Hindawi
Journal of Computer Networks and Communications
Volume 2022, Article ID 5988567, 16 pages
https://doi.org/10.1155/2022/5988567

Research Article

Towards the Development of a Cloud Computing Intrusion Detection Framework Using an Ensemble Hybrid Feature Selection Approach

Noah Oghenefego Ogwara, Krassie Petrova, and Mee Loong Yang

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, 55 Wellesley Street East, Auckland 1010, New Zealand

Correspondence should be addressed to Krassie Petrova; krassie.petrova@aut.ac.nz

Received 18 October 2021; Revised 22 January 2022; Accepted 26 January 2022; Published 27 February 2022

Academic Editor: Liansheng Tan

Copyright © 2022 Noah Oghenefego Ogwara et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Attacks on cloud computing (CC) services and infrastructure have raised concerns about the efficacy of data protection mechanisms in this environment. The framework developed in this study (CCAID: cloud computing, attack, and intrusion detection) aims to improve the performance of intrusion detection systems (IDS) operating in CC environments. It deploys a proposed new hybrid ensemble feature selection (FS) method. The ensemble includes FS algorithms of three different types (filter, wrapper, and embedded algorithms). The selected features used to train the ML (machine learning) model of the intrusion detection component comprised a binary detection engine for the identification of malicious/attack packets and a multi-classification detection engine for the identification of the type of attack. Both detection engines deploy ensemble classifiers. Experiments were carried out using the NSL KDD dataset. The binary model achieved a classification accuracy of 99.55% with a very low false alarm rate of 0.45%. The classification accuracy of the multiclassification model was also high (98.92%). These results compare very favourably with the results reported in the literature and indicate the feasibility of the framework implementation.

1. Introduction

Cloud computing (CC) technology provides users with new and potentially easier ways to access, store, and maintain their data [1]. Efficient resource allocation is key for the CC infrastructure to enable the provision of scalable, affordable, and efficient computing resources on demand with a QoS guarantee [2]. For example, Li et al. [3] proposed a nonlinear optimization model for cloud resource allocation that aims to ensure the operation of an IoT system by maximizing the cost-performance ratio for devices used to access resources in the CC environment. However, cyberattacks on cloud resources and services can have a negative impact on the performance of the CC infrastructure and thus jeopardise meeting the QoS requirements. Furthermore, users of CC services are particularly concerned about security threats that may jeopardise their data availability and/or cause data loss [4]. As cyberattacks are becoming increasingly sophisticated, it is important to develop means of preventing or thwarting such attacks and to ensure the security of the data stored and processed in CC environments [5, 6].

A significant number of the attacks on cloud data confidentiality, integrity, and availability are network based [7, 8]. For example, cloud services may be interrupted and data loss may occur as the result of malicious outsider attacks on the cloud network such as denial of service (DoS) and distributed denial of service (DDoS) [9, 10] attacks. These attacks aim to interrupt the traffic between the cloud and the external networks from which cloud users access cloud services. Attacks such as U2R (user-to-root), R2L (remote to local), and network probes can provide the attacker with unauthorised access to data and system resources [8, 11]. Malicious insiders (e.g., a malicious tenant residing in a virtual machine) may also launch similar attacks and threaten the traffic between virtual machines [12]. Other network attacks such as DNS poisoning and flooding can also affect the cloud network [8]. Successful cloud network attacks may have a significant negative impact on cloud service availability and/or data confidentiality, leading to financial and reputational loss [12, 13]. Therefore, cloud defence systems need to be able to detect attacks and to alert the respective response components.

A number of approaches towards the protection of data residing in CC storage have been reported in the literature, including both access control techniques and intrusion detection and intrusion prevention systems (IDS/IPS). An IDS examines network traffic packets to ascertain harmful activities and raise an alarm when such an activity is detected [5, 11]. Furthermore, multiple IDS can be deployed in the IaaS (infrastructure as a service) layer of the cloud computing infrastructure (e.g., at the back end of each processing cloud server or at each virtual switch) [9, 10, 13].

A comprehensive classification and performance review of IDS suitable for deployment in cloud environments is provided in [11]. The authors classify cloud IDS as host based, network based, distributed, and virtual machine introspection (VMI) based. They also noted that network IDS are commonly deployed on virtual switches and these IDS for cloud environments need to be able to handle the high volume of traffic efficiently. All IDS types are signature or anomaly based, or a hybrid of both. A signature-based IDS can effectively detect an attack if it is able to match the attack signature to a known signature pattern in its signature knowledge database. This type of IDS detects attack packets with very high accuracy and has a low false positive rate (i.e., a low number of benign, or normal network traffic packets "detected" wrongly as malicious). However, a signature-based IDS may exhibit a high false negative rate (i.e., a high number of malicious network traffic packets "detected" wrongly as benign), if an attack signature deviates from the stored patterns. In addition, signature-based IDS will fail to recognize unknown attacks (e.g., zero-day attacks), as its signature would not be available in the IDS knowledge base and also may not be able to cope with the high volume of cloud network traffic.

An anomaly-based IDS may be well suited to the recognition of unknown attacks as they attempt to differentiate between anomalous and normal network activity. Anomaly-based intrusion detection methods implement various supervised and unsupervised machine learning (ML) techniques to analyse traffic data and develop a "normal' network traffic profile [14]. This type of IDS can detect both known and unknown attacks with a low false negative rate. However, raw network traffic data are normally characterised by a large number of features, some of which may be redundant. Therefore, datasets used to train ML-based intrusion detection models may comprise a large number of attributes (dimensions), leading to a high false positive rate and a high computational overhead [8, 12]. The implementation of an efficient feature selection (FS) method can reduce the computational overhead by selecting a subset of features that are most likely to lead to both high classification accuracy and low false alarm rate, thus enhancing the performance of the system by reducing its computational complexity [15]. Furthermore, integrating feature selection and classifier techniques can improve IDS performance [16]; therefore, an appropriate FS method deployed at a preprocessing step can contribute to the building of an efficient intrusion detection model by identifying the network traffic features that are most critical to the IDS.

A number of ML-based models for network intrusion detection intrusion detection have been proposed in the literature with some already being used commercially [17]. Support vector machine (SVM) is one of the most commonly used ML classifiers in IDS developed for CC environments; other often used algorithms include Random Forest (RF), Decision Tree (DT), and K-Nearest Neighbours (KNN) [18]. The performance of the intrusion detection models based on supervised learning classifiers can be improved further by using a feature reduction technique. For example, the multiclass SVM intrusion detection model described in [16] improved the classification accuracy while minimising the time required for training and the testing of the model. Here, the authors applied the Chi-squared metric as a feature selection approach and demonstrated the improved performance characteristics of the model on the NSL-KDD dataset.

While cloud intrusion detection models based on a single ML algorithm are characterised by relatively low computational complexity and thus may be fast, deploying ML models that combine the strengths of several ML algorithms may lead to better results in terms of intrusion detection accuracy and low false alarm rate despite the potentially higher execution times. For example, in their analysis of the results reported in the literature on the application of computational intelligence methods for cloud-based IDS, Shamshirband et al. [8] suggested that IDS models applying a hybrid method which is a combination of different ML algorithms performed consistently better compared to IDS models using a single algorithm. The individual classifiers in a hybrid model may run in sequence or in parallel [19]. For example, the use of ensembles of running in parallel ML classifiers for anomaly detection [20] and feature selection [21] has been proposed; here, the ensemble integrates the outputs of the individual classifiers to approximate the target function. Conversely, in the two-stage anomaly prediction model described in [22], the output of one classifier serves to provide input to another. Recently, Krishnaveni et al. [23] applied an ensemble of ML classifiers to both select features and analyse network traffic behaviour, achieving good performance results in terms of classification accuracy and low false alarm rate.

To enhance IDS performance in terms of intrusion detection accuracy, a class of ML methods known as "deep learning" (DL) has been considered in more recent research [24]. DL methods apply artificial intelligence approaches such as deep neural networks (DNN) and deep ANN where a nonlinear computational structure comprising multiple processing layers facilitates data abstraction and learning [25, 26]. For example, Wang et al. [12] applied a single DL method (tracked contractive autoencoder) to extract the most important network traffic features automatically. An

SVM algorithm then used the extracted features to perform network behaviour analysis to identify malicious traffic. According to their results, this hybrid deep/shallow learning model achieved better detection performance compared to SVM and two other state-of-the-art DL methods; the models were tested on both the KDD Cup 99 and the NSL-KDD datasets. Applying an ensemble approach, Elmasry et al. [27] proposed an integrated framework for a DL-based cloud IDS that included monitoring cloud traffic and capturing and preprocessing data to be used as input to the analysis, prediction, and response modules of the framework. The ensemble comprised three DL methods (DNN, Long Short-Term Memory Recurrent Neural Network-LSTM, and Deep Belief Network-DBN); it was trained and tested on the NSL KDD set.

Results from comparative studies have indicated that DL-based IDS may perform better than classic ML-based ones. For example, the XGBoost ensemble (comprising LSTM, multilayer perceptron (MLP), and back propagation network (BNP)) that was proposed in [24] showed better classification accuracies compared to eight other models advanced in recent research; the research used the benchmark UNSW-NB15 dataset. However, the XGBoost model training time was relatively high. Similarly, the ANN-based intrusion detection model described in [28] that used the MLP classifier performed better than five other ML-based models (RF, KNN, SVM, AdaBoost, and Naïve Bayes); the experimental work was done on the realistic cyber dataset CSE-CIC-IDS2018. A benchmarking review of DL methods used in extant research and a performance comparison of IDS deploying DL and ML approaches (as single methods) can be found in [26]. The experiments conducted over two datasets (CSE-CIC-IDS2018 and Bot-IoT) indicated that the DL-based IDS outperformed the ML-based ones in terms of overall detection rates, with SVM and RM giving best results among the ML-based IDS. However, the false alarm rates of the tested DL-based IDS were relatively high.

Despite the significant work already done in the area of cloud IDS, the development and evaluation of ML-based IDS are still a very active area searching for novel, high-accuracy, and high-speed IDS frameworks for identifying attacks [29]. Elmasry et al. [27] also pointed out that, notwithstanding the important results reported in the last ten years or so, "finding a cloud-based intrusion detection system with effective intrusion detection mechanism" was still a desirable goal. Furthermore, Raj and Pani [19] noted that one of the still open challenges in cloud IDS was the effective minimization of the set of network traffic features used by the cloud IDS.

In this paper, we present a hybrid ensemble feature selection method (HEFSM) and report test results which show improved anomaly detection performance. The method is partly a proposed cloud computing attack and intrusion detection (CCAID) framework which includes an ML-based detection component. The study contributions can be summarised as follows: first, the study proposes and tests an efficient hybrid ensemble method for selecting a minimal set network traffic feature that combines the strengths of the individual classifiers it comprises. Second, the study evaluates the performance of a two-stage model for detecting malicious network behaviour that *t* works on the reduced set of features; the performance evaluation results demonstrated that it was highly effective and outperformed a number of proposed models. Finally, the study presents a comprehensive review of FS approaches for cloud IDS that were proposed and tested in prior empirical work and compares the results using a set of representative evaluation metrics; this addresses a literature gap highlighted in [18].

The rest of the paper is organized as follows: relevant work related to FS is reviewed in the section below. The section following introduces the proposed framework and describes in detail the FS approach adopted in this study. The last two sections present and discuss the experimental results of our method and concludes the paper.

2. Feature Selection Methods for Intrusion Detection Systems

Feature selection (FS) determines what data will be extracted from the network traffic flow for examination by the IDS model [27]. The aim is to improve the overall performance of the IDS by creating an optimal (smaller) set of features that provide an accurate representation of the activity in the original packet flow. Supervised, unsupervised, and semi-supervised FS methods can be used to reduce data redundancy and boost performance [30, 31]. FS methods can be classified as filter, wrapper, and embedded methods [32]. Filter and wrapper methods in particular are widely used in intrusion detection ML models that are trained on network traffic data [33, 34]. Combining different FS methods may enhance the performance of the classifiers used in ML models by identifying and selecting features that may be weak as single predicting entities, but strong as a set of relevant features that are highly associated with the target output class [1].

2.1. Filter Feature Selection Methods.

Filter FS methods operate independently of the classification algorithm of the ML intrusion detection model. Features are selected at the preprocessing stage based on the structural properties of the data [34]. Typically, features are assessed and ranked using measures such as dependency and distance. Filter FS are particularly efficient when applied to relatively large datasets [35, 36]. For example, Ambusaidi et al. [37] proposed a filter FS selection model using mutual information as a relationship measure and trained the ML intrusion detection model with an SVM classifier. The model was tested on several datasets (KDD CUP99, NSL KDD, and Kyoto 2006+) using classification accuracy (i.e., the total number of correctly "detected" network traffic packets compared to the overall number of packets), detection rate (i.e., the number of network traffic packets correctly "detected" as malicious compared to the number of all packets "detected" as malicious, also known as true positive rate), and false positive rate as validation metrics.

Osanayie et al. [1] applied a multifilter FS method to combat DDoS attacks in CC environments. The significant features that enhance the detection accuracy of DDoS attacks were selected by using information gain, gain-ratio, chi-squared, and relief as filter metrics. The ML intrusion detection model used DT classifiers. It was evaluated using the NSL KDD dataset with the following validation metrics: classification accuracy, detection rate, false positive rate (as a measure of the false alarm rate), and the time to build the model. More recently, an ensemble of filter FS methods was proposed by Krishnaveni et al. [23]. Using the same metrics as in [1], the authors evaluated the ML model across several datasets (NSL KDD, the Kyoto 2006 + dataset, and a dataset of real-time network traffic collected by the authors).

An innovative filtering approach based on visual simulation was proposed by Luo and Xia [38]. The experimental data were mapped onto a four-star graph; the computed distances between the coordinates of the data and the star vertexes were used to generate a reduced set of features representing the data. The authors conducted experiments on the KDD CUP99 dataset, using an SVM clustering algorithm; the model was validated for classification accuracy.

The filter FS methods reviewed above apply supervised learning approaches [34]. Selecting features through unsupervised learning was proposed by Idhammad et al. [39]; the filter algorithm selected features by applying the Pearson correlation coefficient (PCC). An ANN was used to build an ML model for the detection of DoS attacks. Its performance was evaluated on the NSL KDD and UNSW-NB15 datasets using classification accuracy, true positive (detection) rate, true negative rate (i.e., the number of normal traffic packets correctly "detected" as normal compared to the overall number of normal network traffic packets), and false alarm rate (the arithmetic average of false negative rate and false positive rate) as evaluation metrics.

2.2. Wrapper and Embedded Feature Selection Methods. Wrapper (and embedded) FS methods examine and evaluate the features of the data in the training dataset with a consideration of the targeted classification output. They work independently of the intrusion detection ML model (normally, at a preprocessing stage). Conversely, in embedded methods, features are selected as part of the training process of the ML detection model. The method used to select features is specific to the ML classifier deployed by the model [34]. Typically, wrapper and embedded methods are more time consuming compared to filter FS methods due to the heavier computational requirements of the classifier used to ascertain the relevance of the feature set [15].

A typical example of a wrapper FS approach is presented in [40]. A modification of the cuttlefish optimization algorithm (CFA) was used to select the optimal set of features. The intrusion detection ML model was deployed with a decision tree classifier and was validated on the KDD CUP99 dataset using classification accuracy and false positive rate as validation metrics. Moreover, using KDD CUP99, Sun et al. [41] proposed a wrapper method that deployed the nature-inspired lightning attachment procedure optimization (LAPO) algorithm to select a subset of relevant best performing features. The intrusion detection ML used an SVM algorithm; the model performance was evaluated using classification accuracy.

Pham et al. [42] presented a wrapper FS approach that used Naïve Bayes and a gain ratio technique to select significant features. The method was tested on two ensemble models (ensemble bagging and ensemble boosting) with tree-based algorithms as the base classifier. The evaluation of the model was done on the NSL KDD dataset based on classification accuracy and false alarm (false positive) rate. An ensemble approach to building the ML detection intrusion model was also used by Besharati et al. [43]. Their proposed host-based IDS for a CC environment implemented logistic regression to remove features that were not relevant to the detection of attacks. The ML intrusion detection model was based on ensemble bagging, including DT, linear discriminant analysis, and ANN. It was validated using classification accuracy, precision (the ratio of the number of malicious network traffic packets "detected" correctly as malicious, to the total number of network packets "detected" as malicious), and F-score (the harmonic mean of precision and recall).

Belouch et al. [44] and Vijayanand et al. [45] attempted to refine the FS process by selecting different sets of features depending on the type of traffic [44] or on the type of attack [45]. In [44], the FS method used information gain to identify the features that were not related to the targeted output by applying an evolutionary search process. The ML intrusion detection model deployed a two-stage classification process using the reduced error pruning tree (REPTree) algorithm. The model was tested and validated on the NSL KDD and UNSW-NB15 datasets using classification accuracy as a performance metric. In [45], a genetic algorithm was used for FS and multiple SVM classifiers were implemented in the intrusion detection ML. The performance evaluation criteria used were detection accuracy, false positive rate, and false negative rate (for each attack type). The model was tested on the ADFA-LD and CICIDS2017 datasets. Aljawarneh et al. [46] also proposed an improvement of the FS process by using information gain and multiple classifiers including Naïve Bayes, J48, AdaBoost, and RT. Features were selected based on the best classification outcome. The ML intrusion detection model was trained on the NSL KDD dataset and validated for classification accuracy.

2.3. Hybrid Feature Selection Methods. Some researchers have attempted to combine the strengths of different feature selection algorithms. For example, CfsSubsetEvalk and WrapperSubsetEval FS algorithms are combined in [5] to identify features that are most strongly related to the targeted classification output. The selected features were used in an ML-based IDS, running several classifiers (RF, J48, KNN, and Naïve Bayes) on the NSL KDD dataset. The validation metrics included classification accuracy, detection rate, true positive rate, false positive rate, false negative rate, Matthews correlation coefficient, and performance time.

Moustafa and Slay [47] also applied a filter and a wrapper; the calculated central points of attribute values and association rule mining were used to reduce the high dimensionality of the two datasets (UNSW-NB15 and NSL

KDD). Expectation-maximisation (EM) clustering, logistic regression, and Naïve Bayes were the classifiers deployed in the network IDS, with evaluation metrics using classification accuracy and false alarm rate (the arithmetic average of false positive rate and false negative rate). Similarly, Mogal et al. [48] used the central points of attribute values and applied the Apriori algorithm to remove features that were not relevant based on the false alarm rate. The ML model used Naïve Bayes and logistic regression as classifiers; the IDS detection engine was validated using classification accuracy and processing time as metrics using the KDD CUP99 and UNSW-NB15 datasets.

Researchers have continued to experiment with involving a larger number of classifiers in the FS selection process, the ML training process, or in both processes as a means of improving the performance of the ML intrusion detection model. More recently, a multiobjective feature selection algorithm that used mutual information and a probabilistic model to search for relevant features was presented in [15]. The attack detection ML model used Naïve Bayes, MLP, NN, SVM, KNN, and DT. It was trained on the NSL KDD dataset using detection accuracy as a validation metric. In the cyber IDS proposed by Mohammadi et al. [31], linear correlation coefficient and CFA were used to extract features relevant to cyber intrusion. The ML model (DT based) was trained on the KDD CUP99 dataset; it was evaluated using classification accuracy, detection rate, false positive rate, statistical fitness, and processing time. Furthermore, Anwer et al. [50] developed a structured framework of strategies to combine filter and wrapper feature selection approaches. Their ML intrusion and anomaly detection model was trained with the J48 and Naïve Bayes classifiers using the UNSW-NB15 dataset; it was validated using classification accuracy.

A hybrid FS method which used three evolutionary search techniques (particle swarm optimization, ant colony optimization, and genetic algorithm) was proposed by Tama et al. [22]. The ensemble ML model was trained on the NSL KDD and the UNSW-NB15 dataset using ensemble bagging with rotation forest. The validation metrics included classification accuracy, false positive rate, sensitivity (true positive rate), precision, and a statistical significance test.

A slightly different approach towards combining FS methods was proposed by Bhattachary and Selvakumar's [51]: a multimeasure, multiweight feature identification method that combined filter, wrapper, and unsupervised learning to rank features relevant to probe and DoS attacks. The ML intrusion detection model used several classifiers (Bayesian network, J48, Simple K Means, and Vote) for predicting the output. The model was evaluated using three different datasets: NSL KDD, SSENet, and ISCX. The FS validation metrics were prediction accuracy, the average number of false positives, the average number of false negatives, and the average training time.

Overall, the summary above shows that in the reviewed studies, FS was used as a means of enhancing ML-based IDS performance. The proposed FS technique was normally applied to well-known benchmark datasets such as NSL KDD, KDD CUP99, and UNSW-NB15. Furthermore, in most cases, a single FS method was used (normally, a filter or a wrapper); similarly, most of the intrusion detection and attack recognition ML models used a single classifier (often, SVM and/or DT). However, the results from deploying a hybrid FS approach indicated that better outcomes may be achieved by combining the strength of the different types of FS methods. The results also show that classification accuracy and other ML performance metrics may benefit from applying an ensemble of classifiers [52].

While all proposed ML models were evaluated using the classification accuracy metric, the false positive rate was used by some as a measure of the false alarm rate. In intrusion detection and attack recognition systems where the majority of the examined network traffic packets are expected to be benign rather than malicious, measuring the false alarm rate by the balanced error rate (the arithmetic average of false positive rate and false negative rate) may be also useful as a performance evaluation criterion [53]. Finally, even though ML model efficiency in terms of computational time and resources used was considered only in some of the proposed models, more consideration should be given to them as these are critical to the performance of the IDS as an integral part of the CC infrastructure.

In the study presented here we aimed to develop a new ML model for intrusion detection and attack recognition that performed with a high overall classification accuracy and low false alarm rate, but without being overly demanding computationally; such models would be suitable to be implemented in IDS operating in high risk CC environments. Given the high traffic volume in CC, it is also essential that the IDS does not impact negatively on the CC service provision. As a way of meeting these requirements, this study focuses on the development of an efficient method for reducing the number of features used in the ML intrusion detection and attack classification models.

3. Materials and Methods

A conceptual overview of the proposed CCAID framework is shown in Figure 1; the specific ML algorithms shown in the figure were selected experimentally as described further in this section. The framework comprises a preprocessing stage in which the proposed hybrid ensemble FS method (HEFSM) is applied to select a subset of relevant features ("ensemble features") from the training dataset. In particular, HEFSM selects a subset of the most significant features by applying a hybrid FS approach comprising a filter, a wrapper, and an embedded algorithm, to identify the highly correlated features that are relevant to the detection of an attack in a CC environment.

The filter FS method considers the properties of the dataset to evaluate and rank features, while the wrapper method works as an ML classifier that adds or removes features aiming to produce a subset of features that best predict the target class [34]. The embedded FS method combines the advantages of the other two methods as features are ranked based on statistical measures that consider the specific classification algorithm used; the features with the highest ranking values are chosen. Adding an embedded

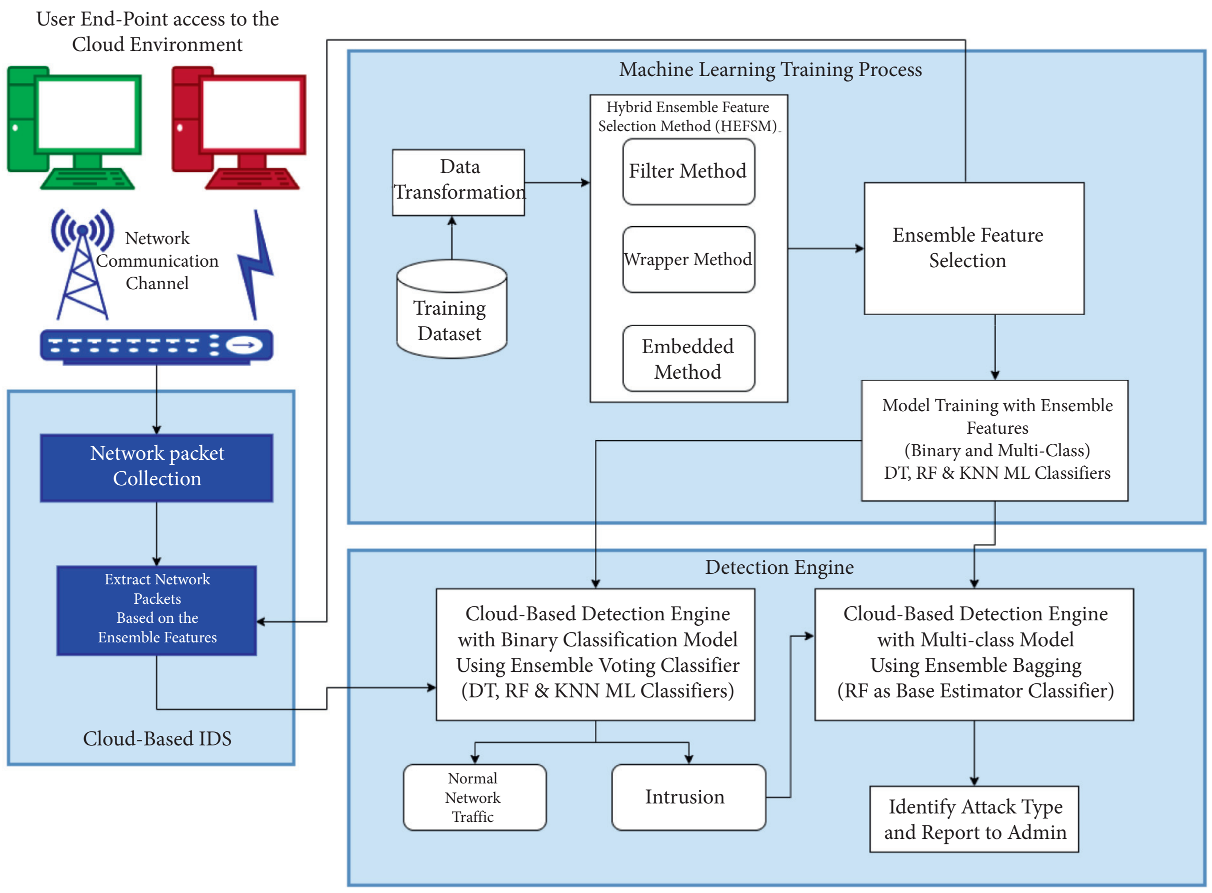
User End-Point access to the Cloud Environment
Network Communication Channel
Network packet Collection
Extract Network Packets Based on the Ensemble Features
Cloud-Based IDS
Machine Learning Training Process
Data Transformation
Training Dataset
Hybrid Ensemble Feature Selection Method (HEFSM)
Filter Method
Wrapper Method
Embedded Method
Ensemble Feature Selection
Model Training with Ensemble Features (Binary and Multi-Class) DT, RF & KNN ML Classifiers
Detection Engine
Cloud-Based Detection Engine with Binary Classification Model Using Ensemble Voting Classifier (DT, RF & KNN ML Classifiers)
Cloud-Based Detection Engine with Multi-class Model Using Ensemble Bagging (RF as Base Estimator Classifier)
Normal Network Traffic
Intrusion
Identify Attack Type and Report to Admin

Figure 1: Cloud computing attack and intrusion detection framework (CCAID). The feature selection process using HEFSM and the ML intrusion detection and attack classification model training are shown in the box labelled "Machine Learning Training Process." The working of the intrusion detection engine is shown in the box labelled "Intrusion Detection Process."

algorithm is particularly important as it identifies the optimal subset of features with high accuracy [54].

The schematic representations of the ML model training and the cloud IDS detection processes can be seen in Figure 1. First, the data in the training dataset are converted to a numeric format. The transformed dataset is used as input by each of the FS methods that constitute the FS ensemble (HEFSM). The output of the ensemble is a subset of features ("ensemble features") that were selected by all three FS methods. The ensemble features are used to train the ML models that comprise the detection component of the CCAID framework.

The CCAID detection component consists of two detection engines—a binary engine and a multiclass engine. During the intrusion detection process, the cloud-based IDS analyses the incoming traffic of a specific virtual machine instance used by a cloud customer as follows. The network packet feature extractor captures incoming traffic and reduces the dimensionality of the incoming traffic. Then, it applies the ensemble feature set to extract data matching the selected features and creates a dataset that contains the selected features' values. The dataset is passed on to the binary detection engine. If a malicious behaviour traffic pattern is identified, the binary detection engine forwards the data to the multiclass detection engine which determines the type of the attack and communicates it to an appropriate component of the cloud environment defence management system.

3.1. Preprocessing Stage.

This research used a dataset named NSL KDD which was obtained from the public repository of the University of New Brunswick (https://www.unb.ca/cic/datasets/index.html). The NSL-KDD set was recently used by Elmasry et al. [27] as the benchmark set for training and testing of a DL-based cloud IDS. The set was also used as one of the three benchmark sets in [23] where an ensemble voting approach was used for selecting features as well as for detecting attacks.

The network traffic records in the NSL KDD dataset contain values for each of the features representing the network packet properties (for a guide to the NSL KDD dataset features as referred to in this study; see Table 1; a detailed description of the dataset can be found in [46]). Most of the feature values are numeric, but three of the features were represented by nonnumeric values. Therefore, the original dataset was first modified by replacing the string values of the three nonnumeric features with numeric values, as shown in Table 2.

There is still a very limited number of publicly available datasets with data collected at virtual machine

Table 1: NSL KDD dataset features as referred to in this study.

F/N	Feature name	Type	Description
F1	Duration	Numeric	The length of the connection process
F2	Protocol_type	String	The connection protocol type of a packet (TCP, UDP, etc.)
F3	Service	String	The type of network service at the destination (e.g., Http)
F4	Flag	String	The normal or error status of the connection
F5	Src_bytes	Numeric	The number of data bytes sent, i.e., source to destination
F6	Dstbytes	Numeric	The number of data bytes received, i.e., destination to source
F7	Land	Numeric	To check if a connection is from the same host or not (1 or 0)
F8	Wrong fragment	Numeric	The total number of wrong fragments of a connection
F9	Urgent	Numeric	The total number of the packet that is urgent
F10	Hot	Numeric	The total number of hot indicators in a packet
F11	Numfailedlogins	Numeric	The total number of times login attempt failed by a connection
F12	Loggedin	Numeric	Login status of a connection (1 successful, 0 failed)
F13	Numcompromised	Numeric	The total number of compromised conditions
F14	Rootshell	Numeric	To determine if a root shell is obtained or not (1 yes 0 no)
F15	Suattempted	Numeric	To check if a super user root command is attempted or not
F16	Numroot	Numeric	The total number of root accesses
F17	Numfilecreations	Numeric	The total number of file creation activities attempted
F18	Numshells	Numeric	The total number of shell prompts recorded
F19	Numaccessfiles	Numeric	The total number of attempts in access control files
F20	Numoutboundcmds	Numeric	The total number of outbound commands in file transfer task
F21	Ishostlogin	Numeric	To check the login belong to the host list or not
F22	Isguestlogin	Numeric	To check if the login is guest or not
F23	Count	Numeric	The total number of connection to the same host as the current connection in the last two seconds
F24	Srvcount	Numeric	The total number of connection to the same service as the current connection in the last two seconds
F25	Serrorrate	Numeric	Total (%) of connection that has “SYN” errors in same-host connection
F26	Srvserrorrate	Numeric	Total (%) of connection that has “SYN” errors in same-service connection
F27	Rerrorrate	Numeric	Total (%) of connection that has “REJ” errors in same-host connection
F28	Srvrerrorrate	Numeric	Total (%) of connection that has “REJ” errors in same-service connection
F29	Samesrvrate	Numeric	The total (%) of connection to the same service connection
F30	Diffsrvrate	Numeric	The total (%) of connection to different services
F31	Srvdiffhostrate	Numeric	The total (%) of connection to a different host
F32	Dsthostcount	Numeric	The total (%) count of connection having the same destination host
F33	Dsthostsrvcount	Numeric	The total (%) count of connection; having the same destination host and using the same service
F34	Dsthostsamesrvrate	Numeric	The total (%) of connection having the same destination host and using the same service
F35	Dsthostdiffsrvrate	Numeric	The total (%) of different services on the current host
F36	Dsthostsamesrcportrate	Numeric	The total (%) of connection to the current host having the same port
F37	Dsthostsrvdiffhostrate	Numeric	The total (%) of connection to the same service coming from different hosts
F38	Dsthostserrorrate	Numeric	The total (%) of connection to the current host that has a SO error
F39	Dsthostsrvserrorrate	Numeric	The total (%) of connection to the current host and specified service that has a SO error
F40	Dsthostrerrorrate	Numeric	The total (%) of connection to the current host that has an RST error
F41	Dsthostsrvrerrorrate	Numeric	The total (%) of connection to the current host and specified service that have an RST error

and hypervisor levels such as the relatively new ISOT-CID dataset [55]. Despite the heterogeneous nature of cloud based networks, conventional and cloud-based networks share numerous common characteristics. Therefore, datasets used to train and test network-based IDS such as the benchmark NSL-KDD dataset are used widely in solutions proposing IDS for cloud environments [18, 56].

All features selected by the proposed feature selection algorithm are applicable to the network traffic in cloud environments with five of them also found in the ISOT-CID dataset. For example, the feature F1 “duration” (representing the length of the connection process in a conventional network environment) is the equivalent of the feature “frame_len” that represents cloud-based hypervisor traffic feature in ISOT-CID. Similarly, F5 and F6 in the NSL-KDD dataset are the equivalent of “traffic_in” and “traffic_out” features in the cloud-based dataset while F3 and F23 match “tcp_seq” and “tcp_ack” in the ISOT-CID dataset.

The complete dataset consisting of 148,517 network traffic records (packets) are labelled either as normal or as attack packets. Four categories of well-known attack packets are represented in the dataset [15]:

(1) Denial of Service Attack (DoS): this is the type of attack that results in the unavailability of resources or services requested by the users of a system

Table 2: Assigning numeric values to nonnumeric features.

Feature	Previous value	New value
F2	ICMP	1
	TCP	2
	UDP	3
F3	AOL	1
	AUTH	2
	BGP	3
	COURIER	4
	. . .	. . .
	Z3950	70
F4	OTH	1
	REJ	2
	RSTO	3
	RSTOS0	4
	RSTR	5
	S0	6
	S1	7
	S2	8
	S3	9
	SF	10
	SH	11

(2) User to Root Attack (U2R): this type of attack leads to the hijacking of the root account as a result of a compromised user account

(3) Remote to Local Attack (R2L): in this type of attack, a network packet is sent to a machine to hijack a user account to gain unauthorised access to the machine

(4) Probe Attack (probe): in this type of attack the host ports are scanned to discover open ports that can be used to exploit potential weaknesses of the system

For the purpose of this research, the original dataset was split into three smaller datasets as follows: 70% of the data (103,961 records) were used for training the ML models, 20% of the data (29,704 records) were used for testing, and 10% of the data (14,852 records) were used for validation. The distribution of the attack instances across the training, testing, and validation datasets is shown in Table 3.

For the purpose of training the multiclass detection engine, the dataset in Table 3 was modified, whereby the current string values of the packet type labels were replaced by numeric values (Table 4). Another modification was created for the purpose of training the binary detection engine where the packet label was modified to indicate either a benign (normal, nonattack) packet or an attack (intrusion) packet (Table 5).

3.2. Feature Selection Stage. The filter, wrapper, and embedded FS methods used in the proposed HEFSM ensemble were SelectKBest, RF, AdaBoost, respectively. The FS algorithm used by HEFSM is presented in Table 6.

The filter method (SelectKBest) applies the chi-squared statistical test to compute a score for each feature in the dataset [23]. The computed score measures the level of dependency between the feature attribute and the target class. SelectKBest deploys this univariate method to rank each of the dataset features: a high score shows a high-dependency relationship that may be useful as part of the prediction model.

Table 3: Number of packets in the training, testing, and validation sets (per packet type).

Packet type	Training set	Testing set	Validation set
Normal traffic	54,148	15,460	7,683
DoS	37,099	10,590	5,403
U2R	8,268	2,384	1,130
R2L	2,734	759	386
Probe	1,712	511	250

The RF algorithm that is used as the second FS method of the ensemble applies a forward selection approach; the algorithm starts with a null feature subset and keeps adding one feature at a time until an optimal feature subset is produced. The selected "most important" features perform best in terms of classification accuracy [5].

The ensemble algorithm AdaBoost used as an embedded FS method selects the features that give the best prediction during the training of the learning algorithm. The boosting procedure combines the output of weaker classifiers into a weighted sum at the end of each iteration to reach a final output [46].

Each of the three FS methods included in HEFSM rank, and select the best features using their respective feature importance metrics. Subsequently, HEFSM selects the features that are common across the three independently produced optimal feature sets.

We conducted the experiments by applying HEFSM to the training dataset to identify the features that were part of the output of each of the three FS methods in HEFSM ensemble. We first applied SelectKBest with a threshold of 0.0 and selected the top 50% of the ranked features of the study dataset (i.e., 20 features). Next, we ran Random Forest

Table 4: Network traffic packet labelling for the multiclassification detection engine.

Packet label	Previous value	New value
Normal traffic	Normal	1
DoS	Back, land, Neptune, pod, smurf, teardrop	2
U2R	Bufferoverf low, Loadmodule, Rootkit, Perl, Ps	3
R2L	Guess Password, Ftp write, Imap, Phf, Multihop, Warezmaster, Warez client, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named	4
Probe	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint	5

Table 5: Network traffic packet labelling for the binary detection engine.

Packet type	Previous value	New value
Normal traffic	Normal	0
Intrusion (malicious) traffic	DoS, U2R, R2L, probe	1

Table 6: The HEFSM algorithm.

Step	Description
Step 1	Let **F** represent the array of feature set in the dataset, where n is the number of features in the dataset such that $n = 41$. $\mathbf{F_n} = \mathbf{F_1}, \mathbf{F_2}, \mathbf{F_3}, \ldots\ldots, \mathbf{F_{41}}$ and let **E** represent an empty array of the selected features from the ensemble FS process.
Step 2	Let **T** represent the target class of the dataset such that **T** is either **0** or **1,** where **0** is a normal traffic and **1** is intrusion.
Step 3	Using the filter FS method with chi-square approach, compute a score for each feature in $\mathbf{F_n}$ and store the results in an array **X**.
Step 4	Using the wrapper FS method with Random Forest ML inbuilt FS algorithm, compute the feature importance score for each feature in $\mathbf{F_n}$ and store the results in Array **Y**.
Step 5	Using the embedded FS method with AdaBoost ML inbuilt FS algorithm, train the model and select the best performing features with the highest classification accuracy and assign scores to each feature in **Fn** and store results in an array Z.
Step 6	Remove features in Arrays **X, Y,** and **Z** where score is equal to zero.
Step 7	Set $\mathbf{j = 1}$ and repeat step 8, step 9, and step 10 while $\mathbf{j \leq 41}$.
Step 8	IF $\mathbf{F_j}$ is in Array **X** and $\mathbf{F_j}$ is in Array **Y** and $\mathbf{F_j}$ is in Array **Z,** then go to step 9 or otherwise go to step 10.
Step 9	Store $\mathbf{F_j}$ in the final array of the ensemble features **E**.
Step 10	Increment **j** by **1**.
Step 11	Output the final feature set in Array **E** as the ensemble features.

with a preset target number of 20 features; the algorithm identified a subset comprising the 20 most important features in the dataset. Finally, AdaBoost ranked the dataset features based on their significance in relation to the targeted output and produced an optimal set of 18 features. The features selected by each of the methods used are shown in Table 7. As seen, the HEFSM outcome comprises 11 features common to the three independently used FS methods (Table 7, bottom row).

4. Results and Discussion

To evaluate the impact of the proposed FS approach on the performance of the ML models for intrusion detection and attack classification, we implemented the binary and multiclass detection engines as a console-based application using a Jupyter Notebook with Python as the language of implementation. We used the scikit-learn and Anaconda ML libraries for the building of the ML models on an Intel(R) Core(TM) i7-8700 CPU @3.20 GHz machine (16 GB RAM, 500 GB HD). The binary detection engine is a binary classifier that applies an ensemble voting approach. The ensemble comprises three ML algorithms (Decision Tree, Random Forest, and K-Nearest Neighbours). The multiclass detection engine applies an ensemble bagging classifier with Random Forest as the base estimator. The ensemble methods for the detection engines were selected experimentally based on their performance.

The scope of the experiments that were conducted included training, testing, and validating each of the two detecting engines in the CCAID framework using the datasets described in the previous section. We obtained results using both the full set of the dataset and the ensemble set of features that were selected by applying HEFSM. For comparison purposes, similar experiments were conducted by deploying the ten well-known ML classifiers shown in Table 8.

All classifiers except C5 were used in prior work reviewed in this study; in particular, C1, C2, C8, C9, and C10 are commonly used in IDS operating in a range of networked environments including CC [15, 18, 28, 57, 58]. Classifiers C4, C6, and C7 were considered by Maza and Touahria [15] for benchmarking the performance of their proposed multiobjective FS method; C4 was used in

Table 7: Feature selection output.

FS method	Selected features
SelectKBest (20)	F1, F3, F4, F5, F6, F10, F12, F13, F16, F23, F24, F25, F26, F29, F32, F33, F34, F36, F38, F39
Random Forest (20)	F1, F3, F4, F5, F6, F7, F9, F10, F17, F20, F23, F28, F29, F30, F32, F34, F35, F38, F39, F40
AdaBoost (18)	F1, F2, F3, F4, F5, F6, F8, F11, F23, F26, F29, F32, F33, F34, F35, F38, F39, F40
HEFSM (11)	F1, F3, F4, F5, F6, F23, F29, F32, F34, F38, F39

Table 8: ML classifiers used for comparison.

Classifier ID	Classifier name
C1	Decision Tree
C2	Random Forest
C3	AdaBoost
C4	Naïve Bayes
C5	Stochastic dual coordinate ascent
C6	Multilayer perceptron
C7	K-Nearest Neighbours
C8	Linear discriminant analysis
C9	Logistic regression
C10	Support vector machine

[47, 48, 50], C6 was used in [24], and C7 was also used in [28]. Classifier C3 was used in [28] and is a preferred classifier for distributed processing environments similar to CC [59]. The algorithm C5 was added as it is one of the primary methods for achieving high accuracy in multi-category classification [60].

4.1. Performance Metrics. We evaluated the performance of the ML models using a confusion matrix using the test dataset and applied a ten-fold cross-validation using the validation datasets. The performance metric definitions are provided as follows.

(1) True positive (TP): the total number of network packets that were classified correctly as attack packets
(2) True negative (TN): the total number of network packets traffic classified correctly as benign (non-attack) packets
(3) False positive (FP): the total number of network packets that were misclassified as attack packets
(4) False negative (FN): the total number of network packets that were misclassified as benign packets

Classification accuracy, true positive rate (recall or detection rate), and false alarm rate are the most widely used criteria for evaluating the performance of ML-based IDS [8, 18]; false alarm is measured either as false positive rate [26] or as the average of false positive rate and false negative rate [47]. Other measures include precision and the harmonic mean of precision and recall (F-score) [23]. In this study, the following performance criteria were used:

(1) Classification accuracy (CA): the percentage of all correctly classified attack and benign packets out of all network packets in the dataset.

$$CA = \frac{TP + TN}{TP + TN + TP + FN} \times 100. \tag{1}$$

(2) Error rate (ER): the percentage of all misclassified network packets out of all network packets in the dataset.

$$ER = \frac{FP + FN}{TP + TN + FP + FN} \times 100. \tag{2}$$

(3) Precision rate (PR): the percentage of all correctly classified attack packets out of all network packets in the dataset classified as attack packets (either correctly or wrongly).

$$PR = \frac{TP}{TP + FP} \times 100. \tag{3}$$

(4) Recall (RC): the percentage of all correctly classified attack packets out of all attack network packets in the dataset.

$$RC = \frac{TP}{TP + FN} \times 100. \tag{4}$$

(5) False positive rate (FPR): the percentage of all benign packets misclassified as attack packets out of all benign network packets in the dataset.

$$FPR = \frac{FP}{FP + TN} \times 100. \tag{5}$$

(6) False negative rate (FNR): the percentage of all attack packets misclassified as benign packets out of all attack network packets in the dataset.

$$FNR = \frac{FN}{FN + TP} \times 100. \tag{6}$$

(7) False alarm rate (FAR): the arithmetic mean of FPR and FNR.

$$FAR = \frac{FPR + FNR}{2}. \tag{7}$$

4.2. Performance Evaluation. We ran experiments with each of the ten classifiers in Table 8 to determine the best performing ones and consider them for inclusion in the detection engines; each classifier was applied twice—first using all features in the dataset and then using only the features selected by the HEFSM ensemble. Based on the outcomes (as shown in Table 9, top ten rows), we included the top three performing classifiers (namely, C1, C2, and C7) in the

Table 9: Intrusion packet identification: performance metrics.

Classifier/feature set	TP	FP	TN	FN	CA	ER	PR	RC	FPR	FNR	FAR (FPR + FNR)/2
	In number of packets				In percentage						
C1-All	14,165	77	15,383	79	99.47	0.53	99.46	99.45	0.5	0.55	0.53
C1-Ens	14,152	100	15,360	92	99.35	0.65	99.3	99.35	0.65	0.65	0.65
C2-All	14,170	53	15,407	74	99.57	0.43	99.63	99.48	0.34	0.52	0.43
C2-Ens	14,163	60	15,400	81	99.53	0.47	99.58	99.43	0.39	0.57	0.48
C3-All	13,739	337	15,123	505	97.17	2.83	97.61	96.45	2.18	3.55	2.86
C3-Ens	13,521	520	14,940	723	95.82	4.18	96.3	94.92	3.36	5.08	4.22
C4-All	274	208	15,252	13,970	52.27	47.73	56.85	1.92	1.35	98.08	49.71
C4-Ens	274	206	15,254	13,970	52.28	47.72	57.08	1.92	1.33	98.08	49.7
C5-All	8,145	14,796	664	6,099	29.66	70.34	35.5	57.18	95.71	42.82	69.26
C5-Ens	8,915	14,693	767	5,329	32.59	67.41	37.76	62.59	95.04	37.41	66.23
C6-All	1	47	15,413	14,243	51.89	48.11	2.08	0.01	0.3	99.99	50.15
C6-Ens	14,244	15,460	0	0	47.95	52.05	47.95	100	100	0	50
C7-All	14,114	144	15,316	130	99.08	0.92	98.99	99.09	0.93	0.91	0.92
C7-Ens	14,067	137	15,323	177	98.94	1.06	99.04	98.76	0.89	1.24	1.06
C8-Al	12,903	737	14,723	1,341	93	7	94.6	90.59	4.77	9.41	7.09
C8-Ens	10,991	279	15,181	3,253	88.11	11.89	97.52	77.16	1.8	22.84	12.32
C9-All	11,868	2,511	12,949	2,376	83.55	16.45	82.54	83.32	16.24	16.68	16.46
C9-Ens	97,51	3,000	12,460	4,493	74.77	25.23	76.47	68.46	19.4	31.54	25.47
C10-All	12,400	3,407	12,053	1,844	82.32	17.68	78.45	87.05	22.04	12.95	17.49
C10-Ens	9,507	1,943	13,517	4,737	77.51	22.49	83.03	66.74	12.57	33.26	22.91
C11-All	14,126	39	15,421	118	99.47	0.53	99.72	99.17	0.25	0.83	0.54
C11-Ens	14,178	68	15,392	66	99.55	0.45	99.52	99.54	0.44	0.46	0.45

All: all features; Ens: features selected by the HEFSM ensemble. C11 is the proposed binary detection engine.

ensemble voting classifier of the binary classification engine (shown as C11 in the bottom row of Table 9).

As seen in Table 9, the results obtained when running the proposed binary engine with the HEFSM ensemble selected features were consistent with and even better than the results obtained when running the same model with the full set of features, except for the false positive rate. Overall, the binary detection engine performed well compared to the best performing classifiers (C1 and C2). The classification accuracy of 99.55% is the second best across all experiments (the best classification accuracy of 99.57% was achieved by C2 running on the full set of features). The false alarm rate of 0.45% was the lowest one achieved across all experiments. Furthermore, the high precision and recall rates of 99.52% and 99.54%, respectively, and the low error rate of 0.45% indicated that the binary detection engine was stable in terms of performance and had the ability to detect accurately intrusion (attack) packets.

The top performing classifier in Table 9 (C2) was chosen for building the ML model of the multiclass detection engine. The experimental data indicated that when applied to classifier C2, ensemble bagging performed better than ensemble voting; therefore, the multiclass detection engine adopted an ensemble bagging approach. The experimental results are shown in Table 10. As seen, the proposed multiclass detection engine (classifier C12) achieved an average attack packet classification accuracy of 98.92% (average calculated across the four attack categories).

It can be seen in Table 9 that the results from running the classifiers with the full feature set and with the ensemble features set are relatively close. This confirms the feasibility of building an intrusion detection model that uses a minimal set of features. Furthermore, the proposed classification models give slightly better results when used with the full set of features compared to using the ensemble selected features only. However, the execution time of the model when run on the features selected by the HEFSM ensemble is much lower compared to the execution time of the same model when run with the full set of features (Table 11). This is an important characteristic of the proposed model: when dealing with network security controls, speed is one of the major factors that need to be considered. The application of an efficient model that is able to analyse the heavy CC network traffic in real time and detect abnormalities in network packets with high detection accuracy and a low false alarm rate will improve significantly the performance of an IDS operating in a CC environment.

The results of the tenfold crossvalidation of the proposed classification models show that judging by the difference between the maximum and minimum CA values of the binary classification model, its stability is higher when the model is run with the HEFSM ensemble selected features compared to running it with the full set of features (Table 12). This result is very encouraging; as CC environments are characterised by extremely high traffic volumes, it is important to deploy a highly reliable system that can accurately differentiate between normal and malicious network traffic packets.

4.3. Comparison with Related Works. We compared the performance evaluation results (binary detection engine)

Table 10: Attack packet classification: classification accuracy.

Feature set/classifier	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C12
	In percentage										
All	99.07	99.22	94.05	9.64	16.91	84.79	98.06	89.16	79.69	79.62	99.27
Ens	98.8	98.96	77.75	9.46	20.6	50.78	97.4	83.35	75.99	84.79	98.92

All: all features; Ens: features selected by the HEFSM ensemble. C12 is the proposed multiclass detection engine.

Table 11: Training and testing times: binary and multiclass detection engines.

FS method	Binary engine training time (in sec)	Binary engine testing time (in sec)	Multiclass engine training time (in sec)	Multiclass testing time (in sec)
All features	22.00	8.5	15.00	0.49
HEFSM	15.00	3.1	5.80	0.17

Table 12: Tenfold crossvalidation results: classification accuracy.

Features	Binary detection engine		Multiclass detection engine	
	Maximum CA (%)	Minimum CA (%)	Maximum CA (%)	Minimum CA (%)
All features	99.46	98.65	99.06	97.91
HEFSM selected features	99.01	98.58	98.92	97.64

obtained in this study with the results reported in studies where the proposed intrusion detection model included an FS process (as reviewed earlier); seven of these studies used a hybrid ensemble approach. It can be seen in Table 13 that the performance of the binary detection engine using HEFSM (99.55%) compares well with the best performing models in terms of classification accuracy: it outperformed all but four of the reviewed models, including five of the hybrid ensemble models (the highest accuracy of 99.91% was reported in [37] where a filter FS algorithm was used).

The FAR metric value of 0.45% (calculated as the average of FNR and FPR) indicated that the binary engine performed with a very low balanced error rate and was better compared to the three other studies which used the same metric. The FPR metric value of.0.45% was the third best among the seven studies that reported it, with the lowest in [37] (0.28%). An extensive comparison regarding the other evaluation metrics used cannot be done due to the lack of reported data. However, the performance of the binary detection engine using HEFSM is comparable to the best performing models, where results are reported.

We also compared our work to some recent results obtained using FS methods over the NSL KDD and KDD'99 datasets that were not part of the initial literature review. Similar to our approach, Krishnaveni et al. [23] applied a hybrid approach to both selecting features and detecting malicious network traffic. The experiment on the NSL KDD set extracted similar number of features (ten), but the reported classification accuracy was lower (96.06%). The ensemble applied majority voting with four classifiers (DT, SVM, Naïve Bayes, and Logistic Regression) but did not include RF which was found to be the best performing classifies in our experiments (Table 9) and also in [61].

In another recent work that used the benchmarking dataset NSL KDD, the authors applied a nature inspired metaheuristic algorithm (Firefly) to select optimal features and tested the model with NN, AdaBoost, and RF [62]. However, even the best accuracy achieved was very low (below 80%). Much better results were achieved in [63] where the authors applied a DL approach (Restricted Bolzmann Machines) to build the detection model; the Random Harmony algorithm was used for feature selection (using the KDD'99 dataset). The accuracy achievement was very high (99.92%), with a very low FPR (0.03%). However, the model detected only one type of attack (DDoS).

It is also worth noting that ML-based cloud IDS operating at the hypervisor layer may be used in conjunction with a non-ML IDS that can ensure normal network activities during prolonged DDoS attacks. For example, Tan et al. [64] proposed a framework that applies time delay control theory to monitor network activities and distinguish normal network traffic from DDoS attacks; the framework involves extracting network traffic data attributes relevant to the monitoring process. While the framework is somewhat similar to the one proposed in this study, Tan et al.'s framework has the additional functionality of being able to automatically reduce the effect of the DDoS attacks. The choking technique that is used to stabilize the network under attack may potentially lead to increased computational overhead in the case of false alarms. However, the ability to maintain system stability against DDoS attacks is extremely important in cloud environments where service availability is a QoS requirement.

Table 13: Comparison with related work.

Source	FS method (type)	ML algorithm (s)	CA	ER	PR	RC	FPR	FNR	FAR (FNR + FPR)/2
Proposed binary detection engine using HEFSM	Hybrid	HEFSM (DT, RF, KNN)	99.55	0.45	99.52	99.54	0.44	0.46	0.45
Luo and Xia [38]	Filter	SVM	94.36	—	—	—	—	—	—
Eesa et al. [40]	Wrapper	DT	91.99	—	—	91	3.917	—	—
Bhattacharya and Sel-vakumar [51]	Hybrid	BN, J48, KM	96.46	—	—	—	—	—	—
Osanaiye et al. [1]	Filter	DT	99.67	—	—	99.76	0.42	—	—
Ambusaidi et al. [37]	Filter	SVM	99.91	—	—	98.76	0.28	—	—
Belouch et al. [44]	Wrapper	REPTree	89.85	—	—	—	—	—	—
Moustafa and Slay [47]	Hybrid	EM, LR, NB	82.1	—	—	—	—	—	17.5
Mogal et al. [48]	Hybrid	NB, LR	99.82	—	—	—	—	—	—
Idhammad et al. [39]	Filter	ANN	99.2	—	—	—	—	—	0.02
Vijayanand et al. [45]	Wrapper	SVM	98.95	—	—	—	4.1	18.5	11.3
Aljawarneh et al. [46]	Wrapper	NB, J48, RT	99.81	—	—	—	—	—	—
Anwer et al. [50]	Hybrid	J48, NB	88	—	—	—	—	—	—
Sun et al. [41]	Wrapper	SVM	91.68	—	—	—	—	—	—
Pham et al. [42]	Wrapper	EC (J48)	84.25	—	—	—	2.79	—	—
Besharati et al. [43]	Wrapper	EC (DT, LDA, ANN)	97.51	—	—	—	—	—	—
Mohammadi et al. [49]	Hybrid	DT	95.03	—	—	95.23	1.65	—	—
Maza and Touahria [15]	Hybrid	NB, MLP, SVM, KNN, DT	99.11	—	—	—	—	—	—
Tama et al. [22]	Hybrid	RoF, CF	85.8	—	—	88	—	—	—

DT: Decision Tree; SVM: support vector machine; REPTree: reduced error pruning tree algorithm; EM: expectation-maximisation clustering; LR: logistic regression; BN: bayesian networks; NB: Naïve Bayes; KM: K means learning algorithm; ANN: artificial neural networks; J48: J48 decision tree; EC: ensemble classifiers, bagging, or boosting; CF: conjunctive rule; LDA: linear discriminant analysis; MLP: multilayer perceptron; RoF: rotation forest.

5. Conclusion

In this paper, we proposed an ML-based intrusion detection and attack identification framework for CC environments (CCAID) that comprises two main processes: an FS process and a two-stage attack detection and classification process. The FS process utilises a new hybrid ensemble method for selecting network traffic features (HEFSM).

The proposed FS method was tested on the NS KDD dataset. The binary detection engine and the multiclass detection engines were trained independently on the same dataset using the set of features selected by HEFSM. The experimental results showed improved performance and stability. We achieved an overall classification accuracy of 99.55% and 98.92% for the binary and multiclass detection engine, respectively. The FAR of 0.45% (calculated as the average of FNR and FPR) indicated that the binary engine performed with a very low balanced error rate. As shown, the results compared very favourably with earlier related works. The classification accuracy of both detection engines is also better than the classification accuracy achieved in [23] for the NSL KDD dataset, where the authors applied an ensemble FS approach. From a more general perspective, in terms of classification accuracy, of the twelve algorithms in [52], our binary classification and multiclass engines outperformed ten and seven of them, respectively.

The set of features on which the ML mode was trained includes the features F1, F3, F5, F6, and F23 (see Table 1 for feature descriptions). As mentioned earlier, these features pertain to the traffic amongst virtual machines; therefore, the proposed IDS will be able to detect attacks that occur at the hypervisor communication layer and may be launched by an external attacker or by a malicious insider. Even though the model was built using a conventional-based dataset, it can be adapted to detect intrusion in both the cloud network and the hypervisor-based network of the cloud computing environment. The eleven selected features are relevant to the detection of attacks in both conventional and cloud environments and, therefore, the proposed model is suitable for implementation in a distributed environment.

These outcomes indicate that the proposed approach to feature selection and attack recognition and classification has the potential to improve significantly the performance of an IDS operating in a CC environment as it will be able to analyse heavy CC network traffic in real time and detect attack packets with high detection accuracy and a low false alarm rate. Directions for further research include validating the model with other network intrusion datasets, including recently-made-available datasets featuring attack packets for threats such as ransomware [34] and implementing it to manage intrusion detection in an experimental CC environment.

Data Availability

This research used a dataset named NSL KDD which was obtained from the public repository of the University of New Brunswick (https://www.unb.ca/cic/datasets/index.html).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 130, 2016.

[2] F. Nzanywayingoma and Y. Yang, "Efficient resource management techniques in cloud computing environment: a review and discussion," *International Journal of Computers and Applications*, vol. 41, no. 3, pp. 165–182, 2019.

[3] X. Li, L. Tan, and F. Li, "Optimal cloud resource allocation with cost performance tradeoff based on Internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6876–6886, 2019.

[4] A. K. Tyagi, M. M. Nair, S. Niladhuri, and A. Abraham, "Security, privacy research issues in various computing platforms: a survey and the road ahead," *Journal of Information Assurance & Security*, vol. 15, no. 1, 2020.

[5] U. Cavusoglu, "A new hybrid approach for intrusion detection using machine learning methods," *Applied Intelligence*, vol. 49, no. 7, pp. 2735–2761, 2019.

[6] I. Mukhopadhyay, "Cyber threats landscape overview under the new normal," *ICT Analysis and Applications*, pp. 729–736, 2022.

[7] U. A. Butt, M. Mehmood, S. B. H. Shah et al., "A review of machine learning algorithms for cloud computing security," *Electronics*, vol. 9, no. 9, p. 1379, 2020.

[8] S. Shamshirband, M. Fathi, A. T. Chronopoulos, A. Montieri, F. Palumbo, and A. Pescapè, "Computational intelligence intrusion detection techniques in mobile cloud computing environments: review, taxonomy, and open research issues," *Journal of Information Security and Applications*, vol. 55, Article ID 102582, 2020.

[9] D. Chaudhary, K. Bhushan, and B. B. Gupta, "Survey on DDoS attacks and defense mechanisms in cloud and fog computing," *International Journal of E-Services and Mobile Applications*, vol. 10, no. 3, pp. 61–83, 2018.

[10] M. Haddadi and R. Beghdad, "DoS-DDoS: taxonomies of attacks, countermeasures, and well-known defense mechanisms in cloud environment," *EDPACS*, vol. 57, no. 5, pp. 1–26, 2018.

[11] A. Riaz, H. F. Ahmad, A. Kiani, J. Qadir, R. Rasool, and U. Younis, "Intrusion detection systems in cloud computing: a contemporary review of techniques and solutions," *Journal of Information Science and Engineering*, vol. 33, pp. 611–634, 2017.

[12] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine," *IEEE Transactions on Cloud Computing*, vol. 1, 2020.

[13] K. Srinivasan, A. Mubarakali, A. S. Alqahtani, and A. Dinesh Kumar, "A survey on the impact of DDoS attacks in cloud computing: prevention, detection and mitigation techniques," *Intelligent Communication Technologies and Virtual Mobile Networks*, Springer, Berlin, Germany, pp. 252–270, 2019.

[14] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "Intrusion detection techniques in cloud environment: a

survey," *Journal of Network and Computer Applications*, vol. 77, pp. 18–47, 2017.

[15] S. Maza and M. Touahria, "Feature selection for intrusion detection using new multi-objective estimation of distribution algorithms," *Applied Intelligence*, vol. 49, no. 12, pp. 4237–4237, 2019.

[16] I. Sumaiya Thaseen and C. Aswani 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, no. 4, pp. 462–472, 2017.

[17] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: a survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, Article ID 102767, 2020.

[18] A. B. Nassif, M. A. Talib, Q. Nasir, H. Albadani, and F. M. Dakalbab, "Machine learning for cloud security: a systematic review," *IEEE Access*, vol. 9, pp. 20717–20735, 2021.

[19] M. G. Raj and S. K. Pani, "A meta-analytic review of intelligent intrusion detection techniques in cloud computing environment," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, pp. 206–217, 2021.

[20] I. S. Thaseen, C. A. Kumar, and A. Ahmad, "Integrated intrusion detection model using chi-square feature selection and ensemble of classifiers," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3357–3368, 2019.

[21] A. Binbusayyis and T. Vaiyapuri, "Identifying and benchmarking key features for cyber intrusion detection: an ensemble approach," *IEEE Access*, vol. 7, pp. 106495–106513, 2019.

[22] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "Tse-ids: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.

[23] S. Krishnaveni, P. Vigneshwar, S. Kishore, B. Jothi, and S. Sivamohan, "Anomaly-based intrusion detection system using support vector machine," *Advances in Intelligent Systems and Computing*, Springer, Berlin, Germany, pp. 723–731, 2020.

[24] I. S. Thaseen, A. K. Cherukuri, B. Poorva et al., "Anomaly detection using XGBoost ensemble of deep neural network models," *Cybernetics and Information Technologies*, vol. 21, no. 3, pp. 175–188, 2021.

[25] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3211–3243, 2021.

[26] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Article ID 102419, 2020.

[27] W. Elmasry, A. Akbulut, and A. H. Zaim, "A design of an integrated cloud-based intrusion detection system with third party cloud service," *Open Computer Science*, vol. 11, no. 1, pp. 365–379, 2021.

[28] V. Kanimozhi and T. P. Jacob, "Artificial Intelligence outflanks all other machine learning classifiers in Network Intrusion Detection System on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," *ICT Express*, vol. 7, no. 3, pp. 366–370, 2021.

[29] Z. Liu, B. Xu, B. Cheng, X. Hu, and M. Darbandi, "Intrusion detection systems in the cloud computing: a comprehensive and deep literature review," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 4, 2022.

[30] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: a new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[31] V. R. Balasaraswathi, M. Sugumaran, and Y. Hamid, "Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms," *Journal of Communications and Information Networks*, vol. 2, no. 4, pp. 107–119, 2017.

[32] R. C. T. de Souza, C. A. de Macedo, L. S. Coelho, J. Pierezan, and V. C. Mariani, "Binary coyote optimization algorithm for feature selection," *Pattern Recognition*, vol. 107, Article ID 107470, 2020.

[33] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo, "Survey and taxonomy of feature selection algorithms in intrusion detection system," in *Proceedings of the International Conference on Information Security and Cryptology*, pp. 153–167, Busan, Korea, November 2006.

[34] M. Di Mauro, G. Galatro, G. Fortino, and A. Liotta, "Supervised feature selection techniques in network intrusion detection: a critical review," *Engineering Applications of Artificial Intelligence*, vol. 101, Article ID 104216, 2021.

[35] P. Bermejo, L. de la Ossa, J. A. Gámez, and J. M. Puerta, "Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 35–44, 2012.

[36] W. Wang, Y. He, J. Liu, and S. Gombault, "Constructing important features from massive network traffic for lightweight intrusion detection," *IET Information Security*, vol. 9, no. 6, pp. 374–379, 2015.

[37] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[38] B. Luo and J. Xia, "A novel intrusion detection system based on feature generation with visualization strategy," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4139–4147, 2014.

[39] M. Idhammad, K. Afdel, and M. Belouch, "Dos detection method based on artificial neural networks," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 4, pp. 465–471, 2017.

[40] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Systems with Applications*, vol. 42, no. 5, pp. 2670–2679, 2015.

[41] S. Sun, Z. Ye, L. Yan, J. Su, and R. Wang, ""Wrapper feature selection based on lightning attachment procedure optimization and support vector machine for intrusion detection," in *Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*, IEEE, Piscataway, NJ, USA, September 2018.

[42] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, and H. F. M. Lahza, "Improving performance of intrusion detection system using ensemble methods and feature selection," in *Proceedings of the Australasian Computer Science Week Multiconference*, p. 2, Brisband, Queensland, Australia, January 2018.

[43] E. Besharati, M. Naderan, and E. Namjoo, "Lr-hids: logistic regression host-based intrusion detection system for cloud environments," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–24, 2018.

[44] M. Belouch, S. El Hadaj, and M. Idhammad, "A two-stage classifier approach using Reptree algorithm for network intrusion detection," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, pp. 389–394, 2017.

[45] R. Vijayanand, D. Devaraj, and B. Kannapiran, “Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection,” *Computers & Security*, vol. 77, pp. 304–314, 2018.

[46] 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.

[47] N. Moustafa and J. Slay, “A hybrid feature selection for network intrusion detection systems: central points,” 2017, https://arxiv.org/abs/1707.05505.

[48] D. G. Mogal, S. R. Ghungrad, and B. B. Bhusare, “Nids using machine learning classifiers on unsw-nb15 and kddcup99 datasets,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 6, no. 4, pp. 533–537, 2017.

[49] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, and H. Karimipour, “Cyber intrusion detection by combined feature selection algorithm,” *Journal of information security and applications*, vol. 44, pp. 80–88, 2019.

[50] H. M. Anwer, M. Farouk, and A. Abdel-Hamid, “A framework for efficient network anomaly intrusion detection with features selection,” in *Proceedings of the 2018 9th International Conference on Information and Communication Systems (ICICS)*, pp. 157–162, IEEE, Irbid, Jordan, 2018.

[51] S. Bhattacharya and S. Selvakumar, “Multi-measure multi-weight ranking approach for the identification of the network features for the detection of dos and probe attacks,” *The Computer Journal*, vol. 59, no. 6, pp. 923–943, 2016.

[52] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, “Building an efficient intrusion detection system based on feature selection and ensemble classifier,” *Computer networks*, vol. 174, Article ID 107247, 2020.

[53] A. A. Cárdenas, J. S. Baras, and K. Seamon, “A framework for the evaluation of intrusion detection systems,” in *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P’06)*, p. 15, IEEE, Los Alamitos, CA, USA, May 2016.

[54] H. P. Vinutha and B. Poornima, “An ensemble classifier approach on different feature selection methods for intrusion detection,” in *Information Systems Design and Intelligent Applications. Advances in Intelligent Systems and Computing*, V. Bhateja, B. Nguyen, N. Nguyen, S. Satapathy, and D. N. Le, Eds., Springer, Singapore, 2018.

[55] A. Aldribi, I. Traore, B. Moa, and O. Nwamuo, “Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking,” *Computers & Security*, vol. 88, Article ID 101646, 2020.

[56] I. F. Kilincer, F. Ertam, and A. Sengur, “Machine learning methods for cyber security intrusion detection: datasets and comparative study,” *Computer Networks*, vol. 188, Article ID 107840, 2021.

[57] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. K. A. A. Khan, “Performance analysis of machine learning algorithms in intrusion detection system: a review,” *Procedia Computer Science*, vol. 171, pp. 1251–1260, 2020.

[58] A. Chiche and M. Meshesha, “Towards a scalable and adaptive learning approach for network intrusion detection,” *Journal of Computer Networks and Communications*, vol. 2021, Article ID 8845540, 2021.

[59] A. Singh, K. Chatterjee, and S. C. Satapathy, “An edge based hybrid intrusion detection framework for mobile edge computing,” *Complex & Intelligent Systems*, vol. 2021, pp. 1–28, 2021, https://link.springer.com/journal/40747/online-first?page=3.

[60] S. Pal, T. Xu, T. Yang, S. Rajasekaran, and J. Bi, “Hybrid-DCA: a double asynchronous approach for stochastic dual coordinate ascent,” *Journal of Parallel and Distributed Computing*, vol. 143, pp. 47–66, 2020.

[61] S. Frarhat arhat, M. Abdelkader, A. Meddeb-Makhlouf, and F. Zarai, “Comparative study of classification algorithms for cloud IDS using NSL-KDD dataset in WEKA,” in *Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC)*, Limassol, Cyprus, June 2020.

[62] P. Ghosh, D. Sarkar, J. Sharma, and S. Phadikar, “An intrusion detection system using modified-firefly algorithm in cloud environment,” *International Journal of Digital Crime and Forensics*, vol. 13, no. 2, pp. 77–93, 2021.

[63] M. Mayuranathan, M. Murugan, and V. Dhanakoti, “Best features based intrusion detection system by RBM model for detecting DDoS in cloud environment,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3609–3619, 2021.

[64] L. Tan, K. Huang, G. Peng, and G. Chen, “Stability of TCP/AQM networks under DDoS attacks with design,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3042–3056, 2020.