IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 2, 2021, accepted September 21, 2021, date of publication September 29, 2021, date of current version October 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3116219

A Novel Ensemble Framework for an Intelligent Intrusion Detection System

SUGANDH SETH, KULJIT KAUR CHAHAL, AND GURVINDER SINGH
Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar 143005, India
Corresponding author: Sugandh Seth (sugandhseth@gmail.com)

ABSTRACT **Background**: Building an effective Intrusion detection system in a multi-attack classification environment is challenging due to the diversity of modern, sophisticated attacks. High-performance classification methods are needed for Intrusion Detection Systems as attackers can establish intrusive methods and easily evade the detection tools deployed in a computing environment. Moreover, it is challenging to use a single classifier to efficiently detect all kinds of attacks. **Aims**: To propose a unique ensemble framework that can effectively detect different attack categories. **Method**: The proposed approach is based on building an ensemble by ranking the detection ability of different base classifiers to identify various types of attacks. The F1-score of an algorithm is used to compute the rank matrix for different attack categories. For final prediction algorithm's output for an attack is only considered if the algorithm has the highest F1-Score in the rank matrix for the particular attack category. This approach contrasts with the voting approach where the final classification is based on the voting of all classifiers in the ensemble irrespective of the fact if the algorithm is efficient enough to detect that attack or not. **Results**: With the proposed method, the final accuracy obtained is 96.97 %, a recall rate of 97.4%, and a better attack detection rate than the baseline classifiers and other existing approaches for different attack categories.

INDEX TERMS Intrusion detection system, machine learning, cybersecurity, ensemble learning, intrusion detection framework, CIC IDS 2018.

I. INTRODUCTION

As cyberspace is evolving, new threats in the realm of computer and internet safety are surfacing too. And this is making the traditional Intrusion Detection Systems (IDS) go obsolete quickly. Whereas earlier IDS-based security systems relied on predefined signatures principles [1], those could not detect freshly developed anomalies and attack variants [2]. Its primary problem was the slow rate of refreshing and scaling the signature database to match the rising pace of threat evolution [2]. As tackling today's cyberattacks is getting more daunting, researchers continually use state-of-the-art techniques to ensure accurate threat detection and security using anomaly-based detection [3]. Researchers are using advanced techniques based on machine learning and deep learning to equip systems with future-proof intrusion detection strategies. But a single machine learning technique is not sufficient to detect all kinds of modern-day attacks effectively.

Much research has been proposed in this field claiming good attack classification performance of the proposed IDS solutions on old datasets. But an Intrusion Detection model's performance in the real world depends upon its efficiency to detect modern-day attacks in real-time. Most of the studies carried out in this area are on old datasets [4]. The commonly used datasets [5] in the IDS research are given in Table 1. With modern-day traffic, the models trained on old datasets are not effective for detecting attacks.

In anomaly-based Network Intrusion Detection Systems (NIDS), machine learning techniques have been extensively used. Diverse machine learning algorithms, including Random Forest (RF), Support Vector Machine (SVM), Decision Trees (DT), k-Nearest Neighbor (kNN), are used to detect intrusions. However, a single machine learning algorithm is no longer sufficient to meet the modern IDS's extensive requirements [7], [4], [8], [9]. Besides the high traffic volume, IDS attacks have diversified and are way more sophisticated [6]. Each machine learning algorithm has its pros and cons. Some algorithms can perform well on one type of attack but perform poorly for other attacks. A recent study by Ferrag *et al.* [7] compared various deep learning techniques on the latest CIC IDS 2018 dataset. CIC IDS 2018 dataset comprises several modern-day attacks. As per

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Cusano.

TABLE 1. Commonly used datasets for network intrusion detection.

Name of the Dataset	Attacks Covered	Year
ADFA [5]	Zero-day attacks, Stealth attack, C100 Webshell attack	2014
CDX [7]	Buffer Overflow	2009
CIC IDS 2017 [9]	Brute force, Portscan, Botnet, DoS, DDoS, Web, Infiltration	2017
CIC IDS 2018 [10]	Brute Force, DoS, DDoS, Web Attack, Infiltration attack, Botnet attack, DoS+Port scan	2018
DARPA [5]	DoS, R2L, U2R, Probe	1998/99
ISCX2012 [7]	DoS, DDoS, BruteForce, Infiltration	2012
ISOT [7]	Botnet	2010
KDD CUP 99 [7]	DoS, R2L, U2R, Probe	1999
KYOTO 2006 [7]	Normal and attack sessions	2006 to 2009
NGIDS-DS [35]	Shellcode, Worms, Backdoors, DoS, Exploits, Generic, Reconnaissance	2016
NSL KDD [5]	DoS, R2L, U2R, Probe	1998
Twente [7]	Normal and Attack Sessions	2008
UNSW-NB 15 [13]	Backdoors, DoS, exploits, fuzzes, generic, port scans, reconnaissance, shellcode, spam, worms	2015

the study's stated results, the Recurrent Neural Network (RNN) gave the highest attack detection rate for seven attack types and the Convolutional Neural Network (CNN) for the other four attack types. Karatas *et al.* [4] analyzed six machine learning-based IDS using k-Nearest Neighbour, RF, Gradient Boosting, Adaboost, DT, and Linear Discriminant Analysis algorithm. Feng *et al.* [8] designed a plug-and-play capture tool to grab the packets and detect three network attacks. To detect DoS attacks, they used Deep Neural Networks (DNN). Two deep learning techniques were used to detect XSS and SQL attacks: the CNN and Long Short-Term Memory (LSTM). None of the single detection approaches could effectively classify all kinds of attacks in all the above research. Therefore, to overcome this problem in this paper, we propose a technique that can effectively combine individual algorithms' attack detection ability, resulting in an increased overall attack detection rate.

So, we propose a novel framework for multi-attack classification to address the above challenges using the latest CIC IDS 2018 [10] dataset. The proposed ensemble approach accurately detects network intrusions and attacks by using a hybrid feature selection approach using RF and Principal Component Analysis (PCA) and a design framework that combines the best of different machine learning techniques for classifying various attacks. It is a rank-based approach based on the F1-score of each algorithm for detecting different attacks. This approach gives promising results, better than many of the previous research.

The significant contributions of our research are as follows:

1. A novel framework that effectively combines different machine learning classifiers based on each algorithm's ability to classify attacks in a multi-attack classification environment.
2. A hybrid approach using SMOTE and undersampling is proposed to address the class imbalance issue.
3. The proposed framework is tested and evaluated on the latest CIC IDS 2018 dataset and is compared with different approaches in the recent studies on the same data set.

This paper is organized as follows. The introduction section highlighted the need of an effective learning method for multi-attack classification for an IDS using the latest CIC IDS 2018 dataset. The next section on the Related Work gives an overview of the different attack classification algorithms for Intrusion Detection and their performance metrics and also highlights the research gaps. The Materials and Methods section presents details of the dataset, the machine learning algorithms, reasoning to use the evaluation metrics in this study. It also presents the proposed framework comprising of different machine learning algorithms for intrusion detection. The Results and Discussions section compares the performance of the proposed framework with standalone algorithms as well as with the recent works in the research literature.

II. RELATED WORK

In the research literature, different machine and deep learning models are proposed for attack classification in IDS. Gao *et al.* [2] proposed an adaptive machine learning model for intrusion detection based on an adaptive ensemble voting classifier. The proposed approach used several base classifiers, including DT, KNN, DNN, and RF on the NSL-KDD Test+ dataset. Boosting methods such as feature selection, sampling, multilayer detection, etc. were used to enhance each classifier's detection rate. They also proposed a multi-tree algorithm, which is an optimized version of the DT. The result of the proposed multitree algorithm outperformed that of DNN. Finally, adaptive voting based on class weights was used to compute the result using the base classifiers. They claim to have achieved an accuracy rate of 85.2%, 86.5% precision, 85.2% recall, and 84.9% F1 score. The disadvantage of the suggested method is that the results are based on an outdated dataset that does not reflect modern-day traffic trends and attacks. Feature selection and data preprocessing techniques used in this approach can be optimized for better results.

Zhou *et al.* [9] proposed an ensemble classifier using the Correlation-based feature selection – Bat algorithm (CFS-BA) dimensionality reduction technique. The proposed CFS-BA approach used a fitness function based on the correlation between the features for dimensionality reduction. After dimensionality reduction, the ensemble approach was used to combined C4.5, RF, and Forest by Penalizing attributes (Forest PA). Finally, the voting technique was used for attack recognition based on the probability distribution of base learners' predictions. This approach gave 99% accuracy for NSL-KDD and CIC- IDS 2017 Dataset. Though the proposed method claims to have achieved an accuracy

rate of 99%, the performance of the proposed approach is evaluated on the subset of the CIC IDS-2017 dataset just using traffic data of Wednesday working hours, a subset of Aegean Wi-Fi intrusion dataset (AWID), and NSL-KDD dataset. These datasets are either subsets (only a few attack categories included) or outdated. Moreover, no method is used to handle the imbalanced nature of the IDS datasets.

Tama *et al.* [11] proposed a two-stage Classifier ensemble for Intelligent Anomaly-based Intrusion Detection. This method used hybrid feature selection using the Genetic Algorithm, Particle Swarm Optimization, and Ant Colony algorithm for reducing the features of the dataset. Features were chosen based on the classification results of the Reduced Error Pruning Tree (REPT) classifier. A two-level ensemble classifier using RF and Bagging was used for attack detection. They claim to have achieved an accuracy rate of 85.8%, 86.8% sensitivity rate, and 88.0% detection rate on the NSL-KDD dataset and improved results on the UNSW-NB15 dataset. The accuracy of the proposed method is 85.8% which could be improved. No technique was used by the author to address the imbalanced nature of the IDS datasets.

Lin *et al.* [12] proposed an anomaly detection system that used LSTM to train the neural network and Attention Mechanism (AM) technique to augment the performance of the proposed approach. The proposed model was trained on the latest CIC IDS 2018 dataset. They claim to have achieved a good accuracy rate of 96.22% and recall rate of 96%. However, the proposed method's detection rate for Infiltration attacks was just 15%. Kumar *et al.* [13] proposed a misused-based technique for attack classification in an IDS. The size of the dataset was reduced by removing redundant data from the dataset. The dataset was divided into 15 clusters, and the samples from dominating classes were selected for processing, resulting in reduced size of the dataset. Further, features were selected using information gain. Finally, an integrated rule-based model was used for final classification. UNSW-NB15 dataset was used for the evaluation of the proposed method. They claimed to have achieved 84.5% Mean F1-score (MFM), 90.32 Attack Detection Rate (ADR), and 2.01% False Alarm Rate (FAR) using this approach. As the suggested solution by the author is based on a misuse-based strategy, it will not be able to detect any publicly unknown zero-day attacks. Roshan *et al.* [14] proposed an adaptive IDS using extreme learning and clustering. They used the NSL-KDD dataset and claimed to have achieved an accuracy rate of 89% for the new attack type sample. The proposed techniques are evaluated on the outdated NSL-KDD dataset and require human intervention to update the data over time. Feng *et al.* [8] designed a plug-and-play capture tool to grab the packets and detect three network attacks. To detect Denial of Service (DoS) attacks, they used DNN. Two deep learning techniques were used to detect XSS(Cross-Site Scripting) and SQL attacks: CNN and LSTM. This model is just trained for the detection of DoS, XSS and SQL attacks. Ma *et al.* [15] proposed a novel approach named SCDNN, which uses Spectral Clustering (SC) and DNN algorithms for classification. KDD-CUP99 and NSL-KDD datasets and a sensor network were used to evaluate the model. They claim that their approach outperforms Back Propagation Neural Network (BPNN), SVM, RF, and Bayes tree models in detection accuracy.

The disadvantage of the SCDNN is that its weight parameters and DNN layer thresholds must be tuned, and the clusters' k and parameters must be determined empirically rather than theoretically. Furthermore, the model is tested on historical and outdated datasets that might fail to perform in real world scenarios.

Ferrag *et al.* [7] compared seven deep learning techniques: DNN, RNN, Restricted Boltzmann Machine, Deep Belief Networks, CNN, Deep Boltzmann Machine, and Deep Autoencoders. The RNN gave the highest attack detection rate for seven attack types, namely Brute Force-XSS, Brute Force-Web, DoS attack Hulk, DoS attack SlowHTTPtest, DoS attack Slowloris, DoS attack Golden Eye, and Infiltration. CNN outperformed other techniques for attacks like DDoS attack HOIC, DDoS attack LOIC-UDP, DDoS attack LOIC-HTTP and Botnet. Though deep learning approaches give promising results on some of the attack categories. However, the entire experiment was conducted on only 5% of the total dataset, yielding a relatively small number of attack occurrences. Karatas *et al.* [4] proposed six machine learning-based IDS using k-Nearest Neighbour, RF, Gradient Boosting, Adaboost, DT, and Linear Discriminant Analysis algorithm. According to the author, CIC IDS 2018 dataset comprises 5 million samples, but the CIC IDS 2018 dataset has 16 million samples. So, the above-proposed work is on a subset of the dataset in which DDoS and web attacks are missing. Besides taking a small subset of the entire dataset, a lot of oversampling is done. SQL injection samples in the given work were oversampled from 53 to 286,191 samples. Considering the infiltration attack, the original dataset had 160,639 samples, whereas the author took just 93,063 samples and then oversampled it to 286,191. Most of the author's experimentation was based on the oversampled data whereas our proposed work is more on the original data. Moreover, the authors evaluated the model based on attack detection accuracy, and other essential metrics like Precision, F1-score, and Specificity were missing.

The advantages and the drawbacks of various approaches discussed above are listed in Table 2.

Though numerous methods are proposed to optimize attack detection algorithms, there is still much room to improve previous research results. Two major concerns considered in this research are as follows:

- In the past, most of the research in this field is on outdated datasets that do not reflect modern-day attack trends [16].
- Many machine learning and deep learning techniques have been proposed for attack detection, but attack wise accuracy obtained using these approaches is not high enough for some of the attack categories. From the previous research, it is also evident that it is challenging

TABLE 2. Comparison of various IDS approaches.

S. no	TITLE	ADVANTAGES	DRAWBACKS
1.	An Adaptive Ensemble Machine Learning Model for Intrusion Detection[2]	The ensemble method is proposed that can take the advantage of various algorithms to detect different attacks more accurately.	The outdated dataset used, feature selection, and data preprocessing techniques could be optimized. No technique was used to balance the skewed IDS datasets
2.	Building an efficient Intrusion Detection System based on feature selection and ensemble classifier[9]	The proposed technique combines the benefits of feature selection and ensemble classifier to enhance the performance of the IDS.	The suggested method is evaluated using subsets of the datasets. These subsets are insufficient to cover all modern-day attacks. Further, no approach is employed to deal with the IDS dataset's imbalanced nature.
3.	TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System [11]	Two-stage classifier is proposed based on a hybrid feature selection methodology combining three methods (Particle Swarm Optimization, Ant Colony algorithm, and Genetic algorithm).	The proposed technique has an accuracy of 85.8%, which can be further improved. The author did not employ any strategies to address the imbalanced nature of the IDS datasets.
4.	Dynamic Network Anomaly Detection System by Using Deep Learning Techniques [12]	LSTM with AM is used to build the neural network model that addresses the time-correlated network traffic's classification issues.	The model is not able to correctly classify all kind of attacks. The accuracy rate for Infiltration attack is just 15%
5.	An integrated rule-based Intrusion Detection System: analysis on UNSW-NB15 data set and the real-time online dataset [13]	The proposed model generates its real time data for the test set	The proposed method is based on a misuse-based strategy, it will not be able to detect any publicly unknown zero-day attacks
6.	Adaptive and online network intrusion detection system using clustering and Extreme Learning Machines [14]	The technique allows for quick learning and detection in real time.	The suggested model is tested on the NSL-KDD dataset, which is obsolete and does not reflect modern-day attack patterns.
7.	Anomaly detection in ad-hoc networks based on deep learning model: A plug and play device [8]	The proposed method can be extended to all other attacks with minor modifications in ad-hoc networks	This model is just evaluated for detection of XSS and SQL attacks
8.	A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks [15]	In real-world security systems, the suggested method may efficiently classify sparse attack situations and improve detection accuracy.	The disadvantage of the SCDNN is that its weight parameters and DNN layer thresholds must be tuned, and the clusters' k and parameters must be determined empirically rather than theoretically. Furthermore, the model is tested using outdated datasets.
9.	Deep learning for cybersecurity intrusion detection: Approaches, datasets, and comparative study [7]	The paper describes 35 well-known cyber datasets. On two recent datasets, the performance of seven deep learning techniques is evaluated.	The experiment was carried out only on 5% of the complete dataset. The imbalance concerns in the skewed dataset were not addressed using any approach. Furthermore, the deep learning models were only assessed for attack detection accuracy, other metrics like precision rate and F1-score were missing.
10.	Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset [4]	The author tries to balance the dataset for a better attack detection rate, the author uses a synthetic data creation approach called Synthetic Minority Oversampling Technique (SMOTE) to balance the skewed CIC IDS 2018 dataset.	The author claims that the CIC IDS 2018 dataset has 5 million samples, while the CIC IDS 2018 dataset contains 16 million. As a result, the work suggested above is based on a subset of the dataset that does not contain DDoS and web attacks.

for a single classifier to efficiently detect all kinds of attacks in a multi-attack classification environment [11]. A classifier that has a good attack detection rate for one attack might perform poorly for other attacks.

To overcome the above challenges, this study proposes a novel intrusion detection framework that takes advantage of each classifier's capability to detect a particular attack. The proposed method is evaluated using the latest CIC IDS 2018 dataset that overcomes the shortcomings of the legacy datasets.

III. MATERIALS AND METHODS

The Proposed Model:

Many researchers have proposed machine and deep learning techniques to detect attacks in anomaly-based IDS.

A single machine learning classifier cannot be used to detect different attacks effectively [9]. An Ensemble approach using a voting scheme is a common approach used to combine the results of different classifiers [9] In voting-based ensembles, predictions are the results of majority vote among the contributing models. In voting, all classifiers in the ensemble contribute to the final output regardless of whether the algorithm is capable of detecting the attack or not. Thus, to overcome the above drawback and increase the attack detection rate for multi-attack classification, we propose a novel framework for attack detection. In this approach, simple machine learning classifiers are ranked based on their efficiency in detecting various attacks. The classifiers based on the computed rank matrix contribute to the detection of different attacks.

The framework for intrusion detection comprises the standard procedure in machine learning:

(1) Data Pre-processing
   a. Data collection,
   b. Data Transformation
   c. Processing Skewed Dataset
   d. Outlier Rejection
(2) Feature Selection
(3) Training the Model
(4) Evaluating model performance

Fig. 1 illustrates the steps the study follows for the proposed framework for classifying attacks in an IDS. The following subsections give a detailed view of the steps mentioned here.

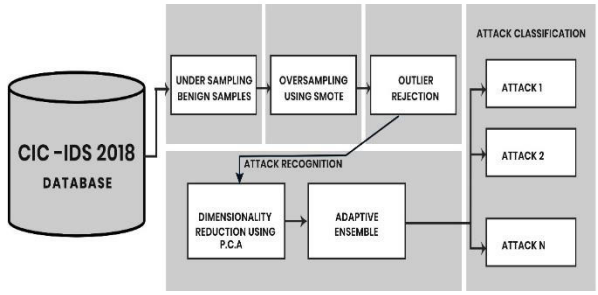

FIGURE 1. Steps for the proposed framework for intrusion detection.

A. DATA PRE-PROCESSING

1) DATA COLLECTION

All the experiments in this research use the CIC IDS 2018 dataset. The CIC IDS 2018 is the latest Intrusion Detection dataset that incorporates all the latest attacks based on famous vulnerabilities. It is published by Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC). This dataset is primarily generated to evaluate IDS, focusing on network-based anomaly detection. The attacking infrastructure had 50 machines, and the victim organization had 420 machines, 30 servers, and 5 departments. There are 80 features and approximately sixteen million rows in the dataset. Some of the features in the dataset are listed in Table 3.

TABLE 3. Some of the features in the CIC IDS 2018 dataset.

Name	Description	Name	Description
fw_pkt_l_avg	Average size of packet in forward direction	fw_psh_flag	Number of times the PSH flag was set in packets traveling in the forward direction (0 for UDP)
fw_pkt_l_std	Standard deviation size of the packet in the forward direction	bw_psh_flag	Number of times the PSH flag was set in packets traveling in the backward direction (0 for UDP)
Bw_pkt_l_max	Maximum size of packet in backward direction	fw_urg_flag	Number of times the URG flag was set in packets traveling in the forward direction (0 for UDP)
Bw_pkt_l_min	Minimum size of packet in backward direction	bw_urg_flag	Number of times the URG flag was set in packets traveling in the backward direction (0 for UDP)

2) DATA TRANSFORMATION

The CIC IDS 2018 dataset is a comprehensive dataset comprising various modern-day attacks. Fig. 2 lists the 14 attack classes present in the CIC IDS 2018 dataset. The 14 attack classes were reclassified into six classes based on the nature of the attack– Bot, Brute Force, DDoS, DoS, Infiltration, and Web attacks. Final dataset comprised of benign samples and samples from the above six attack categories.

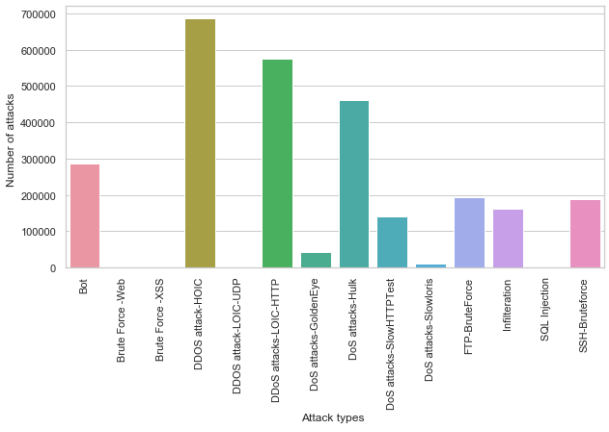

FIGURE 2. Attack distribution in the CIC IDS 2018 dataset.

3) PROCESSING SKEWED DATASET

In an intrusion detection dataset, only a small fraction of the dataset reflects attacks; this makes it challenging to

build efficient models with high attack detection rates [9]. An extremely biased prediction model is not practically useful, as the prediction is highly affected by the majority class samples [16]. CIC IDS 2018 is skewed Big Data with a high imbalance ratio for different attack categories (see Table 4). The mitigation of class imbalance of Big Data poses an even greater problem because of the comparatively diverse and nuanced nature of a large dataset. Many approaches are proposed in the research literature to handle imbalanced data [18].

TABLE 4. Imbalance ratio of the samples in the CIC IDS 2018 dataset.

Class	No of samples	%(volume)	Imbalance Ratio
Benign	13390235	82.98%	Majority Class
Bot	286191	1.77%	46.78775713
BruteForce	380943	2.36%	35.15023245
DDoS	1263933	7.83%	10.5941019
DoS	654300	4.05%	20.46497784
Infiltration	160639	1.00%	83.35606546
Web Attacks	928	0.01%	14429.13254
Total	16137169		

The imbalance ratio [19] is defined as

$$R = S_{maj}/S_{min}$$

where S_{maj} is the number of majority class samples, and S_{min} is the number of samples of the minority class [20]. The greater the value of R, the higher is the imbalance. If $R = 1$, then the dataset is perfectly balanced.

As evident from the above table that CIC IDS 2018 is a highly skewed dataset. One of the most prominent approaches for dealing with such unbalanced datasets is resampling (under sampling or oversampling) of data. Under sampling is deleting samples from the majority class, whereas oversampling is generating new samples in the minority class. SMOTE [21] is one of the most commonly used oversampling approaches in which synthetic samples for the minority class are created. Random oversampling is used to overcome the problem of overfitting. New samples are generated between the positive instances that are close together. The method is efficient as the new synthetic instances of the minority class are generated that are credible as they are similar in feature space to existing minority class samples.

To overcome the skewness in the dataset, both oversampling and undersampling techniques have been explored. It is evident in Table 4 that some of the attack categories, like web attacks, are highly skewed. Web attack samples are oversampled using SMOTE. Web attack samples are increased to 286,191 using SMOTE. As there are huge numbers of samples in normal traffic, it is under-sampled to 1.3 million samples (i.e. 10 % of the original count). After applying under sampling and oversampling, the imbalance ratio obtained is given in Table 5.

TABLE 5. Imbalance ratio of attack samples in the CIC IDS 2018 after pre-processing.

Label	No of samples	Volume (%)	Imbalance ratio
Benign	1390232	31.44%	Majority class
Bot	286191	6.47%	4.857706916
BruteForce	380943	8.61%	3.649448868
DDoS	1263933	28.58%	1.099925392
DoS	654300	14.80%	2.124762341
Infiltration	160639	3.63%	8.654386544
Web Attacks	286191	6.47%	4.857706916
Total	4422429	-	-

The imbalance ratio is considerably reduced from 14429.13 in web attacks before processing to just 4.85 in the newly generated set of rows. There is an overall reduction in the imbalance ratio for different attack classes reducing it to a max imbalance ratio of just 8.65.

4) OUTLIER REJECTION

While building machine learning models, it is very important to clean the dataset to ensure that it correctly represents the problem. The presence of outliers in a dataset can skew the data distribution and other statistical measures. In this paper, we have used the Isolation Forest method for outlier detection. The Isolation Forest method is a machine learning algorithm based on RF and a DT classifier used to classify anomalies in large data sets [22]. Isolation Forest is an unsupervised machine learning algorithm for anomaly detection. It is based on the principles of a DT classifier. It detects the outliers using the following steps:

1. Randomly select a feature from the features in the given feature space.
2. Randomly select a split value between the max and min value of the selected feature.
3. Random partitioning using the above split value results in shorter paths in trees for abnormal data points.
4. Distinguish such points from the rest of the data.

The proposed model uses a 0.1 contamination value for removing the outliers in the dataset using Isolation Forest. After outlier rejection, the dataset got reduced to 4.094,331 samples.

B. FEATURE SELECTION

The performance of a machine learning model heavily depends upon the features selected for training the model. Excessive features can result in poor performance of the model. So, choosing the right set of features is of paramount

TABLE 6. List of selected features using random forest.

Features Selected using the Random Forest			
Dst Port'	'Flow IAT Max'	'Fwd Pkts/s'	'Init Bwd Win Byts'
'Flow Duration'	'Flow IAT Min'	'Bwd Pkts/s'	'Fwd Seg Size Min'
'Tot Bwd Pkts'	'Fwd IAT Tot'	'Pkt Len Max'	'Idle Mean'
'TotLen Fwd Pkts'	'Fwd IAT Mean'	'Pkt Len Mean'	'Idle Max
'TotLen Bwd Pkts'	'Fwd IAT Std'	'RST Flag Cnt'	'Flow Pkts/s'
'Fwd Pkt Len Max'	'Fwd IAT Max'	'ACK Flag Cnt'	'Flow IAT Mean'
'Fwd Pkt Len Mean'	'Fwd IAT Min'	'Pkt Size Avg'	'Fwd Header Len'
'Bwd Pkt Len Max'	'Bwd IAT Std'	'Fwd Seg Size Avg'	'Bwd Header Len'
'Flow Byts/s'	'Bwd IAT Max'	'Subflow Fwd Byts'	'Init Fwd Win Byts'
'Subflow Bwd Pkts'			

importance while building machine learning models. Selecting the right features not only increases the efficiency of the model but also reduces the detection time [23]. Intrusion detection datasets are generally high-dimensional datasets. For better efficiency, selecting the right features in an IDS dataset is very essential.

CIC IDS2018 is a high-dimensional dataset with 80 features. Several features are correlated and redundant. Training on the redundant data will increase the complexity and time and may result in a flawed model. The proposed model used a hybrid feature selection using RF [24] and PCA [25]. First, the features are selected using RF feature selection. Feature importance using RF classifier is computed based on the average impurity of each feature in a tree in the forest. Using this method, 37 most important features were selected (see Table 6).

Second, PCA is used to identify top principal components for further processing. The PCA is a commonly used approach for dimensionality reduction. PCA transforms a dataset with a large number of features into a smaller one while retaining the required information in the dataset.

PCA is based on calculating the eigenvalues and the eigenvectors computed from the given dataset's covariance matrix. Given that x is an eigenvector calculated from the covariance matrix, the features extracted reliant to x, of any random vector v is

$$p = v^T x = \sum_{i=1}^{N} v_i x_i \tag{1}$$

where $x = [x_{1...}x_N]^T$, $v = [v_{1...}v_N]^T$ and N is the dimensionality of sample vectors [25].

The steps followed for dimensionality reduction in PCA are as follows:

1. Standardization of the continuous variables

Standardization of continuous variables is a crucial step while calculating the principal components as this method is overly sensitive with respect to the variance of values between different variables.

2. Computing the Covariance Matrix

An n x n matrix is computed to evaluate how the dataset variables vary with respect to each other. Highly correlated features contain redundant information. Computing covariance matrix helps in the identification of any such correlation. Positive covariance denotes the variables are directly proportional to each other. At the same time, a negative correlation indicates an inverse relationship between the variables.

3. Finding the Principal Components by Computing Eigen Values and Eigen Vectors of the Covariance Matrix

Principal components are the new extracted features that are highly significant and independent of each other. Principal components compress the most significant information scattered in the dataset. To compute the principal components, eigenvector and eigenvalues need to be calculated from the covariance matrix. Eigenvalues represent the magnitude of the extracted new feature space, whereas the eigenvector represents the direction.

4. Feature Vector

Eigenvalues and the corresponding eigenvectors calculated from the covariance matrix are sorted based on eigenvalues' absolute value. Feature vectors are selected from the most significant eigenvalues and eigenvectors computed in the above step. The resulting significant principal components are used for further training the models

After feature selection and dimensionality reduction, 24 most significant principal components were used for further processing.

C. TRAINING THE MODEL

An efficient IDS should have a high attack detection rate for all kinds of attacks. To achieve this, various machine and deep learning algorithms are proposed in the literature [7]. But a single algorithm is unable to detect all kinds of attacks effectively [2]. To accomplish this in this study, the model is trained using multiple machine learning techniques, which are finally combined using the proposed framework which is based on each algorithm's performance for classifying different attacks. The literature offers many algorithms, but we choose the seven most popular machine learning algorithms in our experiments for the classification of attacks in the CIC IDS2018 dataset. The details of the algorithms used here are given below:

1) k-NEAREST NEIGHBOUR

k-Nearest Neighbour (kNN) is one of the simplest and fundamental machine learning classifiers. kNN classification is based upon the classification of data points in a given feature space depending upon the distance between the given sample point X and its k neighbors. Finally, point X is classified according to the majority of its k nearest neighbor [23]. A set of observations $(z_1, y_1), \ldots, (z_n, y_n)$, where observations $z_i \in R^d$ and targets $y_i \in \{c1, c2\}$; then for given i, kNN grades the

neighbors of a test set and uses the class labels of the nearest neighbors to predict the class of the test samples [26].

Euclidean, Manhattan, or Minkowski function can be used for calculating the distance between the points. Equations (2)-(4) show how the above distance functions are computed.

$$Euclidean = \sqrt{\sum_{i=1}^{N} \left(x_i - y_i\right)^2} \tag{2}$$

$$Manhattan = \sum_{i=1}^{N} \left|x_i - y_i\right|^2 \tag{3}$$

$$Minkowski = \left(\sum_{i=1}^{N} \left(\left|x_i - y_i\right|\right)^k\right)^{\frac{1}{k}} \tag{4}$$

where N is the size of the dataset, k is a positive integer, and x_i and y_i are the data points. For evaluating the results, the value of k is taken as 5 and the Minkowski function is used for computing the distance.

2) DECISION TREE

A DT is a popular supervised machine learning algorithm [2]. A DT acquires knowledge from a given dataset in hierarchical structures comprising the root node, internal nodes, and leaf or terminal nodes. A leaf node determines the class to be assigned to a data point in the sample. Non-terminal nodes represent features with discrete rules for further classification. How well the DT classifies entirely depends upon how well the DT is trained. The DT starts from the root node, and then it splits into sub-trees based on some discrete rules for the given feature value [27]. This process is repeated recursively till the leaf nodes are generated. Though DT is simple and easy to implement, it may overfit. The problem of overfitting can be addressed using pruning. Entropy and information gain used in building a DT is calculated using the equations (5) and (6).

$$Entropy\,(D) = \sum_{x \in X} -r\,(a)\,E(a) \tag{5}$$

$$Gain\,(A, D) = Entropy\,(D) - \sum_{x \in S} -r\,(a)\,E\,(a) \tag{6}$$

D is the dataset, X is a set of classes, and r is the ratio of the number of elements in class x, and S is the subset of the dataset D. Gini impurity was used for training the model using the DT.

3) RANDOM FOREST

RF uses a bunch of DT classifiers for classification [24]. Each tree in a RF is trained on a bootstrapped sample from the dataset. The split attribute is randomly selected to divide the sample based on impurity using Gini index/entropy. Further, the results of all the trees are combined by voting for the final prediction. Most of the time, the RF results, even without using hyperparameters, are better than the DT algorithm. It is a widely used algorithm that provides speedy and accurate results.

In a RF algorithm, random samples are drawn from the dataset to build the tree. Split is performed at every cell of every tree to maximize the CART criterion over *mtry* directions [28] randomly selected from the available p options. The building of trees is halted when the count of every cell falls below the *nodesize* points. Where $a_n \in \{1, \ldots, n\}$: the number of sampled data points in every tree in the forest; $mtry \in \{1, \ldots, p\}$: Number of possible ways to split each node of each tree; $nodesize \in \{1, \ldots, a_n\}$: Minimum number of samples in each cell under which the cell is not split. 50 trees are taken to compute the results in our experiments using the RF classifier.

4) EXTRA TREES

Extra trees are extremely randomized trees that aggregate various decor-related trees in the forest for the final result [29]. The extra Trees algorithm works by generating a large number of unpruned decision trees from the training dataset. Both the attribute and cut-points are strongly randomized when a tree node is split, and the tree is grown on a complete dataset rather than the bootstrapped sample. The split in extra trees algorithm is based on two parameters: K, the count of attributes selected at each node, and nmin, the minimum sample size to split a node. The process is repeated several times to generate an ensemble. Extra trees ensemble depends upon three parameters: First M, the number of trees generates; Second, K determines the attribute randomization; and Third, the nmin parameter specifies the degree of smoothing. The extra trees algorithm is computationally fast than RF. As in Extra Trees, the split is selected randomly instead of finding the optimal one. Predictions are based on a majority vote for classification and average in case of regression.

5) EXTREME GRADIENT BOOSTING (XGB)

XGBoost is a tree boosting algorithm that optimally utilizes hardware and memory resources. It is almost ten times quicker than the latest techniques. The three primary gradient boosting techniques, namely Gradient Boosting, Regularized Boosting, and Stochastic Boosting, can be executed by XGBoost. Faster execution using parallel processing, portability, regularization, and tree pruning are the main benefits of the XGBoost algorithm.

This algorithm is based on gradient boosting techniques. The prediction using this algorithm is made using the combination of weighted input features such as $\hat{s_i} = \sum_j \theta_j p_{ij}$ [30]. There can be a number of parameters depending upon the data. Finding the appropriate parameters from a dataset is vital to ensure the performance of the algorithm. The predicted value is used for computing the final output.

6) HISTOGRAM-BASED GRADIENT BOOSTING

Gradient Boosting ensembles are generally not time efficient. Training of the trees can be improved by binning the

continuous variables while training the model. Gradient Boosting ensemble that bins the continuous values to speed up the model is Histogram-Based Gradient Boosting (HBGB). HBGB is inspired by the Light Gradient Boosting machine- by Microsoft [31].

7) LightGBM

Microsoft developed LightBGM as a boosting framework in 2017. LightGBM [31] is an enhanced version of the Gradient Boosting Decision Tree (GBDT) algorithm. This framework outperforms Xgboost in terms of performance, speed, and power. LightGBM, unlike other GBDT techniques, is still effective when the data is large and has many dimensions. This is due to the following two distinct strategies: One-Side Sampling based on Gradients (GOSS) and a Special Feature Bundle (EFB). It is a tree-based method that supports categorical features, making feature numerical transformation and normalization unnecessary during the data preparation step. The tree-growth method based on leaves makes the matching process more efficient during the decision-making process.

Hyperparameter values of Machine Learning algorithms used for experimentation to evaluate the proposed approach are listed in Table 7.

TABLE 7. Machine learning algorithms hyperparameter values.

Algorithms	Hyperparameters
KNN	Neighbors=5,Weights=uniform,Distance=minkowski
DT	Criterion=Gini,Splitter=Best,Min_samples_split=2, Min_samples_leaf=1
RF	n_estimators=50,criterion='gini', max_depth=5, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,max_features='auto', max_leaf_nodes=None,min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True,
Extra Trees	n_estimators = 50, max_features = 5, class_weight= 'balanced'
Extreme Gradient Boosting	max_depth=3, n_estimators=100, learning_rate=0.05
Histogram Based Gradient Boosting	Loss=auto, Max_iter=200, Learning rate=0.1, min_samples_leaf=20, max_bins=255
LightGBM	Num_boost_round=350, learning_rate=0.03

After presenting the commonly used machine learning algorithms, now we present our proposed framework that takes each algorithm's best to classify the attacks as in multi-classification problems, as a single algorithm is not sufficient to effectively detect all classes in a dataset.

8) THE PROPOSED ENSEMBLE FRAMEWORK FOR INTRUSION DETECTION SYSTEM

It is challenging to use a single classifier to predict the attack classes in a multidimensional dataset with multiple attack classes. This paper proposes a novel ensemble framework for effectively detecting attacks of different types in an IDS. Fig. 3 illustrates the essential components for the proposed framework. The proposed framework is based on each classifiers' ability for detecting different attacks.

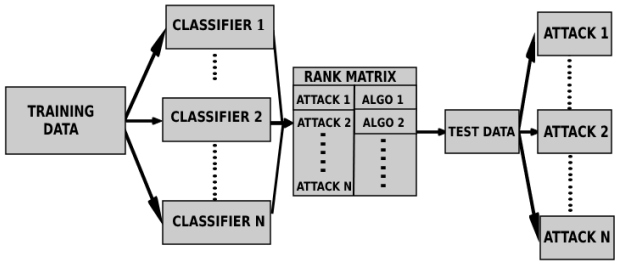

FIGURE 3. The proposed framework for intrusion detection.

Algorithm The Proposed Algorithm

Input:

$$Tr = \{(a_1, b_1, c_1 \ldots .z_1), (a_2, b_2, c_2 \ldots, z_2) \ldots\ldots (a_n, b_n, c_n \ldots, z_n)\}, a_i, b_i, c_i \ldots, z_i \in F,$$

$$Ts = \{(a_1, b_1, c_1 \ldots .z_1), (a_2, b_2, c_2 \ldots, z_2) \ldots\ldots (a_m, b_m, c_m \ldots, z_m)\}, a_i, b_i, c_i \ldots, z_i \in F,$$

$$Yr = \{x_1, x_2 \ldots . x_n\}, xi \in X, C = \{c1, c2 \ldots ct$$

where: F is the set of features in the dataset, Tr is the training set, Ts is the test set, X is the set of labels to be predicted, and c1-ct in C is the set of classifiers.
Output = Labels
Algorithm:

1. Train all the classifiers c_i in C on each row r in the training data Tr.
2. Calculate the F1-score of each classifier c_i for every attack class x_i.
3. Assign the attack detection rank r_{ij} for each attack x_i for each classifier c_j in C. Classifier with the best F1-score for a particular attack gets the highest rank.
4. Predict the class for each row in Testing set Ts for high-ranked classifiers in C.
5. The results of the highest rank classifiers are compared for the final result. Considering c_i as the best classifier for predicting the attack x_i.
   a. The result r_{ci} (prediction result by classifier c for the sample i) is compared to check if it predicts the attack class x_i.
      i. If a match is found, it is added to the result.
      ii. If a conflict is found or no match is found, then the classifier's result with the higher F1-score is considered.

D. EVALUATION METRICS

Several performance metrics exist to evaluate machine learning classification algorithms [32].

Accuracy is the percentage of samples that are correctly classified in (7), as shown at the bottom of the next page.

Sensitivity/Recall is the ratio of samples correctly classified as attack with all attack samples.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (8)$$

Specificity is the ratio of correctly classified benign samples to the total no of benign samples.

$$\text{Specificity} = \frac{TrueNegative}{TrueNegative + FalsePositive} \tag{9}$$

Precision is the ratio of samples correctly classified as attacks to the total samples categorized as attacks.

$$Percision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{10}$$

F- Measure is the harmonic mean of Recall and Precision.

$$F - Measure = 2 * \frac{Percision * Recall}{Percision + Recall} \tag{11}$$

Prediction latency = Average time taken to predict the attack class for the test set. The time is calculated by finding the difference between the start and the end time on the server while training and testing the classifier in each fold during 10-fold cross-validation. There were 52,345 samples in each fold. The calculated results are the mean of the values computed in each fold.

For the proposed classification model, we consider F1-score as the best metric to evaluate the algorithm's performance. The performance of a model may decrease if the classifier is evaluated just based on the attack detection rate. A classifier with a good Recall rate but with poor Precision may correctly classify one attack category but may misclassify other attacks (due to low Precision) resulting in decrease in attack detection rate of an IDS. Thus F1-score metric is used for selecting the best algorithm for each attack category to increase the overall accuracy and the attack detection rate.

Though F1-score is the ideal metric for selecting the best machine learning method for each class in the proposed algorithm, however Accuracy and Recall rate are used in the comparative results. The recall rate indicates the model's ability to recognise True Positives reliably, whereas Accuracy is the ratio of correctly categorised predictions to total predictions. The primary goal of an IDS is to accurately identify assaults and protect the network against intrusions. Thus, accuracy and recall rate are utilised to compare the findings.

Section IV presents the results of the different machine learning algorithms and the proposed framework.

IV. RESULTS AND DISCUSSIONS

This section evaluates the performance of the proposed Intrusion Detection framework on the CIC IDS2018 dataset. The proposed model was implemented using the Scikit-learn library in Python. All the experiments were executed on the AWS cloud using the configuration in Table 8.

In this study, the dataset was trained on seven machine learning algorithms to find the best method to detect different attack categories. Based on the performance result of each machine learning algorithm, a rank matrix was calculated. Based on the rank matrix, the results of the best performing algorithm are considered for the final attack prediction. To reduce variation in the performance results, k-fold cross-validation with k = 10 was used for computing all the results. The proposed model is evaluated using the following performance metrics [32]: Accuracy, Precision, Recall, and F1-score.

TABLE 8. AWS configuration used for the experiments.

Hardware	Properties
VCPU	32
PLATFORM	AMAZON LINUX(Version 31.0)
MEMORY	256 GB
INTERNAL STORAGE	24* 1980 GB
NETWORK PERFORMANCE	25 GIGABIT

TABLE 9. k-nearest neighbor.

Attack	Recall	Precision	F1-score
Benign	95.92%	93.34%	94.61%
Bot	99.99%	100.00%	99.99%
BruteForce	99.64%	84.06%	91.19%
DDoS	99.99%	99.99%	99.99%
DoS	88.80%	99.75%	93.96%
Infiltration	41.40%	54.32%	46.99%
Web attacks	100.00%	99.96%	99.98%

TABLE 10. Decision tree.

Attack	Recall	Precision	F1-score
Benign	93.64%	92.03%	92.83%
Bot	100.00%	99.99%	99.99%
BruteForce	94.48%	83.81%	88.83%
DDoS	99.99%	99.99%	99.99%
DoS	89.30%	96.50%	92.76%
Infiltration	37.03%	42.87%	39.74%
Web attacks	99.94%	99.91%	99.92%

A. RESULTS OF THE PROPOSED MODEL IN COMPARISON TO OTHER MACHINE LEARNING ALGORITHMS

1) ANALYZING INDIVIDUAL PERFORMANCE OF THE BASE ALGORITHMS

The experimentation started with an analysis of the performance of the base classifiers for the dataset. The performance metrics for the seven base algorithms for different attack categories are listed in Tables 9-15.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalseNegative + TrueNegative + FalsePositive} \tag{7}$$

TABLE 11. Random forest.

Attack	Recall	Precision	F1-score
Benign	92.63%	91.10%	91.86%
Bot	99.63%	99.91%	99.77%
BruteForce	99.94%	72.78%	84.23%
DDoS	99.84%	98.62%	99.23%
DoS	77.33%	99.93%	87.19%
Infiltration	31.87%	45.91%	37.62%
Web attacks	98.81%	90.37%	94.40%

TABLE 12. Extra trees.

Attack	Recall	Precision	F1-score
Benign	89.35%	92.57%	90.93%
Bot	100.00%	100.00%	100.00%
BruteForce	95.91%	82.71%	88.82%
DDoS	100.00%	99.99%	99.99%
DoS	88.25%	97.35%	92.58%
Infiltration	44.30%	34.86%	39.02%
Web Attacks	99.94%	99.96%	99.95%

TABLE 13. HBGB classifier.

Attack	recall	precision	F1-score
Benign	99.22%	92.32%	95.64%
Bot	100.00%	100.00%	100.00%
BruteForce	98.43%	84.76%	91.08%
DDoS	100.00%	99.99%	99.99%
DoS	89.51%	98.97%	94.00%
Infiltration	29.31%	81.49%	43.12%
Web attacks	99.96%	99.98%	99.97%

TABLE 14. XGB classifier.

Attack	Recall	Precision	F1-score
Benign	98.90%	90.27%	94.39%
Bot	99.70%	99.93%	99.81%
BruteForce	99.39%	82.53%	90.18%
DDoS	99.93%	99.83%	99.88%
DoS	87.55%	99.26%	93.04%
Infiltration	20.58%	83.46%	33.01%
Web attacks	97.12%	97.28%	97.20%

As given in the above tables, it is evident that a single algorithm is not sufficient to detect all kinds of attacks. It is observed that some of the classifiers have a high Recall rate but a poor Precision rate. For instance, the RF classifier's performance results for detecting Brute Force attack show a high Recall rate of 99.9% but a low Precision rate of 72.78%. Choosing RF as the best algorithm for detecting Brute Force will increase the detection rate for Brute Force but decrease the detection rate of other attacks due to its low Precision rate. So, the F1-score metric is used to evaluate the algorithm's performance that combines both the Recall rate and the Precision rate.

TABLE 15. LightGBM classifier.

Attack	Recall	Precision	F1-score
Benign	99.53%	87.91%	93.36%
Bot	99.98%	100.00%	99.99%
BruteForce	81.78%	98.80%	89.49%
DDoS	99.97%	99.99%	99.98%
DoS	99.43%	90.69%	94.86%
Infiltration	38.79%	95.09%	55.10%
Web attacks	99.97%	99.76%	99.86%

2) CONSTRUCTING A MODEL WITH ACCURACY AS THE OBJECTIVE

To compute the results, classification algorithms are ranked (refer algorithm in Section 3) based on their F1-score (see Tables 9-15). The rank matrix (see Table 16) is calculated based on each classifier's ability to detect different attack categories. Seven algorithms are used for building the ensemble: Histogram based gradient descent, RF, DT, Extra Trees, kNN, XGBoost, and LightGBM. The Histogram Based Gradient Boosting algorithm is listed for Bot, Benign, and DDoS categories in the rank matrix as it has the highest F1-score score for the listed attack categories among all the base algorithms. Similarly, the rank matrix is computed for other attack categories.

TABLE 16. Rank matrix for attack classification.

S.no	Attack Category	Best Ranked Algorithm	F1-score
1	Benign	Histogram Based Gradient Boosting	95.64%
2	Bot	Histogram Based Gradient Boosting	100.00%
3	Brute Force	K-Nearest Neighbour	91.19%
4	DDoS	Histogram Based Gradient Boosting	99.99%
5	DoS	LightGBM	94.86%
6	Infiltration	LightGBM	55.10%
7	Web attacks	K-Nearest Neighbour	99.98%

To predict the attack category, the incoming traffic is classified by all the algorithms in the rank matrix. In contrast to the voting approach [11], where the final class is based on the voting by all the classifiers irrespective of a classifier's ability to detect the attack, our approach uses a technique in which prediction of a classifier is considered only if its attack detection efficiency is the highest for that attack. So, for computing the final class the output of the classifier is considered only if it is the best ranked algorithm for detecting that attack or has the highest F measure.

Table 17 shows the sample classification test results using the proposed approach. Sample 1 is classified as DDoS by Histogram Based Gradient Boosting, LightGBM

TABLE 17. Example of computing results as per the proposed framework.

Samples	Histogram Based Gradient Boosting	K-Nearest Neighbour	LightGBM	Final Result
Sample 1	DDoS	DDoS	DDoS	DDoS
Sample 2	DoS	Bot	DoS	DoS
Sample 3	Benign	Infiltration	Infiltration	Infiltration
Sample 4	Bot	Brute Force	DDoS	Bot
Sample 5	Brute Force	DDoS	Web Attacks	DDoS

and k-Nearest Neighbour. Since Histogram Based Gradient Boosting is the best ranked algorithm for detecting DDoS in the rank matrix, so its result is taken as the final output. Sample 2 is classified as DoS by Histogram Based Gradient Boosting and LightGBM and Bot by kNN. Since LightGBM is the best ranked algorithm for DoS so its output is taken as the final output while dropping the output of the kNN algorithm since kNN is not the best classifier for Bot attacks as per the rank matrix. Similarly, sample 3 is classified as Infiltration since LightGBM is the highest-ranked algorithm for predicting Infiltration attacks. In sample 4, the results of the classifier are as per the highest-ranked algorithms in the rank matrix. To resolve the conflict, the final result is based on the algorithm with the higher F1-score. Since Histogram Based Gradient Boosting has a higher F1-score rate of 100% which is higher in comparison to F1-score score of 91.9% for kNN, so the final attack category is Bot in sample 4. In sample 5, the results of all the classifiers are not as per the highest ranked algorithm in the rank matrix, the final output is based on the algorithm with a higher F1-score Score. The F1-score for detecting brute force attack by histogram based gradient boosting is 91.08% whereas the F1-score of KNN for detecting DDoS is 99.9%, and F1-score for detecting DDoS by LightGBM is 99.98%. So the final output for sample 5 is DDoS.

Table 18 shows the results using the proposed framework based on the rank matrix results in Table 16. The proposed design adjusts the trade-offs of the base classifiers, its results outperform the results of individual algorithms.

TABLE 18. Performance of proposed framework for attack classification.

Attack	Recall	Precision	F1-score
Benign	98.63%	94.15%	96.33%
Bot	100.00%	99.99%	100.00%
BruteForce	98.47%	99.05%	98.76%
DDoS	99.99%	100.00%	100.00%
DoS	99.43%	99.10%	99.26%
Infiltration	63.79%	88.84%	74.26%
Web attacks	100.00%	99.82%	99.91%

3) CONSTRUCTING A MODEL WITH ACCURACY ALONG WITH TIME EFFICIENCY AS AN OBJECTIVE

In real time IDS prediction latency is of paramount importance. Models just based on accuracy fail in real time systems since delay in prediction affects the system's overall performance. Thus, for building time efficient IDS using the proposed approach we consider the prediction latency for each algorithm as listed in Table 19.

TABLE 19. Comparison of prediction time.

Machine Learning Algorithm	Prediction Time (in secs) for Test Set
Histogram Based Gradient Descent	0.300503
ExtraTrees	0.345334
Decision Tree	0.21345
RandomForest	0.33835
XGBoost	0.182613
KNN	186.4265
LightGBM	0.138008

For building time efficient ensemble using the proposed approach the algorithms are filtered on the basis of time efficiency. As give in Table 19, it is observed that kNN is very slow in comparison to the other machine learning algorithms.

Thus, kNN is not considered for building the model as time efficiency is critical in IDS. The new ensemble is built using Histogram Based Gradient Descent, Extra Trees, RF, XGBoost, KNN, and LightGBM as base algorithms. Based on the F1-score (see Tables (9-15). the rank matrix (see Table 20) is computed using the proposed approach.

TABLE 20. Rank matrix for attack classification using HBGB & LightGBM.

S.no	Attack Category	Best Ranked Algorithm	F1-score
1	Benign	Histogram Based Gradient Boosting	95.64%
2	Bot	Histogram Based Gradient Boosting	100.00%
3	Brute Force	Histogram Based Gradient Boosting	91.08%
4	DDoS	Histogram Based Gradient Boosting	99.99%
5	DoS	LightGBM	94.86%
6	Infiltration	LightGBM	55.10%
7	Web attacks	Histogram Based Gradient Boosting	99.97%

The rank matrix in Table 20 lists the best classifiers for detecting particular attack categories based on the F1-score score. As per the rank matrix, Histogram Based Gradient Boosting and LightGBM are the best classifiers for detecting different attack categories in CIC IDS 2018 dataset. The ensemble is built using the two selected classifiers and the proposed algorithm and the results are listed in Table 21 using the rank matrix in Table 20.

Table 22 compares the attack detection rate of the two ensembles based on the proposed approach. Further, the two

TABLE 21. Performance of proposed framework with LightGBM and histogram based gradient descent.

Attack	Recall	Precision	F1-score
Benign	99.63%	92.18%	95.76%
Bot	99.98%	100.00%	99.99%
BruteForce	96.48%	99.04%	97.75%
DDoS	99.99%	100.00%	99.99%
DoS	99.44%	97.95%	98.69%
Infiltration	62.25%	97.46%	75.98%
Web attacks	99.92%	99.91%	99.92%

TABLE 22. Comparison of attack detection rate of proposed ensemble.

ATTACK	RECALL RATE	RECALL RATE
	KNN+ HGBG +LIGHTGBM	HGBG+LIGHTGBM
Benign	98.63%	99.63%
Bot	100.00%	99.98%
Bruteforce	98.47%	96.48%
DDos	99.99%	99.99%
DoS	99.43%	99.44%
Infiltration	63.79%	62.25%
Web Attacks	100.00%	99.92%

ensembles are compared in Table 23 in terms of the time taken for predicting the results.

TABLE 23. Comparison of time efficiency of proposed ensembles.

Ensemble	Prediction Latency in seconds
KNN+HBGB	214 .34
HBGB+LightGBM	6.13

It is evident from the above results that the time taken by the ensemble approach depends upon the base classifiers used for building the ensemble. As an IDS is a real time system and prediction latency is very important, selecting classifiers based on time efficiency is crucial. Thus, the ensemble using LightGBM and HBGB is preferred as it has a comparable attack detection rate in comparison to KNN+HBGB model and low prediction latency.

B. COMPARISON WITH OTHER ENSEMBLE APPROACHES

The ensemble approach is widely used to combine the results of several classifiers for improving the performance of the system. Some commonly used ensemble methods are Voting [9], [33], Boosting [33], and Stacking [33]. Thus, to evaluate the performance of our proposed method, we compare the results of our proposed ensemble using LightGBM and HBGB with Voting and Stacking methods in Tables (24-26). The results of various Boosting methods are evaluated in the table (13-15).

The Voting classifier estimator is created by merging several classification models, resulting in a powerful meta-classifier that compensates for the shortcomings of individual classifiers on a given dataset. The predictions from various models are combined in a voting ensemble as per the majority of the results of the participating models.

TABLE 24. Performance of voting (soft)ensemble for attack classification.

Attack	Recall	Precision	F1-score
Benign	98.95%	86.66%	92.40%
Bot	99.99%	100.00%	99.99%
BruteForce	97.91%	84.72%	90.84%
DDoS	99.98%	99.99%	99.98%
DoS	89.48%	98.62%	93.82%
Infiltration	31.86%	87.44%	46.70%
Web attacks	99.98%	99.80%	99.89%

TABLE 25. Performance of voting (hard) ensemble for attack classification.

Attack	Recall	Precision	F1-score
Benign	99.02%	86.55%	92.37%
Bot	99.99%	100.00%	100.00%
BruteForce	98.39%	82.69%	89.86%
DDoS	99.98%	99.98%	99.98%
DoS	87.73%	98.91%	92.98%
Infiltration	31.22%	87.88%	46.07%
Web attacks	99.91%	99.92%	99.91%

To evaluate our proposed approach the results are evaluated using both hard and soft voting ensembles using LightGBM and HBGB classifier.

On comparing the above results of the voting classifier with the results of our proposed method in Table 21 it is evident that our method outperforms the voting classifier results.

Further, our results are compared with the Stacking ensemble built using Histogram Based Gradient Descent and LightGBM as a base classifier and combining its results using Logistic Regression as the meta classifier. Stacking is based on heterogeneous weak learners, trains them in parallel, and then combines them by training a meta-model to output a prediction based on the predictions of the many weak models. To evaluate the performance of our proposed approach the results of a stacking ensemble with LightGBM and HBGB is Listed in Table 26 and a comparison between the Stacking ensemble, Voting approaches, and our proposed ensemble is done in Table 27.

C. COMPARING THE TIME EFFICIENCY OF THE PROPOSED MODEL WITH OTHER STATE-OF-THE-ART TECHNIQUES ON CIC IDS 2018 DATASET

In real time problems such as IDS, prediction time is very crucial. Algorithms with high prediction lags cannot be deployed in the real world as it can adversely affect the performance of the system.

In our proposed framework, the time efficiency of the ensemble is based on the time efficiency of the base

TABLE 26. Performance of stacking ensemble for attack classification.

Attack	Recall	Precision	F1-score
Benign	98.81%	86.71%	92.37%
Bot	99.98%	98.12%	99.04%
BruteForce	72.10%	98.83%	83.37%
DDoS	99.96%	99.96%	99.96%
DoS	99.52%	86.43%	92.51%
Infiltration	32.05%	86.02%	46.70%
Web attacks	99.60%	99.89%	99.75%

TABLE 27. Comparison of proposed approach with existing ensemble methods.

Approach	Recall	Accuracy
Voting ensemble (Hard)	94.15%	95.49%
Voting Ensemble (Soft)	94.21%	95.52%
Stacking	94.02%	95.38%
Proposed Approach	96.7%	97.5%

TABLE 28. Comparison of proposed approach with existing ensemble methods and deep learning approaches in terms of prediction latency.

Deep Learning Approach	Accuracy	Prediction Latency (Seconds)
Deep Neural Network (DNN)	96.653%	9.496
Recurrent neural networks (RNN)	96.886%	11.3063
Convolutional neural networks (CNN)	96.913%	37.36531
Deep Learning method using LSTM	96.2%	13.880
Voting Ensemble (Hard)	95.49%	403
Voting Ensemble (Soft)	95.52%	394
Stacking Ensemble	95.38%	15
Proposed Model (LightGBM+HBGB)	97.5%	6.13

models. Thus, to evaluate the time efficiency of the proposed model, the prediction latency of various models using state-of-the art techniques is compared to the proposed model in Table 28. Deep learning models were trained with 0.5 learning rate,15 hidden nodes and 1000 batch size.

Prediction latency computed in the table is the prediction time taken by a classifier to predict a test set in a single fold in 10 fold cross validation.

As evident from the table above our approach outperforms the other approaches in terms of both accuracy and time efficiency with highest accuracy rate of 97.5% and lowest prediction time of 6.13 seconds.

D. COMPARATIVE ANALYSIS OF THE PROPOSED MODEL AND THE RECENTLY CITED WORK

A comparison is made between our work and some existing Intrusion Detection research [3], [7], [12] on the CIC IDS2018 dataset in this context to further evaluate the proposed model's efficacy. In some of the research papers analyzed during the literature review, it has been observed that the data size left is too small for the application of deep learning techniques [7]. Whereas in some of the previously proposed methods oversampling issue has been predominately consuming the resources and the real output in terms of efficacy is missing. The data has been more or less synthetic in nature [4]. So, our work is focusing on keeping the originality of the data intact. In our study, we have also tried to make our results consistent, whereas inconsistency is also observed in some of the previous work in the literature [3]. Below we compare our proposed work with recent work in this domain.

Ferrag *et al.* [7] analyzed seven deep learning models for cybersecurity intrusion detection on the CIC IDS 2018 dataset. But the entire experimentation was done on only 5% of the entire dataset, resulting in a very small number of attack instances. Deep learning models learn better on huge datasets but in this study, the dataset was reduced from 16 million rows to 0.2 million rows. No technique was used to address the imbalance issues in the CIC IDS 2018 dataset. Moreover, all the models were evaluated for attack detection Accuracy, but in real world other performance metrics such as Precision rate, F1-score etc are also important. A Comparison of the proposed model with Deep Learning models of the Ferrag *et al.* [7] is listed in Table 29 and 30. Our proposed method outperforms the seven deep learning techniques for Benign samples, Web Attack, DoS, DDoS, and Bot attacks. The authors used a very small test size for evaluation of the deep learning models. For instance, brute force test samples were just 168, whereas in our study 38,015 brute force testing samples were used. Whereas the Deep Learning classification models have achieved reasonable efficiency, but the random testing instability can completely mask the learning curve due to the small test sample size [34].

TABLE 29. Comparison of the proposed model with deep learning models in Ferrag *et al.* [7].

Type	Proposed Approach	DNN	RNN	CNN
Benign	99.63%	96.95%	98.11 %	98.91%
BruteForce	96.48%	100%	100 %	100 %
Web Attacks	99.92%	88.49%	94.50 %	94.36 %
DoS	99.44	94.524 %	96.89 %	96.59 %
DDoS	99.99	97.73 %	97.98 %	98.60 %
Bot	99.98%	96.42 %	98.10%	98.98 %
Infiltration	62.25%	97.518 %	97.874 %	97.76 %

Atefinia and Ahmadi [3] proposed a DNN comprising a feed-forward module, a restricted Boltzmann machine, and two Recurrent Neural Networks based on CIC IDS 2018 dataset. The authors did not use any technique to address the highly skewed CIC IDS 2018 dataset's imbalanced issue. The results in the confusion matrix and the result table are inconsistent. Results of some of the attack types are also missing. A comparison between our results and web

TABLE 30. Comparison of the proposed model with deep learning models in Ferrag *et al.* [7].

Type	RBM	DBN	DBM	DA
Benign	97.316 %	98.212 %	96.215 %	98.101 %
BruteForce	100 %	100 %	100 %	100 %
Web Attacks	88.46 %	94.56 %	94.45 %	96.84 %
DoS	93.4215 %	95.28 %	95.8995 %	94.911 %
DDoS	96.62 %	96.98 %	96.96 %	96.69 %
Infilteration	96.4%	96.71%	96.16%	97.18%

TABLE 31. Confusion matrix of infiltration and web attacks using the proposed framework.

	Actual	Benign	Infiltration	Web attacks
Predicted	Benign	112325	408	5
	Infiltration	9502	15676	0
	Web attacks(Brute Force -Web,'Brute Force -XSS',SQL Injection)	0	0	16413

TABLE 32. Confusion matrix for infiltration attacks as in Atefinia and Ahmadi [3].

	Infiltration attacks		
		Benign	Infilteration
Actual	Benign	154518	371
	Infiltration	31731	397
		Predicted	

TABLE 33. Confusion matrix for web attacks as in Atefinia and Ahmadi [3].

Web Attacks				
SQL Injection	416980	1	0	0
Brute Force-XSS	66	56	0	0
Brute Force-Web	15	31	0	0
Benign	9	9	0	0
Actual	Sql Injection	Brute Force-XSS	Brute-Force-Web	Benign
	Predicted			

attacks confusion matrix is done in tables 31-33. In Web and Infiltration attacks classes 1, 2, 3, and 4 represent Benign, Brute Force-web, Brute Force -XSS, and SQL Injection, respectively. The confusion matrix listed below for our proposed model is the aggregate of results from 10-fold cross validation.

It is evident from the above results that our proposed model clearly outperforms the multi-architectural modular DNN by Atefinia and Ahmadi [3].

Finally, Lin *et al.* [12] proposed Deep Learning method using LSTM +AM.A comparison is made between our work and recent work to further evaluate our proposed method's efficacy [12] in Table 34.

TABLE 34. Comparison of the proposed framework with the deep learning approach of Lin *et al.* [12].

Models	Sensitivity	Accuracy
Deep Learning method using LSTM +AM Reference	96%	96.2%
Proposed framework using machine learning techniques	96.7%	97.5%

The referenced model and the proposed model in Table 34 are trained on the latest CIC IDS2018 dataset. As the results depict, the proposed model has achieved a higher accuracy rate of 97.5 %—a higher recall rate of 96.7% compared to the previously proposed approach. In Table 35, the approaches used in both models are compared.

TABLE 35. Comparison of the methodology of the proposed approach with the deep learning model.

	Lin et al. (2019) [12]	Proposed
Technique used	Deep Learning method using LSTM +AM Reference	Proposed framework using KNN+ Histogram Based Gradient Boosting + Extra Trees
Dataset used	CIC IDS 2018	CIC IDS 2018
No of samples used	2 million random samples	4 million samples
Skewed Dataset treatment	Used SMOTE and under sampling	SMOTE and under sampling

The comparative model is trained on 2 million samples, whereas our proposed model is trained on 4 million samples. The model build using the proposed framework has achieved a rise of 1.3% in accuracy rate and 0.7 % in recall rate compared to the existing model. Moreover, their technique has a very low detection rate of just 15 % for infiltration attacks.

The above results depict that the proposed method is an efficient approach for detecting multiple attacks in IDS. The proposed approach is time efficient and gives promising results on the latest CIC IDS 2018 dataset.

V. CONCLUSION

The study addressed the limitation of a single machine classification algorithm's ability to detect all attack types for Intrusion Detection. Another problem to be solved is that most of the existing machine/deep learning models-based approaches use outdated datasets. In addition, many existing studies which use the latest CIC IDS2018 dataset, don't use

the original data set and fail to address the data quality issues leading to poor classification models.

The study aimed to combine the detection abilities of multiple classifiers to improve attack detection accuracy. This paper proposes a machine learning model featuring a novel framework that brings together the advantages of several base classifiers for this, and different machine learning algorithms have been trained and tested on the latest CIC IDS 2018 dataset. Several algorithms were ranked using the F1-score for their ability to detect specific attacks. Out of them, LightGBM and HBGB were finally used to detect multiple attacks with high attack detection rates and low prediction latency using the proposed framework. The CIC IDS 2018 dataset is highly skewed, so the problem of class imbalance was addressed using a hybrid approach of under sampling of majority class and oversampling some of the attack classes using the SMOTE technique. This dataset balancing was done for training process optimization. The proposed approach enhances the detection accuracy of many attack categories in comparison to Deep Learning, Machine Learning, and the Voting ensemble approaches. Future work can explore unsupervised learning to train models on unlabeled datasets in the security domain.

REFERENCES

[1] L. Lv, W. Wang, Z. Zhang, and X. Liu, "A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine," *Knowl.-Based Syst.*, vol. 195, May 2020, Art. no. 105648, doi: 10.1016/j.knosys.2020.105648.

[2] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019, doi: 10.1109/access.2019.2923640.

[3] R. Atefinia and M. Ahmadi, "Network intrusion detection using multi-architectural modular deep neural network," *J. Supercomput.*, vol. 77, no. 4, pp. 3571–3593, Apr. 2021, doi: 10.1007/s11227-020-03410-y.

[4] 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, doi: 10.1109/access.2020.2973219.

[5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019, doi: 10.1186/s42400-019-0038-7.

[6] S. Zhang, X. Xie, and Y. Xu, "A brute-force black-box method to attack machine learning-based systems in cybersecurity," *IEEE Access*, vol. 8, pp. 128250–128263, 2020, doi: 10.1109/access.2020.3008433.

[7] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419, doi: 10.1016/j.jisa.2019.102419.

[8] F. Feng, X. Liu, B. Yong, R. Zhou, and Q. Zhou, "Anomaly detection in ad-hoc networks based on deep learning model: A plug and play device," *Ad Hoc Netw.*, vol. 84, pp. 82–89, Mar. 2019, doi: 10.1016/j.adhoc.2018.09.014.

[9] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Comput. Netw.*, vol. 174, Jun. 2020, Art. no. 107247, doi: 10.1016/j.comnet.2020.107247.

[10] IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (2018). *UNB*. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[11] B. A. Tama, M. Comuzzi, and K. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019, doi: 10.1109/access.2019.2928048.

[12] P. Lin, K. Ye, and C. Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *Cloud Computing—CLOUD 2019* (Lecture Notes in Computer Science), vol. 11513, D. D. Silva, Q. Wang, and L. J. Zhang, Eds. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-23502-4_12.

[13] 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 Comput.*, vol. 23, no. 2, pp. 1397–1418, Jun. 2020, doi: 10.1007/s10586-019-03008-x.

[14] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and extreme learning machines," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1752–1779, Mar. 2018, doi: 10.1016/j.jfranklin.2017.06.006.

[15] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, Oct. 2016, doi: 10.3390/s16101701.

[16] J. L. Leevy and T. M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 big data," *J. Big Data*, vol. 7, no. 1, pp. 1–19, Dec. 2020, doi: 10.1186/s40537-020-00382-x.

[17] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *J. Big Data*, vol. 5, no. 1, pp. 1–30, Dec. 2018, doi: 10.1186/s40537-018-0151-6.

[18] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 4, pp. 1119–1130, Aug. 2012, doi: 10.1109/tsmcb.2012.2187280.

[19] A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, A. Hawalah, and A. Hussain, "Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study," *IEEE Access*, vol. 4, pp. 7940–7957, 2016, doi: 10.1109/access.2016.2619719.

[20] R. Zhu, Y. Guo, and J.-H. Xue, "Adjusting the imbalance ratio by the dimensionality of imbalanced data," *Pattern Recognit. Lett.*, vol. 133, pp. 217–223, May 2020, doi: 10.1016/j.patrec.2020.03.004.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002, doi: 10.1613/jair.953.

[22] U. Santosuosso, A. Cini, and A. Papini, "Tracing outliers in the dataset of Drosophila suzukii records with the Isolation Forest method," *J. Big Data*, vol. 7, no. 1, pp. 1–11, Dec. 2020, doi: 10.1186/s40537-020-00288-8.

[23] A. I. Saleh, F. M. Talaat, and L. M. Labib, "A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers," *Artif. Intell. Rev.*, vol. 51, no. 3, pp. 403–443, 2017, doi: 10.1007/s10462-017-9567-1.

[24] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/a:1010933404324.

[25] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987, doi: 10.1016/0169-7439(87)80084-9.

[26] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, Jan. 2016, doi: 10.1016/j.asoc.2015.10.011.

[27] S. S. S. Sindhu, S. Geetha, and A. Kannan, "Decision tree based light weight intrusion detection using a wrapper approach," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 129–141, 2012, doi: 10.1016/j.eswa.2011.06.013.

[28] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016, doi: 10.1007/s11749-016-0481-7.

[29] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006, doi: 10.1007/s10994-006-6226-1.

[30] S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective intrusion detection system using XGBoost," *Information*, vol. 9, no. 7, p. 149, Jun. 2018, doi: 10.3390/info9070149.

[31] D. Jin, Y. Lu, J. Qin, Z. Cheng, and Z. Mao, "SwiftIDS: Real-time intrusion detection system based on LightGBM and parallel intrusion detection mechanism," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101984.

[32] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/tetci.2017.2772792.

[33] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Comput. Secur.*, vol. 65, pp. 135–152, Mar. 2017, doi: 10.1016/j.cose.2016.11.004.

[34] C. Beleites, U. Neugebauer, T. Bocklitz, C. Krafft, and J. Popp, "Sample size planning for classification models," *Anal. Chim. Acta*, vol. 760, pp. 25–33, Jan. 2013, doi: 10.1016/j.aca.2012.11.007.
[35] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, Jun. 2017, doi: 10.1016/j.jnca.2017.03.018.

SUGANDH SETH is a certified Ethical Hacker with a specialization in cybersecurity. She has worked as a Subject Matter Expert at Tata Interactive Systems in the field of cybersecurity. Her list of academic achievements includes five publications, one journal, and four conference papers.

KULJIT KAUR CHAHAL is an Associate Professor with the Department of Computer Science, Guru Nanak Dev University, Amritsar. With an experience of 24 years and specialization in open source software and distributed systems, she has supervised four guided and six registered scholars. She has 43 journals and 53 conference papers, eight book chapters, two major research projects, and four professional and academic body memberships on the list of achievements.

GURVINDER SINGH is a Professor with the Computer Science Department, Guru Nanak Dev University, Amritsar, with a specialization in parallel and distributed computing data science. During his experience of around 24 years, he has 60 journals, 23 conference papers, and one book chapter to his name. Besides, he also has five academic body memberships and supervised 12 guided and six registered scholars.