Wiley
Journal of Electrical and Computer Engineering
Volume 2024, Article ID 7302909, 21 pages
https://doi.org/10.1155/2024/7302909

WILEY

Research Article

Network Intrusion Detection Using Knapsack Optimization, Mutual Information Gain, and Machine Learning

Akindele S. Afolabi[1] **and Olubunmi A. Akinola**[2]

[1]*Department of Electrical and Electronics Engineering, Faculty of Engineering and Technology, University of Ilorin, Ilorin, Nigeria*
[2]*Department of Electrical and Electronics Engineering, College of Engineering, Federal University of Agriculture, Abeokuta, Nigeria*

Correspondence should be addressed to Olubunmi A. Akinola; akinolaoa@funaab.edu.ng

Received 1 June 2023; Revised 11 March 2024; Accepted 29 April 2024; Published 1 June 2024

Academic Editor: Kusum Verma

The security of communication networks can be compromised through both known and novel attack methods. Protection against such attacks may be achieved through the use of an intrusion detection system (IDS), which can be designed by training machine learning models to detect cyberattacks. In this paper, the KOMIG (knapsack optimization and mutual information gain) IDS was developed to detect network intrusions. The KOMIG IDS combined the strengths of optimization and machine learning together to achieve a high intrusion detection performance. Specifically, KOMIG IDS comprises a 2-stage feature selection procedure; the first was accomplished with a knapsack optimization algorithm and the second with a mutual information gain filter. In particular, we developed an optimization model for the selection of the most important features from a network intrusion dataset. Then, a new set of features was synthesized from the selected features and combined with the selected features to form a candidate features set. Next, we applied an information gain filter to the candidate features set to prune out redundant features, leaving only the features that possess the maximum information gain, which were used to train machine learning models. The proposed KOMIG IDS was applied to the UNSW-NB15 dataset, which is a well-known network intrusion evaluation dataset, and the resulting data, after optimization operation, were used to train four machine learning models, namely, logistic regression (LR), random forest (RF), decision tree (DT), and K-nearest neighbors (KNN). Simulation experiments were conducted, and the results revealed that our proposed KNN-based KOMIG IDS outperformed comparative schemes by achieving an accuracy score of 97.14%, a recall score of 99.46%, a precision score of 95.53%, and an F1 score of 97.46%.

1. Introduction

Network size and real-time traffic have gotten more sophisticated and vast as a result of the rapid development and widespread adoption of 5G, IoT, cloud computing, and other technologies. Cyberattacks have also developed into a broader range of complex operations, posing significant difficulties to the security of cyberspace [1–3]. A malicious tenant, for example, can stay in a virtual machine (VM), successfully hijack other VMs in the cloud, and then use the puppet machines to distribute malicious software or perform a distributed denial of service (DDoS) attack [4, 5]. The network intrusion detection system (NIDS) is usually the second line of defense behind the firewall and is responsible for identifying malicious network attacks, providing real-time monitoring, implementing dynamic security measures, and developing strategies [1]. NIDSs play an important role in cybersecurity by alerting security managers about malicious activities such as distributed denial-of-service (DDoS), port scans, SQL injection attacks, and others [6]. These activities could degrade the overall network performance and may result in network failures and sometimes endanger human lives [7].

Network intrusion can be defined as any illegal action that compromises the confidentiality, integrity, or availability (CIA) of data inside a network [8–10]. Network intrusion may drastically hurt enterprises by causing monetary losses, reputational damage, legal liabilities, and the loss of sensitive information [11]. Intrusion detection involves the task of observing, analyzing, and identifying activities that compromise a network's security policy [12]. In addition to firewalls, security managers often use password protection systems, encryption methods, and access restriction approaches to secure networks from intrusion [8, 13]. These methods do not, however, go far enough to safeguard the system. To monitor network traffic and identify malicious attempts, many administrators, therefore, choose to utilize intrusion detection systems (IDSs) [8, 14–16].

Network-based IDS and host-based IDS are the two most often used forms of IDS [17, 18]. By examining end-user behavior, an NIDS keeps an eye on the network traffic and looks for suspicious activities. IDSs use two different categories of detection techniques [19]: signature-based [20, 21] and anomaly-based techniques [22–26]. On the one hand, signature-based IDS (or host-based IDS) recognizes patterns (i.e., signatures) in IDS to detect attacks. Anomaly-based IDS, on the other hand, are built on the premise of discriminating between abnormal and regular network events. In this method, the behavior profiles of the system's users are first identified, and behaviors that deviate from normal behavior are classified as abnormal [27]. Subsequently, timely countermeasures can be taken to ensure that the security of the communication network is not compromised [28].

Machine learning (ML) techniques for improved AIDS (MLAIDS) have been considered recently [9, 17, 22]. These algorithms categorize the processed data into normal and abnormal classifications to assess the state of the network. They develop and evaluate AIDS for identifying attacks, and they gauge its flexibility in using various datasets by looking at its false alarm rate and accuracy. However, the bulk of these datasets exhibits a severe cybersecurity imbalance, with the majority (98%) of these datasets being classified as normal and the remaining (2%) as attacks [22, 29].

The existing public datasets for intrusion detection have both benefits and drawbacks [29]. Some of their flaws include (1) failure to reflect contemporary network threats; (2) an augmented minority dataset possessing restricted features to reproduce the nature of network attacks. As a result, real-time capture is required; (3) many of the existing public datasets have not been tested against the recently released domain name system over HTTPS (DoH) protocol. Most published datasets exhibit conventional DDoS attacks and variations; (4) many of the public datasets do not contain layer 3 information in their column characteristics; and (5) a few research works reveal inconsistent analysis and classification findings.

Our contributions to this paper are as follows:

(1) We first defined an optimization problem for the selection of features from a network intrusion dataset.

(2) Then, we transformed the optimization problem into an easy-to-solve form and developed an algorithm for solving it.

(3) Next, we described a procedure for synthesizing new features from the features selected by the optimization model. The synthesized features were combined with the selected features to form the candidate feature set.

(4) Finally, we applied an information gain filter to the candidate features set to pick features that possess high information gain. These features were used to train machine learning models for network intrusion detection.

2. Literature Review

In this study, optimization and machine learning tools were applied to detect the intrusion of a communication network. The decision to combine optimization and machine learning was inspired by the works of Leyva–Pupo et al. [30] in which the degradation of the quality of service (QoS) of a 5G communication network was prevented through the application of machine learning and optimization techniques. The authors used machine learning methods to forecast the QoS status at the next timestamp, which determines whether optimization regarding service latency satisfaction is required. If it is required, optimization algorithms which include integer linear programming (ILP) and a heuristic algorithm are executed. A major difference between their setup and ours is in the order of the optimization and machine learning blocks. In their work, the output of the machine learning block feeds the optimization block whereas in ours (as will be seen later), the output of the optimization block feeds the machine learning block.

Classical optimization and nature-inspired optimization methods have attracted significant attention in the research community. Classical methods such as linear programming, mixed integer programming, quadratic programming, polynomial goal programming, and others are more useful in optimization problems that have certain mathematical properties such as differentiability, continuity, and convexity. Nature-inspired optimization methods, on the other hand, thrive even when these mathematical properties are not satisfied. Examples of nature-inspired optimization algorithms include genetic algorithm [31], artificial bee colony algorithm [32], particle swarm algorithm [33, 34], farmland fertility algorithm [35], mountain gazelle optimizer [36], fruit fly optimization algorithm [37], firefly algorithm [34], simulated annealing [38], flower pollination algorithm [39], whale optimization algorithm [40], manta–ray foraging optimizer [41], African vulture optimization algorithm [42], grasshopper optimization algorithm [43], moth flame algorithm [44], biogeography-based algorithm [45], imperialist competitive algorithm [46], grey wolf optimizer [47], Harris hawks optimization algorithm [48], gravitational search algorithm [49], slime mould algorithm [50], and bat algorithm [51].

Saheed et al. [52] proposed a new strategy that combines the bat metaheuristic algorithm with the residue number system (RNS) for feature selection in intrusion detection systems. In particular, they used a mix of RNS and the bat algorithm to remove extraneous features while principal component analysis (PCA) was used to perform feature extraction. The bat algorithm's longer training and testing periods were addressed by using RNS to reduce processing time. Traffic classification was done using naive Bayes and KNN. Simulation results indicate that combining RNS with the bat algorithm leads to significantly improved detection accuracy, precision, and F scores and, in addition, a reduced processing time.

Gharehchopogh [48] proposed variations of the Harris hawk optimization (HHO) algorithm which included Improved HHO opposition-based learning (IHHOOBL), improved HHO Lévy flight (IHHOLF), and improved HHO chaotic map (IHHOCM). These algorithms were designed such that exploitation and exploration were balanced out and they were tested using normalized mutual information (NMI) and modularity criteria on 12 distinct datasets. The results showed that the IHHOOBL approach outperforms the IHHOLF and IHHOCM methods in terms of detection accuracy. Furthermore, the researchers compared IHHOOBL with an existing cutting-edge algorithm, which has outperformed up to 7.8 percent.

The researchers in [53] suggested the application of a multiagent system along with metaheuristic algorithms (MAMAH) for the detection of e-mail spam. The approach considered several agents as independent actors and each attempted to accomplish its objectives while competing and working with other agents to attain shared objectives. The technique proved to be highly effective in solving high-dimensional optimization problems and outperformed existing metaheuristic algorithms in terms of precision in detecting spam emails.

BFFA (binary farmland fertility algorithm), a binary variant of the farmland fertility algorithm (FFA), was implemented in [35] for feature selection in IDS. In BFFA, a V-shaped function was used to shift the FFA processes to the binary space, thereby, changing the continuous location of the solutions in the FFA algorithm to a binary mode. To achieve a fast and robust IDS, a hybrid solution that incorporates classifiers and the BFFA was developed. The BFFA method was compared with existing classifiers such as K-nearest neighbor (KNN), support vector machine (SVM), decision tree (DT), random forest (RF), AdaBoost (ADA_BOOST), and Naive–Bayes (NB), and it was shown to outperform them in terms of accuracy, precision, and recall; It also has a shorter run time in the FS operation.

In [42], the enhanced African vultures optimization algorithm (AVOA) was used in multithreshold picture segmentation, which used three binary thresholds (Kapur's entropy, Tsallis entropy, and Ostu's entropy). To enhance AVOA performance, the quantum rotation gate (QRG) mechanism boosted population variety in optimization phases and optimized local trap escapes. The association strategy (AS) method was used to find and search for optimum solutions more quickly. Because the AVOA algorithm concentrates on the exploration phase almost entirely in the first half of the iterations, these two processes boosted the variety of production solutions in all optimization phases. So, using this strategy, it is feasible to ensure a broad range of solutions while avoiding the local optimum trap. Standard criteria and datasets were used to assess the proposed algorithm's performance, which was then compared to existing optimization techniques. The experimental findings from the AVOA showed that it has an excellent and substantial performance gain.

Paraneswari et al. [54] used the optimization-enabled deep learning model rat swarm hunter prey optimization-deep maxout network (RSHPO-DMN) method for intrusion detection. They applied Z score data normalization for data preprocessing and then translated the data into a useable format by considering chord distance. The translated data were retrieved using the convolutional neural network (CNN) feature, and the extracted feature was translated into vector format for the network intrusion detection procedure. The DMN was used for intrusion detection, while the RSHPO model was used to improve the intrusion detection rate of the classifier. Results showed that the RSHPO-DMN model achieved a detection accuracy of 90.88%.

The authors of [55] surveyed several applications of quantum computing (QC) in metaheuristics. The paper also provided a taxonomy of quantum-inspired metaheuristic algorithms in optimization issues and analyzed their applications in science and engineering. The slime mould algorithm (SMA) is studied in [50], and the study covered four topics: hybridization, progress, modifications, and optimization. The use of SMA in the aforementioned topics is stated in the paper to be 15%, 36%, 7%, and 42%, respectively. According to the authors, SMA has been routinely used to solve optimization problems, which expectedly will be beneficial to engineers, professionals, and academic scientists.

The article in [31] proposed an enhancement of the cuckoo search optimization (CSO) technique with a genetic algorithm (GA) technique for community detection in complex networks. In the paper, GA operators were used dynamically to improve the speed and accuracy of the CSO. The population size was continually modified depending on the quantity of exploration and exploitation. The modularity objective function and NMI were used as optimization functions. The approach was evaluated with GA, artificial bee colony (ABC), grey wolf optimizer (GWO), and CSO, with varied iterations in modularity and NMI criterion. The findings revealed that the approach outperformed existing algorithms by an average of 54% in modularity and 88% in NMI. It is evident from the papers on optimization discussed so far that the application of optimization algorithms to real-life problems results in improved system performance.

Just as optimization is highly relevant in improving system performance, the field of machine learning has also proven to be highly relevant in achieving the same feat. Machine learning applications are rapidly penetrating our everyday lives and have been applied in the health sector [56], the banking sector [57], the agriculture sector [58], the power sector [59], and the education sector [60]. The successes of machine learning and the rising complexity and severity of security threats on networks have prompted security researchers to adopt various machine learning technologies to safeguard businesses' data and reputation.

The approaches used include supervised learning, unsupervised learning, reinforcement learning, and deep learning [61–64].

In [64], an intrusion detection and prevention system (IDPS) that used machine learning and software-defined networking (SDN) to identify and mitigate IEC 60870-5-104 attacks was presented. The intrusion detection system was based on a classification and regression tree (CART) classifier that used TCP/IP network flow data as well as IEC 60870-5-104 payload flow statistics. The evaluation results indicated that IDPS achieved 81.73% accuracy.

In [65], an efficient wrapper feature selection method, based on the Firefly algorithm, was proposed for selecting important features in wireless sensor network (WSN) traffic. Using a minimum-maximum normalization strategy, traffic data was preprocessed and the firefly algorithm was then utilized for feature dimensionality reduction. Subsequently, the C5.0 algorithm was used for classification, and simulation results show that this approach yields a significantly high detection accuracy.

Kasongo, in [66], developed an IDS that used GA for feature selection and the random forest (RF) model in the GA fitness function. Classifiers used in the intrusion detection procedures included the RF, linear regression (LR), Naive Bayes (NB), decision tree (DT), extra-trees (ET), and extreme gradient boosting (XGB). The UNSW-NB15 was used to evaluate the performance of the IDS, and the results showed that the GA-RF obtained a detection accuracy of 87.61%.

In the research work presented in [67], a wrapper-based feature selection technique, Tabu search-random forest (TS-RF), was used for intrusion detection. The TS-RF wrapper made use of the Tabu search metaheuristic algorithm for feature finding and weighting, as well as RF as a learning method. The evaluation results indicated that TS-RF achieved an intrusion detection accuracy of 83.12%.

A GA-based RF model was developed in [24] to categorize normal and abnormal network traffic for intrusion detection. The GA was used to choose optimal values for two RF parameters. These parameters were the minimum number of instances for each split and the number of trees in the forest, which helped to optimize the RF classifier and improve anomaly classification and intrusion detection accuracy. Results showed that this method achieved an intrusion detection accuracy of 86.70%.

Using the deep learning approach, the authors of [26] developed an anomaly-based IDS solution for IoT networks. A filter-based feature selection deep neural network (DNN) model with strongly correlated features was described in particular. The model was fine-tuned using different hyperparameters. To tackle class imbalance difficulties in the used dataset, generative adversarial networks (GANs) were applied to generate synthetic data of minority attacks. The model's accuracy was found to be 91%.

The study in [68] developed an IDS by using the XGBoost method for feature selection in combination with several ML approaches such as artificial neural network (ANN), KNN, DT, logistic regression (LR), and SVM. The binary and multiclass classification parameters were taken into account in the study. Results showed that XGBoost-ANN, XGBoost-KNN, XGBoost-DT, XGBoost-LR, and XGBoost-SVM achieved intrusion detection accuracies of 84.39%, 84.46%, 90.85%, 77.64%, and 60.89%, respectively.

In [69], the authors presented a dependable IDS model. Their approach relied on the deep transfer learning-based ResNet model, and their major contributions included effective attribute selection, which improved the ability of the design to identify normal and attack situations while making use of only a small size of labeled data. Results showed that the dependable IDS model achieved a detection accuracy of 87%. Some existing works on IDS are summarized in Table 1.

It can be observed in Table 1 that the detection accuracies of most of the existing works on IDS need improvement with many of them having values of 90% or less. To improve on the intrusion detection accuracy, we propose a two-stage feature selection technique such that in the first stage, a knapsack optimization-based algorithm selects a set of high-quality features which are synthesized into new features through a combination process. The synthesized and original features are merged and presented to a mutual information gain filter which constitutes the second stage of the feature selection procedure. The filter passes the most important features to machine learning algorithms and models are developed and trained to detect malicious network intrusion traffic.

3. Materials and Methods

In this section, we present our proposed network intrusion detection system, and its framework is illustrated in Figure 1.

As will be detailed later, it relies on the knapsack optimization technique for initial feature selection and on the mutual information gain method for final feature selection. It is, therefore, named KOMIG (knapsack optimization and mutual information gain) IDS. We assume that the KOMIG IDS operates at layer 2 and detects traffic patterns that deviate from the expected behavior. This category of traffic is subsequently classified as intrusion traffic by some machine learning algorithms.

To enable this capability in KOMIG, annotated data comprising intrusion and normal traffic were used to train some machine learning models. Once the models have been trained and ascertained to be generalizing properly, they gain the capability of classifying new traffic as either normal traffic or intrusion traffic. However, before data can be presented to the machine learning algorithm, it has to be prepared in the format that the algorithm can operate on. Some steps are involved as illustrated in Figure 2.

3.1. Data Preprocessing. Preprocessing operation transforms raw data into a format that is appropriate for machine learning algorithms. Data preparation involves normalizing the data and using data encoding to transform categorical information into numerical form.

Table 1: Some related works on network intrusion detection.

Ref.	Method used	Dataset	Accuracy (%)	Limitation
[64]	CART	Own dataset	81.73	Low accuracy
[66]	GA-RF	UNSW-NB15	87.61	Low accuracy
[67]	TS-RF	UNSW-NB15	83.12	Low accuracy
[24]	GA-RF	UNSW-NB15	86.70	Low accuracy
[26]	GAN-DNN	UNSW-NB15	91.00	Medium accuracy
[68]	XGBoost-DT	UNSW-NB15	90.85	Medium accuracy
[69]	P-ResNet	Own dataset	87.00	Low accuracy
[70]	Specification heuristics IDS	UNSW-NB15	91.77	Medium accuracy
[71]	Integrated classification	UNSW-NB15	84.83	Low accuracy
[27]	CNN + LSTM	UNSW-NB15	93.21	Medium accuracy
[72]	Ensemble	UNSW-NB15	93.88	Medium accuracy

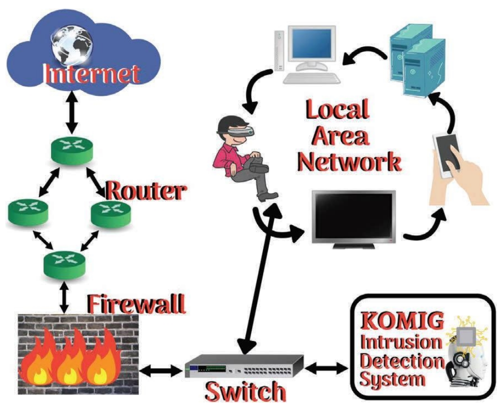


Figure 1: Framework of the proposed KOMIG Intrusion Detection System.

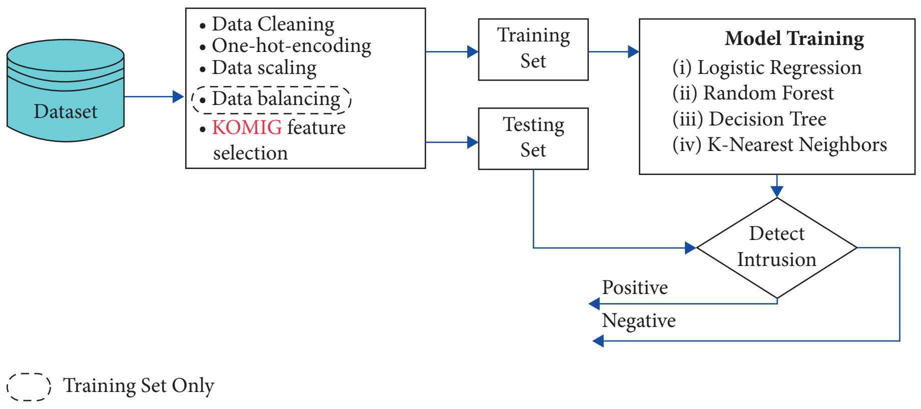


Figure 2: Workflow of the proposed intrusion detection system.

3.1.1. Data Cleaning. Data cleaning is used to correct some flaws in the dataset. This involves handling null samples, case conversion of text data, and encoding of text data which are described as follows:

(i) Samples containing nulls are either deleted or filled with the average or the most frequently occurring value as the case may be

(ii) Text-type data containing both lower and uppercase cases are unified by converting all textual values into a single case

(iii) Text-type data are converted into numbers using one-hot-encoding

3.1.2. One-Hot-Encoding. One-hot encoding represents category (i.e., text-based) variables as binary vectors. First, integer values are mapped to each categorical value. Then, each integer value is represented as an all-zero binary vector (except for the integer's index, which is indicated with a 1).

3.1.3. Normalization. Following the addition of numerical values to a dataset, the scale of each feature (i.e., column) is usually different from the next. This can lead to skewed results when fitted by a machine learning algorithm because features with larger scales might dominate the ones with lower scales. This can be avoided by normalizing the data. The normalization method used in this work was the z score. In this method, the column mean is subtracted from each data entry in the column, and the result is divided by the column standard deviation. The result obtained replaces the original data entry.

3.1.4. Data Balancing. A dataset with an unbalanced number of classes performs poorly when machine learning-based classifiers are used, especially when the ratio of the number of instances in the majority class to the minority class is high. In this case, it is necessary to lower the number of instances in the majority, or raise that of the minority, or do both to balance the dataset. In this work, the SMOTE algorithm was used to raise the number of instances of the minority class to achieve a balanced dataset.

3.2. Feature Processing and Selection. In this section, we present our proposed KOMIG-IDS scheme. Assuming a dataset comprises of M features and K sample instances such that $\mathscr{M} = \{1, 2, 3, \cdots, M\}$ and $\mathscr{K} = \{1, 2, 3, \cdots, K\}$. Given a selected sample instance k, a feature can be expressed as

$$\mathscr{F}_m = \left\{F_{1,m}, F_{2,m}, \cdots, F_{K,m}\right\}, \quad m = 1, 2, \cdots, M, \tag{1}$$

where M is the number of features in the dataset and $\mathscr{F}_m$ is the m^{th} feature in the dataset. Let the set of all features be denoted by

$$\mathscr{G} = \{\mathscr{F}_1, \mathscr{F}_2, \cdots, \mathscr{F}_M\}. \tag{2}$$

Then, the intrafeature variance of feature F_m can be computed as

$$v_m = \frac{\sum_{k=1}^{k}\left(F_{k,m} - \overline{F}_m\right)^2}{K} \quad \forall m \in \{1, 2, 3, \cdots, M\}, F_{k,m} \in \mathscr{F}_m, \tag{3}$$

where $F_{k,m}$ is the k^{th} entry in the m^{th} feature and $\overline{F}_m$ is the mean of all the K entries in the m^{th} feature. The scale of variance v_m of each feature may differ from one feature to another, but each feature's variance is on the same scale with its own average feature value $\overline{F}_m$. We, therefore, normalized each variance by dividing it by the feature average value.

$$v_m \longleftarrow \frac{v_m}{\overline{F}_m}. \tag{4}$$

It is desired to find a set of features $\mathscr{N} \subset \mathscr{G}$ of size D that maximizes the sum as

$$\underset{m}{\text{maximize}} \sum_{m \in \mathscr{M}, \mathscr{F}_m \in \mathscr{N}, |\mathscr{N}| = D} v_m. \tag{5}$$

By introducing a binary decision variable x_m, the optimization problem in (5) can be transformed into

$$\text{maximize} \sum_{m \in \{1,2,\cdots,M\}} x_m v_m, \tag{6}$$

s.t:

$$\sum_{m=1}^{M} x_m \leq D, \tag{7}$$

$$x_m \in \{0, 1\}, \quad m = 1, 2, \cdots, M. \tag{8}$$

Once the associated x_m of each feature is determined, the optimum feature set N can be obtained as

$$\mathscr{N} = \bigcup_{m=1, x_m \neq 0}^{M} \mathscr{F}_m. \tag{9}$$

The transformed optimization problem is a knapsack problem that has capacity D and equal weights. It is an easy-to-solve optimization problem whose solution can be obtained in polynomial time. The solution is presented in Algorithm 1. In Algorithm 1, the features $\mathscr{F}_m \in \mathscr{G}$ are sorted based on the descending order of magnitude of the values of their variances v_m. This is because the weights of all the variance values are 1; hence, sorting in descending order of magnitude and picking the first D variance values maximizes the objective function in (5). As can be observed in Algorithm 1, the corresponding set of features having the highest intrafeature variance values are compiled as $\mathscr{N}$.

Using the features set $\mathscr{N}$, another set of features is synthesized by systematically combining them to form a polynomial of a desired order. The higher the order, the higher the complexity of the resulting machine-learning model. How they are combined will be described shortly. The synthesized features are in 3 categories as follows:

(1) The bias (the value of 1.0)

(2) Each feature value is raised to a power for each degree, e.g., $(F_{k,1})^1$, $(F_{k,2})^2$, $(F_{k,3})^3, \ldots$

(3) A combination of pairs of feature elements in $\mathscr{N}$, e.g., $(F_{k,1}) \times (F_{k,2})$, $(F_{k,1}) \times (F_{k,3}), \ldots$

```
Procedure
  for $m = 1$ to $M - 1$
    for $j = m + 1$ to $M$
      if $(v_j < v_m)$
        swap$(v_j, v_m)$
        swap$(\mathcal{F}_j, \mathcal{F}_m)$
      end if
    end for
  end for
$\mathcal{N} = \bigcup_{\mathbf{m=1}}^{\mathbf{D}} \mathcal{F}_m$
End Procedure
```

ALGORITHM 1: Optimization-based features selection procedure.

The number of synthesized features is computed as

$$R = \frac{(D+E)!}{D! \times E!}, \tag{10}$$

where E is the order of the polynomial used in the synthesis operation and D is the cardinality of $\mathcal{N}$, which is the number of features in $\mathcal{N}$. For example, if the number of selected features is 3 and the element of the selected features at instance k are given as $\{a, b, c\}$, and the selected order of the polynomial is 2, then, $D = 3$ and $E = 2$. Hence, the number of synthesized features, R, is computed as

$$R = \frac{(3+2)!}{3! \times 2!} = \frac{5!}{3! \times 2!} = \frac{5 \times 4 \times 3!}{3! \times 2!} = \frac{5 \times 4}{2 \times 1} = 10. \tag{11}$$

Let the synthesized feature be denoted by $\mathcal{q}_r \in \mathcal{Q}$, where $\mathcal{Q}$ has a dimension $K \times R$. A synthesized feature q_r can be expressed as

$$\mathcal{q}_r = \{q_{1,r}, q_{2,r}, \cdots, q_{K,r}\}, \quad r = 1, 2, \cdots, R, \tag{12}$$

and the set of synthesized features at an instance k is expressed as $\{q_{k,1}, q_{k,2}, \cdots, q_{k,R}\}, k = 1, 2, \cdots, K$. For the case when the element of the selected features at instance k is given as $\{a, b, c\}$, the synthesized features are determined as follows:

$$\{q_{k,1}, q_{k,2}, \cdots, q_{k,R}\} = \{1, a, b, c, a^2, b^2, c^2, a \times b, a \times c, b \times c\}. \tag{13}$$

It can be observed in (13) that the selected features set $\mathcal{N}$ is duplicated in the synthesized feature set $\mathcal{Q}$; they are therefore expunged and $\mathcal{Q}$ is updated as

$$\mathcal{Q} \longleftarrow \mathcal{Q} - \mathcal{N}. \tag{14}$$

The synthesized features are merged with the original features, and the original features set is updated as

$$\mathcal{G} \longleftarrow \mathcal{G} \cup \mathcal{Q}. \tag{15}$$

The resulting total number of features after the synthesized features have been merged with the original features is

$$M \longleftarrow M + R - D. \tag{16}$$

Since the synthesized features that were added to the existing features have been carefully selected through an optimization process, their addition would increase the overall information gain of the dataset although some of them would be redundant and can dropped by a filter that is discussed next. Given feature $\mathcal{F}_m$ and the corresponding target class set Y, the information gain (IG) is computed as [73]

$$IG(\mathrm{Y}, \mathcal{F}_m) = H(Y) - H(Y \mid \mathcal{F}_m), \quad m \in \{1, 2, \cdots, M\}, \tag{17}$$

where

$$H(Y) = -\sum_{b=1}^{B} p(y_b) \log^2 p(y_b), \tag{18}$$

and

$$H(Y \mid \mathcal{F}_m) = -\sum_{z=1}^{Z} p(F_{z,m}) H(Y \mid \mathcal{F}_m = F_{z,m}), \quad m \in \{1, 2, \cdots, M\}. \tag{19}$$

In equation (17), $H(Y)$ is the entropy of the target class set, Y, while $H(Y \mid \mathcal{F}_m)$ is the conditional entropy of Y given the m^th feature, $\mathcal{F}_m$. The information gain of each feature is computed using (17), and W number of features that share the most mutual information gain with the target class set is selected for developing the machine learning model. The entropy of the target class, $H(Y)$, is computed in (18), in which B is the number of different classes in the target class set, and $p(y_b)$ is the probability of value y_b in the target class set. In this paper, $B = 2$ because there are only 2 target classes (i.e., the intrusion class and the normal class). The conditional entropy of the target class, $H(Y \mid \mathscr{F}_m)$, is computed in (19). The number of occurrences of an element $F_{z,m}$ in feature $\mathscr{F}_m$ is denoted by Z in (19), and $p(F_{z,m})$ is the probability of $F_{z,m}$ in $\mathscr{F}_m$. The block diagram of the feature selection steps in KOMIG IDS is illustrated in Figure 3.

A flowchart that summarizes the whole process is illustrated in Figure 4. In the flowchart, data traffic is preprocessed by making it undergo data cleaning, one-hot encoding, normalization, and data balancing operations. After the data preprocessing operation, the data traffic is passed to the KOMIG algorithm where the feature selection operation is performed. This operation involves initial feature selection through knapsack optimization, feature synthesis, and final feature selection using the information gain-based filter. The selected features of the data traffic are then presented to the trained machine learning models where the core operation of intrusion detection is performed. At this stage, a binary classification operation is performed and the traffic is either classified as normal traffic if no threat is detected in the traffic or as malicious traffic if threat is detected. If the data traffic is classified as normal traffic, it is allowed to pass through the network; and if on the other hand, it is classified as malicious traffic, it is dropped.

3.3. Evaluation Metrics. Some metrics, including TP, TN, FP, and FN, are taken into account when assessing the effectiveness of our proposed KOMIG IDS algorithm. In the context of this paper, TP stands for "true positive" and it refers to the number of intrusion traffic instances that were correctly classified; TN stands for "true negative" and refers to the number of normal traffic instances that were correctly classified; FP stands for "false positive" and denotes the number of normal traffic instances that were incorrectly classified as intrusion traffic instances; and FN stands for "false negative," and it refers to the number of intrusion traffic instances that were incorrectly classified as normal traffic. Using these, the following definitions of accuracy, recall, precision, and F1 score are provided as follows [7]:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{20}$$

$$\text{recall of intrusion class} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{21}$$

$$\text{recall of normal class} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \tag{22}$$

$$\text{precision of intrusion class} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{23}$$

$$\text{precision of normal class} = \frac{\text{TN}}{\text{TN} + \text{FN}}, \tag{24}$$

$$F1\text{-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{25}$$

In addition to these metrics, we used an indicator of performance for classification at different threshold levels known as the AUC-ROC (area under the curve-receiver operating characteristics) curve. The ROC curve is a two-dimensional graph that plots a true positive rate (TPR) on the y-axis against a false positive rate (FPR) on the x-axis. To demonstrate how classifiers discriminate between two classes, it draws lines across thresholds obtained while making binary classification judgments. The area under the curve (AUC) is a popular ROC curve statistic with values ranging from 0 to 1.

AUC greater than 0.5 indicates how well-trained classifiers assigned a higher probability to accurate predictions and a lower probability to wrong ones. A poorly trained classifier has an ROC curve with a diagonal line and an AUC value close to 0.5.

3.4. UNSW-NB15 Dataset. The UNSW-NB15 dataset was used in this paper to validate the proposed KOMIG IDS, and it is publicly available at [74]. The UNSW-NB15 dataset was used because it is a realistic and comprehensive dataset for

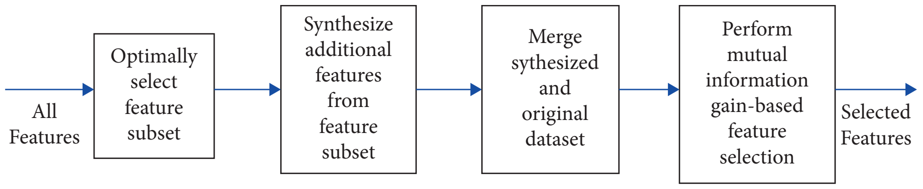

All Features
Optimally select feature subset
Synthesize additional features from feature subset
Merge sythesized and original dataset
Perform mutual information gain-based feature selection
Selected Features

Figure 3: Block diagram of KOMIG IDS.

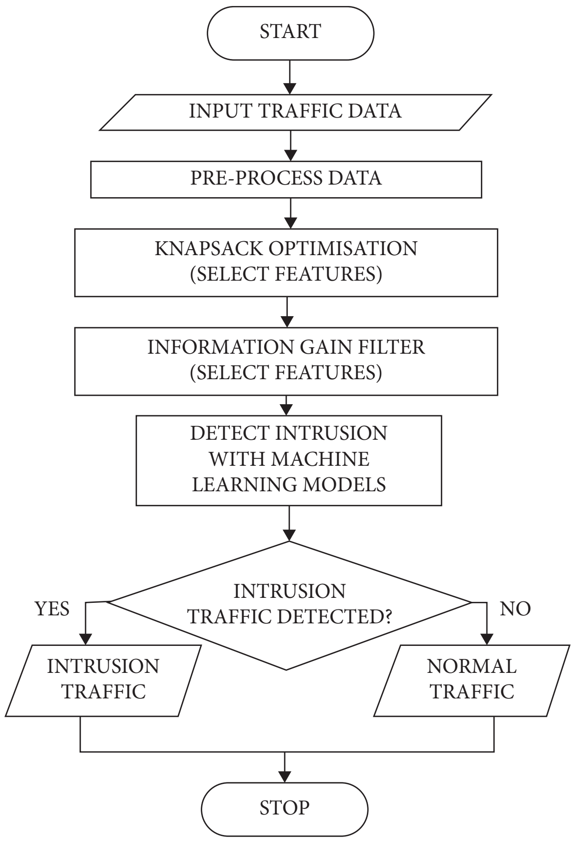

START
INPUT TRAFFIC DATA
PRE-PROCESS DATA
KNAPSACK OPTIMISATION (SELECT FEATURES)
INFORMATION GAIN FILTER (SELECT FEATURES)
DETECT INTRUSION WITH MACHINE LEARNING MODELS
INTRUSION TRAFFIC DETECTED?
YES
NO
INTRUSION TRAFFIC
NORMAL TRAFFIC
STOP

Figure 4: Flowchart of the proposed intrusion detection system.

network intrusion detection [75]. This dataset has 42 features, and it includes contemporary attacks in real networks. The Australian Centre for Cyber Security (ACCS) at UNSW in Canberra has provided the revised UNSW-NB15 dataset, taking into account the drawbacks of the previous dataset. The IXIA PerfectStorm program was used to generate a mix of recent malicious and benign network traffic characteristics. The dataset includes nine kinds of current cyberattacks labeled as Analysis, Backdoors, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms, as well as normal (benign) packets labeled as normal, which have been captured using the Tcpdump program [74, 76]. Details of the attack categories are provided in Table 2.

The most often used datasets, UNSW_NB15_training-set.csv (with 175,341 entries) and UNSW_NB15_testing-set.csv (with 82,332 entries) are partial datasets that are publicly accessible to assist researchers in developing IDSs. In the "label" column of the dataset, all the attacks are grouped as one class while the normal traffic represents the second class. Table 3 displays the sample sizes and percentages for each class of the training and testing sets.

3.5. Experimental Setup. The proposed network intrusion detection system was developed in Python programming language. It was executed on an Intel Core i3 CPU with 8 GB of RAM. The dataset used in this work was the UNSW-NB15 [74]. This dataset was selected because of its benefits over older standard datasets such as the KDD98, KDDCUP99, and NSLKDD datasets. The KDD98, KDDCUP99, and NSLKDD datasets suffer from a lack of contemporary cyberattacks, and normal traffic is skewed in such a manner that stealthy/spy attacks may easily be disguised as normal activity. Other shortcomings include an imbalance in the number of records from various kinds of traffic; noncomprehensive training sets and these training sets do not reflect every attack available in the testing set.

Table 2: The traffic categories of the UNSW-NB15 dataset and their description.

Traffic type	Description
Normal	A threat-free traffic
Fuzzer	An attack in which the attacker tries to identify security flaws in a software, operating system, or network by flooding it with random data in order to cause it to crash
Analysis	A class of invasions that infiltrate online applications via ports (such as port scanning), emails (such as spam), and web scripts (such as HTML files)
Backdoor	A method for getting around a covert normal authentication that prevents unwanted remote access to a device and finding the entry to plain text, while striving to go undetected
DoS	An intrusion that causes computer resources, such as memory, to become very busy in order to block authorized requests from reaching a device
Exploit	A set of instructions that exploits a flaw, fault, or vulnerability produced by inadvertent or unexpected activity on a host or network
Generic	A method for establishing collisions against any block-cipher using a hash function regardless of the block-cipher's setup
Reconnaissance	An attack that obtains information about a computer network in order to circumvent its security protections
Shellcode	An attack in which the attacker gains control of a compromised computer by infiltrating a little piece of code beginning with a shell
Worm	An attack in which the attacker duplicates itself in order to propagate to other systems. It often spreads itself over a computer network, relying on security flaws in the target computer to get access to it

Table 3: Sample distribution of the UNSW-NB15 dataset.

Class type	Training samples	Training samples percentage (%)	Testing samples	Testing samples percentage (%)
Normal	56,000	31.9	37,000	44.9
Attack	119,341	68.1	45,332	55.1
Total	175,341	100	82,332	100

3.5.1. Preprocessing of UNSW-NB15 Dataset. The preprocessing operation performed on the dataset involved the conversion of text datatype entries into numeric datatype entries by using the one-hot encoding method. Also, multiple text cases were unified into the same format by converting them all to their respective lower cases. Three nominal features (proto, state, and service) were one-hot-encoded to generate new columns filled with ones and zeros. After obtaining numeric columns, a z-score was used to normalize the attribute scales for each column. It was observed that the target class was imbalanced with the attack class having 119,341 samples while the normal class had 56,000. To correct this imbalance, the minority class was oversampled using the SMOTE algorithm to raise its figure to 119,341 samples. Our proposed algorithm was deployed on the preprocessed dataset, and the resulting data was the input of 4 machine learning algorithms namely, logistic regression (LR), random forest (RF), decision trees (DT), and K-nearest neighbors (KNN).

3.5.2. Hyperparameter Selection and Tuning. The values of the hyperparameters used for the classifiers were selected based on the random and the grid search methods [77]. Firstly, the random search was used to determine the region of the search space where the best hyperparameters exist while the grid search was used to fine-tune the result of the random search. In the random search, a collection of hyperparameters and their potential values are established and randomly assessed, and the set that gives the highest accuracy among them was selected as the initial set of hyperparameters, which served as the input to the grid search. In the grid search, the hyperparameters set that was determined by the random search were used such that a grid of all possible combinations of its elements was constructed. The models were then trained and assessed on each combination of these hyperparameters, and the one that performed best in terms of accuracy was selected as the final set of hyperparameters for a classifier. Table 4 illustrates the final values of hyperparameters selected for the classifiers after performing the random and the grid searches.

3.5.3. Feature Selection and Model Training. Before the model training operation was performed, the dataset was partitioned into the training set and the test set in the proportion of 70 : 30. This guarantees that 30% of the data are set aside for testing the model with previously unseen data. This is a typical machine learning technique since it enables the validation of training outcomes using previously unseen data. Once the dataset has been partitioned, the feature selection process was initiated by executing

Table 4: Machine learning hyperparameter values.

Algorithms	Hyperparameter values	
	15 features	30 features
Random forest	Number of estimators = 38, maximum depth = 5, criterion = “gini”	The same as 15 features
Decision tree	Minimum samples leaf = 2, maximum depth = 10, criterion = “entropy”	Minimum samples leaf = 2, maximum depth = 2, criterion = “entropy”
K-nearest neighbors	Weights = “uniform,” number of neighbors = 10, distance metric = “euclidean,” leaf size = 5	The same as 15 features

Algorithm 1 on the training data. Next, feature synthesis was performed, the feature selection process was completed by the information gain filter, and the selected features were passed to the machine learning module. In the machine learning module, we used the method used in [19] by applying stratified KFold cross-validation (CV). This procedure is intended to lessen the likelihood of the model overfitting or being impacted by selection bias. The data is separated into *n* subsets in stratified KFold cross-validation, and the class ratio is retained in all of them. In turn, each subset, or fold, is preserved for testing, while the remainder of the data is utilized for training. This means that training is done on the data *n* times, i.e., the number of folds prepared earlier. A 5-fold stratified CV was used in this experiment, which means that the training set was divided into five subsets, each with the same ratio of classes. One of the five subsets was preserved for testing, while the other four were used to train the models. The training procedure is effectively repeated five times, once for each fold. Once this procedure was completed, the remaining 30% of the data was used as test data for a subsequent run. This guaranteed that the models were validated using previously unseen data.

3.6. Machine Learning Algorithms

3.6.1. Logistic Regression. In a logistic regression algorithm, the chance of an event occurring is predicted by fitting data to a logistic function. This function returns a value between 0 and 1. The midpoint number, 0.5, is regarded as the threshold between classes 1 and 0. If the output is larger than 0.5, it is classified as class 1, and if it is less than 0.5, it is classified as class 0.

3.6.2. Random Forest. It is a classifier that is based on decision trees. Random samples are used to form decision trees and then predictions are made from each tree. The predicted class is determined by voting which is cast by the individual trees. Most of the time, even without the usage of a hyperparameter, random forest can provide acceptable results. It produces quick results even for mixed, noisy, and incomplete datasets.

3.6.3. Decision Trees. The structure of a decision tree algorithm is similar to that of a tree, with each internal node representing a test on an attribute, each branch interpreting a test result, and each leaf node displaying a class label. DT can handle both categorical and continuous data and can accomplish classification without needing significant processing.

3.6.4. K-Nearest Neighbors. The K-nearest neighbors machine learning algorithm saves all of the training data upfront. It utilizes this data during categorization to try to detect similarities between the new data and the existing data. Its foundation is the Euclidean distance. The test data is assigned to a class consisting of its K closest neighbors. Accuracy may improve if the value of K is increased.

4. Results and Discussion

Recall that in Section 3, W was the number of features that the mutual information gain filter passed to the machine learning algorithm. We, therefore, investigated how the choice of W affects the performance of KOMIG IDS in terms of accuracy, recall, precision, F1 score, confusion matrix, and AUC. Firstly, we set $W = 30$ features and determined the performances of KOMIG IDS when used with LR, RF, DT, and KNN classifiers.

With W set at 30, Table 5 illustrates its effect on the recall, precision, F1 score, and accuracy scores of LR-based KOMIG IDS, RF-based KOMIG IDS, DT-based KOMIG IDS, and KNN-based KOMIG IDS. For brevity, they have been simply called LR, RF, DT, and KNN in the table. It can be observed in the table that all the classifiers performed well in terms of recall, precision, F1 score, and accuracy. Table 5 contains 7 categories of results, namely, (i) recall of normal class, (ii) recall of intrusion class, (iii) precision of normal class, (iv) precision of intrusion class, (v) F1 score of normal class, (vi) F1 score of intrusion class, and (vii) accuracy. According to Table 5, the KNN algorithm had the top score in 5 out of the 7 performance metrics considered while the logistic regression algorithm had the top score in the remaining 2 (see the bold fonts in the table). In particular, KNN had the highest scores in the recall (0.9946) and F1 score (0.9674) of the intrusion traffic class and also the highest in the precision (0.9930) and F1 scores (0.9674) of the normal traffic class. It also had the highest accuracy score of 97.14%. LR classifier had the highest score in the remaining two metrics, which were the recall of the normal class (0.9904) and the precision of the intrusion class (0.9917).

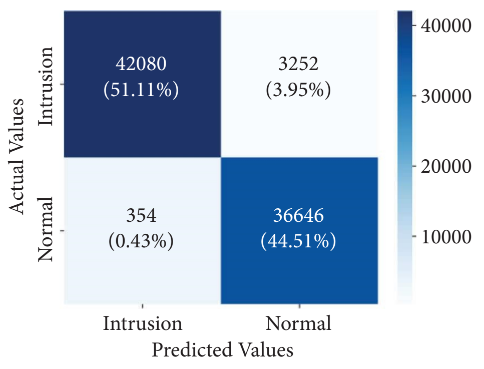
For KNN, a recall value of 0.9946 implies that 99.46% of actual intrusion instances were correctly detected while a precision value of 0.9553 implies that out of all instances that have been classified as intrusion, 95.53% of them are actual intrusion instances. In a like manner, recall values of 0.9283, 0.8617, and 0.8990 for LR, RF, and DT, respectively, imply that 92.83%, 86.17%, and 89.90% of actual intrusion instances were correctly detected for LR, RF, and DT classifiers, respectively. Furthermore, precision values of 0.9917, 0.9849, and 0.9855 for LR, RF, and DT classifiers, respectively, imply that 99.17%, 98.49%, and 98.55% of instances classified as intrusion instances were correct intrusion classifications for LR, RF, and DT classifiers, respectively.

Figures 5(a)–5(d) show the confusion matrices of the LR, RF, DT, and KNN classifiers, respectively. It can be seen that all the algorithms performed relatively well since the misclassification cells (top-right and bottom-left cells) are lightly shaded while the correct classification cells (top-left and bottom-right) are comparatively more boldly shaded. Most of the classifiers such as LR, RF, and DT had difficulty detecting some instances of intrusion traffic; hence, they incorrectly classified such traffic as normal traffic (see the top right cells of Figures 5(a)–5(c)). For example, LR, RF, DT, misclassified, 3,252, 6,271, and 4,582 instances of intrusion traffic as normal traffic, respectively. These amounted to 3.95%, 7.62%, and 5.57% of the total traffic, respectively.

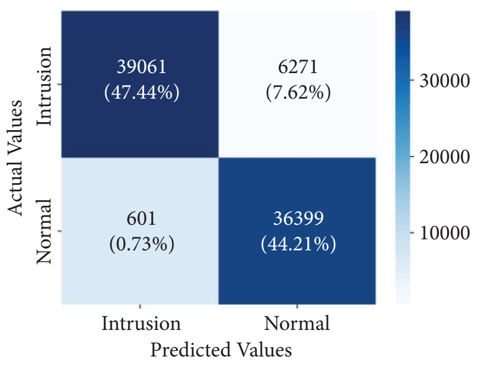
TABLE 5: Recall, precision, F1 score, and accuracy of the normal and intrusion classes when the number of features was set at 30 features.

Classifier	Traffic type	Recall	Precision	F1 score	Accuracy (%)
Logistic regression	Normal	**0.9904**	0.9185	0.9531	95.62
	Intrusion	0.9283	**0.9917**	0.9589	
Random forest	Normal	0.9838	0.8530	0.9137	91.65
	Intrusion	0.8617	0.9849	0.9192	
Decision trees	Normal	0.9838	0.8882	0.9336	93.71
	Intrusion	0.8990	0.9855	0.9402	
K-nearest neighbors	Normal	0.9430	**0.9930**	**0.9674**	**97.14**
	Intrusion	**0.9946**	0.9553	**0.9746**	

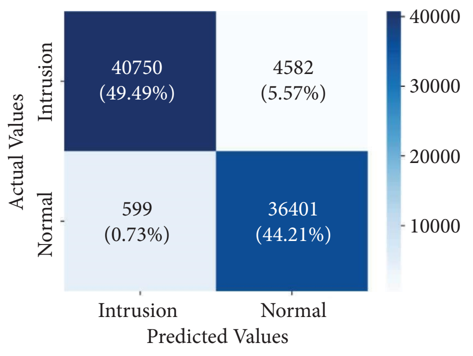
The bold fonts indicate top performance scores across the considered machine learning models for the intrusion and normal traffic categories.



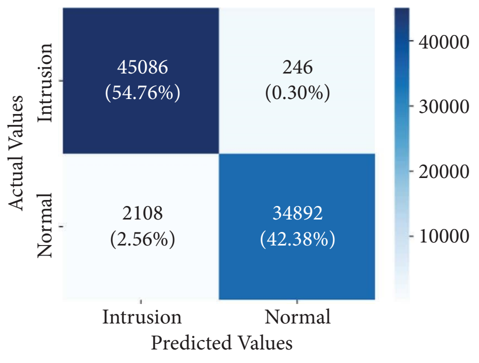
(a)



(b)



(c)



(d)

FIGURE 5: Confusion matrix of classifiers when 30 features are used in the training and testing sets: (a) Logistic regression classifier, (b) random forest classifier, (c) decision trees classifier, and (d) K-nearest neighbor classifier.

The KNN-based KOMIG classifier, on the other hand, had more difficulty classifying normal traffic correctly. For example, it misclassified more normal traffic (2,108 instances) as intrusion traffic than intrusion traffic as normal traffic (246 instances). On a general note, therefore, the KNN-based KOMIG-IDS classifier outperformed all the 3 other variants of KOMIG IDS in terms of correct classification and also in terms of misclassification.

LR, RF, and DT misclassified 4.38%, 8.35%, and 6.3% of the total traffic, respectively. These figures were obtained by summing the misclassified intrusion traffic class percentage (top-right cell), and the misclassified normal traffic class percentage (bottom-left cell). The closer these figures are to zero, the better the performance of the classifier. It is observable that KNN had the best score of total misclassification percentage of 2.86%.

Next, we reduced the number of features by half, i.e., from 30 features to 15 features, and observed the effect of the reduction on the performances of the 4 classifiers. Table 6 illustrates the recall, precision, F1, and accuracy scores of the normal and intrusion classes when the number of features is set at 15 features. It is interesting to note in the table that even though the number of features saw a sharp decline from 30 to 15, there is no corresponding sharp decline in the performances of any of the classifiers. For example, the normal class of the LR classifier had a recall score of 0.9904 at 30 features and 0.9835 at 15 features. This is an insignificant decline in performance. Likewise, the intrusion class of the classifier had a recall of 0.9283 at 30 features and 0.9126 at 15 features. This is also an insignificant reduction in performance.

Table 6: Recall, precision, F1 score, and accuracy of the normal and intrusion classes when the number of features is set at 15 features.

Classifier	Traffic type	Recall	Precision	F1 score	Accuracy
Logistic regression	Normal	0.9835	0.9018	0.9409	94.45
	Intrusion	0.9126	0.9855	0.9477	
Random forest	Normal	**0.9852**	0.8316	0.9019	90.37
	Intrusion	0.8371	**0.9858**	0.9054	
Decision trees	Normal	0.9838	0.8505	0.9123	91.50
	Intrusion	0.8588	0.9849	0.9175	
K-nearest neighbors	Normal	0.9369	**0.9784**	**0.9572**	**96.23**
	Intrusion	**0.9831**	0.9502	**0.9664**	

The bold fonts indicate top performance scores across the considered machine learning models for the intrusion and normal traffic categories.

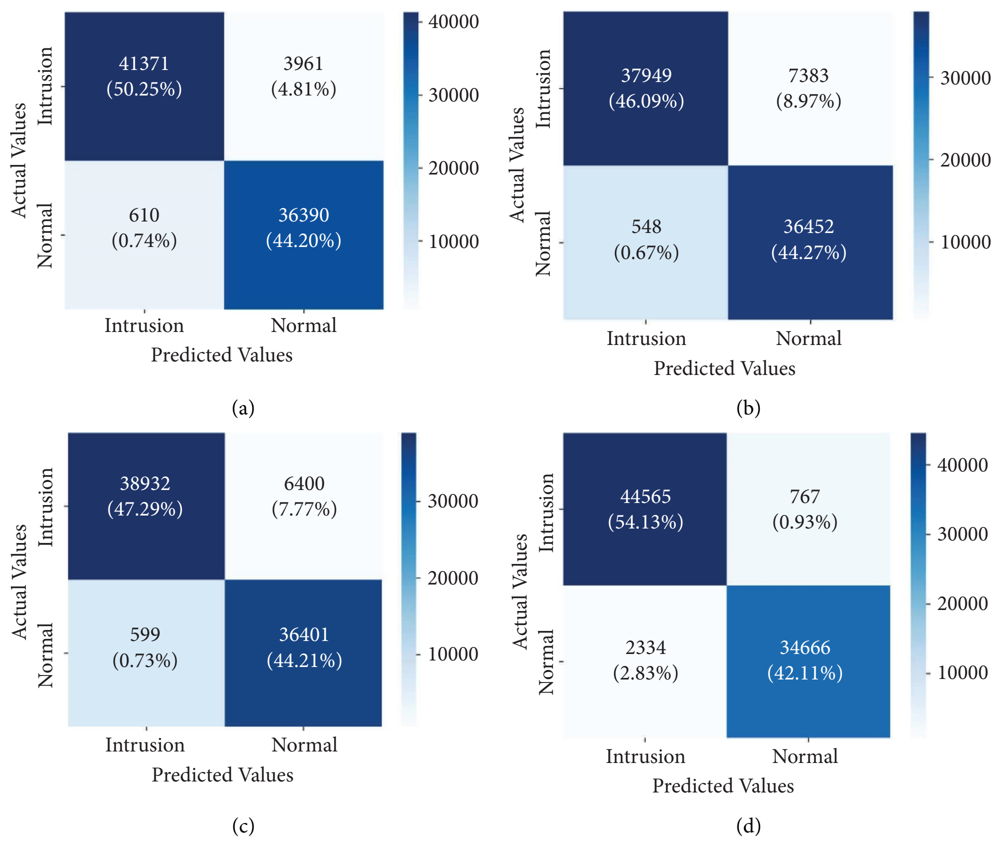
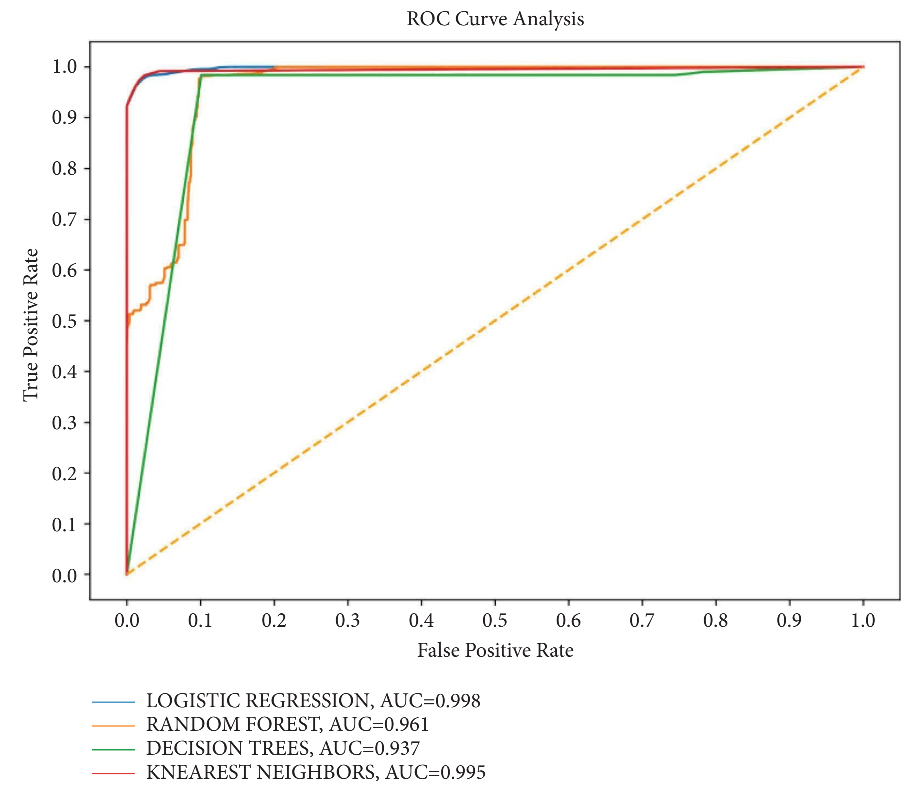


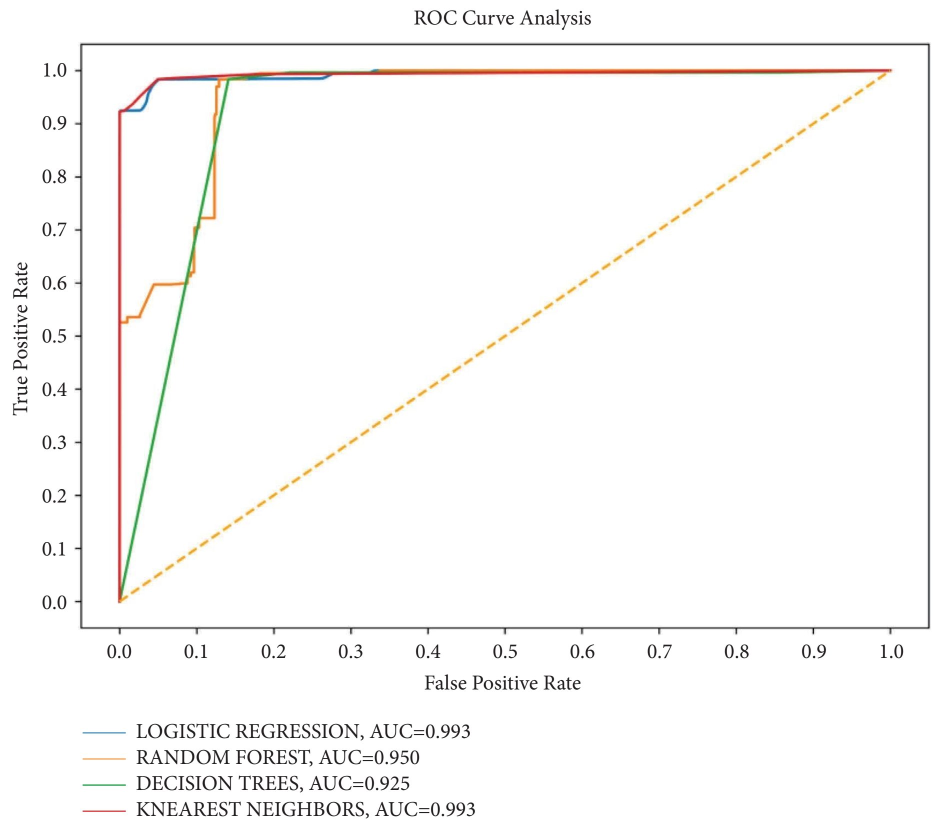
Figure 6: Confusion matrix of classifiers when 15 features are used in the training and testing sets: (a) Logistic regression classifier, (b) random forest classifier, (c) decision trees classifier, and (d) K-nearest neighbor classifier.

Comparing the performances of the classifiers with each other, again, just like in the case of 30 features, KNN had the highest number of best performances. It had the best performances in 5 out of the 7 metrics considered, which are bold in Table 6 for easy identification. These are recall (0.9831) and F1 score (0.9664) of the intrusion class and precision (0.9784) and F1 score (0.9572) of the normal traffic class. It also had the highest accuracy (96.23%) among the classifiers. The performance of the RF classifier was also impressive as it had the best performance in terms of the recall of the normal class (0.9852) and the precision of the intrusion class (0.9858). For the KNN classifier, a recall value of 0.9831 implies that 98.31% of actual intrusion instances were correctly detected while a precision value of 0.9502 implies that out of all instances that have been classified as intrusion, 95.02% of them are actual intrusion instances. In a like manner, recall values of 0.9126, 0.8371, and 0.8588 for LR, RF, and DT classifiers, respectively, imply that 91.26%, 83.71%, and 85.88% of actual intrusion instances were correctly detected for LR, RF, and DT classifiers, respectively. Furthermore, precision values of 0.9855, 0.9858, and 0.9849 for LR, RF, and DT classifiers, respectively, imply that 98.55%, 98.58%, and 98.49% of instances classified as intrusion instances were correct intrusion classifications for LR, RF, and DT classifiers, respectively.

Figures 6(a)–6(d) show the confusion matrices of the LR, RF, DR, and KNN, respectively. It can be seen that all the algorithms performed relatively well. If Figures 5 and 6 are



(a)



(b)

FIGURE 7: AUC-ROC curve for (a) 30 features and (b) 15 features.

Table 7: Comparison between the proposed KNN-based KOMIG IDS and existing IDSs.

Classifier	Data type	Recall	Precision	F1 score	Accuracy
Logistic regression [72]	Normal	**0.9633**	0.8683	0.9134	88.30
	Intrusion	0.7404	0.9191	0.8201	
Ensemble [72]	Normal	0.9531	0.9514	0.9522	93.88
	Intrusion	0.9135	0.9163	0.9149	
Decision trees [72]	Normal	0.8780	0.9375	0.9068	88.44
	Intrusion	0.8960	0.8052	0.8482	
KNN-KOMIG (proposed)	Normal	0.9430	**0.9930**	**0.9674**	**97.14**
	Intrusion	**0.9946**	**0.9553**	**0.9746**	

The bold fonts indicate top performance scores across the considered machine learning models for the intrusion and normal traffic categories.

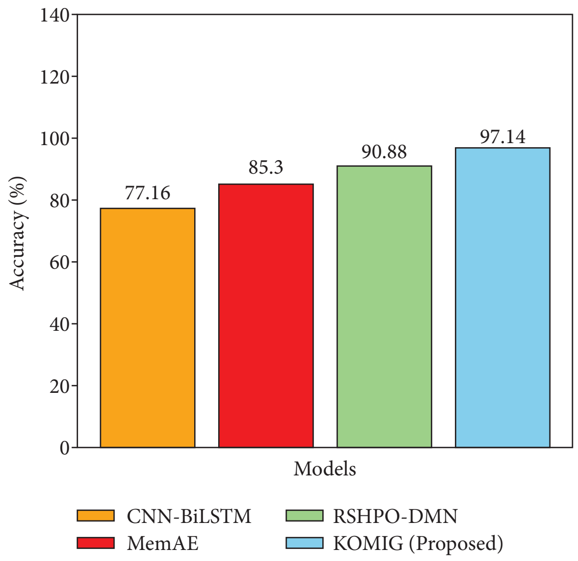


Figure 8: Comparison between KOMIG (proposed), CNN-BiLSTM, MemAE, and RSHPO-DMN IDSs in terms of accuracy.

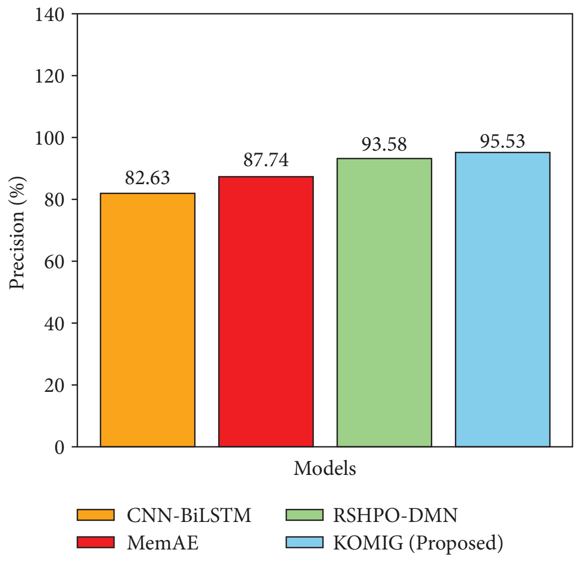


Figure 9: Comparison between KOMIG (proposed), CNN-BiLSTM IDS, MemAE, and RSHPO-DMN IDSs in terms of precision.

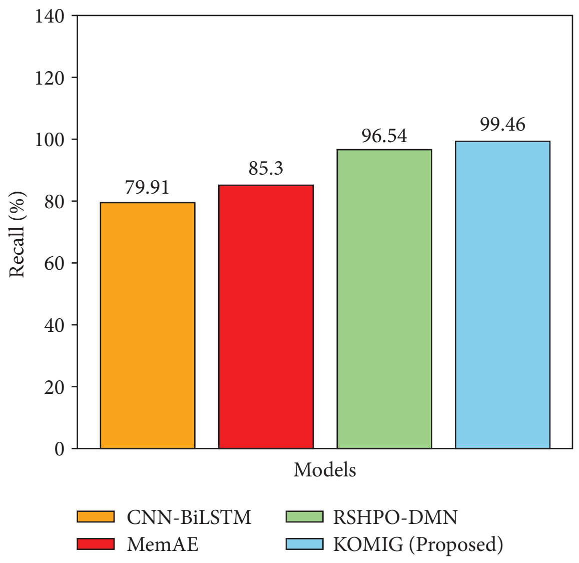


Figure 10: Comparison between KOMIG (proposed), CNN-BiLSTM IDS, MemAE, and RSHPO-DMN IDSs in terms of recall.

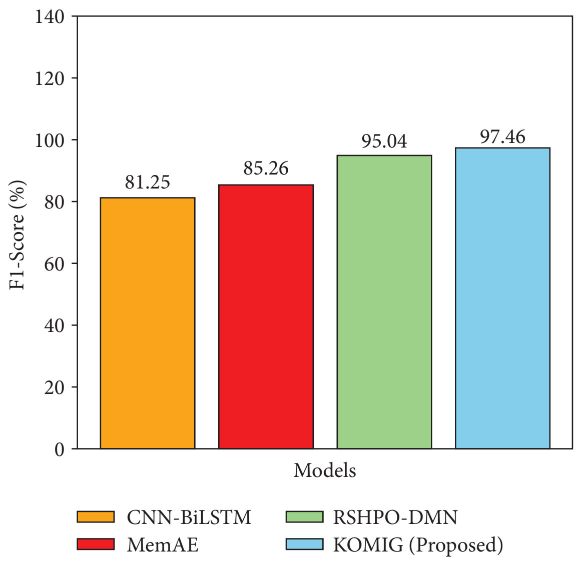


Figure 11: Comparison between KOMIG (proposed), CNN-BiLSTM IDS, MemAE, and RSHPO-DMN IDSs in terms of F1 score.

compared, it can be observed that there is only a small decline in terms of misclassification performance by each of the classifiers. For example, the total misclassification by LR was 4.38% in Figure 5 but 5.55% in Figure 6. This is just about a 1% decline, which is acceptable when we consider that it was caused by a whopping 50% reduction in the number of features. Considering the decline in misclassification performance of other classifiers, RF increased from 8.35% to 9.64%, DT increased from 6.3% to 8.43%, and KNN from 2.86% to 3.76%. In these cases, the highest increase in misclassification performance was just about 2% which is acceptable considering that the number of features has been reduced by 50%.

Next, we compared the classifiers, using the AUC-ROC curve. The AUC-ROC is often regarded as a more essential indicator of an algorithm's quality when compared to the accuracy. This statistic incorporates the trade-offs between precision and recall, while accuracy solely evaluates the number of right predictions. The AUC-ROC curve of 30 features is shown in Figure 7(a) while that of 15 features is shown in Figure 7(b). When the number of features was set at 30 features, it can be seen in Figure 7(a) that the LR classifier had the highest AUC value of 0.998 and is followed very closely by the KNN classifier with an AUC score of 0.995. This shows that both the LR and KNN classifiers have a very high intrusion detection probability when the number of features is set at 30. The RF and DT both had high AUC scores too with values of 0.961 and 0.937, respectively.

The AUC-ROC performances of the 4 classifiers when 15 features were used to train their models are illustrated in Figure 7(b). Just like in Figure 7(a), the LR and KNN both had the highest AUC score of 0.993. Comparing Figure 7(a) with Figure 7(b), it can be seen that the reduced number of features used in Figure 7(b) caused only a small drop in the AUC scores of LR and KNN classifiers. LR dropped from 0.998 to 0.993 while KNN dropped from 0.995 to 0.993. The drop in the case of RF and DT was not significant either. They were from 0.961 to 0.95 and from 0.937 to 0.925, respectively.

Next, we compared the performance of the proposed KNN-based KOMIG IDS method with IDSs existing in the literature as shown in Table 7. The comparative classifiers, which are shown in Table 7, were the logistic regression classifier, the ensemble classifier, and the decision trees classifier. It can be observed that our proposed scheme had the best performances in 6 out of the 7 metrics considered (please see the bold fonts in Table 7). These are the recall (0.9946), precision (0.9553), and F1 score (0.9746) of the intrusion traffic class. It also performed best in terms of the precision (0.9930) and F1 score (0.9674) of the normal traffic class. The highest accuracy of 97.14% was also achieved by our proposed method.

Next, we compare the performance of KOMIG to state-of-the-art IDSs such as CNN-BiLSTM IDS [78], RSHPO-DMN IDS [54], and the memory-augmented deep auto-encoder IDS, MemAE [79]. A comparison of the detection accuracies of the four models is illustrated in Figure 8.

In can be observed in Figure 8 that our proposed KOMIG IDS has the highest detection accuracy of 97.14%. It is ahead of RSHPO-DMN by up to 6.26% since RSHPO-DMN has an accuracy score of 90.88%. RSHPO-DMN is followed by MemAE, which has accuracy score of 85.30%, while the least performing IDS is the CNN-BiLSTM with a score of 77.16%. Figure 9 illustrates the precision scores of the four IDSs. The figure shows that our proposed KOMIG IDS leads all the others by having a precision score of 95.53%. It is closely followed by RSHPO-DMN, which has a precision score of 93.58%, while MemAE and CNN-BiLSTM have precision scores of 87.74% and 82.63%, respectively.

In Figure 10, the recall score of the four IDSs are compared. KOMIG IDS takes the lead with a recall score of 99.46%, and it is followed by RSHPO-DMN, which has a recall score of 96.54%. The recall score of MemAE is 85.30%, while CNN-BiLSTM's recall score is 79.91% Finally, the F1 scores of the four IDSs are compared in Figure 11, where our proposed KOMIG IDS has the highest score of 97.46%. RSHPO-DMN, MemAE, and CNN-BiLSTM have F1 scores of 95.04%, 85.26%, and 81.25%, respectively. KOMIG IDS outperformed all the comparative IDSs shown in Figures 8 through 11 in terms of accuracy, precision, recall, and F1 score. This indicates that the 2-stage feature selection procedure that is adopted in KOMIG (which comprises knapsack optimization and information gain-based filtering) combined with the proposed feature synthesis operation are highly effective and promising when applied in intrusion detection systems.

5. Conclusions and Future Works

This paper discussed KOMIG IDS, which is a new intrusion detection system that relies on selecting appropriate features of a dataset to train machine learning models. To realize KOMIG IDS, we first defined an optimization problem for selecting the best features from the dataset of network traffic. We then transformed the optimization problem into an easy-to-solve form, and an algorithm for its implementation was developed. Following that, we presented a method for synthesizing new features from the features selected by the optimizer. Then, we constructed a candidate features set by combining the synthetic features with the features selected by the optimizer. Next, we used an information gain filter to exclude redundant features from the set of candidate features while picking the set of features with the highest information gain. The size of the feature set after applying the proposed KOMIG feature selection procedure was 30 features. Finally, machine learning models were trained using the chosen features. The models that were trained included logistic regression (LR), random forest (RF), decision trees (DT), and K-nearest neighbors (KNN) models. These resulted in four variants of our proposed method, namely, LR-KOMIG, RF-KOMIG, DT-KOMIG, and KNN-KOMIG. An experiment was set up using a well-known network intrusion detection evaluation dataset (the UNSW-NB15 dataset) to evaluate the performance of KOMIG IDS in comparison to existing state-of-the-art IDSs. Results revealed that the intrusion detection accuracies of LR-KOMIG, RF-KOMIG, DT-KOMIG, and KNN-KOMIG were 95.62%, 91.65%, 93.71%, and 97.14%, respectively. Hence, our proposed KNN-KOMIG which has an accuracy score of 97.14% was able to outperform some state-of-the-art comparative IDSs such as the ensemble learner IDS (which has an accuracy score of 93.88%) and the RSHPO-DMN IDS (which has an accuracy score of 90.88%). Furthermore, it outperformed MemAE and CNN-BiLSTM IDSs which have 85.30% and 77.16% accuracy scores, respectively. The KOMIG IDS is limited to binary classification only. Hence, it is only capable of detecting whether or not a network intrusion has occurred but it is unable to detect the type of attack.

In the future, we will extend the KOMIG IDS to cover multiclass classification so that it will not only be capable of detecting network intrusion but also the type of attack as well. We will also incorporate new optimization algorithms into KOMIG IDS such as the farmland fertility algorithm, African vultures optimization algorithm, mountain gazelle optimizer, and artificial gorilla troops optimizer. These optimization algorithms will be used to replace the knapsack optimizer proposed in the current work so that their performances can be evaluated and compared.

Data Availability

The UNSW-NB15 traffic data supporting this study are from previously reported studies and datasets, which have been cited. The datasets are available at https://research.unsw.edu.au/projects/unsw-nb15-dataset.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Authors' Contributions

A.S.A. conceptualized the study, A.S.A. and O.A.A. provided methodology, A.S.A. collected software, O.A.A validated the study, A.S.A and O.A.A. performed formal analysis, A.S.A. investigated the study, A.S.A. and O.A.A collected resources, A.S.A. prepared the original draft of the study, A.S.A. and O.A.A. reviewed and edited the study, A.S.A. visualized the study, O.A.A. supervised the study, and O.A.A. administrated the project. All authors have read and agreed to the published version of the manuscript.

References

[1] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2021.

[2] M. L. Hernandez-Jaimes, A. Martinez-Cruz, K. A. Ramírez-Gutiérrez, and C. Feregrino-Uribe, "Artificial intelligence for IoMT security: a review of intrusion detection systems, attacks, datasets and Cloud–Fog–Edge architectures," *Internet of Things (Netherlands)*, vol. 23, pp. 100887–100922, 2023.

[3] O. Chakir, A. Rehaimi, Y. Sadqi et al., "An empirical assessment of ensemble methods and traditional machine learning techniques for web-based attack detection in industry 5.0," *Journal of King Saud University-Computer and Information Sciences-Computer and Information Sciences*, vol. 35, no. 3, pp. 103–119, 2023.

[4] 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. 10, no. 3, pp. 1634–1646, 2022.

[5] A. Kim, M. Park, and D. H. Lee, "AI-IDS: application of deep learning to real-time web intrusion detection," *IEEE Access*, vol. 8, pp. 70245–70261, 2020.

[6] M. Data and M. Aritsugi, "T-DFNN: an incremental learning algorithm for intrusion detection systems," *IEEE Access*, vol. 9, pp. 154156–154171, 2021.

[7] L. M. Halman and M. J. Alenazi, "MCAD: a machine learning based cyberattacks detector in software-defined networking (SDN) for healthcare systems," *IEEE Access*, vol. 11, pp. 37052–37067, 2023.

[8] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020.

[9] S. Neupane, J. Ables, W. Anderson et al., "Explainable intrusion detection systems (X-IDS): a survey of current methods, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 112392–112415, 2022.

[10] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, and J. Ahmad, "Enhancing IoT network security through deep learning-powered intrusion detection system," *Internet of Things (Netherlands)*, vol. 24, pp. 100936–36, 2023.

[11] M. A. Hossain and M. S. Islam, "Ensuring network security with a robust intrusion detection system using ensemble-based machine learning," *Array*, vol. 19, pp. 100306–100314, 2023.

[12] N. Ahmad Hamdi Qaiwmchi, H. Amintoosi, and A. Mohajerzadeh, "Intrusion detection system based on gradient corrected online sequential Extreme learning machine," *IEEE Access*, vol. 9, pp. 4983–4999, 2021.

[13] Q. Liu, V. Hagenmeyer, and H. B. Keller, "A review of rule learning-based intrusion detection systems and their prospects in smart grids," *IEEE Access*, vol. 9, pp. 57542–57564, 2021.

[14] J. Lansky, S. Ali, M. Mohammadi et al., "Deep learning-based intrusion detection systems: a systematic review," *IEEE Access*, vol. 9, pp. 101574–101599, 2021.

[15] L. Cerdà-Alabern, G. Iuhasz, and G. Gemmi, "Anomaly detection for fault detection in wireless community networks using machine learning," *Computer Communications*, vol. 202, pp. 191–203, 2023.

[16] K. S. Yakub, O. Harazeem Abdulganiyu, and T. Ait Tchakoucht, "A novel hybrid ensemble learning for anomaly detection in industrial sensor networks and SCADA systems for smart city infrastructures," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 5, pp. 1–18, 2023.

[17] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *IEEE Access*, vol. 9, pp. 22351–22370, 2021.

[18] J. Ashraf, M. Keshk, N. Moustafa et al., "IoTBoT-IDS: a novel statistical learning-enabled botnet detection framework for protecting networks of smart cities," *Sustainable Cities and Society*, vol. 72, pp. 103041–103112, 2021.

[19] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowledge-Based Systems*, vol. 279, pp. 110941–111013, 2023.

[20] M. H. Nasir, J. Arshad, and M. M. Khan, "Collaborative device-level botnet detection for internet of things," *Computers & Security*, vol. 129, pp. 103172–103220, 2023.

[21] Z. Noor, S. Hina, F. Hayat, and G. A. Shah, "An intelligent context-aware threat detection and response model for smart cyber-physical systems," *Internet of Things (Netherlands)*, vol. 23, pp. 100843–100920, 2023.

[22] W. Zhong, N. Yu, and C. Ai, "Applying big data based deep learning system to intrusion detection," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 181–195, 2020.

[23] T. Wisanwanichthan and M. Thammawichai, "A double-layered hybrid approach for network intrusion detection system using combined naive Bayes and SVM," *IEEE Access*, vol. 9, pp. 138432–138450, 2021.

[24] A. Assiri, "Anomaly classification using genetic algorithm-based random forest model for network attack detection," *Computers, Materials & Continua*, vol. 66, no. 1, pp. 767–778, 2020.

[25] S. Alem, D. Espes, L. Nana, E. Martin, and F. De Lamotte, "A novel bi-anomaly-based intrusion detection system approach for industry 4.0," *Future Generation Computer Systems*, vol. 145, pp. 267–283, 2023.

[26] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Anomaly based network intrusion detection for IoT attacks using deep learning technique," *Computers & Electrical Engineering*, vol. 107, 2023.

[27] H. C. Altunay and Z. Albayrak, "A hybrid CNN + LSTMbased intrusion detection system for industrial IoT networks," *Engineering Science and Technology, an International Journal*, vol. 38, pp. 101322–101413, 2023.

[28] S. Wang, W. Xu, and Y. Liu, "Res-TranBiLSTM: an intelligent approach for intrusion detection in the Internet of Things," *Computer Networks*, vol. 235, pp. 109982–110016, 2023.

[29] M. H. M. Yusof, A. A. Almohammedi, V. Shepelev, and O. Ahmed, "Visualizing realistic benchmarked IDS dataset: CIRA-CIC-DoHBrw-2020," *IEEE Access*, vol. 10, pp. 94624–94642, 2022.

[30] I. Leyva-Pupo and C. Cervelló-Pastor, "An intelligent scheduling for 5G user plane function placement and chaining reconfiguration," *Computer Networks*, vol. 237, pp. 110037–110115, 2023.

[31] S. T. Shishavan and F. S. Gharehchopogh, "An improved cuckoo search optimization algorithm with genetic algorithm for community detection in complex networks," *Multimedia Tools and Applications*, vol. 81, no. 18, pp. 25205–25231, 2022.

[32] Z. Zhou, Y. Ning, X. Zhou, and F. Zheng, "Improved artificial bee colony algorithm-based channel allocation scheme in low earth orbit satellite downlinks," *Computers & Electrical Engineering*, vol. 110, 2023.

[33] S. Ding, T. Zhang, C. Chen et al., "An efficient particle swarm optimization with evolutionary multitasking for stochastic area coverage of heterogeneous sensors," *Information Sciences*, vol. 645, 2023.

[34] S. O. Ogundoyin and I. A. Kamil, "Optimal fog node selection based on hybrid particle swarm optimization and firefly algorithm in dynamic fog computing services," *Engineering Applications of Artificial Intelligence*, vol. 121, 2023.

[35] T. S. Naseru and F. S. Gharehchopogh, "A feature selection based on the farmland fertility algorithm for improved intrusion detection systems," *Journal of Network and Systems Management*, vol. 30, pp. 1–27, 2022.

[36] S. Santra and M. De, "Mountain gazelle optimisation-based 3DOF-FOPID-virtual inertia controller for frequency control of low inertia microgrid," *IET Energy Systems Integration*, vol. 5, no. 4, pp. 405–417, 2023.

[37] G. S. Nijaguna, J. A. Babu, B. D. Parameshachari, R. P. de Prado, and J. Frnda, "Quantum Fruit fly algorithm and ResNet50-VGG16 for medical diagnosis," *Applied Soft Computing*, vol. 136, 2023.

[38] Y. Zhang, C. Han, and S. Liu, "A digital calibration technique for N-channel time-interleaved ADC based on simulated annealing algorithm," *Microelectronics Journal*, vol. 133, 2023.

[39] S. Manda and C. Singh, "CVFP: energy and trust aware data routing protocol based on Competitive Verse Flower Pollination algorithm in IoT," *Computers & Security*, vol. 127, 2023.

[40] Q. Yang, J. Liu, Z. Wu, and S. He, "A fusion algorithm based on whale and grey wolf optimization algorithm for solving real-world optimization problems," *Applied Soft Computing*, vol. 146, 2023.

[41] E. H. Houssein, G. N. Zaki, A. A. Z. Diab, and E. M. Younis, "An efficient Manta Ray Foraging Optimization algorithm for parameter extraction of three-diode photovoltaic model," *Computers & Electrical Engineering*, vol. 94, 2021.

[42] F. S. Gharehchopogh and T. Ibrikci, "An improved African vultures optimization algorithm using different fitness functions for multi-level thresholding image segmentation," *Multimedia Tools and Applications*, vol. 83, no. 6, pp. 16929–16975, 2023.

[43] R. C. Jebi and S. Baulkani, "Mitigation of coverage and connectivity issues in wireless sensor network by multi-objective randomized grasshopper optimization based selective activation scheme," *Sustainable Computing: Informatics and Systems*, vol. 35, 2022.

[44] A. Seyfollahi, M. Moodi, and A. Ghaffari, "MFO-RPL: a secure RPL-based routing protocol utilizing moth-flame optimizer for the IoT applications," *Computer Standards & Interfaces*, vol. 82, 2022.

[45] X. Xue, S. K. Palanisamy, D. S. Selvaraj, O. I. Khalaf, and G. M. Abdulsahib, "A Novel partial sequence technique based Chaotic biogeography optimization for PAPR reduction in eneralized frequency division multiplexing waveform," *Heliyon*, vol. 9, no. 9, 2023.

[46] J. Cai, H. Yang, T. Lai, and K. Xu, "A new approach for optimal chiller loading using an improved imperialist competitive algorithm," *Energy and Buildings*, vol. 284, 2023.

[47] V. R. Ekhlas, M. Hosseini Shirvani, A. Dana, and N. Raeisi, "Discrete grey wolf optimization algorithm for solving k-coverage problem in directional sensor networks with network lifetime maximization viewpoint," *Applied Soft Computing*, vol. 146, 2023.

[48] F. S. Gharehchopogh, "An improved Harris Hawks optimization algorithm with multi-strategy for community detection in social network," *Journal of Bionics Engineering*, vol. 20, no. 3, pp. 1175–1197, 2023.

[49] S. Yalçın and E. Erdem, "Effective cluster scheduling scheme using local gravitation method for wireless sensor networks," *Expert Systems with Applications*, vol. 233, 2023.

[50] F. S. Gharehchopogh, A. Ucan, T. Ibrikci, B. Arasteh, and G. Isik, "Slime Mould algorithm: a comprehensive survey of its variants and applications," *Archives of Computational Methods in Engineering*, vol. 30, no. 4, pp. 2683–2723, 2023.

[51] S. Eskandari and M. Seifaddini, "Online and offline streaming feature selection methods with bat algorithm for redundancy analysis," *Pattern Recognition*, vol. 133, 2023.

[52] Y. K. Saheed, T. O. Kehinde, M. A. Raji, and U. A. Baba, "Feature selection in intrusion detection systems: a new hybrid fusion of Bat algorithm and Residue Number System," *Journal of Information and Telecommunication*, vol. 19, 2023.

[53] H. Mohammadzadeh and F. S. Gharehchopogh, "A multi-agent system based for solving high-dimensional optimization problems: a case study on email spam detection," *International Journal of Communication Systems*, vol. 34, no. 3, 2021.

[54] A. Parameswari, R. Ganeshan, V. Ragavi, and M. Shereesha, "Hybrid rat swarm hunter prey optimization trained deep learning for network intrusion detection using CNN features," *Computers & Security*, vol. 139, no. 6, 2024.

[55] F. S. Gharehchopogh, "Quantum-inspired metaheuristic algorithms: comprehensive survey and classification," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5479–5543, 2023.

[56] O. Aromolaran, T. Beder, E. Adedeji et al., "Predicting host dependency factors of pathogens in *Drosophila melanogaster* using machine learning," *Computational and Structural Biotechnology Journal*, vol. 19, pp. 4581–4592, 2021.

[57] E. Domingos, B. Ojeme, and O. Daramola, "Experimental analysis of hyperparameters for deep learning-based churn prediction in the banking sector," *Computation*, vol. 9, no. 3, pp. 34–19, 2021.

[58] M. R. Islam, K. Oliullah, M. M. Kabir, M. Alom, and M. F. Mridha, "Machine learning enabled IoT system for soil nutrients monitoring and crop recommendation," *Journal of Agriculture and Food Research*, vol. 14, pp. 100880–100912, 2023.

[59] U. Sakthi, T. Anil Kumar, K. Vimala Kumar, S. Qamar, G. Kumar Sharma, and A. Azeem, "Power grid based renewable energy analysis by photovoltaic cell machine learning architecture in wind energy hybridization," *Sustainable Energy Technologies and Assessments*, vol. 57, 2023.

[60] G. Ibarra-Vazquez, M. S. Ramirez-Montoya, M. Buenestado-Fernández, and G. Olague, "Predicting open education competency level: a machine learning approach," *Heliyon*, vol. 9, no. 11, 2023.

[61] T. Zoppi, A. Ceccarelli, T. Puccetti, and A. Bondavalli, "Which algorithm can detect unknown attacks? Comparison of supervised, unsupervised and meta-learning algorithms for intrusion detection," *Computers & Security*, vol. 127, pp. 103107–103112, 2023.

[62] N. Omer, A. H. Samak, A. I. Taloba, and R. M. Abd El-Aziz, "A novel optimized probabilistic neural network approach for intrusion detection and categorization," *Alexandria Engineering Journal*, vol. 72, pp. 351–361, 2023.

[63] S. Subbiah, K. S. M. Anbananthen, S. Thangaraj, S. Kannan, and D. Chelliah, "Intrusion detection technique in wireless sensor network using grid search random forest with Boruta feature selection algorithm," *Journal of Communications and Networks*, vol. 24, no. 2, pp. 264–273, 2022.

[64] P. Radoglou-Grammatikis, K. Rompolos, P. Sarigiannidis et al., "Modeling, detecting, and mitigating threats against industrial healthcare systems: a combined software defined networking and reinforcement learning approach," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 2041–2052, 2022.

[65] A. Q. Adeyiola, Y. K. Saheed, and S. Misra, "Metaheuristic firefly and C5.0 algorithms based intrusion detection for critical infrastructures," in *Proceedings of the 3rd International Conference on Applied Artificial Intelligence (ICAPAI)*, pp. 1–7, Halden, Norway, May 2023.

[66] S. M. Kasongo, "An advanced intrusion detection system for IIoT Based on GA and tree based algorithms," *IEEE Access*, vol. 9, pp. 113199–113212, 2021.

[67] A. Nazir and R. A. Khan, "A novel combinatorial optimization based feature selection method for network intrusion detection," *Computers & Security*, vol. 102, 2021.

[68] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," *Journal of Big Data*, vol. 7, no. 1, pp. 105–120, 2020.

[69] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, and R. Islam, "Dependable intrusion detection system for IoT: a deep transfer learning based approach," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 1006–1017, 2023.

[70] M. J. Babu and A. R. Reddy, "SH-IDS: specification heuristics based intrusion detection system for IoT networks," *Wireless Personal Communications*, vol. 112, no. 3, pp. 2023–2045, 2020.

[71] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset," *Cluster Computing*, vol. 23, no. 2, pp. 1397–1418, 2020.

[72] N. Thockchom, M. M. Singh, and U. Nandi, "A novel ensemble learning-based model for network intrusion detection," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5693–5714, 2023.

[73] P. Bhat and K. Dutta, "A multi-tiered feature selection model for android malware detection based on Feature discrimination and Information Gain," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 9464–9477, 2022.

[74] N. Moustafa, "UNSW-NB15 dataset," 2024, https://research.unsw.edu.au/projects/unsw-nb15-dataset.

[75] J. Zhu and X. Liu, "An integrated intrusion detection framework based on subspace clustering and ensemble learning," *Computers & Electrical Engineering*, vol. 115, pp. 109113–109122, 2024.

[76] S. M. Kasongo, "A deep learning technique for intrusion detection system using a recurrent neural networks based framework," *Computer Communications*, vol. 199, pp. 113–125, 2023.

[77] K. E. Hoque and H. Aljamaan, "Impact of hyperparameter tuning on machine learning models in stock price forecasting," *IEEE Access*, vol. 9, pp. 163815–163830, 2021.

[78] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access*, vol. 8, pp. 32464–32476, 2020.

[79] B. Min, J. Yoo, S. Kim, D. Shin, and D. Shin, "Network anomaly detection using memory-augmented deep autoencoder," *IEEE Access*, vol. 9, pp. 104695–104706, 2021.