

RESEARCH

Open Access



Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset

Sydney M. Kasongo*  and Yanxia Sun

*Correspondence:
sydneybleuops@gmail.com
Department of Electrical
and Electronic Engineering
Science, University
of Johannesburg, Kingsway
Ave, Johannesburg 2006,
South Africa

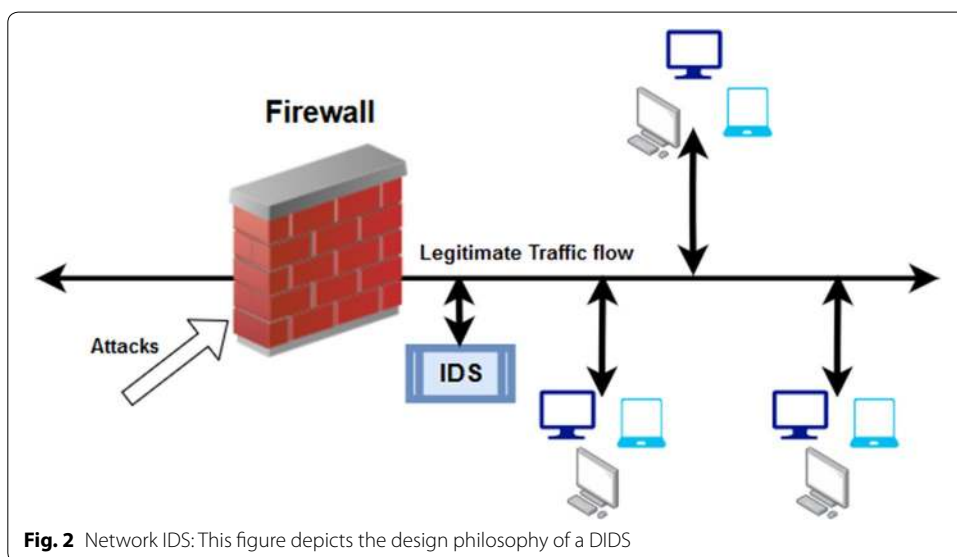
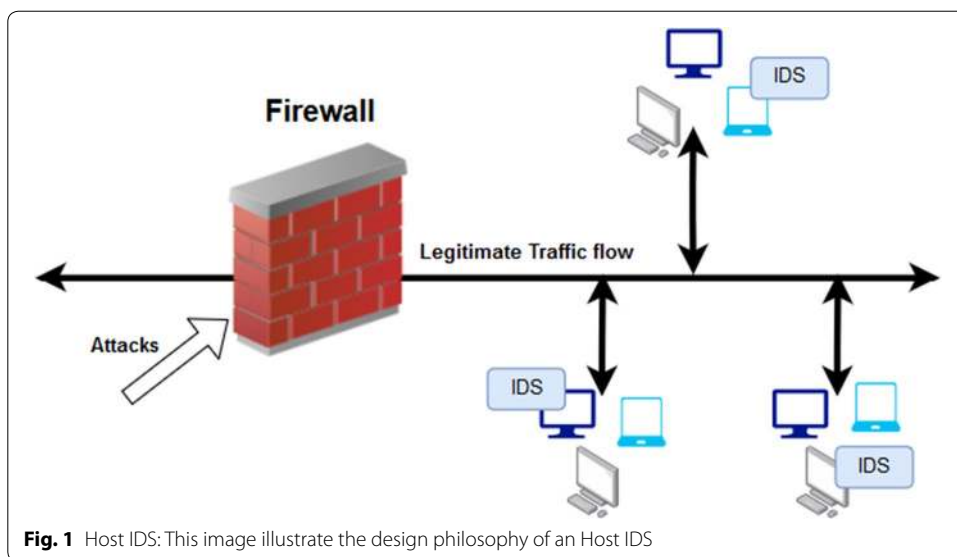
Abstract

Computer networks intrusion detection systems (IDSs) and intrusion prevention systems (IPSS) are critical aspects that contribute to the success of an organization. Over the past years, IDSs and IPSS using different approaches have been developed and implemented to ensure that computer networks within enterprises are secure, reliable and available. In this paper, we focus on IDSs that are built using machine learning (ML) techniques. IDSs based on ML methods are effective and accurate in detecting networks attacks. However, the performance of these systems decreases for high dimensional data spaces. Therefore, it is crucial to implement an appropriate feature extraction method that can prune some of the features that do not possess a great impact in the classification process. Moreover, many of the ML based IDSs suffer from an increase in false positive rate and a low detection accuracy when the models are trained on highly imbalanced datasets. In this paper, we present an analysis the UNSW-NB15 intrusion detection dataset that will be used for training and testing our models. Moreover, we apply a filter-based feature reduction technique using the XGBoost algorithm. We then implement the following ML approaches using the reduced feature space: Support Vector Machine (SVM), k-Nearest-Neighbour (kNN), Logistic Regression (LR), Artificial Neural Network (ANN) and Decision Tree (DT). In our experiments, we considered both the binary and multiclass classification configurations. The results demonstrated that the XGBoost-based feature selection method allows for methods such as the DT to increase its test accuracy from 88.13 to 90.85% for the binary classification scheme.

Keywords: Machine learning, Feature engineering, Computer networks, Intrusion detection

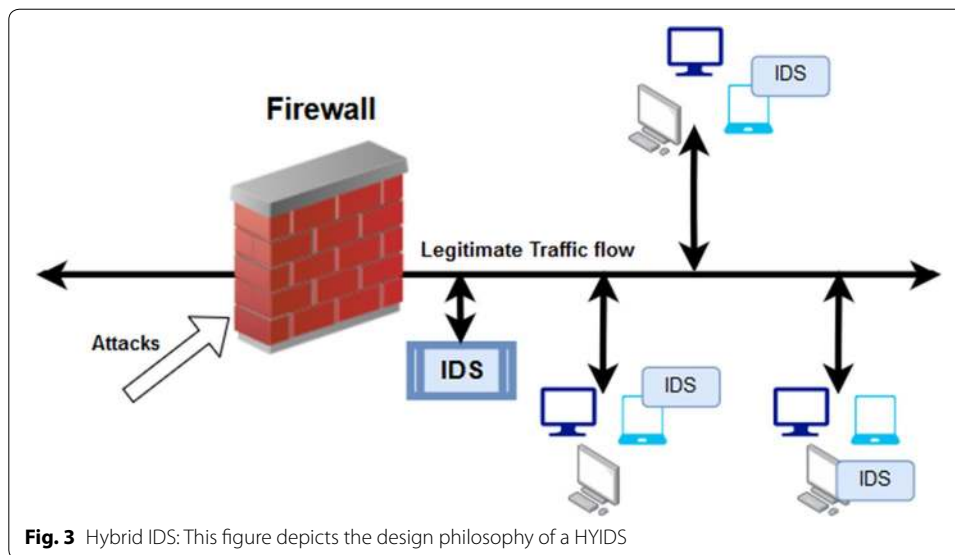
Introduction

The rapid pace at which technologies such as the Internet, Internet-of-Things (IoT) and communication systems is advancing has caused hackers to evolve with a higher velocity in terms of their capabilities. These criminals strive to discover new ways to compromise computer networks security. Therefore, Intrusion Detection Systems (IDSs) inherently become critical components in a computer network. An IDS is defined as a hardware or a software system that monitors an organization's computer network for imminent and potential threats or attacks. Moreover, an IDS is capable



of responding to any malicious transactions and report them accordingly [1]. IDSs are categorized in 3 broad classes, namely, node or host based IDS (HIDS) (Fig. 1), network or distributed IDS (DIDS or NIDS) (Fig. 2) and Hybrid IDS (HYIDS) (Fig. 3). This classification is based on the specific IDS operation philosophy. HIDS runs on a single node computer system and it is concerned with the security related to its host. In contrast, a DIDS runs on a distributed computer network and analyses the traffic at the network level for suspicious activities. HYIDS is a combination of DIDS and HIDS [2, 3].

Moreover, all classes of IDSs can be assorted into the following categories: signature-based, anomaly-based and hybrid-based. A signature-based IDS uses an existing data store of previously intercepted attacks in order to detect intrusions. In contrast,



an anomaly-based IDS is concerned with the behaviour of the network. It constantly checks for activities that constitute unusual behaviour and flags them. Finally, hybrid-based IDS is a combination of both anomaly-based and signature-based IDSs [4–6].

The primary philosophical design goal of an IDS is to minimise instances of false positive alarms and to increase its detection accuracy. Therefore, the design and implementation of any IDS must take that philosophy in consideration [7]. In recent years, Machine Learning (ML) based IDSs have emerged as the leading systems in the intrusion detection research domain. ML gives systems the ability of learning and improving by using previous data. In other words, ML based computer programs do not need to be explicitly engineered (programmed). They are capable of learning by themselves [8]. Broadly speaking, there exist the following sorts of ML philosophies, namely, supervised ML and unsupervised ML. In the case of supervised ML, models are learned using data that is labelled [9]. In the instance of unsupervised ML, the data that is used to train models is unstructured (unlabelled) [10].

In this research, supervised ML methods and more specifically, the binary and multiclass classifications tasks, are considered. The classification procedure happens when a supervised ML model is tasked to predict a discrete value [11]. The datasets used to train the models in this set-up are generally large and possess a high dimensional feature space. Due to that dimensionality, it is often time consuming to train and test supervised ML models. Therefore, it is critical to perform feature engineering processes to drop the number of features required for the training and testing phases [2, 12]. For the work presented in this paper, a filter-inspired feature reduction technique using the feature importance measures generated by the XGBoost Algorithm [13] is considered. Unlike the wrapper-based feature extraction approach; the filter-inspired feature extraction technique performs the feature space reduction process without the intervention of the classifier used for the final predictions [14]. Furthermore, a thorough literature review is conducted and the results obtained by the various ML methods that were surveyed are compared to those obtained in this research.

In our experimental processes, we applied the following supervised ML methodologies for IDS: k-Nearest-Neighbour (kNN), Logistic Regression (LR), Artificial Neural Network (ANN), Support Vector Machine (SVM) and Decision Tree (DT). We apply the XGBoost methodology over the UNSW-NB15 dataset [15, 16] to calculate the feature importance measure for each feature so as to generate a reduced and optimal feature vector. Furthermore, we have conducted a performance analysis that involves the binary classification and the multiclass classification schemes for each of the selected ML algorithms.

The remainder of this work is structured as follows. Firstly, an account of related work is presented. An highlight of all the ML techniques that are used in this research is provided. Moreover, the details about the UNSW-NB15 dataset are given. Furthermore, the proposed IDS approach is explained as well as all the algorithms involved during the experimental processes. A summary of all the experimental processes and discussions is then presented. The last section of this paper is the conclusion.

Related Work

The researchers in [17] implemented the Genetic Algorithm (GA) in conjunction with the Logistic Regression(LR) wrapper based feature selection methodology over the UNSW-N15 and KDDCup99 datasets. In this research, the Weka simulation tool was used. After multiple simulations, the results showed that the GA-LR coupled with the DT classifier attained a detection score of 81.42% as well as a FAR of 6.39% with 20 features out the 42 features present within the UNSW-NB15 feature space. With regards to the KDDCup99 dataset, the GA-LR in conjunction with the DT classifier obtained a detection score of 99.90% and a FAR rate of 0.105% using 18 features.

The authors in [18] presented a filter-based approach using multiple filters for Distributed Denial of Service (DDoS) detection. The filter methods that were used includes: Information Gain, Chi-Square and Gain Ratio, ReliefF. In a bid to demonstrate the capability of this system, the researchers employed the NSL-KDD attack detection dataset. For the classification process the authors employed the Decision Tree (DT) algorithm trained and validated with the help of the k -fold cross-validation method where $k = 10$. The experimental results showed that with only 13 features out of the 42 features (full feature space); the DT classifier was able to obtain a detection accuracy score equal to 99.67% as well as a false alarm rate (FAR) equating 0.42%. However, this research did not dive deep into the study of the multiclass classification problem of the NSL-KDD.

In [19], an IDS coupled to a filter-inspired input reduction approach was introduced. In this work, the different datasets including the Kyoto 2006, the KDDCup99 as well as the NSL-KDD were utilized for assessment. The authors in this work considered a Flexible Mutual Information (FMI) technique in order to assert the correlation that could exist between different input variables. The FMI is a non-linear correlation measure. The classifier used in the experiments is the Least Square SVM (LS-SVM). The results showed that for the NSL-KDD, with 18 features, the LS-SVM FMI achieved a FAR 0.28% and an accuracy 99.94%. In the instance of KDD Cup 99, the LS-SVM FMI obtained an overall accuracy of 78.86% and in the case of Kyoto 2006 +, the LS-SVM FMI attained a detection rate of 97.80% with a FAR rate of 0.43% on iteration 10.

The authors in [20] implemented an IDS using a filter-based method methodology with the aim to drop the number of input attributes (features) required to perform the training and testing of their model. The DT classifier was used in together with a correlation input selection technique. The dataset utilized in the experimental processes is the NSL-KDD. 14 features were selected after the filter was applied on the feature space. Moreover, the author considered the multiclass classification setting that included all five classes of attacks within the NSL-KDD as well as the binary classification configuration. The experimental outcome demonstrated that the system yielded an accuracy of 83.66% for the multiclass configuration and 90.30% for the binary setting.

In [21], the researchers implemented a feature reduction method for intrusion detection system using the Pigeon Inspired Optimizer (PIO). The PIO is a bio-inspired algorithm influenced from the flight of white pigeons. These birds constantly regulate their flight position by referring to that of the best bird in the flock [22]. In this work, the following two types of PIO were considered: Sigmoid PIO and Cosine PIO. The the UNSW-NB15, the NSL-KDD and the KDDCup99 intrusion detection sets were considered. The Sigmoid PIO selected 10 features and the Cosine PIO selected 7 features of the KDDCup99. The Sigmoid PIO selected 18 features and the Cosine PIO selected 5 features of the NSL-KDD. The Sigmoid PIO selected 14 features and the Cosine PIO selected 5 features of the UNSW-NB15. With regards to accuracy, the Sigmoid PIO achieved an accuracy of 94.7% using the KDDCup99, 86.9% over the NSL-KDD and 91.3% on the UNSW-NB15. In contrast, Cosine PIO obtained 96.0% on the KDDCup99, 88.3% over the NSL-KDD and 91.7% on UNSW-NB15.

Janarthanan and Zargari [23] implemented a number of feature selection algorithms through the UNSW-NB15 in an effort to select an optimal feature space. Using the Weka tool, the following algorithms were implemented: attribute evaluator (CfsSubsetEval), Greedy Stepwise and Information Gain and the Ranker Method. After various simulations, two subset were considered. The authors used the Kappa Statistic measure to evaluate the effectiveness of each subset. Many classifiers were considered during the experiments; however, the RF classifier was selected as the best method in terms of its overall performance. The first subset with 8 significant features achieved a Kappa Score of 0.6891 and an accuracy of 75.6617% over test dataset. The second subset using only 5 significant features achieved a Kappa score of 0.7639 and an accuracy of 81.6175%.

Vikash and Ditipriya [24] implemented an IDS system that was validated using the UNSW-NB15 dataset. In this research, the authors considered a feature reduction technique that was inspired from the Information Gain methodology. This filter-based feature extraction technique resulted in the selection of 22 most important attributes. Moreover, the IDS proposed in this work utilized an integrated rule-based model that used multiple Tree-based classifiers in order to carry out the the classification process. The performance of this system was mainly measured using the Attack Accuracy (AAc) achieved on the test data, the F-Measure (FM) and False Alarm Rate (FAR). The results show that the proposed IDS obtained an AAc of 57.01%, an FM of 90% and a FAR of 2.01%. The results obtained in this research can be improved when considering alternative ML algorithms that could be integrated in replacement to a strict adherence to Tree based methods.

In [25], the researchers introduced a feature extraction algorithm based on Particle Swarm Optimization (PSO), Firefly Optimization (FO), Grey Wolf Optimization (GO) and Genetic Algorithm (GA). These methods were applied iteratively on the UNSW-NB15 dataset in order to discover which feature subset would produce an optimal attack detection accuracy. After several experiments, a feature subset containing 30 features was selected. Furthermore, the J48 Tree based model and the SVM models were used in order to conduct the classification process. The main performance criteria considered in this study were the Accuracy, the False Positive Rate (FPR), the False Negative Rate (FNR) as well as the FM. The experiments were carried out on the UNSW-NB15 training subset and the binary classification scheme was implemented. The results demonstrated that the proposed J48 model achieved a training accuracy of 90.484%, an FPR of 14.950% and an FM of 90.172%. Additionally, the proposed SVM model achieved a training accuracy of 90.119%, an FM of 89.808%, an FPR 15.391% and an FNR of 3.130%.

Maajid and Nalina [26] implemented a feature reduction method based on the RF algorithm in order to generate the Feature Importance (FI) score of each attribute present in the UNSW-NB15 dataset. The FI algorithm is a ranking system whereby the feature with the highest FI is the most important attribute in the classification process with regards to the target variable (class). After several experiments, a feature subset containing 11 attributes was selected. For the classification procedure, the authors considered the following ML approaches: kNN, DT, Bagging Meta Estimator (BME), XGBoost and RF. In order to assess the performance of these methods, the accuracy and the F-Measure scores obtained on test data were used as the principal metrics. The RF algorithm obtained the best results with an AC of 75.56% and an FM of 73.00%.

In [27], a two-stage ensemble model for IDS using the Rotation Forest and Bagging algorithms was proposed. This multiple stage meta classifier model used a majority voting approach to decide on the best course of action. Additionally, a feature selection method using the Particle Swam Optimization (PSO), the GA and the Ant Colony Optimization (ACO) was implemented. The PSO-GA-ACO technique was applied on UNSW-NB15 and 19 optimal features were extracted. In order to validate their two-stage model, the researchers used the 10-fold cross-validation technique in conjunction with the hold-out method. The performance metrics considered in this work were the AC, the FPR, the sensitivity and the Precision. Therefore, for the binary classification scheme, the methodology proposed in this work achieved an AC of 91.27%, a precision of 91.60% and a sensitivity of 91.30% over the UNSW-NB15 test subset.

Zong et al. [28] proposed an IDS using a two-stage (TS) classifier model based on the RF classifier. The Information Gain (IG) method was used to select the attributes required for the binary classification process. The authors used the one-hot encoding technique in order to transform the categorical features of the UNSW-NB15 dataset. In the first stage, the IG-TS is concerned with the detection of minority classes and in the second phase, the IG-TS carries out the detection of majority classes. The results of each stage in then combined to provide the final prediction. The researchers used the AC and the FAR as performance metrics. After several experiments, the IG-TS achieved an AC of 85.78% and a FAR of 15.78%.

In [29], a two-stage model using the RepTree algorithm for NIDS was presented. To assess the performance of this method, the authors used the UNSW-NB15 dataset

amongst others. In the first stage of the framework, the classifier subdivides the network traffic into the following 3 categories: UDP, TCP and Other (Other kinds of network traces). In the second stage, the RepTree classifies the anomalies recorded in the initial stage and performs the attacks predictions. Moreover, the framework proposed in this research used the IG and Consistency (IGC) method to reduce the feature space. The results for the binary classification scheme showed that the TS-RepTree achieved an accuracy of 88.95% over the test dataset with about 20 most relevant features. The drawback of this research is the fact that the authors only considered one metric to evaluate the performance of their framework.

Gao et al. [30] proposed an IDS using an incremental approach of the Extreme Learning Machine (IELM) in conjunction with an Advanced Principal Component (APCA) algorithm. The APCA has the role to adaptively select the most relevant features required by the IELM so as to perform an optimal attack prediction. The researchers used the UNSW-NB15 in to evaluate the performance of the proposed IDS framework. The accuracy obtained on the test data was the most important metric considered in order to arrive at their conclusion. In addition, the authors also contemplated other metrics such as the detection rate (DR) and the FAR. The experimental results showed that the IELM-APCA obtained an accuracy of 70.51%, a DR of 77.36% and a FAR of 35.09%.

In [31], an IDS adapted to the Edge-of-Things (EoT) was presented. The EoT is a paradigm that takes the Internet of Things (IoT) to the next level whereby it becomes possible for data to be processed and transferred from Cloud Servers to Edge devices such as hand-held computers, mobile phones, laptops, IoT devices, etc. with minimal delay. The EoT-IDS is composed a feature selection module that uses the correlation method to select the best attributes. Moreover, once the feature subset is picked, it is fed into a Deep Belief Network (DBN) classifier to carry out the predictions. The authors used the UNSW-NB15 to assess the efficiency of their proposed system. The accuracy obtained on test data was used as the main performance metric. The authors conducted the experiments using several structures of DBN networks whereby the best configuration (layer 1 = 64 hidden units, layer 2 = 60 hidden units and the serial number of the DNB architecture was 28) achieved an accuracy of 85.73% for the binary classification process.

Kaiyuan et al. [32] proposed an NIDS framework that used the One-Side Selection (O-SS) method to reduce the amount of noisy data records within the majority classes. Moreover, the authors utilized a synthetic approach (Synthetic Minority Over Sampling—SMOS) to increase the number of minority examples within the dataset. Furthermore, Convolutional Neural Networks (CNN) were employed for extracting the spatial attributes and Bidirectional Long-Short Term Memory (Bi-LSTM) models were used to select the temporal attributes. The combination of CNN-Bi-LSTM is the DL model that the authors used to conduct the predictive tasks. To evaluate the proposed framework, the researchers used the UNSW-NB15 and NSL-KDD intrusion detection datasets, and they considered the accuracy obtained on test data as the main performance metric. The results showed that the CNN-Bi-LSTM attained accuracies of 77.16% and 83.58% for each dataset respectively.

Overview of ML Methods

The following sections provide a review of the supervised ML approaches that are used in this work.

Support Vector Machine

Support Vector Machine (SVM) happens to be one of the most flexible ML model that is able to conduct both classification and regression tasks. SVM is amongst one of the foremost well known models in ML research. The objective of the SVM model is to partition a given dataset into different classes in a bid to discover the optimal hyperplanes. Some of the pros of using SVM include the fact that it tends to perform effectively for high dimensional input spaces. Furthermore, the SVM model allows for selection of a variety of Kernel functions in the decision making process. One of its disadvantages is that, it requires careful tuning in general and especially when the input dimension is greater than the number of examples [33].

Logistic Regression

Logistic Regression (LR) is a ML technique that is primarily employed for the binary classification task although it is termed as “regression”. The LR can also be used for the multiclass classification tasks whereby the learning algorithm employs the one-vs-rest methods. The LR model applies the sigmoid function or its variations to a linear ML model. The output of this operation is squashed between [0, 1]. An output that is closer to 1 determines the probability of a given class. The mathematical formulation is described in Eqs. 1, 2 and 3 [34].

$$\text{Linear Model: } y = b + w_1x_1 + w_2x_2 + \dots + w_nx_n \quad (1)$$

$$\text{Sigmoid Expression: } \sigma(k) = \frac{1}{1 + e^{-k}} \quad (2)$$

$$\text{Logistic Regression: } \sigma(y) = \frac{1}{1 + e^{-y}} \quad (3)$$

In the above formulation, y is processed by σ .

Artificial Neural Network

Artificial Neural Network (ANN) is a ML philosophy that mirrors the inner workings of the biological neurons. The most basic component of an ANN is referred to as a node or a neuron. An ANN with an input layer that is processed by one hidden layer to provide an output is alluded to as a Perceptron. ANN that possesses multiple hidden (middle) layers are sometimes called Feed-forward ANN or Multilayer Perceptrons (MLP) [35]. Neurons in an ANN are processed via an activation expression such as the Sigmoid $\sigma(k) = \frac{1}{1+e^{-k}}$, a Rectified Linear Unit (*ReLU*) $f(k) = \max(0, k)$, an

Hyperbolic Tangent $\tanh(t) = \frac{1-e^{-2t}}{1+e^{-2t}}$ or a variation of those functions. A visual representation of an ANN is illustrated in Fig. 4 [36, 37].

Decision Tree

Decision Tree (DT) is a supervised ML technique that is employed for classification and regression tasks. The model resulting from the decision making process is shaped like a tree. This makes DT less complex to interpret by the user. Moreover, various ML tools allow for the visualization of the resulting trees [38].

k-Nearest-Neighbour

The k-Nearest-Neighbour (kNN) ML algorithm is capable of conducting both supervised and unsupervised processes. For instance, kNN is the basis of many of the clustering algorithms in use today [39]. In this research, we used the kNN method in its supervised ML flavour. This technique bases itself on the Standard Euclidean Metric (EM). The EM is a distance that separates two points (instances) in a space [40]. Lets p and q denote two instances in an Euclidean space Z , the EM between p and q , $\Delta(p, q)$, is computed as follows:

$$\Delta(p, q) = \sqrt{\sum_{i=1}^r (q_i - p_i)^2} \tag{4}$$

r denotes the maximum number of instances within the space Z . The kNN technique finds the identity (label) of an example r_0 within Z by calculating the EM separating r_0 and its k closest instances within Z and the label (class) of r_0 is selected with reference to the class of its k related neighbouring instances.

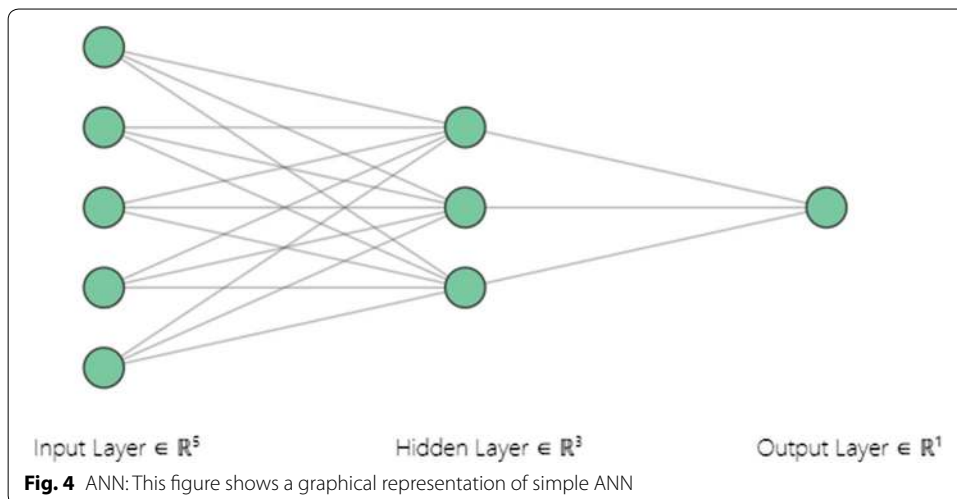


Table 1 UNSW-NB15 list of features

Feature number	Feature	Format	Feature number	Feature	Format
f1	dur	Float	f22	dtcpb	Integer
f2	proto	Categorical	f23	dwin	Integer
f3	service	Categorical	f24	tcprrt	Float
f4	state	Categorical	f25	synack	Float
f5	spkts	Integer	f26	ackdat	Float
f6	dpkts	Integer	f27	smean	Integer
f7	sbytes	Integer	f28	dmean	Integer
f8	dbytes	Integer	f29	trans_depth	Integer
f9	rate	Float	f30	response_body_len	Integer
f10	sttl	Integer	f31	ct_srv_src	Integer
f11	dttl	Integer	f32	ct_state_ttl	Integer
f12	sload	Float	f33	ct_dst_ltm	Integer
f13	dload	Float	f34	ct_src_dport_ltm	Integer
f14	sloss	Integer	f35	ct_dst_sport_ltm	Integer
f15	dloss	Integer	f36	ct_dst_src_ltm	Integer
f16	sinpkt	Float	f37	is_ftp_login	Binary
f17	dinpkt	Float	f38	ct_ftp_cmd	Integer
f18	sjit	Float	f39	ct_flw_http_mthd	Integer
f19	djit	Float	f40	ct_src_ltm	Integer
f20	swin	Integer	f41	ct_srv_dst	Integer
f21	stcpb	Integer	f42	is_sm_ips_ports	Binary

UNSW-NB15 Dataset

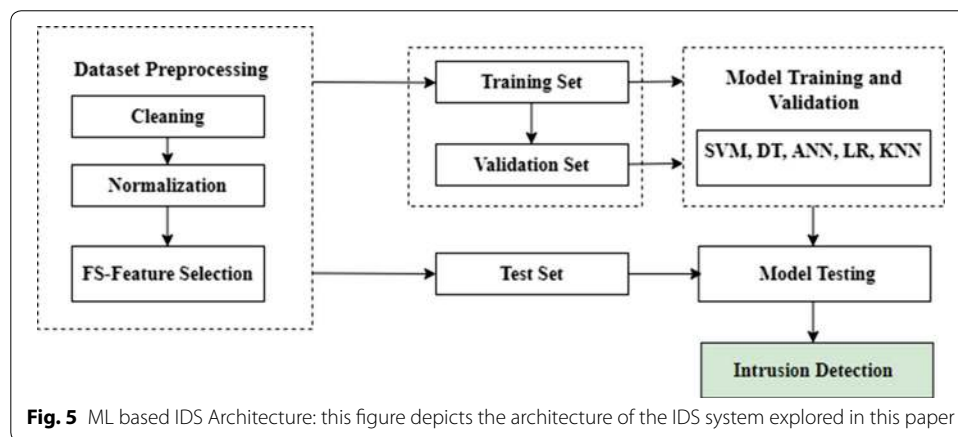
For our experimental processes we utilize the UNSW-NB15 attacks dataset [41]. In its clean format, the UNSW-NB15 contains 42 features as shown in Table 1. Out of the 42 features, 3 instances are non-numeric (categorical) features and 39 are numeric in nature.

The UNSW-NB15 is subdivided in the following main datasets: UNSW-NB15-TRAIN, which is employed for training various models and the UNSW-NB15-TEST (100%) which is employed for testing the trained models. In our work, we further split the UNSW-NB15-TRAIN in the following two partitions: the UNSW-NB15-TRAIN-1 (75% of the full training set) for training and the UNSW-NB15-VAL (25% of the full training set) for validation before testing. This second partition is used as a sanity check for the results obtained during the training process. When using this strategy, it is crucial to avoid that a model trains on the evaluation or the test set because it could lead to a phenomenon known as data leakage. Data leakage occurs during the training process when a model sees information that is not supposed to and therefore introducing a bias in the final model. This in turn leads to a poor model performance on previously unseen data [42].

The UNSW-NB15 contains instances with the following categories of network attacks: Backdoor, Shellcode, Reconnaissance, Worms, Fuzzers, DoS, Generic, Analysis, Shellcode and Exploits. Moreover, Table 2 provides the details and the values distribution of each attack class within the data subsets.

Table 2 UNSW-NB15 instances repartition

Attack Type	UNSW-NB15	UNSW-NB15-TRAIN-1	UNSW-NB15-VAL	UNSW-NB15-TEST
Normal	56,000	41,911	14,089	37,000
Generic	40,000	30,081	9919	18,871
Exploits	33,393	25,034	8359	11,132
Fuzzers	18,184	13,608	4576	6062
DoS	12,264	9237	3027	4089
Reconnaissance	10,491	7875	2616	3496
Analysis	2000	1477	523	677
Backdoor	1746	1330	416	583
Shellcode	1133	854	279	378
Worms	130	99	31	44



The Proposed Approach

The architectural design of the system proposed in this research is detailed in Fig. 5. The first block is all about data processing. This process is often referred to as the data engineering. This step is critical for a successful learning process. Data processing has three steps, namely, cleaning, normalization and feature selection. The feature selection process is conducted using a filter-based method inspired by the XGBoost algorithm for generating feature importances scores. Once the required feature vector is selected; the next process involves model training using the training set. A trained model is then validated with the validation set. Finally, the test dataset is used to test the validated model. The procedure described above happens in an iterative manner till a tuned and fit model is found.

Feature Normalization

The high numerical value of various features have an impact on the learning process on ML approaches such as SVM, LR, ANN and kNN. Also, the training of high dimensional datasets requires a great deal of computational resources. To mitigate those issues, data is often scaled using methods such as the Z-score standardization,

Decimal scaling, Max normalization and Min–Max scaling [43]. The choice of which method to use is often based on the application. We use the Min–Max scaling (Eq. 5) in our data processing step.

$$\text{Min-Max scaling of feature } F: F_{norm} = \frac{F - F_{min}}{F_{max} - F_{min}} \tag{5}$$

Given a dataset with an input vector (feature space) represented by $U(f_1, \dots, f_n)$, $1 < n < N$, where N is the total number of instances (features) in the space, the standardization computation happens as described in Algorithm 1.

Algorithm 1 Min-Max Scaling Algorithm

Input: $U(f_1, \dots, f_n)$, where $1 < n < N$

Output: $U_{normalized}(f_1^{norm}, \dots, f_n^{norm})$:

for i from 1 to k do

 if (f_i a non-numeric input) then

 Step 1: encode using one-hot-encode or scikit-learn feature mapping

 Step 2: Compute Min-Max Scaling: $f_n^{norm} = \frac{f_n - (f_n)_{min}}{(f_n)_{max} - (f_n)_{min}}$

 end if

 Step 1: Compute Min-Max Scaling : $f_n^{norm} = \frac{f_n - (f_n)_{min}}{(f_n)_{max} - (f_n)_{min}}$

end for

XGBoost Technique

The basic concept with the XGBoost algorithm is to train an Ensemble (Group) Tree Model in an incremental fashion with a penalty parameter, θ , that is added to reduce the complexity of the model. The goal is to minimize the expression in Eq. 6. The mathematical formulation is explained below:

Let $\hat{y}^{(t)}$ designate the prediction of the k -th example at the t -th increment, the objective function is then expressed below:

$$L^{(t)} = \sum_{k=1}^n l(y_k, \hat{y}_k^{(k-1)} + f_t(X_i)) + \theta(f_t) \tag{6}$$

where L is the loss function. The goal of the XGBoost method consist in minimizing the above objective function using the following simplified second-order expression:

$$\tilde{L}^{(t)} = \sum_{k=1}^n [g_k f_t(X_k) + \frac{1}{2} h_k f_t^2(X_k)] + \theta(f_t) \tag{7}$$

where $g_k = \partial_{\hat{y}^{(t-1)}} l(y_k, \hat{y}^{(t-1)})$ and $h_k = \partial_{\hat{y}^{(t-1)}}^2 l(y_k, \hat{y}^{(t-1)})$. g_k and h_k denote the first and the second order expressions of the gradients with regards to the loss function. The XGBoost algorithm has the ability to compute a score for each feature given a dataset. This score metric is labelled as the Feature Importance (FI) [44]. In this research, we use the FI measure as a means to decide which inputs are important for the learning and classification procedures. The pseudo-code explaining the XGBoost method of generating the feature importances using Scikit Learn is described in Algorithm 2.

Algorithm 2 XGBoost Algorithm: Generate Feature Importances

Input: $U_{normalized}(f_1^{norm}, \dots, f_n^{norm})$
Output: $U_{optimal}$: the selected feature vector
 Step 1: Load the normalized feature vector
 Step 2: Create an empty dictionary S to save the scores.
 Step 3: Instantiate a *GradientBoostingClassifier* as clf
 Step 4: fit clf
 Step 4: Generate FIs
 Step 5: Determine the FI threshold FI_p
 Step 6:
 for n from $U_{normalized}$ do
 if $(FI(x^i) \geq FI_p)$ then
 append $FI(x^i)$ into S
 end if
 end for
 Step 5: Use the scores in S to generate $U_{optimal}$

We have applied the XGBoost method over the UNSW-NB15 dataset and the outcome have yielded 19 features as detailed in Table 3.

Experiments, Results and Discussions

Hardware and Environment Setting

The experiment presented in this work are conducted on a DELL 153000 Inspiron 3580 loaded with the Windows 10 Operating System with the following processor: Intel(R) Core(TM) i7-8568U CPU @ 1.80GHz to 1.99 GHz. The ML models are built, trained, evaluated and tested on the Scikit-Learn ML Python framework [45]. Scikit-Learn is a versatile open source platform that is constructed on top of matplotlib, NumPy and

Table 3 UNSW-NB15 selected features using the XGBoost method

Feature number	Feature	Format	Feature importance score
f10	sttl	Integer	0.803374
f41	ct_srv_dst	Integer	0.039387
f7	sbytes	Integer	0.037377
f27	smean	Integer	0.019878
f2	proto	Integer	0.018848
f32	ct_state_ttl	Integer	0.016783
f14	sloss	Integer	0.012008
f25	synack	Float	0.010125
f35	ct_dst_src_ltm	Integer	0.007203
f28	dmean	Integer	0.007134
f31	ct_srv_src	Integer	0.006745
f3	service	Categorical	0.006305
f13	ct_dst_sport_ltm	Integer	0.003717
f8	dbytes	Integer	0.002706
f15	dloss	Integer	0.001793
f4	state	Categorical	0.001548
f24	tcprtt	Float	0.001224
f34	ct_src_dport_ltm	Integer	0.000526
f9	rate	Float	0.000503

Scipy Python libraies. Moreover, Classification, Regression and Clustering tasks can all be conducted using Scikit-Learn.

Performance Metrics

There exist an number of metrics to evaluate ML based IDS systems; however, this research aims to maximize the correct predictions of instances in the test dataset. The main measure to look at is the *Accuracy*(AC) defined below:

$$Accuracy = \frac{TN + TP}{FP + FN + TP + TN} \quad (8)$$

whereby the TP stand for True Positive and is the rate of examples correctly identified as attacks. TN , True Negative, is the rate of legitimate traffic classified as legitimate. FP , False Positive, sometimes referred to as Type I error, is the rate of legitimate traffic classified as attacks. FN , sometimes referred to as Type II error, is the rate of legitimate traffic classified as intrusions. Additional metrics we consider in this paper are the *Recall*, the *Precision* and $F1_{score}$ defined below:

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

$$F1_{score} = 2 \frac{Precision \cdot Recall}{Precision + Recall} \quad (11)$$

The $F1Score$ is defined as the harmonic mean of the expression in (9) and (10). Moreover, some applications are designed to be *Precision* or *Recall* oriented. In order to achieve that in the training process, the F_{β} defined below can be tuned:

$$F_{\beta} = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta FN + FP} \quad (12)$$

where the parameter β can adjusted based on the application. In our research however, we use the $F1_{score}$ which is a trade-off between the *Precision* and the *Recall*.

Experiments and Results

Our experimental design was categorized in two main processes whereby we considered the following ML techniques: ANN, LR, kNN, SVM and DT. In the first step of the experiments, we employed the full feature space (42 features) of the UNSW-NB15 for the binary setting and for the multiclass configuration. In the second phase, we introduced the feature selection method using the XGBoost algorithm. Subsequently, a reduced feature vector containing 19 attributes (Detailed in Table 3) was generated. Using the optimal feature vector, we conducted the experiments that involved the binary classification step along with the multiclass classification procedure. The results of our experimental processes are listed in Tables 4, 5, 6, 7 whereby Table 4 provides the results obtained by the ML methods for the binary classification technique using the full feature

Table 4 Results using 42 features—binary classification

ML method	Tr. AC (%)	Val. AC (%)	Test AC (%)	Precision (%)	Recall (%)	F1-Score (%)
ANN	94.49	94.21	86.71	81.54	98.06	89.04
LR	93.22	92.87	79.59	73.32	98.94	84.22
KNN	96.76	93.60	83.18	79.15	94.30	86.06
SVM	70.98	70.63	62.42	60.91	88.58	71.18
DT	93.65	93.37	88.13	83.91	96.47	90.00

Table 5 Results using 19 features—binary classification

ML Method	Tr. AC (%)	Val. AC (%)	Test AC (%)	Precision (%)	Recall (%)	F1-Score (%)
ANN	93.75	93.66	84.39	78.56	98.53	87.42
LR	89.21	89.25	77.64	73.18	93.74	82.20
KNN	95.86	94.73	84.46	80.31	95.09	87.08
SVM	75.42	75.51	60.89	58.89	95.88	72.97
DT	94.12	93.81	90.85	80.33	98.38	88.45

Table 6 Results using 42 features—multiclass classification

ML Method	Tr. AC (%)	Val. AC (%)	Test AC (%)	Precision (%)	Recall (%)	F1-Score (%)
ANN	79.91	79.61	75.62	79.92	75.61	76.58
LR	75.51	73.93	65.53	76.91	65.54	66.62
KNN	81.75	76.83	70.09	75.79	70.21	72.03
SVM	53.43	52.67	61.09	47.47	62.00	53.77
DT	77.69	77.38	66.03	79.82	66.04	51.12

Table 7 Results using 19 features—multiclass classification

ML Method	Tr. AC (%)	Val. AC (%)	Test AC (%)	Precision (%)	Recall (%)	F1-Score (%)
ANN	79.46	78.91	77.51	79.50	77.53	77.28
LR	72.53	71.81	65.29	70.88	65.29	65.96
KNN	82.66	79.87	72.30	77.24	72.30	73.81
SVM	53.60	52.97	61.53	53.95	61.52	51.31
DT	78.75	78.43	67.57	79.66	67.56	69.26

space of the UNSW-NB15. Table 5 lists the results attained by the ML algorithms for the binary classification scheme using the reduced feature vector. Tables 6 and 7 list the results achieved by the ML algorithms for the multiclass classification task using the full feature space and the reduced feature vector, respectively. In each table, Tr. AC is the accuracy obtained on training data. Val. AC denotes the accuracy obtained on validation data and Test AC represents the accuracy obtained on test data.

With regards to ANNs, all our experiments used the Adam solver which is stochastic gradient-based method that is suitable for large datasets in lieu of the standard Stochastic Gradient Descent (SGD). The ANNs were designed using a single hidden layer that could have the following number of neurons: $hidden_layer_size = \{5, 10, 15, 30, 50, 100\}$.

The learning rate was adaptive. As depicted in Table 4, the ANN network that achieved the highest performance obtained an accuracy rate of 86.71% over the test set with the help of 150 neurons at an adaptive learning rate of 0.02.

For the LR method, the random state was set at 10 with a maximum iteration number of 1000. The results show that it obtained 79.59% and 77.64% of accuracy rates over the test set with regards to the binary classification utilizing the entire and reduced feature space respectively as detailed within Tables 4 and 5.

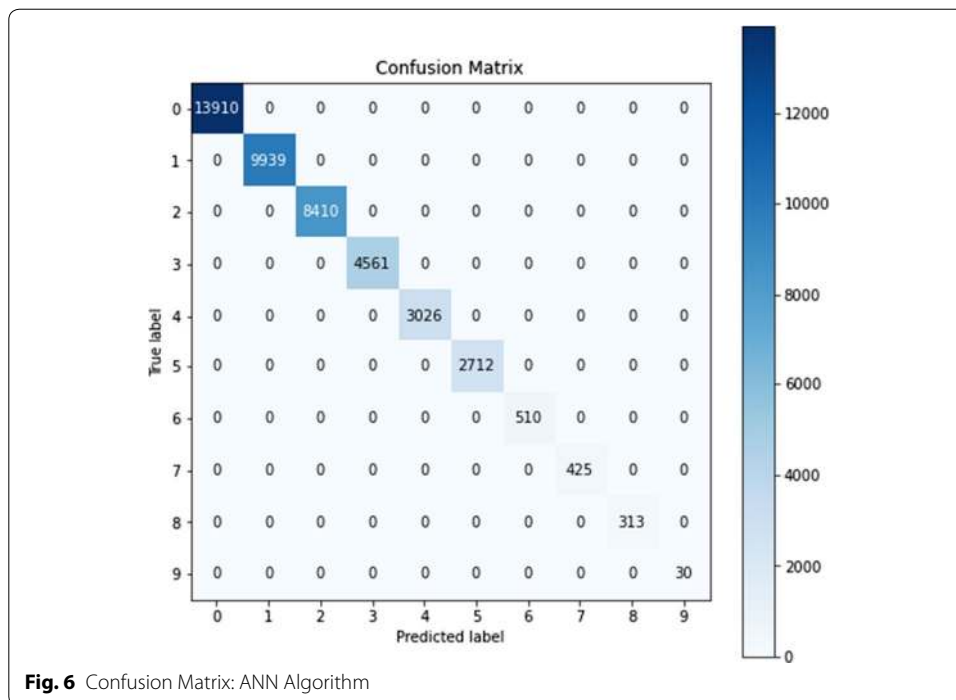
Using the kNN method, we trained multiple models with the number of neighbours set as follows: $number_of_neighbours = \{3, 5, 7, 9, 11\}$. The results show that in the case of the multiclass classification setup, a kNN classifier with 3 neighbours achieved a test score of 83.18% utilizing the entire feature space. In the instance of the reduced feature dimension, the kNN method obtained a test accuracy of 84.46% using 9 neighbours without overfitting.

The SVM method using the Radial Basis Function (RBF) as its kernel obtained a test accuracy of 62.42% and 60.89% using the full and reduced input space, respectively, with regards to the binary detection. In contrast, for the multiclass detection, the SVM obtained test accuracies of 61.09% and 61.53% using 42 and 19 features, respectively. The tuned parameters of the SVM classifier were as follows: $C = 1.12$ (regularization parameter), $gamma = 'scale'$ and $kernel = 'rbf'$.

In the case of the DT classifier, we conducted experiments using different models based on the trees maximum depth. The depth values could assume of the following numbers: $maximum_depth_values = \{2, 5, 7, 8, 9\}$. The outcome shows that the DT obtained a test accuracy of 88.13% using 42 features in the instance of the binary classification setup whereas it obtained a test score of 85.85% utilizing 19 features. The DT obtained a better test accuracy score compared to other ML methods when running the binary classification utilizing the entire feature dimension as well as the reduced one.

Discussions

After several iterations of experiments, the majority of ML methods used in this research improved on their test accuracies for the multiclass classification scheme by reducing the dimension of the feature vector from 42 to 19. In the instance of ANN, there has been an increase from 75.62% (42 features with 30 neurons in the hidden layer) to 77.51% (19 features and 15 neurons in the hidden layer). This shows that using only the most important features as reduced the complexity of the ANN method by halving the number of nodes required in the ANN network. This can also be seen in the trade-off between the precision (75.50%) and the recall (77.53%) through the test set. This produced a F1-Score of 77.28%. Moreover, the ANN method performed the best in terms of the AC, the Precision and the Recall obtained on test data. Additionally, a confusion matrix (Fig. 6) for the ANN multiclass classification process has been computed to evaluate the performance of individual attack classes present in the UNSW-NB15 whereby class 0 = *Normal*, class 1 = *Generic*, class 2 = *Exploits*, class 3 = *Fuzzers*, class 4 = *DoS*, class 5 = *Reconnaissance*, class 6 = *Analysis*, class 7 = *Backdoor*, class 8 = *Shellcode*, class 9 = *Worms*. The diagonal in Fig. 6 (From the top left to the bottom right) determines the efficiency of the ANN ML method for each class. The darker the square, the more accurate the prediction with regards to a given class. The results show that the



classifier performs well for class 0–5. However, class 7–9 are minority classes within the UNSW-NB15 and therefore, the classifier underperformed.

In the instance of kNN ML approach, the accuracy for multiclass classification increased from 70.09% (using the 42 features) to 72.30% (using the 19 features generated by the XGBoost method) on the test set. Whereas, the XGBoost-LR did not improve on the accuracy of the stand alone LR (using the full feature vector) for the multiclass classification schemes as well as the binary one.

In the case of the DT classifier, the test accuracy has increased from 66.03 to 67.57% using the 42 and 19 features, respectively, for the multiclass classification scheme. Moreover, for the binary classification process, the DT has increased the test accuracy from 88.13 to 90.85% employing the entire and the reduced feature dimensions of the UNSW-NB15 respectively. The results demonstrated the XGBoost helped in improving the DT algorithm predictive capabilities.

The use of the SVM method with the optimal input vector (19 features) has resulted in a test accuracy increase from 61.09 to 61.53% in the instance of the multiclass configuration. Moreover, for the binary classification scheme, the SVM approach increased its validation accuracy from 70.63 to 75.51%.

Moreover, in comparison to the work presented in [17] whereby the GA-LR-DT was used; the XGBoost-DT attained a test accuracy score of 90.85% as compared to 81.42% obtained by the GA-LR-DT (input dimension of 20 attributes of the UNSW-NB15). Furthermore, the results obtained in this paper are superior to those obtained in [21] whereby the Sigmoid PIO selected 14 optimal features of the UNSW-NB15 and obtained an accuracy score of 91.30% through the validation dataset. In comparison, the XGBoost-ANN, the XGBoost-kNN and the XGBoost-DT obtained validation accuracies of 93.66%, 94.73% and 93.81%, respectively, as detailed in Table 5. Finally, in Table 8, a

Table 8 Methods

ML methods	Feature extraction technique	Training AC (%)	Binary AC (%)	Multiclass AC (%)
DT [24]	IG	–	–	57.01
J48 [25]	PSO-FO-GO-GA	90.48	–	–
SVM [25]	PSO-FO-GO-GA	90.11	–	–
RF [26]	FI	–	–	75.56
Bagging Forest [27]	PSO-GA-ACO	91.27	–	–
RF [28]	IG	–	85.78	–
RepTree [29]	IGC	–	88.95	–
IELM [30]	APCA	–	–	75.36
DBN [31]	Corr.	–	85.73	–
CNN-BiLSTM [32]	O-SS-SMOS	–	–	77.16
DT	XGBoost	94.12	90.85	67.57
ANN	XGBoost	94.21	84.39	77.51
LR	XGBoost	89.20	77.64	65.29
kNN	XGBoost	95.86	84.46	72.30
SVM	XGBoost	75.42	60.89	53.95

comparison between the methods proposed in this paper against those surveyed in the literature show that the DT-XGBoost outperformed all other ML methods for the binary scheme and in the instance of the multiclass configuration, the ANN-XGBoost was the better option.

Conclusion

This research explored the application of the XGBoost algorithm for feature selection in conjunction with multiple ML techniques including ANN, kNN, DT, LR and SVM in order to implement accurate IDSs. The UNSW-NB15 dataset was utilized to assess the performance of these methods. In this work, the binary and the multiclass classification settings were considered. Furthermore, the XGBoost based attribute selection method was applied over the UNSW-NB15 and as a result, 19 optimal features were selected. To put our research into perspective, we carried out a thorough literature review whereby various techniques for feature selection applied to the UNSW-NB15 dataset were reviewed. Moreover, we compiled a summary of the performance results obtained by the various classifiers in the literature and we compared them to those obtained in our proposed methodology. Initially, we carried out the experiments using the proposed ML approaches over the full feature space of the UNSW-NB15 dataset. Thereafter, we ran the experiments using the reduced feature vector that was generated by the XGBoost feature extraction algorithm proposed in this work. The experimental results demonstrated that using a reduced (optimal) feature vector has its merits in terms of reducing the models complexity as well as increasing the detection accuracy on test data. A notable instance is the XGBoost-ANN. It managed to reduce number of neurons in the hidden layer by 50 percent the number of nodes in the stand alone ANN. Moreover, the XGBoost-kNN and the XGBoost-DT have also registered an improvement of their performance over unseen data (test data). Additionally, a study of the performance of individual classes within the UNSW-NB15 using the ANN-XGBoost algorithm demonstrated that this method

performs optimally when predicting the majority classes (class 0–6) and it underperforms with regards to the minority classes (class 7–9). In further work, we intend to use a synthetic oversampling algorithm to increase the occurrence of the minority classes within the UNSW-NB15 dataset during the training process. Furthermore, we plan to implement the XGBoost-based feature selection technique utilizing the NSL-KDD dataset and assess its performance in comparison to existing state of the art methods.

Acknowledgements

Not applicable.

Author's contributions

SM Kasongo wrote the algorithms related to this research and he interpreted the results that the Machine Learning based IDS systems generated. YS provided guidance in terms of validating the obtained results. All authors read and approved the final manuscript.

Funding

This research is funded by the University of Johannesburg, South Africa.

Availability of data and materials

The datasets used during the current study are available in the UNSW repository page, <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/> [46].

Competing interests

The authors declare that they have no competing interests

Received: 30 July 2020 Accepted: 6 November 2020

Published online: 25 November 2020

References

1. Wang Z: Deep learning-based intrusion detection with adversaries. *IEEE Access*. 2018;6:38367–384.
2. Kasongo SM, Sun Y. A deep gated recurrent unit based model for wireless intrusion detection system. *Cakovec: ICT Express*; 2020.
3. Ribeiro J, Saghezchi FB, Mantas G, Rodriguez J, Abd-Alhameed RA. Hidroid: prototyping a behavioral host-based intrusion detection and prevention system for android. *IEEE Access*. 2020;8:23154–168.
4. Van NTT, Tinh TN. Accelerating anomaly-based IDS using neural network on GPU. In: 2015 international conference on advanced computing and applications (ACOMP). IEEE; 2015. pp. 67–74.
5. Jabez J, Muthukumar B. Intrusion detection system (IDS): anomaly detection using outlier detection approach. *Procedia Comput Sci*. 2015;48:338–46.
6. Neelakantan S, Rao S. A threat-aware anomaly-based intrusion-detection approach for obtaining network-specific useful alarms. In: International conference on distributed computing and networking. Springer. 2009; pp. 175–180.
7. Kasongo SM, Sun Y. A deep learning method with filter based feature engineering for wireless intrusion detection system. *IEEE Access*. 2019; 7:38597–607.
8. El Naqa I, Murphy MJ. What is machine learning? In: Machine learning in radiation oncology. Berlin: Springer; 2015. p. 3–11.
9. Khatri S, Arora A, Agrawal AP. Supervised machine learning algorithms for credit card fraud detection: a comparison. In: 2020 10th international conference on cloud computing, data science & engineering (confluence), IEEE; 2020. pp. 680–83.
10. Singh P. Supervised machine learning. In: Learn PySpark. Springer; 2019. pp. 117–59.
11. Harrington P. Machine learning in action. New York: Manning Publications Co.; 2012.
12. Dong G, Liu H. Feature engineering for machine learning and data analytics. Boca Raton: CRC Press; 2018.
13. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016; pp. 785–94.
14. Zhu Z, Ong Y-S, Dash M. Wrapper-filter feature selection algorithm using a memetic framework. *IEEE Trans Sys Man Cybern Part B (Cybern)*. 2007;37(1):70–6.
15. Moustafa N, Turnbull B, Choo K-KR. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet Things J*. 2018;6(3):4815–830.
16. Moustafa N, Slay J. 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*. 2016;25(1–3):18–31.
17. Khammassi C, Krichen S. A GA-LR wrapper approach for feature selection in network intrusion detection. *Comput Secur* 2017;70:255–77.
18. Osanaiye O, Cai H, Choo K-KR, Dehghantanha A, Xu Z, Dlodlo M. Ensemble-based multi-filter feature selection method for DDOS detection in cloud computing. *EURASIP J Wirel Commun Netw*. 2016;2016(1):130.
19. Ambusaidi MA, He X, Nanda P, Tan Z. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans Comput*. 2016; 65(10):2986–98.
20. Ingre B, Yadav A. Performance analysis of NSL-KDD dataset using ANN. In: 2015 international conference on signal processing and communication engineering systems, IEEE; 2015. pp. 92–6.

21. Alazzam, H., Sharieh, A., Sabri, K.E.: A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. *Expert Syst Appl.* 2020;148:113249.
22. Deng Y, Duan H. Control parameter design for automatic carrier landing system via pigeon-inspired optimization. *Nonlinear Dyn.* 2016; 85(1):97–106.
23. Janarthanan T, Zargari S. Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In: 2017 IEEE 26th international symposium on industrial electronics (ISIE). IEEE; 2017. pp. 1881–1886.
24. Kumar V, Sinha D, Das AK, Pandey SC, Goswami RT. An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset. *Cluster Comput.* 2020; 23(2):1397–1418.
25. Almomani O. A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms. *Symmetry.* 2020;12(6):1046.
26. Khan NM, Negi A, Thaseen IS, et al. Analysis on improving the performance of machine learning models using feature selection technique. In: International conference on intelligent systems design and applications. Springer; 2018. pp. 69–77.
27. Tama BA, Comuzzi M, Rhee, K-H: TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access* 2019; 7:94497–507.
28. Zong W, Chow Y-W, Susilo W. A two-stage classifier approach for network intrusion detection. In: International conference on information security practice and experience. Springer; 2018. pp. 329–340.
29. Belouch M, El Hadaj S, Idhammad M. A two-stage classifier approach using reptree algorithm for network intrusion detection. *Int J Adv Comput Sci Appl.* 2017;8(6):389–94
30. Gao J, Chai S, Zhang B, Xia Y. Research on network intrusion detection based on incremental extreme learning machine and adaptive principal component analysis. *Energies* 2019;12(7):1223.
31. Almogren AS. Intrusion detection in edge-of-things computing. *J Parallel Distrib Comput.* 2020;137:259–65.
32. Jiang K, Wang W, Wang A, Wu H. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access.* 2020; 8:32464–476.
33. Scikit-Learn, Support Vector Machines. <https://scikit-learn.org/stable/modules/svm.html>. Accessed 25 Sept 2020.
34. Dreiseitl S, Ohno-Machado L. Logistic regression and artificial neural network classification models: a methodology review. *J Biomed Inform.* 2002;35(5–6):352–59.
35. Basheer IA, Hajmeer M. Artificial neural networks: fundamentals, computing, design, and application. *J Microbiol Methods.* 2000;43(1):3–31.
36. Li Y, Yuan Y. Convergence analysis of two-layer neural networks with relu activation. In: Advances in neural information processing systems; 2017. pp. 597–607.
37. Graupe D. Principles of artificial neural networks, vol. 7. Singapore: World Scientific; 2013.
38. Safavian SR, Landgrebe D. A survey of decision tree classifier methodology. *IEEE Trans Syst Man Cybern.* 1991;21(3):660–74.
39. Kuang Q, Zhao L. A practical GPU based KNN algorithm. In: Proceedings. The 2009 international symposium on computer science and computational technology (ISCCSI 2009). Citeseer; 2009. p. 151.
40. Schouten TE, Van den Broek, EL. Fast exact euclidean distance (feed): a new class of adaptable distance transforms. *IEEE Trans Pattern Anal Mach Intell.* 2014;36(11):2159–72.
41. Moustafa N, Slay J. 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; 2015. pp. 1–6.
42. Shabtai A, Elovici Y, Rokach L. A survey of data leakage detection and prevention solutions. Berlin: Springer; 2012.
43. Liu Z, et al. A method of SVM with normalization in intrusion detection. *Procedia Environ Sci.* 2011;11:256–62.
44. Scikit-Learn, Gradient Boosting Classifier. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>. Accessed 26 Sept 2020.
45. Scikit Learn, Machine Learning in Python. <https://scikit-learn.org/stable>. Accessed 26 Sept 2020.
46. UNSW-NB15, Intrusion Detection Dataset. <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>. Accessed 26 Sept 2020.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
