel SVM-Knn-PSO Ensemble Method for Intrusion Detection System. Applied Soft Computing, 2016, 38, 360-372. https://doi.org/10.1016/j.asoc.2015.10.011
2. Aburomman, A. A., Reaz, M. B. I. Review of IDS Development Methods in Machine Learning. International Journal of Electrical and Computer Engineering (IJECE), 2016, 6(5), 2432-2436. https://doi.org/10.11591/ijece.v6i5.pp2432-2436
3. Ahmad, I. Feature Selection Using Particle Swarm Optimization in Intrusion Detection. International Journal of Distributed Sensor Networks, 2015, 11(10). https://doi.org/10.1155/2015/806954
4. Alazab, M., Venkatraman, S., Watters, P., Alazab, M. Zero-Day Malware Detection Based on Supervised Learning Algorithms of API Call Signatures. Proceedings of 9th Australasian Data Mining Conference (AusDM'11), Ballarat, Australia. Conferences in Research and Practice in Information Technology (CRPIT), 2011, 121, 171-181.
5. Alazzam, H., Ahmad, S., EddinSabri, K. A Feature Selection Algorithm for Intrusion Detection System Based on Pigeon Inspired Optimizer. Expert Systems with Appications, 2020, 148. https://doi.org/10.1016/j.eswa.2020.113249
6. Ali, J., Khan, R., Ahmad, N., Maqsood, I. Random Forests and Decision Trees. International Journal of Computer Science Issues, September, 2012, 9 (5), 272-278.
7. Aljawarneh, S., Yassein, M. B., Aljundi, M. An Enhanced J48 Classification Algorithm tor the Anomaly Intrusion Detection Systems. Cluster Computing, 2019, 22, 10549-10565. https://doi.org/10.1007/s10586-017-1109-8
8. AV-Test Institute. Malware, https://www.av-test.org/en/statistics/malware/. Accessed on Januray 14 2022.
9. Berhane, T. M., Lane, C. R., Wu, Q., Autrey, B. C., Anenkhonov, O. A., Chepinoga, V. V., Liu, H. Decision-Tree, Rule-Based, And Random Forest Classification of High-Resolution Multispectral Imagery for Wetland Mapping and Inventory. Remote Sensing, 2018, 10(4), 580. https://doi.org/10.3390/rs10040580
10. Breiman, L. Random Forests. Machine Learning, October 2001, 45, 5-32. https://doi.org/10.1023/A:1010933404324
11. Chen, T., Guestrin, C. Xgboost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, August, 2016, 785-794. https://doi.org/10.1145/2939672.2939785
12. Diplaris, S., Tsoumakas, G., Mitkas, P. A., Vlahavas, I. Protein Classification with Multiple Algorithms. Panhellenic Conference on Informatics, Advances in Informatics, Lecture Notes in Computer Science, Berlin, Heidelberg. 2005, 3746, 448-456. https://doi.org/10.1007/11573036_42
13. Džeroski, S., Ženko, B. Is Combining Classifiers with Stacking Better than Selecting the Best One? Machine Learning, 2004, 54(3), 255-273. https://doi.org/10.1023/B:MACH.0000015881.36452.6e
14. Esmaily, H., Tayefi, M., Doosti, H., Ghayour-Mobarhan, M., Nezami, H., Amirabadizadeh, A. A Comparison Between Decision Tree and Random Forest in Determining the Risk Factors Associated with Type 2 Diabetes. Journal of Reseasrch in Health Sciences, 2018, 18 (2), 412.
15. Farnaaz, N., Jabbar, M. A. Random Forest Modeling for Network Intrusion Detection System. Procedia Computer Science, 2016, 89, 213-217. https://doi.org/10.1016/j.procs.2016.06.047
16. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D. Do We Need Hundreds of Classifiers to Solve Real World Classification Problems? Journal of Machine Learning Research, 2014, 15(1), 3133-3181.
17. Ganapathy, S., Kulothungan, K., Muthurajkumar, S., Vijayalakshmi, M., Yogesh, P., Kannan, A. Intelligent Feature Selection and Classification Techniques for Intrusion Detection in Networks: A Survey. EURASIP Journal on Wireless Communications and Networking, 2013, 1, 1-16. https://doi.org/10.1186/1687-1499-2013-271
18. Ghazi, M. R., Gangodkar, D. Hadoop, MapReduce and HDFS: A Developers Perspective. Procedia Computer Science, 2015, 48, 45-50. https://doi.org/10.1016/j.procs.2015.04.108
19. Ghazi, M. R., Raghava, N. S. MapReduce Based Analysis of Sample Applications Using Hadoop. International Conference on Application of Computing and Communication Technologies, Applications of Computing and Communication Technologies, Singapore, 899, 34-44. https://doi.org/10.1007/978-981-13-2035-4_4
20. Govindarajan, M., Chandrasekaran, R. M. Intrusion Detection Using Neural Based Hybrid Classification Methods. Computer networks, 2011, 55(8), 1662-1671. https://doi.org/10.1016/j.comnet.2010.12.008
21. Gupta, B. B., Tewari, A., Jain, A. K., Agrawal, D. P. Fighting Against Phishing Attacks: State of the Art and Fu-

ture Challenges. Neural Computing and Applications, 2016, 28(12), 3629-3654. https://doi.org/10.1007/s00521-016-2275-y

22. Gupta, D., Rani, R. A Study of Big Data Evolution and Research Challenges. Journal of Information Science, 2018, 45(3), 322-340. https://doi.org/10.1177/0165551518789880
23. Gupta, D., Rani, R. Big Data Framework for Zero-Day Malware Detection. Cybernetics and Systems, February 8, 2018, 49(2), 103-121. https://doi.org/10.1080/01969722.2018.1429835
24. Hansen, L. K., Salamon, P. Neural network Ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, 12(10), 993-1001. https://doi.org/10.1109/34.58871
25. Haykin, S. Neural Networks: A Comprehensive Foundation, Prentice Hall, Englewood Cliffs, 1994. https://dl.acm.org/doi/10.5555/975792.975796
26. Holte, R. C. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. Machine Learning, 1993, 11(1), 63-903. https://doi.org/10.1023/A:1022631118932
27. Juniper Research. Cybercrime and the Internet of Threats 2019. https://www.juniperresearch.com/white-papers/cybercrime-and-the-internet-of-threats-2019. Accessed on January 15, 2022.
28. Karegowda, A. G., Manjunath, A. S., Jayaram, M. S. Comparative Study of Attribute Selection Using Gain Ratio and Correlation Based Feature Selection. International Journal of Information Technology and Knowledge Management, 2010, 2(2), 271-277.
29. Lin, W. C., Ke, S. W., Tsai, C. F. Top 10 Data Mining Techniques in Business Applications: A Brief Survey. Kybernetes, 2017, 46(7), 1158-1170. https://doi.org/10.1108/K-10-2016-0302
30. Liu, Z. A Method of SVM with Normalization in Intrusion Detection. Procedia Environmental Sciences, 2012, 11, 256-262. https://doi.org/10.1016/j.proenv.2011.12.040
31. Madi, M., Jarghon, F., Fazea, Y., Almomani, O., Saaidah, A. Comparative Analysis of Classification Techniques for Network Fault Management. Turkish Journal of Electrical Engineering and Computer Sciences, 2020, 28(3), 1442-1457. https://doi.org/10.3906/elk-1907-84
32. McIntosh, T., Jang-Jaccard, J., Watters, P., Susnjak, T. The Inadequacy of Entropy-Based Ransomware Detection. International Conference on Neural Information Processing, December 5, 2019, 1143, 181-189. https://doi.org/10.1007/978-3-030-36802-9_20
33. Mell, P., Grance, T. The NIST Definition of Cloud Computing. NIST Special Publication, 2011, 800-145. https://doi.org/10.6028/NIST.SP.800-145
34. Mirza, A. H. Computer Network Intrusion Detection Using Various Classifiers and Ensemble Learning. 2018 26th Signal Processing and Communications Applications Conference (SIU) IEEE, 2018, 1-4. https://doi.org/10.1109/SIU.2018.8404704
35. Moustafa, N., Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). 2015 Military Communications and Information Systems Conference IEEE, 10 December, 2015, 1-6. https://doi.org/10.1109/MilCIS.2015.7348942
36. Mukkamala, S., Sung, A. H., Abraham, A. Intrusion Detection Using an Ensemble of Intelligent Paradigms. Journal of Network and Computer Applications, 2005, 28(2), 167-182. https://doi.org/10.1016/j.jnca.2004.01.003
37. Noor, U., Anwar, Z., Amjad, T., Choo, K. K. A Machine Learning-Based Fintech Cyber Threat Attribution Framework Using High-Level Indicators of Compromise. Future Generation Computer Systems, 2019, 96, 227-242. https://doi.org/10.1016/j.future.2019.02.013
38. Negandhi, P., Trivedi, Y., Mangrulkar, R. Intrusion Detection System Using Random Forest on the NSL-KDD Dataset. Emerging Research in Computing, Information, Communication and Applications. Advances in Intelligent Systems and Computing, 2019, 906, 519-531. https://doi.org/10.1007/978-981-13-6001-5_43
39. Panigrahi, A., Patra, M. R. Fuzzy Rough Classification Models for Network Intrusion Detection. Transactions on Machine Learning and Artificial Intelligence, 2016, 4(2), 7-22. https://doi.org/10.14738/tmlai.42.1882
40. Panigrahi, A., Patra, M. R. Anomaly Based Network Intrusion Detection Using Bayes Net Classifiers. International Journal of Scientific and Technology Research, 2019, 8(9), 481-485.
41. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J. Scikit-learn: Machine learning in Python. The Journal of machine Learning Research, 2011, 12, 2825-2830.
42. Pham, N.T., Foo, E., Suriadi, S., Jeffrey, H., Lahza, H. F. Improving Performance of Intrusion Detection System Using Ensemble Methods and Feature Selection. Proceedings of the Australasian Computer Science Week Multi-conference, Brisband, Queensland, Australia, 2018, 1-6. https://doi.org/10.1145/3167918.3167951

43. Prusti, D. Efficient Intrusion Detection Model Using Ensemble Methods. Dissertation, 2015. http://ethesis.nitrkl.ac.in/7304/

44. Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufman Publishers, San Mateo, CA, Elsevier, 1993.

45. Rajagopal, S., Kundapur, P. P., Hareesha, K. S. A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets. Security and Communication Networks, 2020, 2020(4586875), 1-9. https://doi.org/10.1155/2020/4586875

46. Rashid, M., Kamruzzaman, J., Imam, T., Wibowo, S., Gordon S. A Tree-Based Stacking Ensemble Technique With Feature Selection for Network Intrusion Detection. Applied Intelligence, 2022, 52, 9768-9781. https://doi.org/10.1007/s10489-021-02968-1

47. Rashid, M. M., Kamruzzaman, J., Hassan, M. M., Imam, T., Gordon, S. Cyberattacks Detection in IoT-Based Smart City Applications Using Machine Learning Techniques. International Journal of Environmental Research and Public Health, 2020, 17(24), 9347. https://doi.org/10.3390/ijerph17249347

48. Rashid, M. M., Kamruzzaman, J., Ahmed, M., Islam, N., Wibowo, S., Gordon, S. Performance Enhancement of Intrusion detection System Using Bagging Ensemble Technique with Feature Selection. 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Gold Coast, Australia, 2020, 1-5. https://doi.org/10.1109/CSDE50874.2020.9411608

49. Reinstein, I. Xgboost, A Top Machine Learning Method on-Kaggle,Explained. https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html. Accessed on February 9, 2022.

50. Salo, F., Nassif, A. B., Essex, A. Dimensionality Reduction with IG-PCA and Ensemble Classifier for Network Intrusion Detection. Computer Networks, 2019, 148, 164-175. https://doi.org/10.1016/j.comnet.2018.11.010

51. Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., Ng, A. Cybersecurity Data Science: An Overview From Machine Learning Perspective. Journal of Big Data, 2020, 7(41), 1-29. https://doi.org/10.1186/s40537-020-00318-5

52. Sarker, I. H., Abushark, Y. B., Alsolami, F., Khan, A. I. Intrudtree: A Machine Learning Based Cyber Security Intrusion Detection Model. Symmetry, 2020, 12(5), 754. https://doi.org/10.3390/sym12050754

53. Saxena, A. K., Sinha, S., Shukla, P. General Study of Intrusion Detection System and Survey of Agent Based Intrusion Detection System. 2017 IEEE International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, May 6, 2017, 421-471. https://doi.org/10.1109/CCAA.2017.8229866

54. Shaw, A. Data Breach: From Notification To Prevention Using PCI DSS. Columbia Journal of Law and Social Problems, 2009, 43(4), 517-562.

55. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A. A. Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection. Computers & Security, 2012, 31(3), 357-374. https://doi.org/10.1016/j.cose.2011.12.012

56. Shrivas, A. K., Dewangan, A. K. An Ensemble Model for Classification of Attacks with Feature Selection Based on KDD99 and NSL-KDD Data Set. International Journal of Computer Applications, 2014, 99(15), 8-13. https://doi.org/10.5120/17447-5392

57. Sikora, R., O'la, A. A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms. Handbook of Research on Organizational Transformations Through Big Data Analytics, IGI-Global, Hershey, PA, 43-53. https://doi.org/10.4018/978-1-4666-7272-7.ch004

58. Sun N., Zhang J., Rimba P., Gao S., Zhang L. Y., Xiang Y. Data-Driven Cybersecurity Incident Prediction: A Survey. IEEE Communications Surveys & Tutorials, December, 2018, 21(2), 1744-1772. https://doi.org/10.1109/COMST.2018.2885561

59. Tama, B. A., Rhee, K. H. An In-Depth Experimental Study of Anomaly Detection Using Gradient Boosted Machine. Neural Computing and Applications, 2017, 31(4), 955-965. https://doi.org/10.1007/s00521-017-3128-z

60. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A. A Detailed Analysis of the KDD CUP 99 Data Set. IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 2009, 1-6. https://doi.org/10.1109/CISDA.2009.5356528

61. Tsai, C. F., Hsu, Y. F., Lin, C. Y., Lin, W. Y. Intrusion Detection by Machine Learning: A Review. Expert Systems with Applications, 2009, 36(10) 11994-12000. https://doi.org/10.1016/j.eswa.2009.05.029

62. Wankhade, A., Chandrasekaran, K. Distributed-Intrusion Detection System Using Combination of Ant Colony Optimization (ACO) and Support Vector Machine (SVM). 2016 IEEE International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), Ghaziabad, India, 2016, 646-651. https://doi.org/10.1109/ICMETE.2016.94

63. Wolpert, D.H. Stacked Generalization. Neural Networks, 1992, 5(2), 241-259. https://doi.org/10.1016/S0893-6080(05)80023-1

64. Yerima, S. Y., Sezer, S. Droidfusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. IEEE Transactions on Cybernetics, 2019, 49(2), 453-466. https://doi.org/10.1109/TCYB.2017.2777960

65. Ying, X. An Overview of Overfitting and Its Solutions. Journal of Physics: Conference Series, 2019, 1168(2), 1-7. https://doi.org/10.1088/1742-6596/1168/2/022022

66. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M. J., Shenker, S., Stoica, I. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, April, 2012, 15-28.

67. Zenko, B., Todorovski, L., Dzeroski, S. A Comparison of Stacking with Meta Decision Trees to Bagging, Boosting, and Stacking with Other Methods. Proceedings 2001 IEEE International Conference on Data Mining, San Jose, CA, USA, 2002, 669-670. https://doi.org/10.1109/ICDM.2001.989601

68. Zhao, G., Shen, Z., Miao, C., Gay, R. Enhanced Extreme Learning Machine with Stacked Generalization. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 2008, 1191-1198. https://doi.org/10.1109/IJCNN.2008.4633951

69. Zhou, Y., Cheng, G., Jiang, S., Dai, M. Building an Efficient Intrusion Detection System Based on Feature Selection and Ensemble Classifier. Computer Networks, 2020, 174, 107247. https://doi.org/10.1016/j.comnet.2020.107247