Cybersecurity

RESEARCH

Open Access

Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique

Raisa Abedin Disha[1*] and Sajjad Waheed[2]

Abstract

To protect the network, resources, and sensitive data, the intrusion detection system (IDS) has become a fundamental component of organizations that prevents cybercriminal activities. Several approaches have been introduced and implemented to thwart malicious activities so far. Due to the effectiveness of machine learning (ML) methods, the proposed approach applied several ML models for the intrusion detection system. In order to evaluate the performance of models, UNSW-NB 15 and Network TON_IoT datasets were used for offline analysis. Both datasets are comparatively newer than the NSL-KDD dataset to represent modern-day attacks. However, the performance analysis was carried out by training and testing the Decision Tree (DT), Gradient Boosting Tree (GBT), Multilayer Perceptron (MLP), AdaBoost, Long-Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) for the binary classification task. As the performance of IDS deteriorates with a high dimensional feature vector, an optimum set of features was selected through a Gini Impurity-based Weighted Random Forest (GIWRF) model as the embedded feature selection technique. This technique employed Gini impurity as the splitting criterion of trees and adjusted the weights for two different classes of the imbalanced data to make the learning algorithm understand the class distribution. Based upon the importance score, 20 features were selected from UNSW-NB 15 and 10 features from the Network TON_IoT dataset. The experimental result revealed that DT performed well with the feature selection technique than other trained models of this experiment. Moreover, the proposed GIWRF-DT outperformed other existing methods surveyed in the literature in terms of the F1 score.

Keywords: Cyber security, Feature selection, Intrusion Detection System, Machine learning, Network security

Introduction

In this era of modernization, information system has become a prominent asset for companies and organizations to collect, store, process, manage and distribute information effectively in an organized way. As soon as information system technology has been introduced to organizations, the risk of security breaches has also emerged. To protect sensitive data as well as the Information Technology (IT) infrastructure from cyber attackers, several information security measures have been taken by organizations. One of the essential components of cybersecurity is the Intrusion Detection System (IDS). Based upon the knowledge it is provided, an IDS can identify the anomalous traffic and alert the network administrator accordingly (Scarfone and Mell 2007). In general, the architecture of an intrusion detection system (IDS) is developed based on four functional modules—(1) E block (Event-boxes), (2) D block (Database-boxes), (3)

*Correspondence: raisaabedin@gmail.com
[1] Department of Information and Communication Technology, Bangladesh University of Professionals, Mirpur Cantonment, Dhaka 1216, Bangladesh
Full list of author information is available at the end of the article

A blocks (Analysis-boxes), and (4) R blocks (Response-boxes) (Garcia-Teodoro et al. 2009). In a nutshell, E block (Event-Boxes) contains sensor elements for monitoring the system, and it acquires information of events for further analysis. This acquired information needs to be stored for processing. For this purpose, D block (Database-boxes) elements store the information that coming from the E block. A block (Analysis-boxes) is the processing module, where the detection of malicious behavior is carried out by analyzing the events. After detecting the hostile behavior, the most important task is to prevent that threat. So, if any kind of intrusion occurs, the R block (Response- boxes) takes initiative to thwart the malicious event with appropriate response execution (Chandola et al. 2009). An illustration of the IDS framework is given in Fig. 1. Based on information source (E block), an IDS can be categorized into Host-based IDS (HIDS) and Network-based IDS (NIDS). HIDS is related to operating system information (system calls and process identifiers), whereas NIDS performs analysis on the network events (IP address, protocols, service ports, traffic volume, etc.). Based on the analysis performed in A block, IDS can be classified into Signature-based IDS (misuse-based) and Anomaly-based IDS. In Signature-based IDS (SIDS), a database of known attack signatures is maintained, and the IDS matches the analyzed data with the database to find out the intrusion (Khraisat et al. 2019). It is the best fit for detecting known attacks, but unable to detect new types (previously unseen attacks). The false-positive rate is considerably lower in Signature-based IDS. On the contrary, Anomaly-based IDS (AIDS) tries to understand the normal behavior of the system and sets a threshold value. When the given observation at an instant deviates from normal behavior exceeding the preset threshold value, an anomaly alarm is raised (Liao et al. 2013). As Anomaly-based IDS tries to find out suspicious events, it is a good solution for detecting previously unseen attacks. However, the false-positive rate of intrusion detection is higher in AIDS than in SIDS.

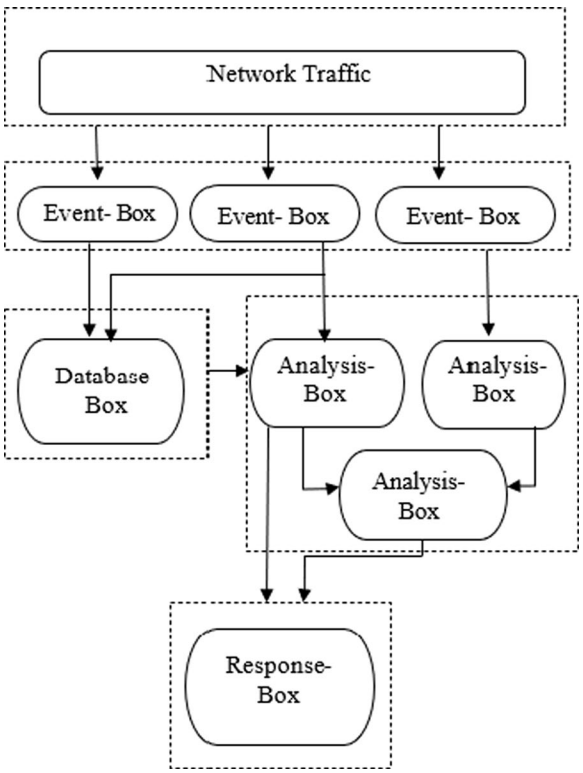

Fig. 1 Common intrusion detection architecture for IDS

Due to the tremendous usage of cloud services and the Internet of things (IoT), network traffic is also upraising every day in an enormous amount. It makes it quite difficult for the IDS to distinguish between normal and anomalous behavior of network traffic, especially while detecting zero-day attacks (previously unseen attacks). To address this issue, Machine Learning (ML) methods have become a convenient and effective technique for identifying and categorizing multiple network attacks. Machine Learning technology makes the IDS able to learn and improve the system's performance by analyzing previous data. Moreover, computer programs that use ML do not require to be explicitly programmed as they can learn by themselves (Naqa and Murphy 2015). The intrusion detection technique is still the core of many types of research works as the detection rate and accuracy score of the machine learning models are not up to the mark for classifying the intrusion. Furthermore, many solutions are not effective enough for a large number of data using the full dataset (Yin et al. 2017). Also, a lot of intrusion detection works have been conducted using NSL-KDD 99 or KDD 99 dataset, which is now considered outdated to identify modern cyber-attacks (Moustafa and Slay 2015; Labonne 2020; Divekar et al 2018).

In this research, a binary classification task implementing supervised machine learning models had been conducted. Binary classification happens when the supervised machine learning model is assigned to predict a discrete value as a 'normal' or 'attack' instance (Harrington 2012). The dataset used in this study is relatively large and owns a high dimension of feature space. For a large dataset, it is important to choose a feature selection method for dropping the irrelevant features, and use only the important set of features in both the training and testing step (Dong and Liu 2018). Feature selection is the process of choosing a subset of relevant features by reducing the number of input variables while building a predictive model. It is desired in most cases to improve the performance of predictive models. Also, irrelevant

features have a negative impact on the performance of the model. However, as the IDS deals with numerous data instances which have irrelevant or redundant features, it not only decreases the detection accuracy score but also increases the computational cost (Zaman and Karray 2009). Feature selection gives the solutions to some frequently occurred mistakes in IDS by identifying the relevant features which consist of essential information for the classification task (Mohammadi et al. 2019).

In our proposed approach, a Gini Impurity-based weighted Random Forest (GIWRF) model was developed to select the important and relevant features based on the importance score. The feature importance scores were calculated by adjusting the weight in the Random Forest algorithm (Breiman 2001) for imbalanced class distribution, and employing the Gini impurity criterion for splitting the trees. However, the Decision Tree (DT), Gradient Boosting Tree (GBT), Multilayer Perceptron (MLP), Adaptive Boosting (AdaBoost), Long-Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) were developed over two datasets for ML-based IDS framework. Data preprocessing or Data engineering was required before training and testing the models. The data preprocessing phase included three steps which would be discussed later in this paper. However, performance analysis for the binary classification task was conducted using UNSW-NB 15 dataset (Moustafa et al. 2018; Moustafa and Slay 2016), and the Network TON_IoT dataset (Moustafa 2021) which are relatively new datasets containing modern attack patterns. The rest of the paper is organized as follows. First of all, related research works have been discussed in "Literature review" section. After that, a brief discussion of the other existing datasets for IDS and the experimental datasets (UNSW-NB 15 and Network TON_IoT) have been presented in "Datasets" section. The proposed methodology of the experimental process has been explained subsequently in "Proposed methodology" section. "Experiments and results" section has focused on the experiment environment and the result of the models. Based on the experimental result, a discussion segment has been added in "Discussion" section, and finally, the conclusion ("Conclusion" section) has ended the paper.

Literature review

Different Intrusion detection techniques have been explained and suggested over the last 2 decades (Catania and Garino 2012). To build robust and efficient IDS many types of research have been proposed with machine learning approaches and made some improvements in this area. An approach was introduced by Ingre and Yadav (2015) for ML-based IDS that combined the Decision Tree (DT) classifier and correlation input selection method for the classification task. A filter-based feature selection method was used to train and test their models. For their experimental process, the NSL-KDD dataset was analyzed. 14 significant features had been selected by filter-based feature selection technique to reduce the time complexity. However, this study was conducted for both binary and multiclass classification tasks. The result showed that they obtained an accuracy of 83.66% for a multiclass classification task that successfully identified five different attacks and 90.30% accuracy for binary classification.

Another filter-based approach was introduced to detect Distributed Denial of Service (DDoS) attacks by Osanaiye et al. (2016). They used multiple filters: Chi-Square, Information Gain, Gain Ratio, and ReliefF algorithm for selecting an optimum number of features. For performance analysis of the system, they trained and evaluated the models on the NSL-KDD dataset. The classifier used in this approach was DT. The model had been validated with a tenfold cross-validation technique before testing. Their result indicated that DT was capable of detecting DDoS attacks with an accuracy of 99.67% while using 13 important features out of the whole feature set.

Alazzam et al. (2020) compared the performance of IDS using Decision Tree (DT) as a binary classifier. Pigeon Inspired Optimizer (PIO) is considered as the feature reduction technique in this study. PIO is a swarm intelligence algorithm inspired by pigeon's homing behavior. When a flock of pigeons flies a long way home, they use different navigation tools in different phases of their flight (Liu et al. 2019). The pigeons continuously change their position according to the navigation tool (map and compass operator) following the best pigeon that has the best position. Based on this philosophy of search and optimization, PIO has been further developed. In this experiment, two types of PIO were used for feature reduction: Cosine PIO and Sigmoid PIO. The authors carried out the simulation on UNSW-NB 15, NSL-KDD, and KDDCup99 datasets. Using Sigmoid PIO, 10 features from KDDCup99, 14 features from the UNSW-NB dataset, and 18 features from NSL-KDD were selected. Also, they selected 7 features from KDDCup99, 5 features from UNSW-NB-15, and NSL-KDD using Cosine PIO for comparing the result. Sigmoid PIO obtained an accuracy of 94.7% for KDDCup99, 86.9% using the NSL-KDD, and 91.3% over the UNSW-NB 15 dataset. On the contrary, Cosine PIO achieved 91.7% accuracy over UNSW-NB 15, 88.3% on NSL-KDD, and 96% on KDDCup99.

Another experiment was conducted by Khan et al. (2018) to enhance the performance of Random Forest, Extreme Gradient Boosting (XGBoost), Decision Tree, Bagging Meta Estimator, and K-Nearest Neighbors (KNN). The goal of this study was to decrease

the computational time with reduced features while improving the accuracy of models. The subset of 11 features had been chosen according to the feature importance calculated by the Random Forest model. Random Forest obtained 74.87% accuracy, XGBoost achieved 71.43% detection accuracy, 74.64% for Bagging Meta Estimator using UNSW-NB Dataset. Also, Decision Tree and KNN attained an accuracy of 74.22% and 71.10% respectively. However, the training time for XGBoost was comparatively higher than other models in this experiment. Decision Tree required the lowest prediction time and KNN needed significantly higher prediction time than other models.

An improved anomaly detection technique was proposed by Tama and Rhee (2019) using Gradient Boosting Machine (GBM) with complete features of NSL KDD, UNSW-NB 15, and GPRS dataset. The authors statistically assessed the superiority of GBM over other models: Support Vector Machine (SVM), Random Forest, Classification and Regression Tree (CART), and Deep Neural Network (DNN) in terms of area under the receiver operating characteristic curve (AUC), specificity, false-positive rate, sensitivity, and accuracy. The experiment was simulated in a Python environment with machine learning libraries. GBM outperformed other methods with an accuracy score of 91.31% for the UNSW-NB Dataset, 91.82% for KDDTest+, and 86.51% for KDDTest-21.

An SVM and Artificial Neural Network (ANN) based technique was proposed by Aboueata et al. (2019) for intrusion detection systems in cloud environments. Additionally, they considered Univariate and Principal Component Analysis (PCA) together for choosing an optimal set of features. They trained and tested the models on UNSW-NB 15 dataset and performance evaluation is carried out in terms of F1 score, Precision, Recall as well as Accuracy. However, feature engineering along with parameter tuning was performed by the authors to obtain maximum accuracy while reducing the time complexity of the models. The study indicated that with an appropriate set of features, ANN and SVM techniques were capable of detecting anomalies with 91% and 92% accuracy scores respectively.

Jing and Chen (2019) introduced another SVM-based anomaly detection approach for both binary classification and multiclass classification. A nonlinear feature scaling method was applied in their simulation. However, they used Radial Basis Function (RBF) to map low dimensional space to high dimensional space in SVM. RBF kernel is suitable for large dataset evaluation in terms of computational time and accuracy. They achieved 85.99% testing accuracy for binary classification and 75.77% of accuracy for multiclass classification. Their preferred dataset for offline performance analysis was also the UNSW-NB 15 dataset.

Kasongo and Sun (2020) implemented five supervised models: Support Vector Machine (SVM), Logistic Regression (LR), k-Nearest-Neighbour (kNN), Decision Tree (DT), and Artificial Neural Network (ANN) using a filter-based feature reduction technique. The Extreme Gradient Boosting (XGBoost) technique was used for reducing the feature vector from 42 to 19 based on the feature importance score. This study compared the performance of models by training and testing them on UNSW-NB 15 dataset. However, they conducted the work for both binary and multiclass classification. With reduced features, they observed an increment in accuracy from 88.13 to 90.85% for the DT model while performing the binary classification task.

Another intrusion detection framework was proposed by Meftah et al. (2019) that included two stages. In the first stage, their approach performed binary classification. In the second stage, the attack traffic output was fed to multiclass classifiers for identifying each attack. The authors used Random Forest for the feature selection technique for the binary classification task. They trained the Gradient Boost Machine, Logistic Regression, and Support Vector Machine (SVM) and the test result revealed that SVM obtained the maximum detection accuracy of 82.11%. The Gradient Boost Machine and Logistic Regression obtained the accuracy score of 61.83% and 77.21% respectively. For multiclass classification, they applied multinomial Support Vector Machine, Decision Trees (C5.0), and Naïve Bayes. However, the Decision Tree achieved the highest accuracy of 74%, and 86% F1 score for multiclass classification. Moreover, they presented constructive criticism on UNSW-NB 15 dataset, which had been used in their experimental analysis.

Injadat et al. (2020) proposed an ML-based NIDS framework that studied the oversampling technique on the training data and selected a suitable sample size for training the KNN and RF. They applied correlation-based (CBFS) and information gain (IGBFS) feature selection techniques and compared the performance of classifiers using CICIDS 2017 and UNSW-NB 2015 datasets. However, unlike other research works which used the benchmark UNSW-NB 15 training and UNSW-NB 15 test datasets, this study used the full UNSW-NB 15 data (2,540,044 instances) and made a random split for training and testing. Moreover, they used the oversampling technique, SMOTE to synthetically produce more instances to a smaller class. They analyzed the effect of feature selection on the training sample and feature set size. The performance of the models was evaluated based on Accuracy, Precision, Recall, and FAR (False Alarm Rate) score. The oversampling method as well as the

selection of an optimum train size helped to achieve an accuracy score of 99% for both datasets.

Belgrana et al. (2021) proposed two approaches for NIDS. Firstly, they applied Condensed Nearest Neighbors (CNN) to reduce the dimensionality of features as well as the computational time. Secondly, the authors implemented a Radial Basis Function (RBF) neural network to realize the performance training on the NSL-KDD dataset. Then, they compared the performance of KNN, C4.5, and IBK (version of KNN in Weka) with CNN and RBF model with all features of NSL-KDD and with reduced features in terms of True Positive Rate (TPR), False Alarm Rate (FAR) and Missed Attacks Rate (MAR). Their proposed RBF and CNN obtained the accuracy (success rate) score of 94.28% and 95.54% respectively. However, they found CNN faster than RBF, though the MAR in CNN was lower than RBF.

A hybrid approach was proposed by Lee et al. (2020), where the authors proposed a deep sparse autoencoder for feature selection in the data pre-processing step. Autoencoder is widely used in image processing to compress the image without decreasing the number of features, so this research used autoencoder to compress the features of training and test datasets without having any loss. They compared the performance of a single RF and their proposed Deep Sparse Autoencoder Random Forest (DSAE-RF) to realize the effect of the proposed feature compression technique. The authors evaluated their approach in terms of Accuracy, Precision, Recall, and F1 score by training the DSAE-RF on CICIDS2017 Dataset. The accuracy score for DSAE-RF was 99.83%, which was slightly lower than single RF (99.86%), but the precision and recall score was comparatively higher than single RF. In addition, they analyzed the performance of the model according to the hidden layer's structure. However, they claimed that their proposed DSAE-RF model requires less training and testing time than the single RF.

Based on Neural Network approaches an experimental review for Network Intrusion Management was provided by Mauro et al. (2020). The authors offered a full view of some promising neural networks which are very relevant for applying to ML-based IDS. For their experimental analysis, they developed Convolutional Neural Network, MLP, Recurrent Neural Network (RNN), Wilkes Stonham and Aleksander Recognition Device (WiSARD), Learning Vector Quantization (LVQ), Self-Organizing Maps (SOM) using CIC-IDS-2017/2018 dataset released from Canadian institute for cybersecurity. As per their experimental analysis, MLP is much slower than other neural networks due to the backpropagation property, though it performed well in terms of detection performance. LVQ performed inferior to other networks, whereas WiSARD showed the best trade-off between performance and time complexity.

Gu and Lu (2021) proposed an effective approach for IDS using SVM with Naïve Bayes algorithm together to classify intrusion and normal instances. In that work, the Naïve Bayes algorithm was used to transform the original features into new data. After that, the newly transformed data was used to train the SVM model for the classification task. They applied this approach on UNSW-NB 15, CICIDS2017, NSL-KDD, and Kyoto 2006 + datasets to evaluate the performance. Their experimental result showed that they achieved good performances with 98.92% accuracy for the CICIDS2017 dataset, 99.35% accuracy for the NSL-KDD dataset, 93.75% accuracy on UNSW-NB 15 dataset, and 98.58% accuracy applying Kyoto 2006 + dataset.

Moustafa (2021) created a new dataset called, Network TON_IoT dataset and used a Wrapper Feature Selection technique-based RF to select the important features. They developed the GBM, RF, NB, and DNN to evaluate the performance. After feature selection, the GBM, RF and DNN achieved the accuracy score of 93.83%, 99.98%, and 99.92% respectively and for NB they obtained the AUC score of 91.28%.

A summary of the related works that used the ML techniques for IDS has been presented in Table 1.

Datasets

Existing datasets

Several datasets are available to evaluate the security systems of cybersecurity. However, many datasets do not contain modern cyber-attack patterns for intrusion detection systems such as NSL-KDD. In this segment, the most recent and widely used datasets for ML-based IDS have been discussed as follows:

1. *CAIDA datasets* CAIDA datasets (Hick et al. 2007) are a set of enormous different data sources for verifying IDSs performance with very few numbers of attack vectors (e.g., DDoS). The dataset is available as CAIDA 2007, which contains anonymized network data for DDoS attacks without payload. However, the datasets neither possess the audit traces of telemetry data of IoT sensors and operating systems nor have a ground truth of the security events.
2. *The Kyoto dataset* It was built at Kyoto University collecting the network traffic from a honeypot environment. The Kyoto 2006 + (Song et al. 2011) dataset was generated by using the Zeek tool which extracted 24 features from the original KDD99 dataset. However, this dataset did not have the testbed configuration with the IoT system, and network components.

Table 1 Related works on IDS using ML models

Proposed by	Used dataset	Feature selection	Algorithm	Accuracy
Khan et al. (2018)	UNSW-NB 15 dataset	Feature importance (RF)	XGBoost, RF, Bagging, KNN, DT	71.43%, 74.87%, 74.64%, 74.22%, 71.10%
Tama and Rhee (2019)	NSL KDD, UNSW-NB 15 and GPRS	Complete feature	GBM	91.31% (UNSW-NB), 91.82% (KDDTest +), 86.51% (KDDTest-21)
Jing and Chen (2019)	UNSW-NB 15	All features	SVM	85.99% (binary classification), 75.77% (multiclass classification)
Kasongo and Sun (2020)	UNSW-NB 15	XGBoost algorithm	SVM, Logistic Regression, KNN, DT, ANN	60.89, 77.64, 84.46, 90.85, 84.39
Ingre and Yadav (2015)	NSL-KDD	filter-based	DT	83.66% (multiclass classification), 90.30% (binary classification)
Osanaiye et al. (2016)	NSL-KDD	Chi-Square, information gain, gain ratio, and relieff algorithm	DT	99.67%
Alazzam et al. (2020)	UNSW-NB 15, NSL-KDD, KDDCup99	Sigmoid PIO Cosine PIO	DT DT	91.7% (UNSW-NB 15); 88.3% (NSL-KDD); 96% (KDDCup99); 91.3% (UNSW-NB) 86.9% (NSL-KDD) 94.7% (KDDCup99)
Aboueata et al. (2019)	UNSW-NB 15	Univariate and principal component analysis (PCA)	ANN, SVM	91%, 92%
Meftah et al. (2019)	UNSW-NB 15	Random forest	GBM, LR, SVM	61.83%, 77.21%, 82.11%
Injadat et al. (2020)	CICIDS 2017, UNSW-NB 15	Correlation based (CBFS) and information gain (IGBFS)	KNN RF	99% for both datasets
Belgrana et al. (2021)	NSL-KDD	Condensed nearest neighbors (CNN)	Radial basis function (RBF), CNN	94.28%, 95.54%
Lee et al. (2020)	CICIDS2017 dataset	Autoencoder	Deep sparse autoencoder random forest (DSAE-RF)	99.83%
Gu and Lu (2021)	UNSW-NB 15, CICIDS2017, NSL-KDD, and Kyoto 2006 +	Feature transformation with Naïve Bayes	SVM	98.92% (CICIDS2017), 99.35% (NSL-KDD), 93.75% (UNSW NB15), 98.58% (Kyoto 2006 +)
Moustafa (2021)	Network TON_IoT	Wrapper feature selection technique-based RF	GBM RF DNN	93.83%, 99.98%, 99.92%

3. *ISCX dataset* This dataset (Shiravi et al. 2012) was created by capturing normal traffic and a synthetic attack simulator, which was carried out in a real-time simulation environment for more than 1 week. However, there was no ground truth of the intrusions, and the proofing concept requires enormous computational resources making it unsuitable to be applied in real-time.
4. *CICIDS 2017* The dataset (Sharafaldin et al. 2018) was created at the Canadian Institute for Cybersecurity (CIC) which involves several normal and attack (hack) scenarios employing data profiling concept, similar to ISCX dataset. The network traffic of this dataset was scrutinized by the CICFlowMeter. The flow identifiers and the time-stamp were utilized by the CICFlowMeter with the labeled data to investigate the network traffic. However, CICIDS 2017 also has the same shortcomings as the ISCX dataset.
5. *The UNSW-NB 15 dataset* The dataset (Moustafa and Slay 2015) was designed at the University of New

South Wales, Canberra, especially to evaluate the Network Intrusion Detection System. It used IXIA traffic generator which captured 2,540,044 observations. Out of the whole data, a portion is available online as two separated.csv files called UNSW_NB15_training-set and UNSW_NB15_test-set. However, this dataset does not contain any security events against the operating system and IoT.

6. *TON_IoT Datasets* It was also created at the University of New South Wales, Canberra deploying a novel testbed architecture by Moustafa (2021). The data was collected and labeled by executing real-world normal and intrusion scenarios. The final dataset was named TON_IoT as the data was gathered from different telemetry datasets of IoT services, the dataset of network traffic, and Windows and Linux-based datasets.

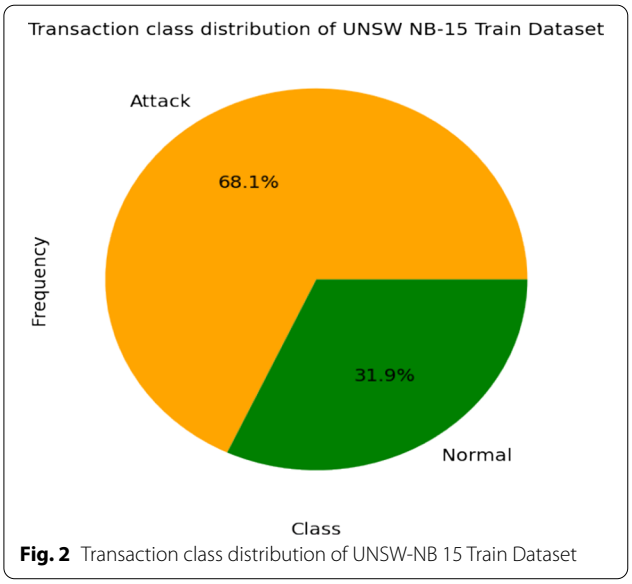

Fig. 2 Transaction class distribution of UNSW-NB 15 Train Dataset

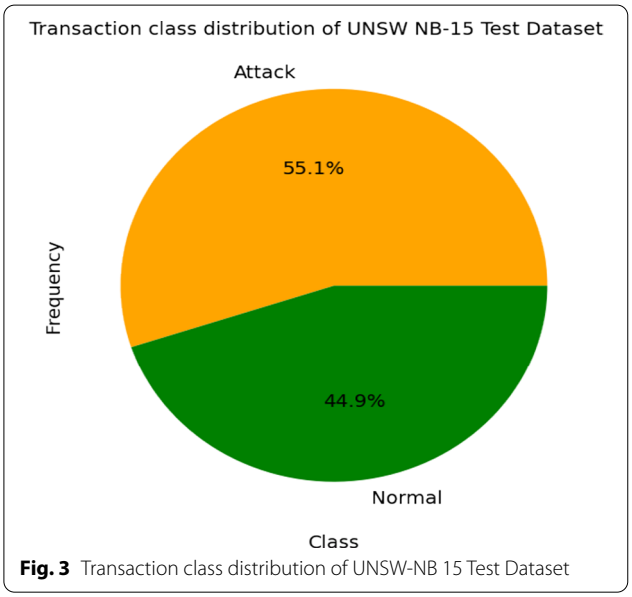

Fig. 3 Transaction class distribution of UNSW-NB 15 Test Dataset

Discussion about experimental datasets

UNSW-NB 15 dataset

For our experimental process of Intrusion Detection System (IDS) the first dataset we used for the offline analysis was UNSW-NB 15 (Moustafa and Slay 2015). UNSW-NB 15 dataset is comparatively newer than NSL KDD 99 or KDD 99, CAIDA, Kyoto 2006+, and ISCX dataset. It contains modern network traffic for both normal and anomalous instances including present-day low footprint attacks. It is available in a clean format, and there is no redundancy in the data, thus making it more suitable for reliable evaluation for Network Intrusion Detection systems. The total number of data instances is 2,540,044 which are kept in four.csv files. From these records, a partition of 175,341 instances is considered as a training set and 82,332 data instances as the test set. In UNSW-NB 15 training set, a significant portion (68.1%) of data instances is attack types and just less than one-third (31.9%) of data instances are normal transactions, as shown in Fig. 2. Similarly, in the test set, 55.1% of total data is attack traffic, and 44.9% of instances contain normal transactions, as shown in Fig. 3. Attacks are represented by '1', whereas normal traffic is indicated by '0' in the class label of the dataset for binary classification. The dataset does not have any duplicate records to make the classifiers being biased. However, attack types in the UNSW-NB 15 train and test dataset are similar. Figure 4 has illustrated the distribution of attack categories in both datasets. In its clean format, the dataset contains 44 features, where 'attack_cat' and 'label' are output variables (labeled feature). The data instances can be categorized into float, categorical and binary format as shown in Table 2. Both of train and test set contains nine different types of attacks (Moustafa and Slay 2016), which can be described as in the following points:

1. *Fuzzers* It is a malicious activity by the perpetrators in which they explore the security vulnerabilities in applications, programs, operating systems, or networks by flooding it with enormous random data with the aim of crashing it.
2. *Analysis* It involves a wide range of intrusions that penetrate the web application via emails (using spam), web scripts (through HTML files), and ports (via port scan techniques).

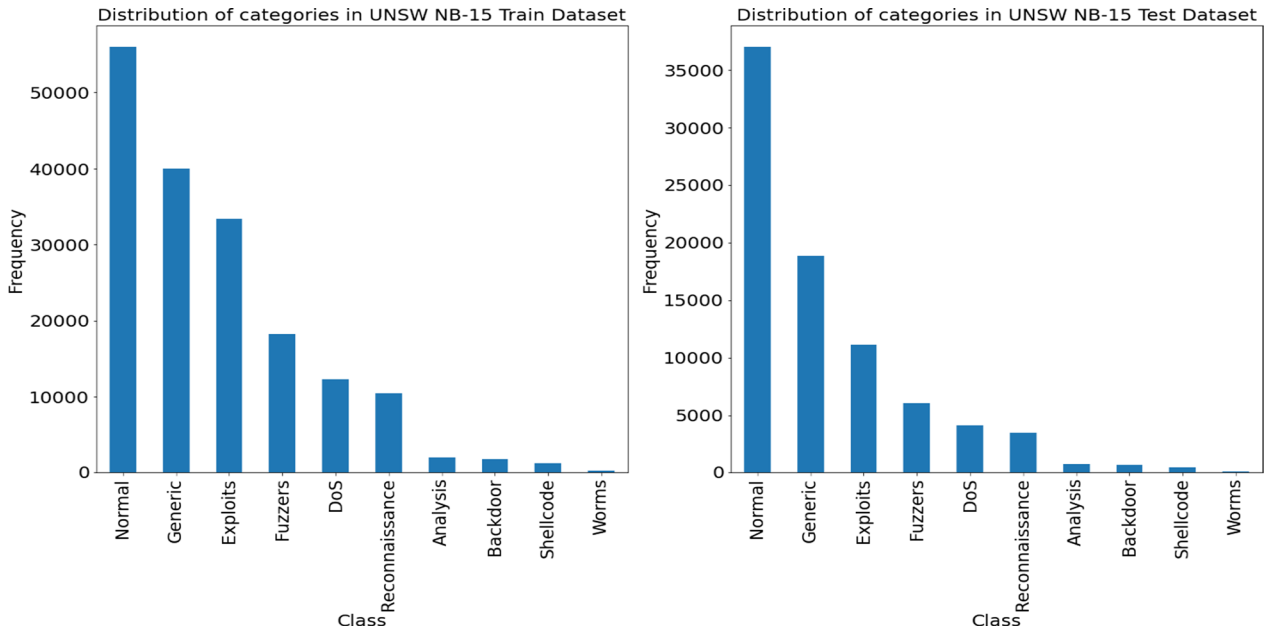

Fig. 4 Distribution of categories in UNSW-NB 15 Train and Test dataset

3. *Backdoor* This technique is used to bypass the normal authentication process, and let unauthorized users (attackers) access a computer or device, which helps them to execute commands remotely.
4. *DoS (Denial of Service)* In DoS, the attacker attempts to make the computer resources unavailable to authorized users temporarily or indefinitely by shutting down theservices provided by the target system or network.
5. *Exploit* It is a series of instructions given by the perpetrators, which takes advantage of any vulnerability, bug, or glitch that exists in the network or system.
6. *Generic* It is a malicious activity in which the attackers do not care about the cryptographical implementation of any primitives. It works successfully against all block ciphers using the hash function to cause a collision, no matter what configurations the block ciphers have.
7. *Reconnaissance/ Probe* This attack collects information about the computer network in order to dodge the security controls.
8. *Shellcode* For the purpose of taking control over the compromised device or machine, attackers write the code and inject it into the application that activates the command shell.
9. *Worms* It is a self-replicating code that exploits the system vulnerabilities or uses social engineering techniques to gain access to the system. It reduces the availability of the system by consuming memory and network bandwidth.

Network TON_IoT dataset

TON_IoT network dataset is the most recent dataset created by Moustafa (2021). This dataset was collected by the Intelligent Security Group of the Cyber Range and IoT Labs of UNSW, which contains nine very recent attacks traffic. The full data record contains 22,339,021 instances, from where a portion of 461,043 instances is being considered as 'Train_Test_Network_dataset' for evaluating new Artificial Intelligence-based cybersecurity solutions. The dataset was published for applying into different machine learning models and to handle the challenges of class imbalance, which made it suitable for using in this research. However, the dataset has 44 features excluding the target variable 'label' and 'type'. As the author suggested in Moustafa (2021) four features: source IP, destination IP, source ports and destination ports were removed from the dataset before applying the feature selection technique in the proposed approach, and from this dataset, 80% of data (368,834) was selected for training the models, and other 20% (92,209) was used for testing the models. As shown in Table 3, the features are categorized into four types of format: String, Number, Boolean, and Time, which belong to four service groups, such as connection, statistics, user attributes (i.e. HTTP, SSL activities and DNS), and violation attributes. Unlike UNSW NB 15, where the number of attack traffic is higher than normal instances, the TON_Train_Test_Network dataset posses 65.1% normal instances and 34.9% attack traffics as shown in Fig. 5. The distribution

Table 2 List of features exist in UNSW-NB 15 Dataset

Feature number	Format	Feature
1	Float	dur
2	Categorical	proto
3	Categorical	service
4	Categorical	state
5	Integer	spkts
6	Integer	dpkts
7	Integer	sbytes
8	Integer	dbytes
9	Float	rate
10	Integer	sttl
11	Integer	dttl
12	Float	sload
13	Float	dload
14	Integer	sloss
15	Integer	dloss
16	Float	sinpkt
17	Float	dinpkt
18	Float	sjit
19	Float	djit
20	Integer	swin
21	Integer	stcpb
22	Integer	dtcpb
23	Integer	dwin
24	Float	tcprtt
25	Float	synack
26	Float	ackdat
27	Integer	smean
28	Integer	dmean
29	Integer	trans_depth
30	Integer	response_body_len
31	Integer	ct_srv_src
32	Integer	ct_state_ttl
33	Integer	ct_dst_ltm
34	Integer	ct_src_dport_ltm
35	Integer	ct_dst_sport_ltm
36	Integer	ct_dst_src_ltm
37	Binary	is_ftp_login
38	Integer	ct_ftp_cmd
39	Integer	ct_flw_http_mthd
40	Integer	ct_src_ltm
41	Integer	ct_srv_dst
42	Binary	is_sm_ips_ports
43	Categorical	attack_cat
44	Binary	label

Table 3 List of features exist in TON_IoT Train_Test Network Dataset

Feature number	Feature	Format
1	dns_AA	Boolean
2	dns_RD	Boolean
3	dns_RA	Boolean
4	dns_rejected	Boolean
5	ssl_resumed	Boolean
6	ssl_established	Boolean
7	weird_notice	Boolean
8	src_port	Number
9	dst_port	Number
10	duration	Number
11	src_bytes	Number
12	dst_bytes	Number
13	missed_bytes	Number
14	src_pkts	Number
15	src_ip_bytes	Number
16	dst_pkts	Number
17	dst_ip_bytes	Number
18	dns_qclass	Number
19	dns_qtype	Number
20	dns_rcode	Number
21	http_trans_depth	Number
22	http_request_body_len	Number
23	http_status_code	Number
24	http_response_body_len	Number
25	http_user_agent	Number
26	label	Number
27	src_ip	String
28	dst_ip	String
29	proto	String
30	service	String
31	conn_state	String
32	dns_query	String
33	ssl_version	String
34	ssl_cipher	String
35	ssl_subject	String
36	ssl_issuer	String
37	http_method	String
38	http_uri	String
39	http_referrer	String
40	http_version	String
41	http_orig_mime_types	String
42	http_resp_mime_types	String
43	weird_name	String
44	weird_addl	String
45	type	String
46	ts	Time

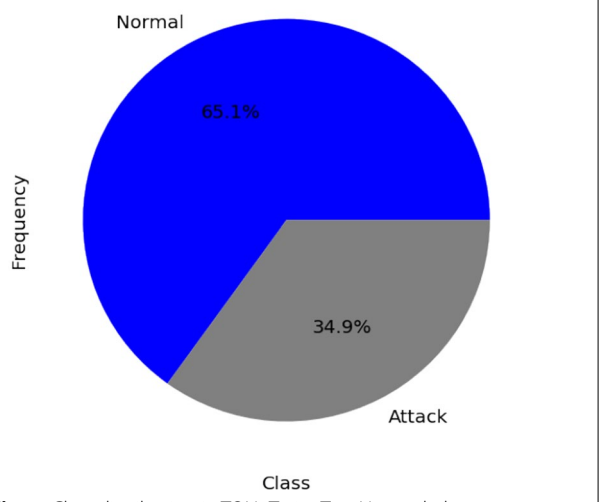

Fig. 5 Class distribution in TON_Train_Test Network data

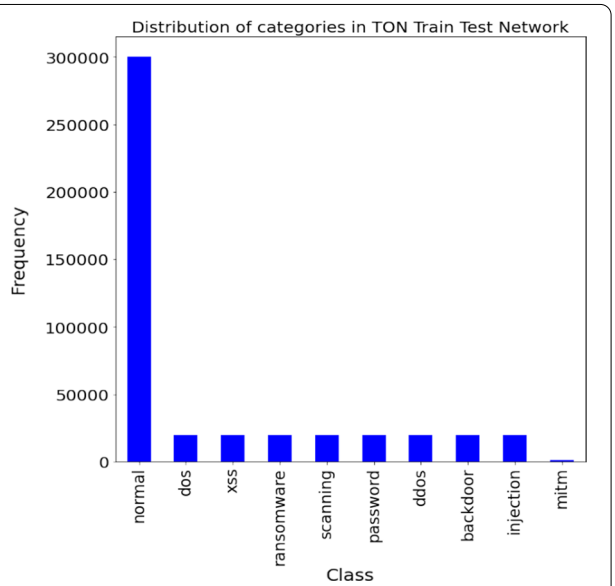

Fig. 6 Distribution of categories in TON_Train_Test Network data

of attack categories as shown in Fig. 6 has illustrated that except Man-in-the-middle (MITM) other attacks have an equal distribution of records. The MITM attack has the lowest, and normal traffic has the highest records in the dataset.

However, the nine different attacks that exist in the dataset can be summarized as following points:

1. *Scanning attack* The goal of a scanning attack is to gather information about the target system. Attackers try to find out the active IP addresses and the open port of the vulnerable target system using Nmap or Nessus or any other scanning tool.
2. *Denial of Service (DoS)* DoS attack is one of the frequently occurred invasions in the network that induces fake flooding to target network for making services unavailable to its users.
3. *Distributed Denial of Service (DDoS)* In DDoS attack, multiple DoS attacks are launched together to corrupt the system. Usually, an attacker infects many vulnerable systems with malware and turns each system into a bot or zombie. Then, taking control over the botnet it attacks the target from all of the compromised bot systems.
4. *Ransomware attack* It is a complicated malware that infects any system or service by encrypting those and makes the authorized user unable to access until they pay the attacker ransom money.
5. *Backdoor attack* It bypasses the normal authentication process and gains high-level user access to the vulnerable system for stealing user's personal data, financial information, installing other malware, or hijacking the devices.
6. *Injection attack* This attack inserts or injects any malicious input or any forged data from the client to a web application, and forces it to execute some commands for changing the operation.
7. *Cross-site Scripting (XSS) attack* It is a type of vulnerability where the attackers inject client-side scripts into the web pages that are accessed by other users, and employs the web pages to transmit the malicious code to the other end user's system in the form of browser-side script.
8. *Password cracking attack* Password cracking refers to any hacking technique that continuously tries to discover the correct password by employing brute force or dictionary attacks.
9. *Man-In-The-Middle (MITM) attack* MITM is a kind of eavesdropping attack, where the attackers put themselves between two parties (e.g. users and applications), and impersonate as one of the parties to steal personal information (e.g. login credentials, credit card information etc.) from them.

Proposed methodology

The main motivation of this study was to apply a suitable feature selection technique for imbalanced data and make a comparison of the performance of the ML models described in "Machine learning methods under scrutiny" section by training those over the datasets summarized in "Discussion about experimental datasets" section. The diagram of the proposed approach has been illustrated in Fig. 7.

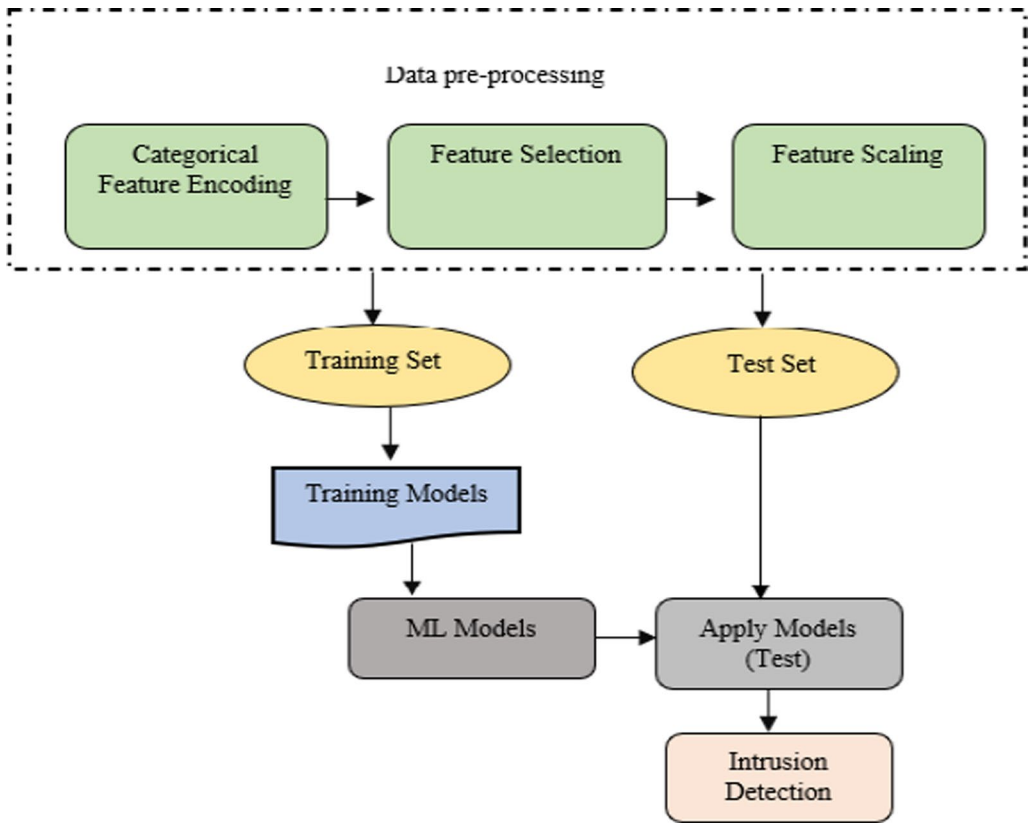

Fig. 7 Proposed ML-based IDS framework

Data preprocessing

The first step of the experiment was data preprocessing or data engineering. Feeding the models raw data may give a misleading prediction sometimes. As the datasets do not have any null or duplicate value, the data preprocessing step included only Categorical Feature Encoding, Feature Selection, and Feature Scaling.

Coding for categorical feature

The UNSW-NB 15 dataset contains three categorical features: State, Proto, and Service, and the Network TON_IoT Dataset has nineteen string features as shown in Table 3. To encode the categorical features of UNSW-NB 15, Response Coding was applied, and to transform the string features of Network TON_IoT dataset Label Encoding was used. In Label Encoding, each categorical feature is given a unique integer value based on alphabetical order (Sethi 2020). On the other hand, Response Coding is an encoding technique, which represents the probability of data instances that belongs to a specific class given a category. For N-class classification, N new features are generated for each category that inserts the probability (P) of data instance belonging to each class based upon the value of categorical data (Dharmik 2019). A mathematical expression can be presented by:

$$P(Y|A) = P(A \cap Y)/P(A) \tag{1}$$

where Y is denoted as a class and the category of feature is represented as A. It requires considerable memory space that made it unsuitable for transforming string features of the Network TON_IoT dataset, as it contains several categorical features.

Feature scaling

The datasets contain some features which have highly fluctuated magnitudes, ranges, and units. Due to this highly varying characteristic of the datasets, some algorithms such as the Multilayer Perceptron predicts wrongfully. Besides, it requires high computational resources. So, feature scaling is one commonly used method that normalizes the range of independent variables. Feature scaling is mandatory for some machine learning models which calculate the distance between data. Normalization and Standardization are two widely applied feature scaling methods in

machine learning. In this experiment, Normalization or Min–Max scaling was used to normalize the data between the range of 0 and 1. The general formula of normalization can be shown as below:

$$f^{-} = (f - \min(f))/(\max(f) - \min(f)) \quad (2)$$

where f is the original feature and f^{-} is the normalized feature. The minimum and maximum values of the feature are represented as $\min(f)$ and $\max(f)$ respectively.

Gini Impurity-based Weighted Random Forest (GIWRF) for feature selection

Random Forest (Breiman 2001) is an ensemble classifier that is built on a number of decision trees and supports various feature importance measures. One of those is to derive the importance score by training the classifier. The traditional machine learning classification algorithm expects that all the classes in the training set come up with similar importance, and the models are built without considering that there may exist an imbalance class distribution in training data (Krawczyk 2016). To understand the relevance of features with the output of the imbalanced data, this feature selection technique employed a weight adjustment technique in RF once the classifier measured the Gini impurity, i(τ). Gini impurity reveals how well a split is to divide the total samples of binary classes in a specific node. Mathematically it can be written as:

$$i(\tau) = 1 - p_p^2 - p_n^2 \quad (3)$$

Table 4 Selected features of UNSW-NB 15 and Network TON_IoT dataset

UNSW-NB 15 (20 features)		Network TON_IoT (10 features)	
feature name	**Importance score**	**Feature name**	**Importance score**
sttl	0.123	ts	0.258
ct_state_ttl	0.099	proto	0.146
sload	0.059	src_ip_bytes	0.137
rate	0.058	src_pkts	0.067
dload	0.057	dst_ip_bytes	0.062
dttl	0.045	dst_pkts	0.048
sbytes	0.039	conn_state	0.047
ct_srv_dst	0.037	dst_bytes	0.034
smean	0.034	src_bytes	0.032
dmean	0.032	duration	0.030
dbytes	0.028	–	–
ackdat	0.027	–	–
ct_dst_src_ltm	0.027	–	–
ct_srv_src	0.026	–	–
dur	0.026	–	–
tcprtt	0.025	–	–
synack	0.024	–	–
dinpkt	0.023	–	–
sinpkt	0.022	–	–
dpkts	0.021	–	–

where p_p is the fraction of positive samples and p_n is the fraction of negative samples out of the total number of

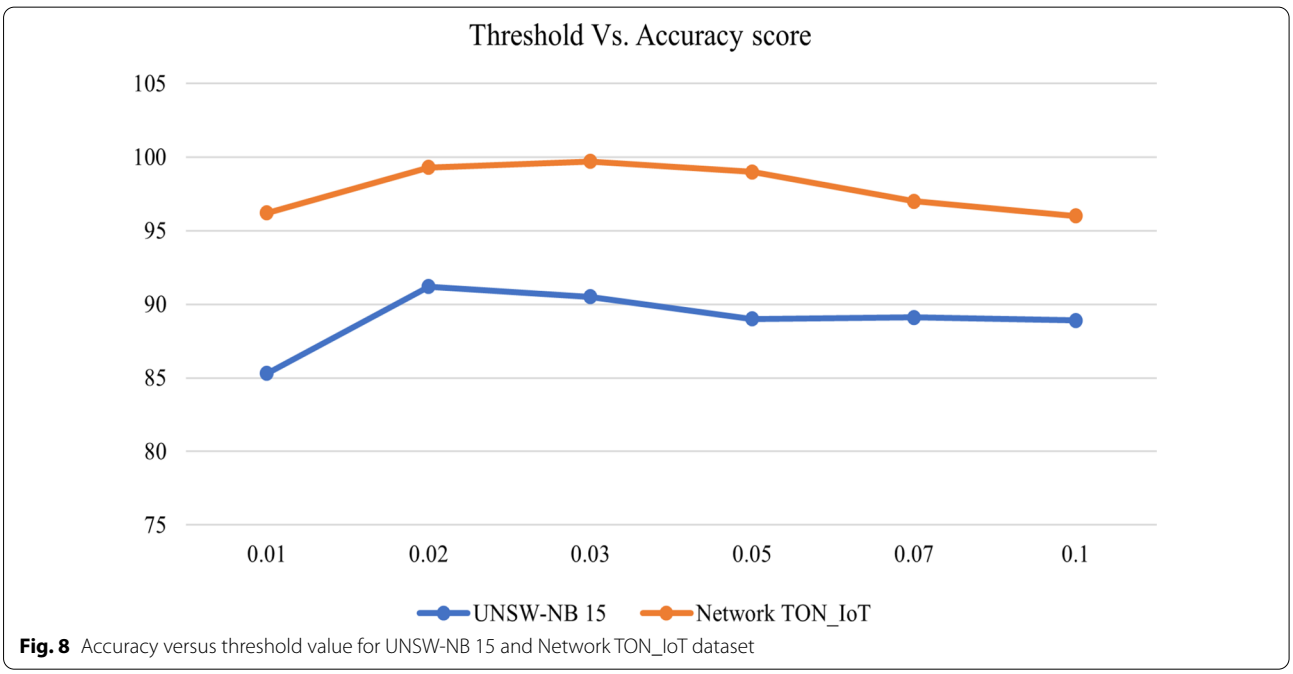

Fig. 8 Accuracy versus threshold value for UNSW-NB 15 and Network TON_IoT dataset

samples (N) at node τ. The reduction in Gini impurity deriving from any optimal split $\Delta i_f(\tau, M)$ is gathered together for all the nodes τ in the M number of weighted trees in the forest, individually for all of the features. Mathematically it can be written as:

$$I_g(f) = \sum_{M} w_{p,n} \sum_{\tau} \Delta i_f(\tau, M) \tag{4}$$

where I_g is the Gini importance, which specifies the frequency of a particular feature (f) being selected for the split and the significance of the feature's overall discriminative value for the binary classification task. Assigning the weight $w_{p,n}$ defines the imbalanced class distribution in the learning algorithm. The weight adjustment can be written as:

$$\text{weight for positive class, } w_p = \frac{n_n}{N} \tag{5}$$

$$\text{weight for negative class, } w_n = \frac{n_p}{N} \tag{6}$$

$w_p + w_n = 1$ and for imbalanced class data $w_p \neq w_n$.

The number of negative instances is represented as n_n and the positive instances are denoted as n_p. N is the total number of instances in the training dataset. The pseudo-code for selecting the features using the *sklearn* library has been explained in Algorithm 1. Considering the threshold as 0.02 and 0.030 for UNSW-NB 15 and Network TON_IoT respectively maximum accuracy was observed as shown in Fig. 8, and the selected features for both datasets have been listed in Table 4.

Algorithm 1 Gini Impurity based weighted Random Forest**:** Feature selection

Input: $X_{encoded}$ ($f_1^{encoded}$, …, $f_i^{encoded}$)

Output: $X_{selected}$: the selected features

Step 1: Load the encoded feature vectors

Step 2: Create a blank dictionary, *D* for saving the scores

Step 3: Instantiate a *RandomForestClassifier* as *model*

Step 4: Define parameters: *criterion='gini'*, *n_estimators=500*, *class_weight= {p:* w_p*, n:*w_n*}*

Step 5: fit *model*

Step 6: Generate feature importance scores, $F_{importance}$

Step7: Determine the threshold, F_{th}

Step 8: **for** *i* from $X_{encoded}$ **do**

If ($F_{importance}(s^j) \geq F_{th}$) **then**

append $F_{importance}(s^j)$ into *D*

end if

end for

Step 9: Using scores from *D* generate $X_{selected}$

Machine learning methods under scrutiny

In this section, a summary of ML techniques that were applied to the experimental analysis has been discussed. The methods were selected in such a way that a comparison between traditional ML technique (DT), Ensemble method (Adaboost, GBT), Artificial Neural Network (MLP), and Deep Neural Network (LSTM, GRU) would be performed to detect intrusions over two imbalanced datasets.

Decision tree

Decision Tree (Quinlan 1986) is a non-parametric supervised machine learning model which is implemented for both classification and regression tasks. It repeatedly splits the data following a specific attribute. Decision Tree learns the decision rules that are deduced from the data attributes. Based on that rules, it predicts the value of the target variable. The decision-making process of this model can be shaped like a tree and makes it easier for the user to interpret. Numerous ML tools are available to visualize the output of DT (Safavian and Landgrebe 1991). Two units: decision nodes and leaves are the fundamental concepts of DT. In the decision node, data is split based on a particular parameter, and in the leaves unit, the outcome or decisions are obtained. However, the splitting criterion in this experimental study was entropy (Shannon 1948), which measures the impurity of the split. For every single internal decision node of the Decision Tree, the entropy equation can be given by the following formula:

$$Entropy, E(j) = -\sum_{k=1}^{j} P_k \log_2 P_k \tag{7}$$

where j is the number of unique classes in the dataset and P_k is the probability of each particular class. For binary classification the entropy equation can be written as:

$$Entropy, E = -P_p \log_2 P_p - P_n \log_2 P_n \tag{8}$$

where P_p is the probability of positive event and P_n is the probability of the negative event.

Adaptive boosting (AdaBoost)

AdaBoost algorithm was firstly introduced by Schapire (2003). It is an ensemble learning method, which utilizes an iterative approach to correct the mistakes of weak learners. It calls a given base learning algorithm or a weak learner repeatedly in a series of rounds to boost the model's performance. The fundamental concept of AdaBoost is to reassign the weights to each instance, and giving higher weights to the misclassified instances. In brief, while training the Adaboost model, firstly it trains the base classifier (such as DT) and utilizes that classifier to predict over the training set. Then, increasing the weight of incorrectly classified training instances, it trains the second classifier, using the newly updated weights, again it makes a prediction on the training set. Then, again updates the weights of instances, and so on. This process will be continued until it reaches the very last base learner.

Gradient boosting tree (GBT)

Gradient Boosting Tree (Mason et al. 1999) is a widely used machine learning model that identifies the shortcomings of weak learners and overcomes their limitations by boosting gradient descent with each weak learner in the loss function. The Loss function is calculated from the difference between the true value and estimated value. It transforms the weak learners into stronger ones by adding up the predictors to the ensemble, sequentially as well as gradually. Each predictor in the ensemble corrects the mistakes made by its predecessor. GBT is an effective technique to reduce noise, variance as well as bias (Felix and Sasipraba 2019).

Multilayer perceptron (MLP)

MLP (Rosenblatt 1961) is a feed-forward Artificial Neural Network (ANN) that passes the information from input to output in a forward direction. As the name suggests, Multilayer Perceptron contains three layers of nodes: Input Layer, Hidden Layer, and Output Layer. However, in MLP each node except the input unit is a neuron, and these neurons utilize a nonlinear activation function to transform the weighted sum of input into an output. The input unit receives the input signal, and the desired task of regression or classification is conducted in the output layer (Abirami and Chitra 2020). A mathematical expression of MLP can be written as the following equation:

$$y = \emptyset\left(\sum_{i=1}^{k} w_i x_i + c\right) \tag{9}$$

where each neuron estimates the weighted sum of the inputs k, then adds a bias c, and after that uses an activation function ($\emptyset$) for producing the output y. MLP uses back-propagation to train the neurons.

Long Short-Term Memory (LSTM)

LSTM (Hochreiter and Schmidhuber 1997) is a type of artificial Recurrent Neural Network (RNN) that is adapted to store the information for a longer period. It was developed to handle the vanishing gradient problem, which can be experienced by RNN during the training

phase. An LSTM unit is a composition of a memory cell, and three gates: input, output and forget gates to control the flow of information towards the memory cell or out of the cell. The input gate takes the decision of updating the cell state, the forget gate determines what information would be kept or discarded based on estimating a coefficient computed by the input data and the earlier hidden state. Based on the previous hidden state and the input data, the output gate highlights which information should be conveyed to the next hidden state. Mathematically, if $o^j(t)$ is an output gate and $s^j(t)$ denotes the cell state of the j^{th} LSTM unit, the equation for the hidden state $h^j(t)$ at any time t can be represented as follows:

$$h^j(t) = o^j(t).\tanh\left(s^j(t)\right) \tag{10}$$

where *tanh* is the activation function.

Gated Recurrent Unit (GRU)

Gated Recurrent Unit (Cho et al. 2014) is considered as a variation or a lightweight version of LSTM, but it is simpler to implement than LSTM. It has two gates: The update gate and Reset gate, which have been introduced to solve the vanishing gradient problem of RNN. The update gate is similar to the input and the forget gate used in the LSTM, which helps the model to decide what information should be passed to the next state. On the other hand, the Reset gate determines how much past information should be forgotten. Mathematically, the hidden state of jth GRU unit can be represented as a linear interpolation between the previous activation at time t and the candidate activation $\widetilde{h^j}(t+1)$, where $z^j(t+1)$ is the update gate and makes a decision regarding the updating amount of the candidate activation:

$$h^j(t+1) = \left(1 - z^j(t+1)\right)h^j(t) + z^j(t+1)\widetilde{h^j}(t+1) \tag{11}$$

Evaluation metrics

The goal of this experiment was to increase the number of correct predictions in the test set for binary classification. To evaluate the performance of ML-based IDS several performance metrics are used in Machine Learning, such as Accuracy, False Positive Rate, Precision, Recall, etc. (Kumar 2014). However, the Accuracy (AC) score is the most commonly used metric to evaluate the performance of models in binary classification problems. It can be defined as below equation:

$$AC = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{12}$$

where TP stands for True Positive, and TN represents the count of True Negative. True Positive is the number of correctly detected attack instances. True Negative counts the correctly classified normal instances. On the other hand, False Positive (denoted as FP) is the number of legitimate traffic that is misclassified as an attack. False Negative (denoted by FN) counts the number of attack instances that are wrongfully considered as normal instances. In the case of detecting intrusion, less FN is always expected, as it is more hazardous than FP. However, for imbalanced class data, especially, where the aim is to efficiently detect the intrusion (positive instances) four other metrics: False Positive Rate (FPR), Precision, Recall, and F1 score are also considered in addition to the Accuracy measure.

1. *False Positive Rate (FPR)* The FPR is measured as the ratio of the negative events that are misclassified as positive (FP) and the total amount of truly negative events. The expression can be given as below:

$$FalsePositiveRate(FPR) = \frac{FP}{(FP + FN)} \tag{13}$$

2. *Precision* Precision is the measure that evaluates a model's performance by calculating how often the model's prediction is correct when it positively predicts an instance. Mathematically, it can be expressed as below equation:

$$Precision = \frac{TP}{(TP + FP)} \tag{14}$$

3. *Recall/ Detection Rate (DR)* It is the measure of the Machine Learning model correctly detecting True Positive instances. Also, Recall measures how accurate the model is to identify relevant data. This is why it is referred to as Sensitivity or True Positive Rate. The mathematical expression can be shown as below:

$$Recall/DR = \frac{TP}{(TP + FN)} \tag{15}$$

4. *F1 Score* It is the tradeoff between Recall and Precision that takes both FP and FN into account and measures the overall accuracy of the ML model. F1 score can be expressed as below equation:

$$F1score = \frac{(2 * Recall * Precision)}{(Recall + Precision)} \tag{16}$$

Table 5 Result of the models with all features (42) for UNSW-NB 15 dataset

Models	AC. (%)	Precision (%)	Recall (%)	F1 score (%)	FPR (%)
DT	90.15	**87.45**	95.85	91.46	**16.84**
AdaBoost	**90.51**	87.07	97.19	**91.85**	17.67
GBT	87.56	82.49	98.25	89.68	25.54
MLP	84.11	78.34	98.31	87.20	33.28
LSTM	87.90	85.01	94.71	89.60	20.44
GRU	82.87	76.78	**98.75**	86.39	36.57

The best result per column has been written in boldface to have better understanding

Table 6 Result of the models with reduced features (20) for UNSW-NB 15 dataset

Models	AC.(%)	Precision (%)	Recall (%)	F1 score (%)	FPR (%)
DT	**93.01**	**92.69**	94.76	**93.72**	**09.14**
AdaBoost	90.51	87.59	96.43	91.80	16.73
GBT	87.08	82.25	97.58	89.26	25.78
MLP	87.26	82.02	**98.44**	89.48	26.43
LSTM	88.99	87.59	93.21	90.31	16.17
GRU	90.11	86.73	96.84	91.51	18.14

The best result per column has been written in boldface to have better understanding

Table 7 Result of the models with all features (39) for Network TON_IoT dataset

Models	AC. (%)	Precision (%)	Recall (%)	F1 score (%)	FPR (%)
DT	99.50	99.83	98.74	99.28	0.09
AdaBoost	99.88	**99.99**	99.67	99.83	**0.001**
GBT	**99.98**	99.98	**99.95**	**99.97**	0.006
MLP	98.35	97.60	97.68	97.64	1.2
LSTM	94.51	91.28	93.18	92.22	4.7
GRU	95.69	91.23	96.99	94.02	5.0

The best result per column has been written in boldface to have better understanding

Table 8 Result of the models with selected features (10) for Network TON_IoT dataset

Models	AC.(%)	Precision (%)	Recall (%)	F1 score (%)	FPR (%)
DT	99.90	99.84	99.87	99.85	0.08
AdaBoost	**99.98**	**99.98**	99.96	99.97	**0.006**
GBT	**99.98**	**99.98**	**99.97**	**99.97**	0.008
MLP	97.13	93.64	98.49	96.00	3.5
LSTM	88.99	76.44	99.00	86.27	16.37
GRU	95.02	90.95	95.24	93.04	5.08

The best result per column has been written in boldface to have better understanding

Experiments and results

The experiment was conducted on an HP Pavilion 14-AL143TX loaded with the Windows 10 Operating System with the following processor: Intel(R) Core(TM) i5-7200U CPU @ 2.5–3.1 GHz. The building, training, and evaluation of the Machine Learning model were performed by Pandas, NumPy, Scikit-Learn (sklearn), etc. Machine Learning libraries in the python environment of Jupyter Notebook, an open-source tool.

Considering six ML models: Decision Tree (DT), Adaptive Boosting (AdaBoost), Gradient Boosting (GBT), ANN (Multilayer Perceptron), Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU), this experiment was carried out into two states. In the first state, all the features of the UNSW-NB 15 and Network TON_IoT dataset were utilized, and the performance of ML models was evaluated for binary classification. In the second state, using the proposed feature selection method only 20 features for UNSW-NB 15 and 10 features from the Network TON_IoT dataset were chosen and the binary classification was performed again with six models. For training all the models, parameter tuning was one common approach that assisted this experiment to improve the accuracy and F1 score as well as decrease the FPR score. Table 5 (for UNSW-NB 15) and Table 7 (for Network TON_IoT) have shown the result obtained by machine learning algorithms for binary classification using the whole feature set, and Table 6 (for UNSW-NB 15) and Table 8 (for Network TON_IoT) have indicated the performance of the same models applying only the selected features. The best result per column has been written in boldface. Visual representation of the performance comparison for Network TON_IoT and UNSW-NB 15 has been illustrated in Figs. 9 and 10 respectively in terms of Accuracy, F1 score and FPR.

For the Decision Tree (DT) classifier, *entropy* was selected as the *criterion* during parameter tuning. Entropy is the measure of information gain, based on what DT decides to split the data. To define each class label another important parameter, *class_weight* was set to '*balanced*'. Other tuned parameters of DT were set as follows: *random_state* = 10, *max_depth* = 11, *max_leaf_nodes* = 162, *min_samples_leaf* = 20, and *min_impurity_decrease* = 0.00006. However, for UNSW-NB 15, the result indicated that DT achieved the test accuracy score of 90.15% while applying the full feature space (42 features), whereas it achieved the accuracy of 93.01% using 20 selected features. A 16.84% FPR score and 91.46% F1 score were achieved applying the full features, and a 09.14% FPR score and 93.72% F1 score were observed while using only the reduced features. For Network

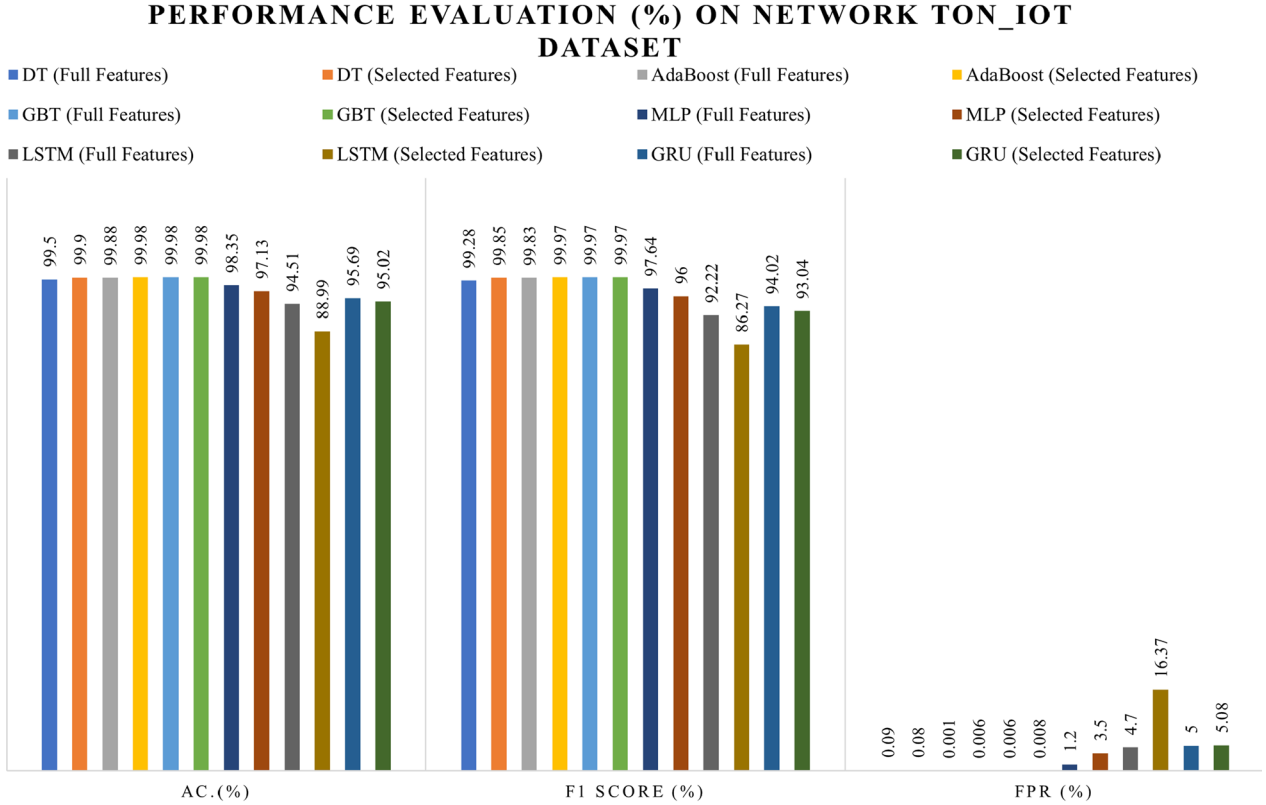

Fig. 9 The performance comparison between ML models with full feature and with reduced feature for Network TON_IoT dataset

TON_IoT dataset, the accuracy score for DT was 99.50%, F1 score was 99.28% and FPR score was 0.09% using full features, and using only the selected features the accuracy, F1 and FPR score became 99.90%, 99.85% and 0.08% respectively.

During hyperparameter tuning, DecisionTreeClassifier was selected as the base estimator of the AdaBoost classifier. In addition, the number of trees (n_estimators) chosen for the AdaBoost model was 3300 with a learning rate of 0.3. Other than that, the 'SAMME.R' algorithm was used as another important parameter that achieves comparatively lesser test errors with a fewer number of boosting iterations. The base estimator, Decision Tree, was also tuned with some important attributes as follows: criterion = 'gini', random_state = 10, class_weight = 'balanced', max_depth = 11, max_leaf_nodes = 162, min_samples_leaf = 20, and min_impurity_decrease = 0.00006. The criterion, 'gini' is the measure of impurity that calculates the probability of incorrect classification of an observation. Based on the criterion, DT takes the decision to split the data. However, intending to perform binary classification task for UNSW-NB 15, the obtained result for AdaBoost classifier exhibited that the accuracy score was unchanged (90.51%) in both cases. However, the F1 score and FPR of detection was 91.85% and 17.67% respectively using full features, and became 91.80% and 16.73% respectively applying the selected features. In the case of Network TON_IoT dataset, the accuracy, F1 score and FPR score were 99.88%, 99.83%, and 0.001% respectively using full features, whereas it became 99.98%, 99.97% and 0.006% using selected features.

The Gradient Boosting (GBT) model used *'deviance'* as the loss function. Multiple models were trained with different number of trees and different learning rate as follows: n_estimators = {2500, 2700, 3000, 3200, 3500} and learning_rate = {0.1, 0.08, 0.07, 0.05, 0.01}. There is always a tradeoff between n_estimators and learning_rate. The experimental result revealed that *n_estimators* set as 3200 with a learning rate of 0.05 provided the highest test accuracy score for GBT. Using the full features of UNSW-NB 15, the accuracy score for detecting intrusion was 87.56%, F1 score was 89.68% and the FPR was 25.54%. In contrast, the accuracy score was 87.08%, F1 score was 89.26% and the FPR score became 25.78% with only the selected features.

For the Network TON_IoT dataset, in both cases of whole features and selected features the accuracy, and F1

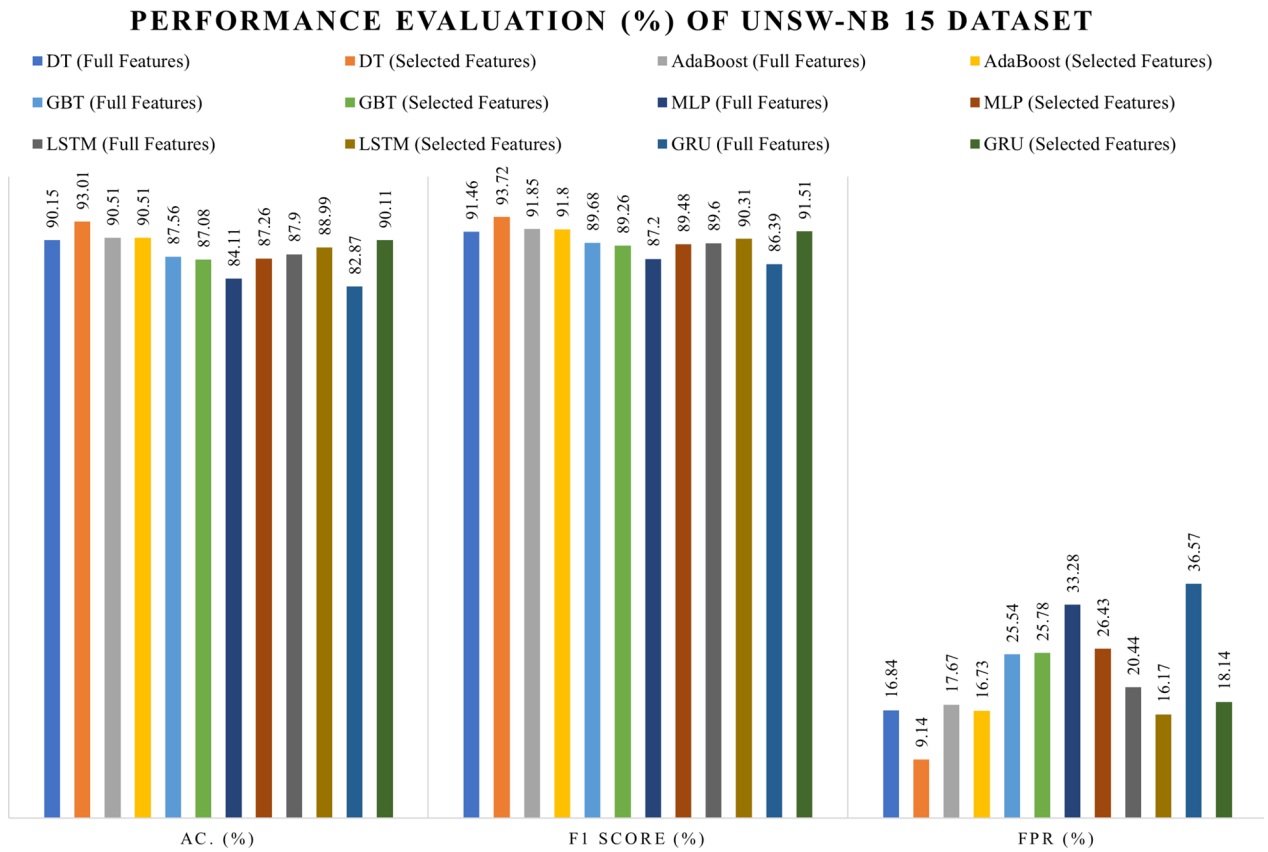

Fig. 10 The performance comparison between ML models with full feature and with reduced feature for UNSW-NB 15 dataset

score did not change, but FPR was increased to 0.008% from 0.006% using selected features.

In the case of the Multilayer Perceptron (MLP), 'Adam' solver was used in the experiments for weight optimization. 'Adam' solver is a stochastic gradient-based optimizer that works efficiently for large datasets. This is why it was selected instead of Stochastic Gradient Descent (SGD) or Limited memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) solver. The MLP was developed with only one hidden layer with the following number of neurons: hidden_layer_sizes = (15, 30, 60). The activation function, which decides whether a neuron will be triggered or not for the hidden layer was set at 'Relu' in all of our experiments. The maximum iteration for the solver was set at 2000 and the learning rate was 'adaptive'. In addition, *learning_rate_init* was tuned into 0.002 to control the step size for upgrading the weight. However, for the binary classification task using the whole feature set of UNSW-NB 15, the accuracy score for detection was 84.11% and it became 87.26% while using the subset of features selected through the feature selection method. In the case of F1 score and FPR, the scores were 87.20% and 33.28% respectively when all of the features were used, and those became 89.48% and 20.86% respectively using only selected features. For the Network TON_IoT dataset, the accuracy, F1 score and FPR were respectively 98.35%, 97.64% and 1.2% with whole features, and became 97.13%, 96.00% and 3.5% with the selected features.

A stacked LSTM model with four hidden layers and one output layer was trained on both UNSW-NB 15 and Network TON_IoT dataset. In the hidden layers, *tanh* was used as the activation function, wherein *sigmoid* was employed in the output layer as the activation function. Other selected parameters were *optimizer = 'adam', loss = 'binary_crossentropy', epochs = 50*, and *batch_size = 64*. The layers contained 300, 200, 100, 80 LSTM units sequentially with a dropout of 0.4. Using the full feature set of UNSW-NB 15, the accuracy score was 87.90%, F1 score was 89.60% and FPR score was 20.44%, and after using selected features the scores became 88.99%, 90.31%, and 16.17% respectively. In the case of Network TON_IoT dataset, the accuracy, F1 score and FPR were respectively 94.51%, 92.22% and 4.7% with full features, and those became 88.99%, 86.27% and 16.37 with reduced features.

Similar to the LSTM model, the same setting was also considered for the GRU model. While employing the full features of UNSW-NB 15, the obtained accuracy, F1 and FPR scores were 82.87%, 86.39% and 36.57% respectively, which became 90.11%, 91.51%, and 18.14% after applying the selected features. In the case of Network TON_IoT dataset, with all features the accuracy score was 95.69%, the F1 score was 94.02% and the FPR was 5.0%, which became 95.02%, 93.04% and 5.08% respectively.

Discussion

Observation for UNSW-NB 15 dataset

After numerous trials and errors, the experimental result revealed that with the GIWRF feature selection technique, Decision Tree outperformed all other models obtaining an accuracy score of 93.01% and an F1 score of 93.72%. As the accuracy score climbed to 93.01% from 90.15% using the 20 features, it is clear that the feature selection method had a great impact on the detection process of DT-based IDS. Also, the feature selection technique helped to reduce the FPR by 7.7%. There was a minor decrement in the accuracy score of GBT while using the reduced features for binary classification. In the case of GBT, the accuracy score was 0.48% higher when we used the full feature set and it dropped to 87.08% using 20 features. Similarly, F1 score was decreased by 0.42% and the FPR score was increased by 0.24% using the reduced feature. The result indicated that the feature selection technique did not improve the prediction capability of GBT. Surprisingly, the accuracy of Adaboost model remained at 90.51% for both cases of selected features and full features, though the FPR was decreased by 0.94% after feature selection, F1 score was reduced by a negligible percentage of 0.05.

For UNSW-NB 15, the feature selection technique significantly improved the performance of the Neural Network. In the case of using full features, MLP, LSTM and GRU classified the intrusion and normal instances with an accuracy score of 84.11%, 87.9% and 82.87% respectively, whereas the feature selection technique boosted this accuracy score to 87.26%, 88.99 and 90.11% respectively. Likewise, the F1 scores for the aforementioned models were improved by 2.28%, 0.71% and 5.12% respectively with reduced features. Similarly, the FPR score for MLP, LSTM and GRU were decreased by 6.85%, 4.27% and 18.43% respectively with the selected features. The GRU came out as the best performer out of other two neural networks.

Observation for network TON_IoT dataset

In the case of the Network TON_IoT dataset, the experimental result showed that DT and Adaboost models performed better with the feature selection technique than the full features. Training the models on selected features, the accuracy score was improved for DT by a percentage of 0.40, and became 99.90%, F1 score was increased by 0.5% and FPR was decreased by 0.001%. In the case of Adaboost, the accuracy score was boosted by 0.1% and became 99.98% with selected features, F1 score was also improved by 0.14%, but the FPR was increased by 0.005%. Surprisingly, no change in accuracy or F1 score was observed for GBT with feature selection, but FPR score was increased to 0.008% from 0.006%.

Feature selection did not improve the prediction capability for the neural networks in the case of the Network_TON_IoT dataset. The accuracy score for MLP, LSTM and GRU were respectively 98.35%, 94.51% and 95.69% without feature selection technique, which reduced to 97.13%, 88.99% and 95.02% respectively. Similarly, the F1 score was reduced by 1.64%, 5.95% and 0.98% respectively for the aforementioned models. Feature selection did not decrease the FPR score for the neural networks. It increased the score by 2.3%, 11.67% and 0.08% respectively for the above-mentioned models.

General consideration

Overall, it is clear that the weight adjustment technique along with the Gini impurity criterion that split the trees in RF picked a set of important features considering the skewed class distribution of training data, and it enhanced the learning ability of DT over two imbalanced datasets. As DT obtained the improved Accuracy and F1 score for both datasets with the selected features, it can be claimed as the best model of the proposed approach to detect intrusions. Another observation should be pointed out that during the parameter tuning, setting the class weight as *'balanced'* improved the prediction ability of DT. As this work did not consider any oversampling or undersampling technique to handle the skewed class distribution of training data, setting the class weight as '*balanced*' implicitly put a higher weight on the minor class and a lower weight on the major class while training the model.

While feature selection significantly improved the performance of neural networks over the UNSW-NB 15 dataset, it showed opposite behavior when trained on the Network TON_IoT dataset, where normal traffic is higher than the attack traffic. For this dataset, without feature selection, the ANN and DNN learned well, but in the case of UNSW-NB 15 where attack traffic is higher than the normal instances, the ANN and DNN required important features to train on. Similarly, AdaBoost performed better in terms of accuracy, F1 score and FPR with selected features from the Network TON_IoT dataset, but in the case of UNSW-NB 15 feature selection did not make any negative or positive impact on the

Table 9 Comparison between different existing methods and our proposed approach for UNSW-NB 15

Proposed by	ML methods used for IDS	Accuracy(%)	Recall/detection rate (DR) (%)	F1 score (%)
Alazzam et al. (2020)	Cosine-PIO DT	91.7	–	90
Khan et al. (2018)	RF	75.65	76	73
Jing and Chen (2019)	SVM	85.99	–	–
Kasongo and Sun (2020)	DT	90.85	98.38	88.45
Meftah et al. (2019)	SVM	82.11	–	–
Tama and Rhee (2019)	GBM	91.31	–	–
Aboueata et al. (2019)	SVM	92	92	91
Gu and Lu (2021)	NB-SVM	93.75	94.73	–
Proposed approach	GIWRF-DT	93.01	94.76	93.72

Table 10 Comparison between existing methods and our proposed approach for Network TON_IoT

Proposed by	ML methods used for IDS	Accuracy (%)	Recall/detection rate (DR) (%)	F1 score (%)
Moustafa (2021)	RF	99.98	99.99	99.97
Proposed Approach	GIWRF-DT	99.90	99.87	99.85

prediction ability of AdaBoost. Also, feature selection did not improve the performance of GBT for both datasets.

However, choosing the best model from each literature that used the UNSW NB-15 dataset, Table 9 has compared the performance with the best model (DT) presented in this paper. It has affirmed that DT with our proposed feature selection (GIWRF) achieved satisfactory performance than other works in terms of F1 score, which is a suitable evaluation criterion for imbalanced data. Also, the accuracy score was found to be higher than other existing works except the one proposed by Gu and Lu (2021). In this case, the proposed GIWRF-DT obtained slightly a lowered accuracy score (0.74%) than the NB-SVM (Gu and Lu 2021), but the Recall/Detection Rate was still higher (0.03%) than theirs. That proved the effective intrusion detection capability of the proposed DT. However, they used a different training size than the one used in the proposed work.

To expand the comparison scope, Table 10 has depicted the performance of the best model proposed by Moustafa (2021) versus our proposed DT using the newest dataset, Network TON_IoT. Models developed in the proposed approach were also capable to classify normal and malicious traffic effectively compared to the proposed work by Moustafa (2021). DT developed in the proposed study obtained a slightly lowered accuracy score (0.08) than their proposed RF in Moustafa (2021), which was quite expected as the authors used the source IP, destination IP, source port, and destination port attributes for developing the models but suggested other future works to remove those features for demonstrating the difficulty of the security event.

It should be mentioned that Tables 9 and 10 have just illustrated a picture of a comparison between the proposed intrusion detection technique and other existing frameworks. Hence, it should not be claimed that the proposed framework is superior to other intrusion detection techniques, but the effectiveness of this study may bring stimulation for future works in the active research area of IDS.

Conclusion

The motivation of this study was to train and evaluate the ML models: DT, AdaBoost, GBT, MLP, LSTM, and GRU for performing the binary classification task of ML-based IDS. To select a suitable set of features from two imbalanced datasets: UNSW-NB 15 and Network TON_IoT, a Gini Impurity-based Weighted RF was introduced as the feature selection process based upon the assumption that imbalanced class distribution might have an influence on the feature selection process. The feature selection technique reduced the features of the UNSW-NB 15 and Network TON_IoT dataset to 20 (from 42) and 10 (from 41) respectively. The performance of the models was evaluated in terms of Accuracy, FPR, Precision, Recall, and F1 score to detect the intrusion. In the beginning, the experiment was conducted using the whole feature set over both datasets. After that, the experiment was carried out again using only the selected features extracted through the feature selection method. A comparison of the

model's performance with full feature space and with reduced features for both datasets was performed. In both cases, the Decision Tree behaved better with the feature selection technique for both datasets. However, this work did not perform multiclass classification and time complexity analysis, hence, a multiclass classification scheme for IDS considering time complexity analysis can be carried out as future work.

Abbreviations

IDS: Intrusion Detection System; ML: Machine learning; RF: Random forest; DT: Decision tree; GBT: Gradient Boosting Tree; MLP: Multilayer Perceptron; FPR: False positive rate; HIDS: Host-based IDS; NIDS: Network-based IDS; SIDS: Signature-based IDS; AIDS: Anomaly-based IDS; IoT: Internet of Things; DDoS: Distributed Denial of Service; PIO: Pigeon Inspired Optimizer; XGBoost: Extreme Gradient Boosting; LSTM: Long-Short Term Memory; GRU: Gated Recurrent Unit; KNN: K-Nearest Neighbors; GBM: Gradient Boosting Machine; RBF: Radial Basis Function; SVM: Support Vector Machine; CART: Classification and Regression Tree; DNN: Deep Neural Network; AUC: Area under the receiver operating characteristic curve; ANN: Artificial Neural Network; PCA: Principal Component Analysis; DoS: Denial of Service; SGD: Stochastic Gradient Descent; LBFGS: Limited memory Broyden–Fletcher–Goldfarb–Shanno.

Acknowledgements

Not applicable.

Authors' contributions

RAD designed the feature selection technique, performed the experiments, interpreted the results, and drafted the manuscript. SW participated in problem discussions and improvements of the manuscript. The author(s) read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

The UNSW-NB 15 dataset can be found at the website of the University of New South Wales, (https://research.unsw.edu.au/projects/unsw-nb15-dataset). The Network TON_IoT dataset can be found at the website of the University of New South Wales, (https://cloudstor.aarnet.edu.au/plus/s/ds5zW91vdgjEj9i?path=%2FTrain_Test_datasets%2FTrain_Test_Network_dataset).

Declaration

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Information and Communication Technology, Bangladesh University of Professionals, Mirpur Cantonment, Dhaka 1216, Bangladesh. [2]Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University, Santosh, Tangail 1902, Bangladesh.

Received: 2 August 2021 Accepted: 17 November 2021
Published online: 04 January 2022

References

Abirami S, Chitra P (2020) Energy-efficient edge based real-time healthcare support system. In: Advances in computers. Elsevier, pp 339–368

Aboueata N, Alrasbi S, Erbad A, Kassler A, Bhamare D (2019) Supervised machine learning techniques for efficient network intrusion detection. In: 2019 28th international conference on computer communication and networks (ICCCN). IEEE, pp 1–8

Alazzam H, Sharieh A, Sabri KE (2020) A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. Expert Syst Appl 148:113249

Belgrana FZ, Benamrane N, Hamaida MA et al (2021) Network intrusion detection system using neural network and condensed nearest neighbors with selection of NSL-KDD influencing features. In: 2020 IEEE international conference on internet of things and intelligence system (IoTaIS). IEEE, pp 23–29

Breiman L (2001) Random forests. Mach Learn 45:5–32

Catania CA, Garino CG (2012) Automatic network intrusion detection: current techniques and open issues. Comput Electr Eng 38:1062–1072

Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41:1–58

Cho K, Van Merriënboer B, Gulcehre C et al (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv14061078

Dharmik (2019) Response coding for categorical data. https://medium.com/@thewingedwolf.winterfell/response-coding-for-categorical-data-7bb8916c6dc. Accessed 23 July 2021

Di Mauro M, Galatro G, Liotta A (2020) Experimental review of neural-based approaches for network intrusion management. IEEE Trans Netw Serv Manag 17:2480–2495

Divekar A, Parekh M, Savla V, et al (2018) Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. In: 2018 IEEE 3rd international conference on computing, communication and security (ICCCS). IEEE, pp 1–8

Dong G, Liu H (2018) Feature engineering for machine learning and data analytics. CRC Press

Felix AY, Sasipraba T (2019) Flood detection using gradient boost machine learning approach. In: 2019 international conference on computational intelligence and knowledge economy (ICCIKE). IEEE, pp 779–783

Garcia-Teodoro P, Diaz-Verdejo J, Maciá-Fernández G, Vázquez E (2009) Anomaly-based network intrusion detection: techniques, systems and challenges. Comput Secur 28:18–28

Gu J, Lu S (2021) An effective intrusion detection approach using SVM with naïve Bayes feature embedding. Comput Secur 103:102158

Harrington P (2012) Machine learning in action. Simon and Schuster

Hick P, Aben E, Claffy K, Polterock J (2007) The CAIDA DDoS attack 2007 dataset. 2012) [2015-07-10]. http//www. caida. org

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780

Ingre B, Yadav A (2015) Performance analysis of NSL-KDD dataset using ANN. In: 2015 international conference on signal processing and communication engineering systems. IEEE, pp 92–96

Injadat M, Moubayed A, Nassif AB, Shami A (2020) Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans Netw Serv Manag

Jing D, Chen H-B (2019) SVM based network intrusion detection for the UNSW-NB15 dataset. In: 2019 IEEE 13th international conference on ASIC (ASICON). IEEE, pp 1–4

Kasongo SM, Sun Y (2020) Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. J Big Data 7:1–20

Khan NM, Negi A, Thaseen IS (2018) Analysis on improving the performance of machine learning models using feature selection technique. In: International conference on intelligent systems design and applications. Springer, pp 69–77

Khraisat A, Gondal I, Vamplew P, Kamruzzaman J (2019) Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2:1–22

Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. Prog Artif Intell 5:221–232

Kumar G (2014) Evaluation metrics for intrusion detection systems-a study. Evaluation 2:11–17

Labonne M (2020) Anomaly-based network intrusion detection using machine learning. https://tel.archives-ouvertes.fr/tel-02988296/. Accessed 30 Sept 2021

Lee J, Pak J, Lee M (2020) Network intrusion detection system using feature extraction based on deep sparse autoencoder. In: 2020 international conference on information and communication technology convergence (ICTC). IEEE, pp 1282–1287

Liao H-J, Lin C-HR, Lin Y-C, Tung K-Y (2013) Intrusion detection system: a comprehensive review. J Netw Comput Appl 36:16–24
Liu H, Yan X, Wu Q (2019) An improved pigeon-inspired optimisation algorithm and its application in parameter inversion. Symmetry (basel) 11:1291
Mason L, Baxter J, Bartlett P, Frean M (1999) Boosting algorithms as gradient descent in function space. In: Proc. NIPS, pp 512–518
Meftah S, Rachidi T, Assem N (2019) Network based intrusion detection using the UNSW-NB15 dataset. Int J Comput Digit Syst 8:478–487
Mohammadi S, Mirvaziri H, Ghazizadeh-Ahsaee M, Karimipour H (2019) Cyber intrusion detection by combined feature selection algorithm. J Inf Secur Appl 44:80–88
Moustafa N (2021) A new distributed architecture for evaluating AI-based security systems at the edge: network TON_IoT datasets. Sustain Cities Soc 72:102994
Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 military communications and information systems conference (MilCIS). IEEE, pp 1–6
Moustafa N, Slay J (2016) The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf Secur J A Glob Perspect 25:18–31
Moustafa N, Turnbull B, Choo K-KR (2018) An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet Things J 6:4815–4830
El Naqa I, Murphy MJ (2015) What is machine learning? In: Machine learning in radiation oncology. Springer, pp 3–11
Osanaiye O, Cai H, Choo K-KR, Dehghantanha A, Xu Z, Dlodlo M (2016) Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. EURASIP J Wirel Commun Netw 2016:1–10
Quinlan JR (1986) Induction of decision trees. Mach Learn 1:81–106
Rosenblatt F (1961) Principles of neurodynamics. Perceptrons and the theory of brain mechanisms. Cornell Aeronautical Lab Inc, Buffalo
Safavian SR, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern 21:660–674
Scarfone K, Mell P (2007) Guide to intrusion detection and prevention systems (idps). NIST Spec Publ 800:94
Schapire RE (2003) The boosting approach to machine learning: an overview. Nonlinear Estim Classif 149–171
Scikit Learn, Machine Learning in Python. https://scikit-learn.org/stable. Accessed 6 July 2021
Sethi (2020) One-hot encoding vs. label encoding using scikit-learn. https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/. Accessed 30 Sept 2021
Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27:379–423
Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. Icissp 1:108–116
Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA (2012) Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput Secur 31:357–374
Song J, Takakura H, Okabe Y, et al (2011) Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In: Proceedings of the first workshop on building analysis datasets and gathering experience returns for security, pp 29–36
Tama BA, Rhee K-H (2019) An in-depth experimental study of anomaly detection using gradient boosted machine. Neural Comput Appl 31:955–965
Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5:21954–21961
Zaman S, Karray F (2009) Features selection for intrusion detection systems based on support vector machines. In: 2009 6th IEEE consumer communications and networking conference. IEEE, pp 1–8

Publisher's Note