Scrutinizing Attacks and Evaluating Performance Appraisal Parameters via Feature Selection in Intrusion Detection System

Navroop Kaur (Knavroop7488@gmail.com)
Punjabi University
Meenakshi Bansal
Yadavindra College of Engineering
Sukhwinder Singh Sran
Yadavindra College of Engineering

Research Article

Keywords: AB, DoS, DT, IDS, kNN, LR, NB, Probe, R2L, SVM, U2R

Posted Date: November 23rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-748765/v1

Scrutinizing Attacks and Evaluating Performance Appraisal Parameters via Feature Selection in Intrusion Detection System

Navroop Kaur[*#1] Meenakshi Bansal[#2] Sukhwinder Singh Sran[#3]

[#1]Research Scholar (Ph.D.), Punjabi University, Patiala, Punjab, India.

[#2,3]Assistant Professor, CSE, Yadavindra College of Engineering, Talwandi Sabo, Punjab, India.

Abstract

In modern times the firewall and antivirus packages are not good enough to protect the organization from numerous cyber attacks. Computer IDS (Intrusion Detection System) is a crucial aspect that contributes to the success of an organization. IDS is a software application responsible for scanning organization networks for suspicious activities and policy rupturing. IDS ensures the secure and reliable functioning of the network within an organization. IDS underwent huge transformations since its origin to cope up with the advancing computer crimes. The primary motive of IDS has been to augment the competence of detecting the attacks without endangering the performance of the network. The research paper elaborates on different types and different functions performed by the IDS. The NSL KDD dataset has been considered for training and testing. The seven prominent classifiers LR (Logistic Regression), NB (Naïve Bayes), DT (Decision Tree), AB (AdaBoost), RF (Random Forest), kNN (k Nearest Neighbor), and SVM (Support Vector Machine) have been studied along with their pros and cons and the feature selection have been imposed to enhance the reading of performance evaluation parameters (Accuracy, Precision, Recall, and F1Score). The paper elaborates a detailed flowchart and algorithm depicting the procedure to perform feature selection using XGB (Extreme Gradient Booster) for four categories of attacks: DoS (Denial of Service), Probe, R2L (Remote to Local Attack), and U2R (User to Root Attack). The selected features have been ranked as per their occurrence. The implementation have been conducted at five different ratios of 60-40%, 70-30%, 90-10%, 50-50%, and 80-20%. Different classifiers scored best for different performance evaluation parameters at different ratios. NB scored with the best Accuracy and Recall values. DT and RF consistently performed with high accuracy. NB, SVM, and kNN achieved good F1Score.

Keywords – AB, DoS, DT, IDS, kNN, LR, NB, Probe, R2L, SVM, U2R.

I. INTRODUCTION

In the present time, the primary concern for all organizations and institutions is to offer high-level security to their respective networks. The advent of IDS (Intrusion Detection System) is dedicated to the weakness of firewalls and access control in providing an acceptable guard against attack. The concept can be imitated as the car has been stolen despite been locked. Despite putting optimum efforts into securing the information, the intruders get successful in stealing critical information. IDS is a solution to such malicious activities (García-Teodoro *et al.*, 2009). IDS is accomplished by sensing the attacks before, during, and after their occurrence. The IDS monitors and examines the activities of the user and system, investigates configuration and weaknesses of the system, accesses the reliability of system and file, identifies patterns emblematic of attacks, and keeps track of policy violations at the user end. The information is passed via wires between hosts and is denoted as packet sniffers. If captured, the packets are analyzed in several manners mentioned below (Buczak and Guven, 2016) (Agrawal and Agrawal, 2015).

- Signature-based detection - This type of detection is good enough against known attacks. The effectiveness of this detection relies on receiving consistent pattern updates and fails to detect the new or unidentified previous threats.

- Anomaly-based detection – This detection classifies a network based on heuristics or rules rather than signatures or patterns. Before implementing anomaly detection, one should be familiar with the usual behavior of the network (Belavagi and Muniyal, 2016).
- Specification-based detection – Specification-based detection relies on observing the processes and cross-checking the actual data with the program. If any sort of abnormality is detected, an alert is issued indicating the need for removal of such anomaly and apprising the system to normal.

IDS can be classified based on their point of installation as mentioned below(Dhanabal and Shantharajah, 2015) .

- Host-based IDS – This IDS is located on a workstation or server where data is composed and examined locally.
- Network-based IDS – This IDS is placed at an intentional point on network infrastructure. Network-based IDS is also denoted as "sniffer" because its sniffs upon the medium.
- Hybrid-based IDS – Hybrid-based IDS is a combination of Host-based and Network-based IDS and acts as a centralized control unit.
- Signature-based IDS (SIDS) – Signature-based IDS makes use of patterns to detect familiar attacks. It fails to detect formerly unidentified attacks. The competence hinge on the newness and size of the signature file. SIDS effectively recognizes intrusions with minimum false alarms. The number of imprecise results is very rare on none.
- Anomaly-based IDS (AIDS) – Anomaly-based IDS makes use of system features, heuristics, and statistical measures. It is smart enough to detect beforehand unidentified attacks. Competence hinge on how the IDS develops itself as time progresses. It is prone to generating false alarms (high false-positive rate) (Aburomman and Reaz, 2017).

AIDS can be further classified into three categories as mentioned below (Sharafaldin *et al.*, 2017)(Panigrahi and Borah, 2018).

- Techniques based on Statistics – Statistics-based IDS constructs a distribution model for standard behavior profiles and works on identifying events with low probability and declare them as potential intrusions. This technique makes use of different statistical metrics like mean, mode, median, and standard deviation(Hassan *et al.*, 2020) . This method monitors each packet rather than the entire data traffic. The technique is good enough to differentiate the current behavior from the normal behavior. The different statistical-based techniques models are XGBoost, Multivariate, and Time Series models (Khan and Gumaei, 2019)(Wang *et al.*, 2018).
- Techniques based on Knowledge - Knowledge-based techniques are also denoted as expert system methods. This method involves constructing a knowledge base that imitates the genuine traffic profile and activities which differ from this typical outline are stated as an intrusion (Teng *et al.*, 2018). Knowledge-based techniques are shaped by human knowledge. The noteworthy advantage of techniques knowing its base is their competence to curtail false-positive alarms as the system has information about all the normal conducts. But, in a vigorously varying computing scenario, this kind of IDS needs a reliable description of knowledge for the expectable typical conduct which is a time-consuming job as compounding information about all typical events is very stimulating(Liu and Xu, 2019)(Wang *et al.*, 2010) . The diverse knowledge-based techniques models are Finite-state machine, Description languages, and Expert systems.
- Techniques based on Machine learning – Machine learning technique comprises mining of huge data volumes intended for mining knowledge. Machine learning prototypes incorporate a set of directives, events, or multilayered "transfer functions" that can be smeared to determine captivating data outlines or to classify or predict enactment (Alaparthy and Morgera, 2018). Machine learning methods have been used largely in the area of AIDS. Abundant methods and procedures such as genetic algorithms, decision trees, nearest neighbor methods, clustering, association rules, and neural networks have been applied for ascertaining the information from invaded datasets(Alaparthy and Morgera, 2019) .

In the year 1999, it was decided to gather the traffic records. The motive was to come up with a predictive model to perform intrusion detection to distinguish between bad connections and good connections. The bad connections were the intrusions or attacks which hinder the performance of the overall system. The efforts were made and an enormous amount of data related to internet traffic records was gathered into a data set referred to as KDD'99. As few flaws were encountered in the KDD'99 dataset, a revised and clean dataset was extracted from KDD'99 which was given

the name of the NSL KDD dataset. The NSL KDD data set comprises four sub-datasets, popularly known as KDDTest+, KDDTest-21, KDDTrain+, and KDDTrain+_20Percent. Each record is composed of 43 features out of which 41 features denote traffic input itself and the remaining two denote labels and score. The label indicates whether the feature is a normal input or an attack and the score indicates the severity of the traffic input. The attacks are further classified into four categories: DoS (Denial of Service), Probe, U2R (User to Root), and R2L (Remote to Local). DoS attempts to halt the traffic flow to and from the target system. Probe tries to get information from a network. U2R initiates with a normal user account and attempts to upgrade to super-user to expand admittance to the system or network. R2L tries to expand local admittance to a remote machine.

The prominent functions performed by IDS are mentioned below.

- Authentication: Authentication can be demarcated as the individuality which will tell about the authorization of the user whether the present or logged user is legitimate or genuine. Authentication is the individuality that asserts that the user is legitimate and should be allowed to connect with the data sources which are demanded (Amouri, Alaparthy and Morgera, 2018).
- Access control: Access control denotes a safety mechanism that is supposed to defend critical information from unauthorized access.
- Data confidentiality: This service delivers the secrecy of information. When executed appropriately, confidentially permits only authorized/authentic users to have admittance to information and guards the critical data against unsanctioned users.
- Data Integrity: The job of data integrity is to ensure secure data transmission. It keeps an eye on the data been transmitted so that unauthorized users are not able to alter the data. It offers the exactness of the data that was communicated (Amouri, Alaparthy and Morgera, 2018)(Networks, 2003).
- Non-repudiation: The task assigned to non-repudiation is to make sure that the user transmits the message. It operates in two forms. The first is non-repudiation with proof of origin, i.e. the data is communicated with proof of origin so that the user cannot deny the transmission later. The second form is non-repudiation with delivery proof, i.e. the delivery report is issued indicating that the data is sent by the legitimate user so that later users cannot deny the transmission of the data(Thamilarasu and Chawla, 2019) .

Machine learning is a group of mathematical equations that need to be solved fast at high speed. The majority of machine earning algorithms comprise parameter functions, weight values, and cost functions that one can transact based on the data employed. Majority of the machine learning and deep learning is concerned with the classification of images, managing a huge volume of data from sensors, language translation, and prediction of future values based on current values. One is free to opt for the strategy as per the problem domain.

The two approaches of machine learning are supervised learning and unsupervised learning.

- Supervised learning – In supervised learning the machine learning model is trained using categorized data. The data have accurate classification linked with them. Supervised learning assists in predicting the values for new data. The models are rebuilt according to new data and the job is to make sure that predictions are still accurate. For instance, supervised learning would be provided to categorize the images of mammograms as benign or malignant. One could have a dataset committed to mammograms to explain to the model what is a benign mammogram and what is a malignant mammogram.
- Unsupervised learning – In unsupervised learning, the model is trained using unlabeled data. The model is supposed to discover its features and predict depending upon the classification of the data. For instance, the dataset would have typewritten images in English, Hindi, and Punjabi and one might use diverse algorithms to acquire the model to recognize the images of English typewritten text deprived of any labels.

Feature selection has vital importance in constructing machine learning models. Inappropriate features in data hinder the overall performance of the model and lead to prolonged training time in building the model. A machine learning algorithm makes use of a training dataset to control weight features that can be smeared to unobserved data for making

predictions (Riahi Sfar *et al.*, 2018). The selection of relevant features is necessary for the implementation of the machine learning algorithm. Feature selection plays an important role in Intrusion Detection System. The reasons for adopting feature selection are mentioned below (Surendar and Umamakeswari, 2016).

- Training the machine at high speed.
- To enhance the accuracy of the trained model. Training a lesser number of relevant features is much better than training inappropriate features.
- To evade the curse of dimensionality i.e. to avoid the situations when existing machine learning algorithms do not perform or scale well on the increased number of features.
- To avoid or minimize overfitting.
- To enhance the efficiency and accuracy of the classifier.

The research focuses on analyzing different classification algorithms with their positive and negative aspects concerned with intrusion detection. The implementation has been conducted at five different ratios of 60-40%, 70-30%, 90-10%, 50-50%, and 80-20% for computing the readings of different performance evaluation parameters comprising accuracy, precision, recall, and F1Score. The working of XGB (Extreme Gradient Booster) has been elaborated using a flowchart and algorithm depicting the procedure to perform feature selection in NSL KDD datasets considering the four categories of attacks: DoS (Denial of Service), Probe, R2L (Remote to Local Attack), and U2R (User to Root Attack). The nominated features have been graded as per their occurrence. The proposed model for the implementation of the feature selection in IDS has been elaborated via flowchart and algorithm followed by the results achieved. The *KDDTrain+_2.csv* and *KDDTest+_2.csv* have been used for training and testing. Finally, the obtained results for different classifiers at different ratios have been compared.

II. STATE OF ART

(Naseer *et al.*, 2018) investigated the usability of deep learning techniques in IDS (intrusion detection system). The authors came up with anomaly recognition models constructed on diverse DNNs (Deep Neural Networks), comprising CNNs (Convolution Neural Networks), autoencoders, and RNN (recurrent neural networks). The models under discussion were trained on the NSL-KDD training data set and appraised on both test data sets, NSL KDD Test+ and NSLKDDTest21. Researchers evaluated both conventional and deep learning models using classification metrics comprising Precision recall curve, ROC, Area Under Curve, and accuracy. The authors implemented a few of the ELMs (Extreme Learning Machines) techniques like NB (Naïve Bayes), RF (Random Forest), DT (Decision-Tree), kNN (k Nearest Neighbors), and SVM (Support Vector Machine). They presented a comparison in which DCNN and LSTM models performed well with an accuracy of 85% and 89% respectively. The authors stated that upcoming research will be focused on inspecting deep learning as a feature extraction tool to study competent data representations for abnormality detection glitches. (Roshan *et al.*, 2018) proposed an adaptive design of IDS based on extreme learning machines that offer the competence of spotting recognized and novel attacks and being restructured according to new inclinations of data patterns economically delivered by security experts. The proposed IDS consists of 3 main components, clustering manager, and decision-maker and update manager. The goal of these components is to provide an IDS system that allows for an expert to be "in the loop" and review, apprise and correct the model. The authors evaluated by using the NSL KDD data set in supervised and unsupervised mode and the anticipated method identify 89% of the samples of the new attack type. Detection rates provided by the updated system were 70% and 77% in supervised and unsupervised modes. Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C. Suh, Ikkyun Kim, Kuinam J. Kim, 2019 (Kwon *et al.*, 2019); proposed an effective deep learning model by using a Fully Connected Network (FCN) for analyzing network traffic and for improving accuracy. They consider the NSL KDD dataset for evaluating different parameters like precision, accuracy, recall and F1score. A set of trials with four amalgamations of the NSL-KDD datasets (i.e. Train+, Train20, Test+ and Test-)were piloted. Accuracy results of the research are above 90% with FCN that is much improved in comparison to 50.3-82.5 by using SVM, Random forest, and AdaBoost. The authors also presented an overview of different deep learning techniques and also investigated past work in NIDS using deep learning methods. Besides FCN they endorse exhausting a generative adversarial network (GAN) to advance the overall performance for NIDS. (X. Zhang *et al.*, 2019) proposed an intrusion detection framework consisting of three parts data fragmentation, new data formation, and ensemble-based intrusion detection model for the binary class. In this framework data split is done in two ways one by using random split with the help of kernel density estimation and the other one is by using FCM (Fuzzy C-Means) clustering technique, ratio transformation is done on original data to collect highly competent data and in the ensemble-based model, three layers

of SVM classifiers are used, so it is called decision tree ensemble SVM. Experiments are done on the NSL KDD data set to evaluate the accuracy, detection rate, false alarm rate. 10 fold cross-validation method was used for training and testing of the proposed model. The authors compared various parameters for DT-ENSVM (for both random split and FCM), ENSVM (for both random split and FCM), and SVM. Accuracy of the Proposed model was 99.41% with FCM and 99.16 with the random split while ENSVM had 97.88% with FCM and 97.13 with random split and SVM had 97.39%, false alarm rate OFDT-ENSVM was below 0.4% but for ENSVM and SVM was more than 0.8 %. The authors concluded that the FCM scheme for splitting was better than the random split method. For the future, the authors suggested the research be generalized for different attack types. (Gao *et al.*, 2019) elaborated different deep learning models for intrusion detection are proposed using fully connected networks, Variational AE, and Seq2Seq structures. They used a set of public data set NSLKDD, Kyoto-honeypot, UNSW-NB15, IDS2017, and MAWILAB for the experiment and to evaluate performance parameters like accuracy, precision, recall, and F1score. The Seq2Seq model with LSTM performs best among other models with an accuracy of more than 99% over the various datasets with different characteristics. For future studies, researchers suggest examining the feasibility of CNN for network anomaly detection.(Kasongo and Sun, 2020) proposed an adaptive ensemble learning model by using a decision tree, SVM, logical regression, KNN, AdaBoost, random forest and deep neural network as an alternative classifier. By modifying the percentage of training data, the authors built multi-tree and adaptive voting algorithm. They used the NSL-KDD dataset for the experiment to evaluate detection rate, accuracy, precision, recall, F1 factor and training time. The accuracy of the proposed adaptive voting algorithm was 85.2% and precision, Recall and F1 were 86.5%, 85.2%, 84.9% respectively. DNN has a good detection rate as compared to others but took a long training time. (Gu *et al.*, 2019) proposed multiple machine learning techniques including ANN, KNN, DT, LR, and SVM in conjunction with the XGBoost algorithm for feature selection to implement an accurate intrusion detection system. All the Binary and multiclass experiments are done using the UNSW-NB15 dataset. The authors worked on both the full feature space of the dataset and reduced feature space with 19 features of the same dataset. The results demonstrated that the accuracy increased while experimenting on reduced feature space. For future reference, the authors proposed to implement the same XGBoost feature selection technique using the NSL KDD data set. G. Andresini, A. Appice, N. D. Mauro, C. Loglisci, and D. Malerba, 2020(Andresini *et al.*, 2020) ; proposed the multi-channel deep feat Ure learning (MINDFUL) for intrusion detection which combines an unsupervised approach for multichannel feature construction with two autoencoders and a supervised on exploding cross channel feature co-relations. With the help of two autoencoders each sample was replaced by multichannel sample then multichannel sample was fed into one dimensional CNN as input with one filter and again the output of 1D CNN was given to two stacked fully connected layers as input. Three benchmarks datasets 10% of KDD99 train and KDD99 test, UNSWNB15 train, and test and CICIDS2017 train and test were used for evaluating the accuracy of MINDFUL, NN, ANN, CNN, ACNN. The accuracy achieved by the experiment for KDD99 was 92.49, UNSWNB15 was 93.40and CICDS2017 was 97.90 but there was a great absence of minor attacks and the model was unable to detect the structure characteristics and architecture of attacks.(Karatas, Demir and Sahingoz, 2020) implemented six machines learning-based intrusion detection system by using K-nearest neighbor, random forest, gradient boosting, AdaBoost, Decision tree and, linear discriminate analysis algorithm using a CSECIC IDS2018 dataset. The author's foremost emphasis was on eliminating the influence of asymmetry between classes in the dataset for refining the accuracy of the system. The authors used the synthetic minority oversampling technique (SMOTE) for decreasing the imbalance ratio. The authors used the k-fold (k=5) cross-validation method for the experiment to evaluate the performance metric accuracy, precision, recall, F1 score, and error rate values. The results showed that the accuracy increased in sampled data by using random forest, KNN, GB, LDA. Y. (Yu and Bian, 2020) ; purposed multi-stage deep feature learning for intrusion detection of few short learning patterns for DNN, CNN as embed function for feature extraction and dimensionality reduction with random sampling technique for balancing dataset for both binary and multiclass classification, using 1% NSLKDD train + for training and UNSWNB15 dataset, attained high accuracy of 85.75% for KDD test 21 and 92.34% for KDD test+. The detection rates for U2R and R2 L were increased from 13 to 81.50 and 44.41 to 75.93%.

III. MACHINE LEARNING CLASSIFIERS

This section explains the different machine learning classifiers used in Intrusion Detection Systems.

A. LR (Logistic Regression)

Logistic Regression is a classification algorithm for finding the chances of success or failure of the event. Logistic Regression is used for binary variables. The data is categorized into discrete classes by learning the association from a given set of labeled data. It studies a linear relationship from the given dataset and then familiarizes a non-linearity in the form of the Sigmoid function.

Advantages

- Logistic Regression is easy to use, understand, and very effective to train.
- Logistic Regression can effortlessly range to multiple classes and a normal probabilistic observation of class predictions.
- Logistic Regression is very fast at categorizing unidentified records.
- Logistic Regression offers good accuracy for simple data sets and does good in the case of the linearly separable dataset.
- Logistic Regression is less prone to over-fitting.

Disadvantages

- The main restraint of Logistic Regression is the assumption of linearity amid the independent and dependent variables.
- Logistic Regression can merely forecast disconnected functions.
- Logistic Regression must not be favored when the observations are lesser than the available features. This may result in overfitting.
- Logistic Regression often results in creating linear boundaries.

The equations for LR are mentioned in equation 1 below where P refers to the probability of the event whose value lies between 0 and 1.

$$Z_i = \ln (P_i / (1 - P_i)) = \alpha + \beta_1 x_1 + \ldots + \beta_n x_n \quad \text{---------- (1)}$$

Taking exponents on both sides of the equation provides equation 2.

$$P_i = E (y = 1|x_i) = e^z / (1 + e^z) = e^{\alpha+\beta ixi} / (1 + e^{\alpha+\beta ixi}) \quad \text{--------- (2)}$$

B. NB(Naive Bayes)

Naïve Bayes is a comprehensively basic Bayesian probability model. It functions on a robust independence supposition. For n attributes, the Naïve Bayes creates 2n! independent assumptions. The Naïve Bayes provides accurate results.

Given a feature vector *X=(x1, x2,…, xn)* and a class variable *Ck*, Bayes Theorem states that:

$$P(Ck|X)=P(X|Ck)P(Ck)/(P(X)), \text{ for } k=1,2,\ldots,K \quad \text{---------- (3)}$$

Where

P(Ck/X) - refers to the posterior probability

P(X/Ck) – denotes the likelihood,

P(Ck) - refers *to* the prior probability of a class

P(X) – indicates the prior probability of the predictor.

The motive is to calculate the posterior probability from the likelihood and prior probabilities.

The process of calculating the set of probabilities for all values repeatedly is a tedious one. With the Naïve conditional independence assumption, the probability of one attribute has zero effect on the probability of the other attribute.

The conditional independence is given as

$$P(y|x_1,\ldots,x_n) = (P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)) / (P(x_1)P(x_2)\ldots P(x_n)) \quad \text{---------- (4)}$$

And the posterior probability is equated as

$$P(y|x_1,\ldots,x_n) \alpha P(y) \pi_{i=1}^{n} P(x_i|y) \quad \text{---------- (5)}$$

subsequently, the denominator rests constantly for all values.

The Naïve Bayes classifiers are categorized into three categories: Gaussian, Multinomial, and Bernoulli. Gaussian is intended for classification and admits features following a normal distribution. Multinomial is used for isolated counts. Bernoulli is used for binary feature vectors.

Advantages of Naïve Bayes

- In case, when the supposition of independent predictors stands true, the performance of the Naïve Bayes classifier is ranked above the other classifiers. It converges faster when compared with other models like Logistic Regression.
- It is sufficiently smart and a small training data is enough to estimate the test data.
- It is simple and easy to implement. Due to its simplicity, it performs at high speed and makes probabilistic predictions.

- Naïve Bayes linearly measures with the multiple data points and predictor features.
- Naïve Bayes can handle multi-class and binary classification problems and can handle discrete and continuous data.

Disadvantages of Naïve Bayes

- Naïve Bayes indirectly accepts that entire attributes are conjointly independent, but in real life, this is not true.
- If a categorical variable is not detected in the training dataset but has a category in the test dataset, the zero probability will be allotted by the model and will not be capable to make a prediction.

C. DT (Decision Trees)

A Decision Tree acts as a support tool having a tree-like structure for modeling possible outcomes and consequences. It offers a way of presenting algorithms with conditional statements. The branches of the tree represent the decision-making steps that may lead to a favorable result.

DT is smart enough to handle both classification and regression complications. The flowchart comprises internal nodes representing attributes at different stages. Every branch refers to an outcome for the attributes. The route from the leaf to the root indicates the rules for classification.

Advantages

- The Decision Tree is simple and easy to understand. It can be easily interpreted and visualized. It is a kind of if-else statement.
- Decision Tree is capable of handling both categorical and continuous variables.
- A Decision Tree is a rule-based approach and does not require any standardization and normalizing.
- Decision Tree outperforms other curve-based algorithms when handling non-linear parameters.
- Decision Trees can effectively handle the missing values.
- The Decision Tree is robust to outliers.
- Training time is less in the case of the Decision tree as there is a single tree.

Disadvantages

- Decision Tree suffers from overfitting of data resulting in wrong predictions. To cope with the data, it continuously generates new nodes which leads to the tree becoming too complex. This results in a loss of generalization capabilities.
- Decision Trees do not perform well on unseen data.
- The overfitting leads to chances of high variance and results in higher inaccuracy.
- Whenever a new data point is added, it causes to regenerate the tree and all the nodes need to be recalculated and recreated.
- The decision Tree becomes unstable with a small amount of noise and makes wrong predictions.
- It is not suitable for large data as large data may grow one single tree too complex.

D. AB (AdaBoost)

AdaBoost stands for Adaptive Boosting. An AdaBoost algorithm is dedicated to boosting the performance of machine learning algorithms. The boosting is a procedure of converting the weak classifiers into strong classifiers. A weak classifier uses a simple threshold on a single feature. If the feature is above the threshold then predicted, it belongs to positive otherwise belongs to negative.

Firstly, there is a need to assign a sample weight for each sample which can be accomplished by using the formula depicted in equation 6 mentioned below.

$$sample\ weight = 1\ /\ number\ of\ samples \quad \text{---------- (6)}$$

Then the Gini impurity for each variable is to be calculated using the formula mentioned in equation 7.

$$Gini\ impurity = 1 - (the\ probability\ of\ True)^2 - (the\ probability\ of\ False)^2 \quad \text{---------- (7)}$$

The Gini Impurity is a method that finds its application in decision tree algorithms in deciding the best split from a root node and succeeding splits. After calculating the Gini Impurity of each node, one can get the total Gini Impurity for each variable which is the weighted average of the impurities of each node.

Finally, one needs to calculate the amount of say which can be done using the formula as stated in equation 8.

$$Amount\ of\ say = \tfrac{1}{2} \log\left((1 - total\ error)\ /\ total\ error \right) \quad \text{---------- (8)}$$

Where Total Error is equal to the sum of the weights of the incorrectly classified samples.

Advantages

- An AdaBoost algorithm is simple, fast, and easy to program.
- It is flexible enough to be combined with any machine learning algorithm.
- It can be used for binary, text, and numeric data.

Disadvantages

- AdaBoost is vulnerable to uniform noise.
- Weak classifiers lead to overfitting and low margins.
- Boosting technique learns gradually, it is significant to guarantee that one has quality data.

E. RF (Random Forest)

Random Forest is used as both classifier and regressor. As the name suggests, RF contains a lot of trees. RF comprises a package of DTs when it comes to classification and is reflected as a saving method as far as overfitting of DT is concerned. DT has high variance and low bias leading to undesired outputs. RF can focus on both, the interpretations and variables of training data for evolving distinct decision trees and yield concentrated voting for classification and the total average for regression problems respectively(Farnaaz and Jabbar, 2016) . RF makes use of the bagging technique that randomly considers observations and chooses the columns incompetent of signifying noteworthy variables at the root for all the DTs. When RF is used for solving regression problems, it makes use of MSE (Mean Squared Error). The formula for calculating MSE is depicted in equation 9.

$$MSE = 1/N \sum_{i=1}^{N} (f_i - y_i)^2 \quad \text{---------- (9)}$$

Where N refers to the number of data points. f_i is the value returned by the model and y_i is the actual value for data point i.

In the case of classification data, one uses the Gini index. The formula for calculating the Gini index is depicted in equation 10.

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2 \quad \text{---------- (10)}$$

Where p_i refers to the relative frequency of the class in the dataset and c refers to the number of classes.

Advantages of RF

- RF enhances the accuracy by minimizing the overfitting in DTs.
- RF can handle problems related to both classification and regression.
- RF performs well with both continuous and categorical values.
- RF is smart enough to automate the omitted value in the data.
- RF makes use of a rule-based approach and hence normalization of data is not needed.

Disadvantages of RF

- RF needs high computational power and resources to build several trees for combining the outputs.
- RF requires an adequate training time as a combination of several DTs takes place.
- RF undergoes interpretability and fails to decide the importance of each variable.

F. KNN (K-Nearest Neighbors)

K-nearest Neighbors falls under the category of supervised ML (machine learning) methods used for the classification and regression of predictive problems. But it finds the majority of its application in classification predictive problems. KNN does not have a dedicated training phase and makes use of all the available data for training purposes during classification(Y. Zhang *et al.*, 2019). KNN does not assume anything relevant to the available data and depicts a non-parametric feature. KNN operates on the concept of feature similarity to assign a value to new data points based on their resemblance with the data points existing in the training set. The working of KNN is mentioned below.

- Firstly, the training and testing data is loaded.
- Select the value of the nearest data points referred to as K which can be some integer.

- Compute the distance amid test data and each row of training data using apposite distance calculation methods like Euclidean, Manhattan, or Minkowski shown in Equations 11, 12, and 13 respectively.

$$Euclidean = Sqrt\ (\sum_{i=1}^{k}(x_i - y_i)^2 \quad \text{---------- (11)}$$

$$Manhattan = \sum_{i=1}^{k}|x_i - y_i| \quad \text{---------- (12)}$$

$$Minkowski = \ (\sum_{i=1}^{k}(|x_i - y_i|)^q)^{1/q} \quad \text{---------- (13)}$$

  Where x and y are the variables under consideration.
- Perform sorting of distance value in increasing order.
- Select the upper K rows from the arranged array.
- Allocate a class to the test point centered on the most repeatedly occurring class of these rows.

Advantages of KNN

- KNN is easy to comprehend.
- There are no conventions related to data.
- KNN is valuable for both classification and regression
- KNN works effortlessly in case of multiclass problems.

Disadvantages of KNN

- KNN is highly memory exhaustive and is parallel computationally expensive.
- The performance of KNN is sensitive to the amount of data.
- KNN may not perform well in the rare event target variable.
- KNN scuffles with a higher number of independent variables.

G. Support Vector Machine (SVM)

Support Vector Machines are a group of supervised learning procedures intended for performing regression, classification, and outliers' detection. SVMs can be used as per the problems encountered. The uniqueness of SVM lies in the manner it selects the decision boundary that exploits the distance from the neighboring data points of the classes under study. The created decision boundary is referred to as the maximum margin hyperplane or maximum margin classifier. SVM classifier operates via constructing a straight line amid two classes. The data points on one side of the constructed line refer to a particular category and the data points on the other side of the line refers to a different category. The choice is to be made from an infinite number of lines. The uniqueness of SVM is that it chooses the best line for classifying the data points. The line responsible for dividing the data and is farthest from the closest data points is selected. For instance, consider having some data points on a grid. The purpose is to separate these data points as per the category in which they fit maintaining that no data point falls in the incorrect category. So one is trying to find the line amid two closest points keeping the other data points detached. Such a line is referred to as a decision boundary. SVMs are used for face detection, biometrics, categorization of text and hypertext, handwriting recognition, etc.

In SVM, the data points are plotted as points in an n-dimensional space where the value of every feature is considered to be the value of a specific coordinate. For performing the classification into individual categories there is a need to find the optimal hyperplane that optimally distinguishes the two classes. The generalization of a plane is referred to as a hyperplane which can be a line in two dimensions, a plane in three dimensions, and a hyperplane in more dimensions. Consider a two-dimensional space. The function of the line is written as $y = ax + by$ where x and y are the features which are named as x1, x2,…xn, the equation ca be rewritten as $ax1 – x2 + b = 0$.On defining x = (x1,x2) and w = (a, -1), one gets the equation w.x + b = 0. Support vectors are the data points that are nearer to the hyperplane and have an influential impact on the position and orientation of the hyperplane. Consider a hypothesis function h defined as depicted below in equations 14a and 14b.

$$h\ (xi) = +1 \text{ if } w.x+b>=0 \quad \text{---------- (14a)}$$

$$h\ (xi) = -1 \text{ if } w.x + b<0 \quad \text{---------- (14b)}$$

SVM classifier minimizes the expression to the equation 15.

$$[1/n\sum_{i=1}^{n} \max\ (0,1 - y_i\ (w.x_i - b))\ ] + \lambda||w||^2 \quad \text{---------- (15)}$$

Advantages of the SVM classifier

- Assured optimality – Due to convex optimization, the result will be a global minimum and not a local minimum.
- Ease of implementation – SVM can be accessed using Matlab or Python.
- Effective use – SVM proves effective in managing linearly separable and non-linearly separable data. Linearly separable data is a hard boundary and non-linearly separable data is a soft boundary.
- Wide compliance – SVM can be utilized with labeled as well as unlabeled data in semi-supervised learning models using minimization problem referred to as Transductive SVM.
- Feature Mapping – SVM performs feature mapping by means of a simple dot product to moderate the load on the computational complication of the training performance of the model.

Disadvantages of SVM

- Fails to handle text structures – SVM is not competent for managing text structures. The loss of sequential information makes the performance worse.
- Difficult to choose the kernel – The presence of numerous kernels makes it difficult to make the right choice and is a major limitation of SVM.

IV. BOOSTING FOR FEATURE SELECTION

Feature selection denotes the procedure of selecting features that are of the utmost value for prediction. Though it sounds modest, it is one of the most multifaceted difficulties for generating a novel machine learning model. The significant challenge in machine learning is to select the appropriate set of features as inputs to a model(Hussain and Lalmuanawma, 2016)(Vasilomanolakis *et al.*, 2015) . The features selected to train a model have a massive impact on the accomplished performance(Khraisat *et al.*, 2019) . Inappropriate features can negatively influence the performance of the constructed model and may convert an aching point for the data scientist. The algorithms intended for feature selection assist in overwhelming this glitch via recognizing appropriate features from the originally available features without dropping a significant proportion of information (da Costa *et al.*, 2019)(Liu and Lang, 2019).

Boosting is one of the most influential learning concepts presented in the last twenty years. It was initially premeditated for classification glitches but has been profitably extended to regression as well. The inspiration for boosting was a technique that pools the outputs of several weak classifiers to harvest an authoritative committee (Ahmim, Derdour and Ferrag, 2018)(Dhaliwal, Nahid and Abbas, 2018). Gradient Boosting fits into the category of machine learning called Ensemble Learning. Ensemble Learning is a subdivision of machine learning approaches that is capable of training and predicting several models at once intended for fabricating a single superior output(Karatas, Demir and Sahingoz, 2020)(Yan and Han, 2018). For instance, imagine planning out different routes to a new destination. As one analyzes the probable routes to the destination, one comes to know which traffics lights are for more periods and how the time influences one route over the other, enabling one to craft the perfect route. This practice of investigating with and joining different models to reach the best conclusion is similar to Ensemble Learning.

Ensemble learning is classified into three primary categories mentioned as under.

- Bootstrap Aggregation (Bagging) – Bagging comprises two dissimilar features of training and predicting. To conduct training, Bagging influences a Bootstrap technique to isolate the training data into diverse random subsamples, which diverse iterations of the model used to train on (Yao *et al.*, 2019). To conduct prediction, the bagging classifier uses the prediction having the most votes from participating models to harvest its output. The bagging regression is responsible for calculating the average of all the models to generate an output.
- Stacking – Stacking controls the outputs received from multiple models as input features that are considerably different. For instance, stacking trains KNN, LR, and DT with the training data and then gather the received outputs and merge them with LR(Gao *et al.*, 2019). The purpose is to minimize the overfitting and enhancing the accuracy.
- Boosting – Boosting is a technique responsible for transforming weak learners into strong learners. Boosting is characteristically applied to trees(Yin *et al.*, 2017) . Boosting augments iterations of the model sequentially via regulating the weights of the weak learners along the way. This decreases prejudice from the model and characteristically increases accuracy (Sheikhan, Jadidi and Farrokhi, 2012)(Naseer *et al.*, 2018). Prevalent boosting algorithms are Extreme Gradient Boosting (XGBoost) and AdaBoost.

XGBoost is a collection of gradient boosting algorithms boosted for tackling modern data science problems and tools. It influences the techniques stated with boosting and comes enfolded in an easy-to-use library (Derhab, 2019). XGBoost wires tree-pruning, parallel processing, regularization to avoid overfitting, and handling missing values. It inevitably computes feature importance for all the features and makes the final feature scores available(Wei *et al.*, 2019) . A few of the foremost benefits of the XGBoost technique are mentioned below.

- It is extremely accessible, swift to perform, and stereotypically outdoes other algorithms.
- It can execute on single and distributed systems.
- It can be used in supervised learning comprising regression and classification problems.
- It supports parallel processing and permits cache optimization.
- It performs effective tree-pruning, built-in regularization, and parallelization.
- It offers well-organized memory management for large datasets.
- It offers a range of regularizations assisting in reducing overfitting.
- It can effectively handle missing values.
- It handles outliers to a certain extent.

Few drawbacks of XGBoost are

- It is disposed to over-fitting on small datasets.
- It does not deal with unconditional features directly.

The flowchart for XGBoost for performing feature selection is shown in Fig. 1.

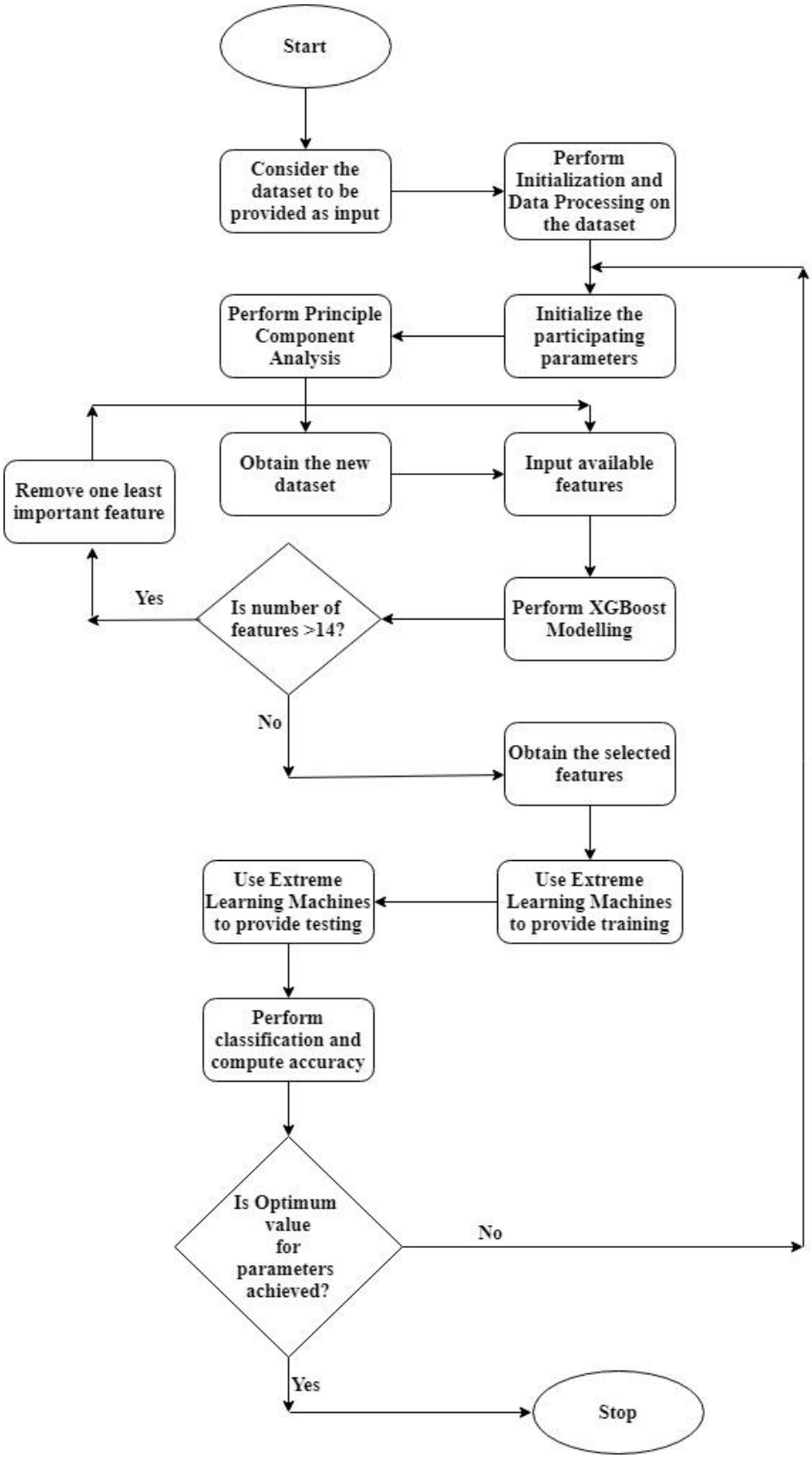

Fig. 1 Flowchart illustrating adopted procedure to perform feature selection via XGBoost to be used in research methodology

Algorithm

1. Contemplate the dataset to be provided as input.
2. Initiate processing on the considered dataset.
3. Provide values of the participating parameters.
4. Perform Principle Component Analysis for reducing the dimensionality and increasing interpretability with minimum information loss for obtaining the processed dataset.
5. Consider the available features.
6. Execute XGBoost modeling for detecting and dealing with the missing values and assist in the classification and ranking of the participating features.
7. If the number of features is more than 14
   Eradicate one least important feature and Return to 5
   Else
   Obtain the selected features and Goto 8
8. Use Extreme Learning Machines to provide training and testing
9. Perform classification to optimize accuracy.
10. If the optimum value of participating parameters is achieved
    Goto 11
    Else
    Goto 3
11. End

XGboost has established itself to be the utmost well-organized Scalable Tree Boosting Technique. It has revealed exceptional outcomes through dissimilar use cases such as stock sales predictions, motion detection, customer behavior analysis, malware classification, etc. The system executes much faster on a single machine than any other machine learning method with competent data and memory handling (Xiao *et al.*, 2019). The optimization techniques of the algorithm offer higher performance and deliver high speed using a minimum amount of resources.

V. CONTRIBUTION AND IMPLEMENTATION

This section elaborates the flowchart and algorithm for the procedure adopted to implement the feature selection in IDS followed by the results achieved. The *KDDTrain+_2.csv* and *KDDTest+_2.csv* have been used for training and testing. Different performance evaluation parameters used are briefly defined below.

- Accuracy - The accuracy is defined as the percentage of correctly classified instances

  $$Acc = (TP + TN) / (TP + TN + FP + FN)$$

  where *TP, TN, FN, and FP* represent the number of true positives, true negatives, false negatives, and false positives respectively.
  For good classifiers, *TPR* and *TNR* both should be nearer to 100%
- Precision (*PR*): It is the fraction of data instances predicted as positive that is positive.

  $$PR = TP / (TP + FP)$$
- Recall - Recall is calculated as the number of TPs divided by the total number of TPs and FNs. The outcome is an assessment amid 0.0 for no recall and 1.0 for full or perfect recall.

  $$Recall = TP / (TP + FN)$$
- F1Score – F1Score is the harmonic mean of the precision and recall at that threshold.

  $$F1Score = 2 / ((1/PR) + (1/Recall))$$

The flowchart for performing feature selection in IDS is illustrated in Fig. 2.

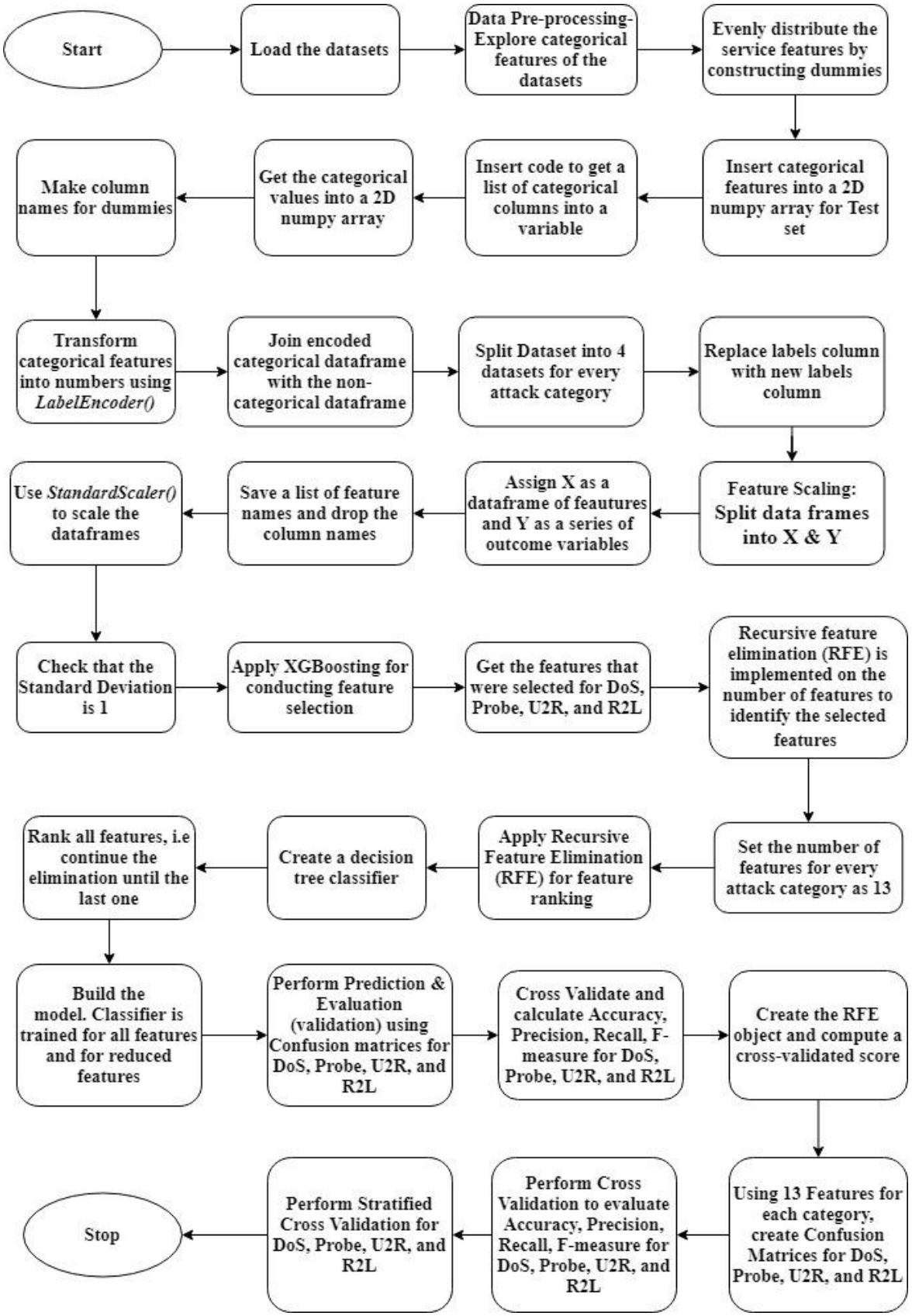

Fig. 2 Flowchart depicting adopted research methodology for IDS

Algorithm

- Select the appropriate datasets as input and load them.
- Perform data preprocessing. Explore the categorical features of the datasets which can have a limited number of possible values. *(Categorical data generally takes a limited number of possible values).*
- Distribute the service features evenly by constructing dummies. *(a dummy variable is a placeholder for a variable that will be combined over, summed over, or marginalized).*
- Insert categorical features into a 2D NumPy array for the Test set. *(NumPy is a python library accumulating facility for multidimensional arrays and matrices besides high-level mathematical functions to control these arrays).*
- Get the code inserted to get a list of categorical columns into a variable.
- Obtain the categorical values into a 2D NumPy array.
- Assign column names for dummies.
- Renovate categorical features into numbers using *LabelEncoder(). (LabelEncoder () are used to convert categorical data, or text data, into numbers, which are easily understood by the predictive models).*
- Connect encoded categorical data frame with the non-categorical data frame.
- Split Dataset into 4 datasets for every category of attacks.
- Replace labels column with new labels column.
- Perform Feature Scaling and split data frames into X and Y. Assign X as a data frame of features and Y as a series of outcome variables. *(Feature scaling or data normalization is responsible for normalizing the range of independent variables or features of data).*
- Save a list of feature names and drop the column names. Use *StandardScaler()* to scale the data frames. *(StandardScaler is intended to perform the conversion on the data such that the distribution will have a 0 as mean value and 1 as standard deviation).*
- Check that the Standard Deviation is 1.
- Apply XGBoost for conducting feature selection. Get the features that were selected for DoS, Probe, U2R, and R2L.
- After procuring the satisfactory number of features from the XGBoost, RFE (Recursive Feature Elimination) is activated with the number of features approved as a parameter to identify the features selected. (The job of *RFE is eliminating the weakest feature (or features) till the quantified number of features is touched*).
- Set the number of features for every category of attacks as 13. Apply Recursive Feature Elimination (RFE) for feature ranking. *(Feature ranking selects the k highest-ranked features from the dataset)*.
- Initiate with the formation of a decision tree classifier. Assign rank to all the features and keep on eliminating the one with the least rank.
- Construct the model. A classifier is trained for all features and reduced features. Accomplish Prediction & Evaluation (validation) using Confusion matrices for all four attack types. *(A Confusion matrix is an N x N (N is the number of target classes) matrix required for evaluating the performance of a classification model).*
- Cross Validate and calculate Accuracy, Precision, Recall, F-measure for DoS, Probe, U2R, and R2L.
- Create the RFE object and compute a cross-validated score. *(Cross-validation is a technique for evaluating ML models. The number of ML models is trained on subsets of the prevailing input data and appraising them on the corresponding subset of the data).*
- Using 13 Features for each category, create Confusion Matrices for DoS, Probe, U2R, and R2L.
- Perform Cross-Validation to evaluate Accuracy, Precision, Recall, F-measure for DoS, Probe, U2R, and R2L. Perform Stratified Cross-Validation for DoS, Probe, U2R, and R2L. (*Stratification is the procedure of reorganizing the data to guarantee each fold is a decent description of the whole*).
- End.

Results obtained

The results obtained after the successful implementation of the procedure adopted in the above-discussed flowchart in Fig. 2 and the algorithm are mentioned below.

Label Distribution for Training Set

Table 1 demonstrates the name of the attacks and the number in which they exist in the NSL KDD dataset. The normal (non – suspicious) are 67343 which is the maximum number and Spy is 2 in minimum number. Table 1 demonstrates the label distribution for the training set.

Table 1. Label distribution for Training Set

Attacks	**In Number**
Normal	67343
Neptune	41214
Satan	3633
Ipsweep	3599
Portsweep	2931
Smurf	2646
Nmap	1493
Back	956
Teardrop	892
Warezclient	890
Pod	201
Guess_passwd	53
Buffer_overflow	30
Warezmaster	20
Land	18
Imap	11
Rootkit	10
Loadmodule	9
ftp_write	8
Multihop	7
Phf	4
Perl	3
Spy	2

Table 2 demonstrates the name of the attacks and the number in which they exist in the NSL KDD dataset. The normal (non – suspicious) are 9711 which is the maximum in number and the minimum is Imap which is 1. Table 2 demonstrates the label distribution for the Test Set.

Table 2. Label distribution for Test Set

Attacks	**In Number**
Normal	9711
Neptune	4657
Guess_passwd	1231
Mscan	996
Warezmaster	944
Apache2	737
Satan	735
Processtable	685
Smurf	665
Back	359
Snmpguess	331
Saint	319
Mailbomb	293
Snmpguess	331

Saint	319
Mailbomb	293
Snmpgetattack	178
Portsweep	157
Ipsweep	141
Httptunnel	133
Nmap	73
Pod	41
Buffer_overflow	20
Multihop	18
Named	17
Ps	15
Sendmail	14
Xterm	13
Rootkit	13
Teardrop	12
Xlock	9
Land	7
Xsnoop	4
ftp_write	3
Udpstorm	2
Sqlattack	2
Phf	2
Worm	2
Loadmodule	2
Perl	2
Imap	1

Table 3 depicts the dimensions of different attacks in accordance with Training Set.

Table 3. Table depicts the dimensions of different attacks as per Training Set

Attack	**Dimensions**
DoS	113270, 123
Probe	78999, 123
R2L	68338, 123
U2R	67395, 123

Table 4 depicts the dimensions of different attacks with the Test Set.

Table 4. Table depicts the dimensions of different attacks as per Testing Set

Attack	**Dimensions**
DoS	17171, 123
Probe	12132, 123
R2L	12596, 123
U2R	9778, 123

Table 5 lists the different selected features using XGBoost Feature Selection in the case of DoS.

Table 5. Table shows the selected features for DoS using XGBoost Feature Selection

Features selected
Logged_in
Count
Serror_rate
Srv_serror_rate
Same_srv_rate
Dst_host_count
Dst_host_srv_count
Dst_host_same_srv_rate
Dst_host_serror_rate
Dst_host_srv_serror_rate
Service_http
Flag_S0
Flag_SF

Table 6 lists the different selected features using XGBoost Feature Selection in the case of Probe.

Table 6. Table shows the selected features for Probe using XGBoost Feature Selection

Features selected
Logged_in
Rerror_rate
Srv_rerror_rate
Dst_host_srv_count
Dst_host_diff_srv_rate
Dst_host_same_src_port_rate
Dst_host_srv_diff_host_rate
Dst_host_rerror_rate
Dst_host_srv_rerror_rate
Protocol_type_icmp
Service_eco_i
Service_private
Flag_SF

Table 7 lists the different selected features using XGBoost Feature Selection in the case of R2L.

Table 7. Table shows the selected features for R2L using XGBoost Feature Selection

Features selected
Src_bytes
Dst_bytes
Hot
Num_failed_logins
Is_guest_login
Dst_host_srv_count
Dst_host_same_src_port_rate
Dst_host_srv_diff_host_rate
Service_ftp
Service_ftp_data
Service_http

Service_imap4
Flag_RSTO

Table 8 lists the different selected features using XGBoost Feature Selection in the case of U2R.

Table 8. Table shows the selected features for U2R using XGBoost Feature Selection

Features selected
Urgent
Hot
Root_shell
Num_file_creations
Num_shells
Srv_diff_host_rate
Dst_host_count
Dst_host_srv_count
Dst_host_same_src_port_rate
Dst_host_srv_diff_host_rate
Service_ftp_data
Service_http
Service_telnet

Table 9 lists the different selected features using XGBoost Feature Selection in the case of DoS sorted by rank.

Table 9. Table shows the selected features for DoS using XGBoost Feature Selection by rank

Rank	**Features selected**
1	Same_srv_rate
2	Count
3	Flag_SF
4	Dst_host_serror_rate
5	Dst_home_same_srv_rate
6	Dst_host_srv_count
7	Dst_host_count
8	Logged_in
9	Serror_rate
10	Dst_host_srv_serror_rate
11	Srv_serror_rate
12	Service_http
13	Flag_S0

Table 10 lists the different selected features using XGBoost Feature Selection in the case of Probe sorted by rank.

Table 10. Table shows the selected features for Probe using XGBoost Feature Selection by rank

Rank	**Features selected**
1	dst_host_same_src_port_rate
2	dst_host_srv_count
3	dst_host_rerror_rate
4	service_private
5	logged_in
6	dst_host_diff_srv_rate

7	dst_host_srv_diff_host_rate
8	flag_SF
9	service_eco_i
10	rerror_rate
11	Protocol_type_icmp
12	dst_host_srv_rerror_rate
13	srv_rerror_rate

Table 11 lists the different selected features using XGBoost Feature Selection in the case of R2L sorted by rank.

Table 11. Table shows the selected features for R2L using XGBoost Feature Selection by rank

Rank	**Features selected**
1	src_bytes
2	dst_bytes
3	Hot
4	dst_host_srv_diff_host_rate
5	service_ftp_data
6	dst_host_same_src_port_rate
7	dst_host_srv_count
8	num_failed_logins
9	service_imap4
10	is_guest_login
11	service_ftp
12	flag_RSTO
13	flag_RSTO

Table 12 lists the different selected features using XGBoost Feature Selection in the case of U2R sorted by rank.

Table 12. Table shows the selected features for R2L using XGBoost Feature Selection by rank

Rank	**Features selected**
1	Hot
2	dst_host_srv_count
3	dst_host_count
4	root_shell
5	num_shells
6	service_ftp_data
7	dst_host_srv_diff_host_rate
8	num_file_creations
9	dst_host_same_src_port_rate
10	service_telnet
11	srv_diff_host_rate
12	service_http
13	urgent

Cross-validation executes exhausting stratified sampling for ensuring that the proportion of the feature of interest is uniform throughout the original input data, training set, and testing set. This is done to make sure that none of the value is over or underrepresented in training and test sets. This offers higher accuracy for the estimation of performance and error.

Table 13 shows the Stratified Cross-Validation Accuracy for DoS.

Table 13. Table depicts the reading of Stratified Cross-Validation Accuracy for DoS

Attacks	Accuracy
DoS	0.99738 (+/- 0.00267)
Probe	0.99085 (+/- 0.00559)
R2L	0.97459 (+/- 0.00910)
U2R	0.99652 (+/- 0.00278)

Table 14 displays the maximum and minimum Stratified Cross-Validation accuracy of each of the four attacks.

Table 14. Table depicts the maximum and minimum Stratified Cross-Validation accuracy of the attacks

Accuracy	Maximum	Minimum
DoS	1.00005	0.99471
Probe	0.99644	0.98526
R2L	0.98369	0.96549
U2R	0.9993	0.99374

Fig. 3 displays the maximum and minimum of the Stratified Cross-Validation accuracy of the four attacks as per readings of Table 14. The green-colored line denotes the maximum accuracy and the blue color denotes the minimum accuracy. DoS attack has maximum Stratified Cross-Validation accuracy of 1.00005 and the R2L attack has the minimum Stratified Cross-Validation accuracy of 0.96549.

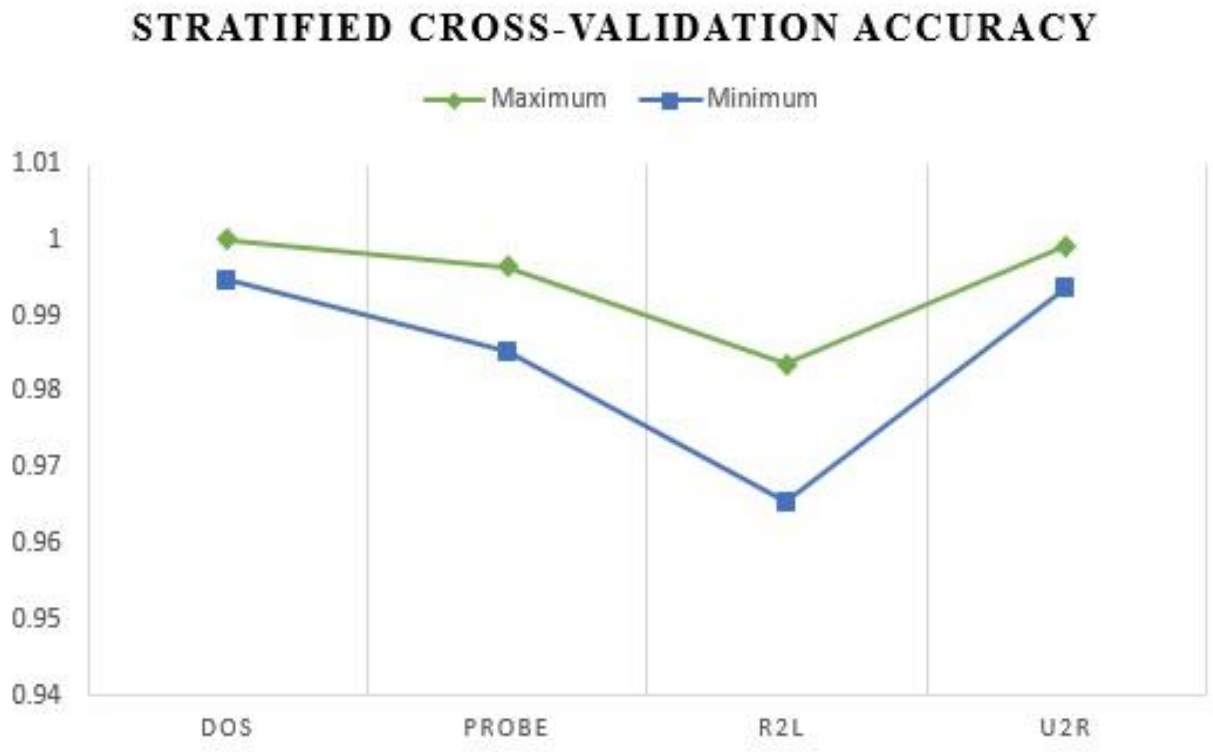

Fig. 3 Figure displays the maximum and minimum of the Stratified Cross-Validation accuracy of the four attacks as per readings of Table 14

Table 15 displays the Cross-Validation readings of four performance evaluation parameters Accuracy, Precision, Recall, and F1-Measure for the four attacks DoS, Probe, U2R, and R2L.

Table 15. Table depicts the Cross-Validation readings of four performance evaluation parameters for the four attacks

Parameters	DoS	Probe	R2L	U2R
Accuracy	0.99639 (+/- 0.00341)	0.99571 (+/- 0.00328)	0.97920 (+/- 0.01053)	0.99652 (+/- 0.00228)
Precision	0.99505 (+/- 0.00477)	0.99392 (+/- 0.00684)	0.97151 (+/- 0.01736)	0.86295 (+/- 0.08961)
Recall	0.99665 (+/- 0.00483)	0.99267 (+/- 0.00405)	0.96958 (+/- 0.01379)	0.90958 (+/- 0.09211)
F1-measure	0.99585 (+/- 0.00392)	0.99329 (+/- 0.00512)	0.97051 (+/- 0.01478)	0.88210 (+/- 0.06559)

Table 16 depicts the maximum and minimum Cross-Validation Accuracy, Precision, Recall, and F1-measure of DoS.

Table 16. Table demonstrates the maximum and minimum Cross-Validation of the four performance evaluation parameters of DoS

Parameters	Maximum	Minimum
Accuracy	0.9998	0.99298
Precision	0.99982	0.99028
Recall	1.00148	0.99182
F1-measure	0.99977	0.99193

Fig. 4 displays the maximum and minimum values for the four performance evaluation parameters for Cross-Validation of DoS. The Recall has a maximum value of 1.00148 and precision has a minimum value of 0.99028.

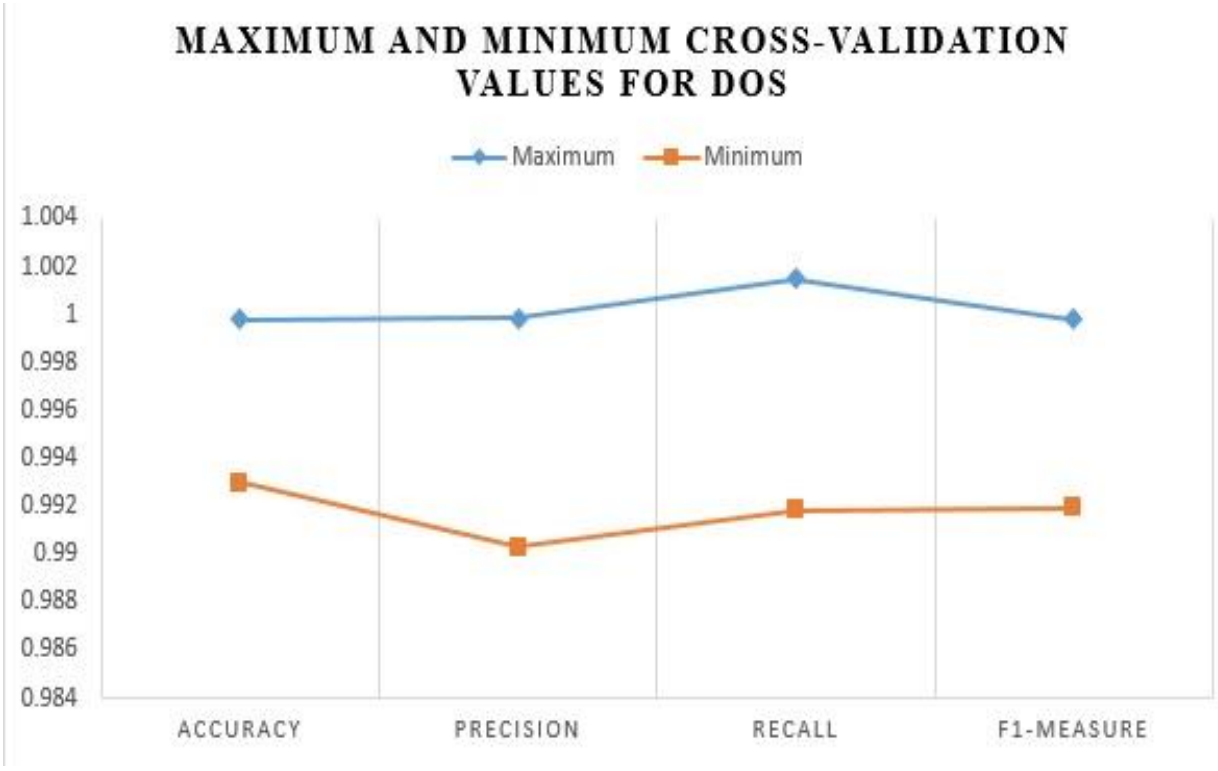

Fig. 4 Figure displays the maximum and minimum values for the four performance evaluation parameters for Cross-Validation of DoS

Table 17 depicts the maximum and minimum Cross-Validation Accuracy, Precision, Recall, and F1-measure of Probe.

Table 17. Table demonstrates the maximum and minimum Cross-Validation of the four performance evaluation parameters of Probe

Parameters	Maximum	Minimum
Accuracy	0.99899	0.99243
Precision	1.00076	0.98711
Recall	0.99672	0.98862
F1-measure	0.99841	0.98817

Fig. 5 displays the maximum and minimum values for the four performance evaluation parameters for the Cross-Validation of Probe. The Precision has a maximum value of 1.00076 and a minimum value of 0.98711.

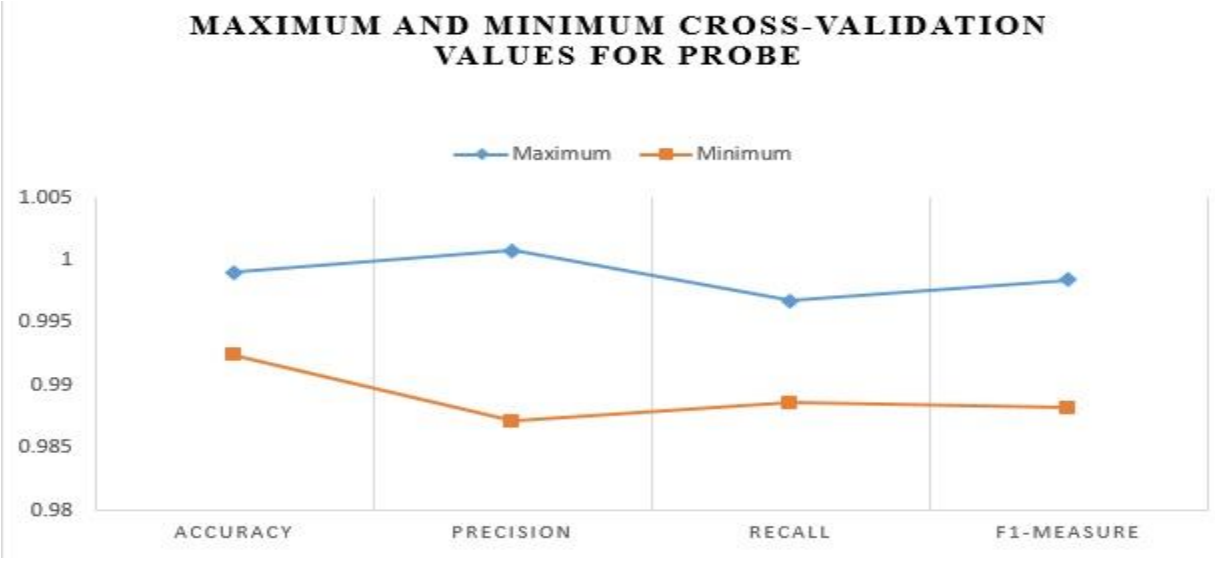

Fig. 5 Figure displays the maximum and minimum values for the four performance evaluation parameters for Cross-Validation of Probe

Table 18 depicts the maximum and minimum Cross-Validation Accuracy, Precision, Recall, and F1-measure of R2L.

Table 18. Table demonstrates the maximum and minimum Cross-Validation of the four performance evaluation parameters of R2L

Parameters	Maximum	Minimum
Accuracy	0.98973	0.96867
Precision	0.98887	0.95415
Recall	0.98337	0.95579
F1-measure	0.98529	0.95573

Fig. 6 displays the maximum and minimum values for the four performance evaluation parameters for Cross-Validation of R2L. The Accuracy has a maximum value of 0.98973 and Precision has a minimum value of 0.95415.

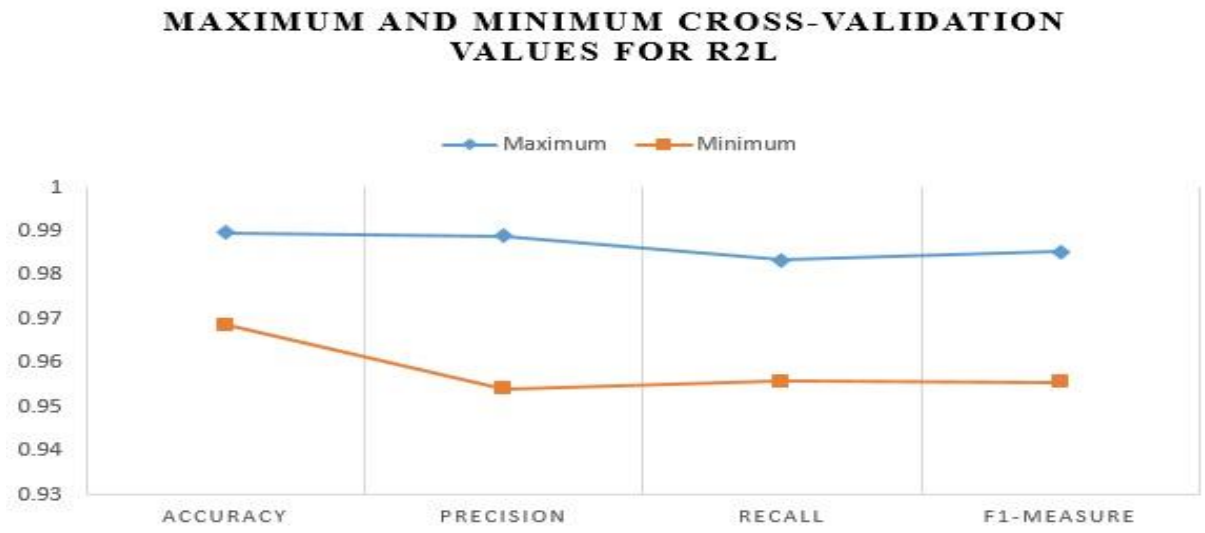

Fig. 6 Figure displays the maximum and minimum values for the four performance evaluation parameters for Cross-Validation of R2L

Table 19 depicts the maximum and minimum Cross-Validation Accuracy, Precision, Recall, and F1-measure of U2R.

Table 19. Table demonstrates the maximum and minimum Cross-Validation of the four performance evaluation parameters of U2R

Parameters	Maximum	Minimum
Accuracy	0.9988	0.99424
Precision	0.95256	0.77334
Recall	1.00169	0.81747
F1-measure	0.94769	0.81651

Fig. 7 displays the maximum and minimum values for the four performance evaluation parameters for Cross-Validation of U2R. The Recall has a maximum value of 1.00169 and Precision has a minimum value of 0.77334.

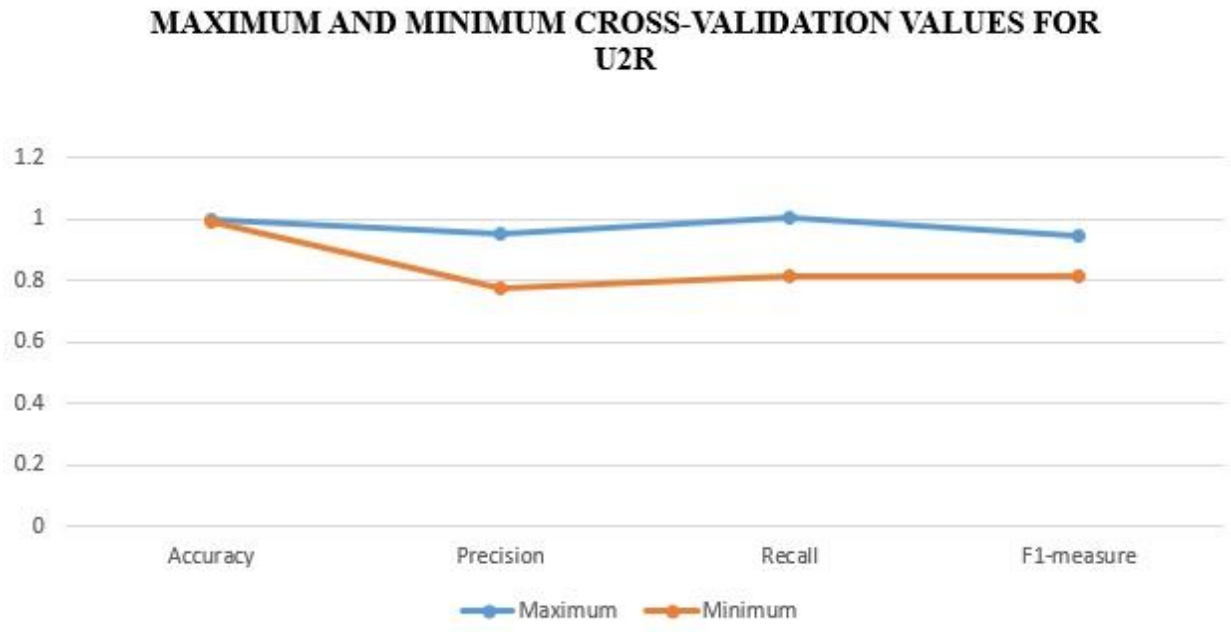

Fig. 7 Figure displays the maximum and minimum values for the four performance evaluation parameters for Cross-Validation of U2R

Table 20 shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for DoS.

Table 20. Table shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for DoS

Accuracy at	Value
2 folds	0.99662 (+/- 0.00116)
5 folds	0.99709 (+/- 0.00064
10 folds	0.99738 (+/- 0.00267
30 folds	0.99726 (+/- 0.00430
50 folds	0.99703 (+/- 0.00622

Table 21 shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for DoS.

Table 21. Table shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for DoS

Accuracy	Maximum	Minimum
2 folds	0.99778	0.99546
5 folds	0.99773	0.99645
10 folds	1.00005	0.99471
30 folds	1.00156	0.99296
50 folds	1.00325	0.99081

Fig. 8 shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for DoS as per the reading of Table 21. The maximum accuracy of 1.00325 is achieved on 50 folds and the minimum accuracy of 0.99081 is also achieved on 50 folds.

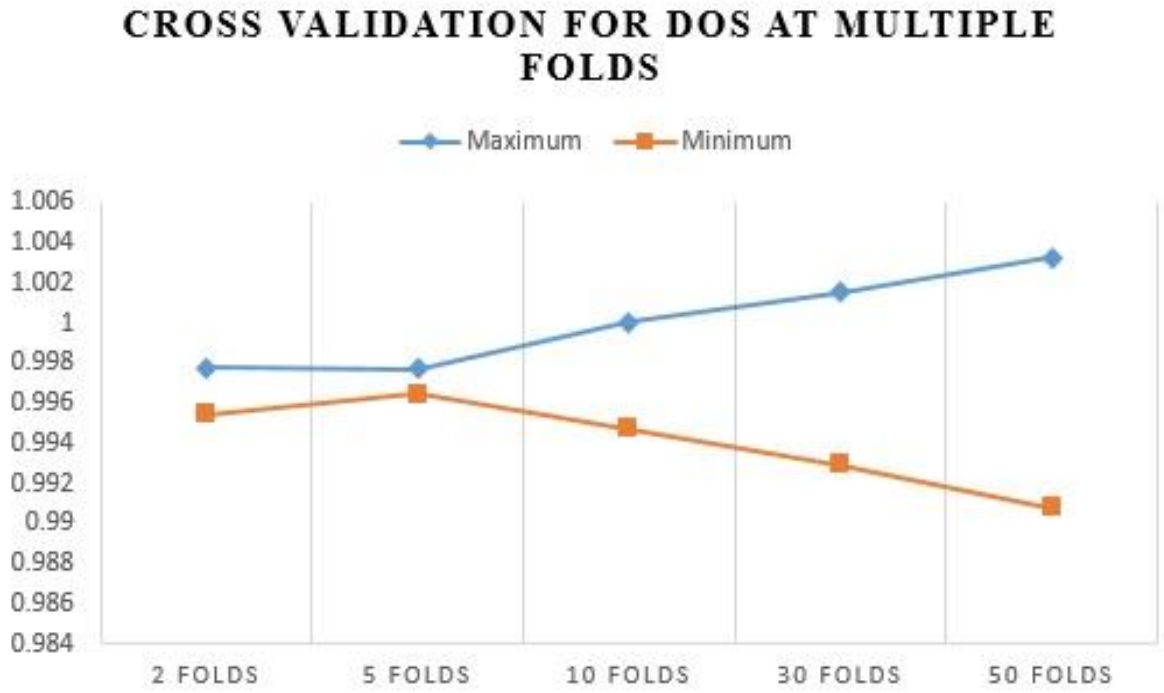

Fig. 8 Figure shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at multiple folds for DoS

Table 22 shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for Probe.

Table 22. Table shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for Probe

Accuracy at	Value
2 folds	0.99060 (+/- 0.00165)
5 folds	0.99093 (+/- 0.00233)
10 folds	0.99085 (+/- 0.00559)
30 folds	0.99118 (+/- 0.00742)
50 folds	0.99085 (+/- 0.01122)

Table 23 shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for the Probe.

Table 23. Table shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for Probe

Accuracy	Maximum	Minimum
2 folds	0.99225	0.98895
5 folds	0.99326	0.9886
10 folds	0.99644	0.98526
30 folds	0.9986	0.98376
50 folds	1.00207	0.97963

Fig. 9 shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for Probe as per the reading of Table 23. The maximum accuracy of 1.00207 is achieved at 50 folds and the minimum accuracy of 0.97963 is also achieved at 50 folds.

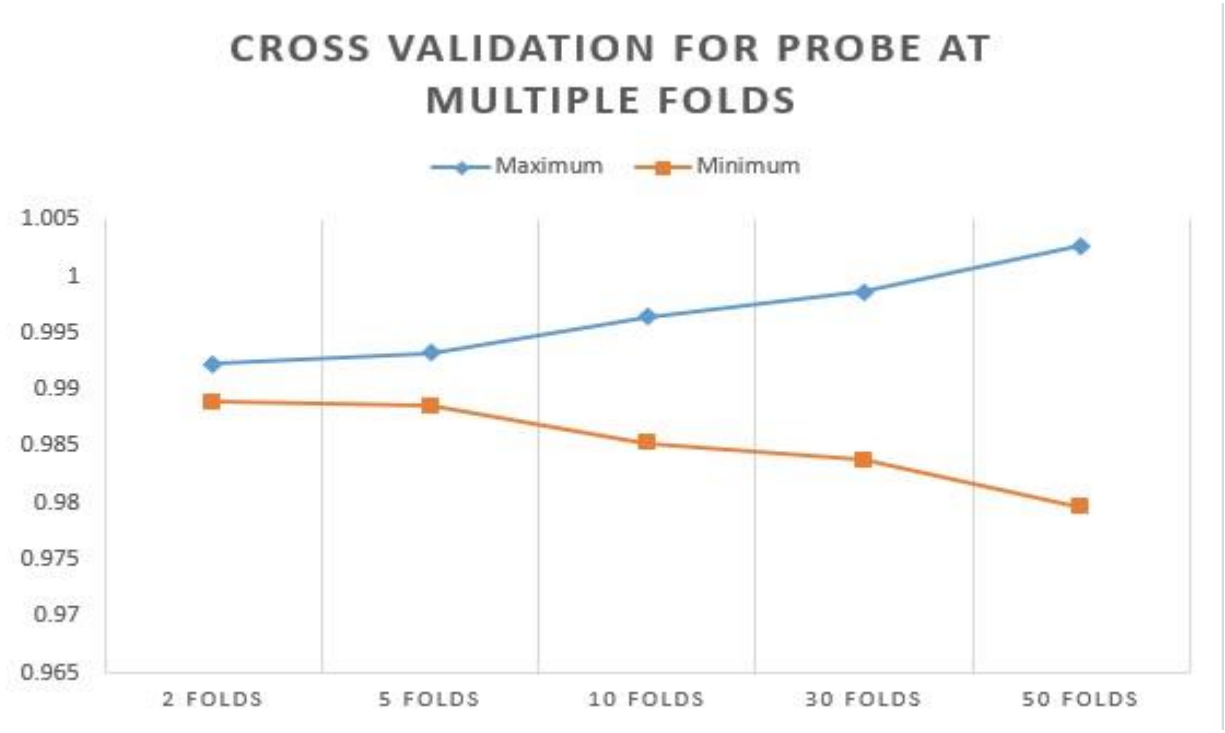

Fig. 9 Figure shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at multiple folds for Probe

Table 24 shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for R2L.

Table 24. Table shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for R2L

Accuracy at	Value
2 folds	0.97118 (+/- 0.00143)

5 folds	0.97388 (+/- 0.00624)
10 folds	0.97459 (+/- 0.00910)
30 folds	0.97467 (+/- 0.01644)
50 folds	0.97523 (+/- 0.01795)

Table 25 shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for R2L.

Table 25. Table shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for R2L

Accuracy	**Maximum**	**Minimum**
2 folds	0.97261	0.96975
5 folds	0.98012	0.96764
10 folds	0.98369	0.96549
30 folds	0.99111	0.95823
50 folds	0.99318	0.95728

Fig. 10 shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for R2L as per the reading of Table 25. The maximum accuracy of 0.99318 is achieved at 50 folds and the minimum accuracy of 0.95728 is also achieved at 50 folds.

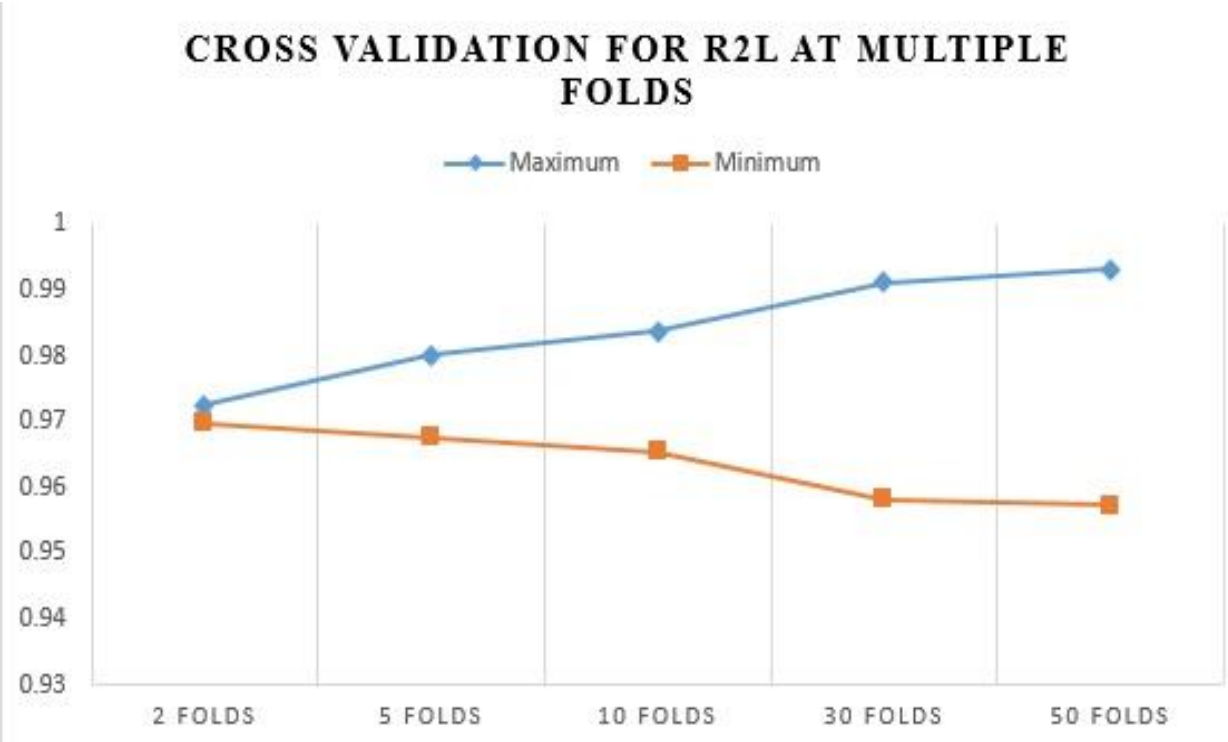

Fig. 10 Figure shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at multiple folds for R2L

Table 26 shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for U2R.

Table 24. Table shows the achieved Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for U2R

Accuracy at	**Value**
2 folds	0.99519 (+/- 0.00184)
5 folds	0.99714 (+/- 0.00153)
10 folds	0.99652 (+/- 0.00278)

30 folds	0.99693 (+/- 0.00571)
50 folds	0.99662 (+/- 0.00755)

Table 27 shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for U2R.

Table 27. Table shows the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for U2R

Accuracy	Maximum	Minimum
2 folds	0.99703	0.99335
5 folds	0.99867	0.99561
10 folds	0.9993	0.99374
30 folds	1.00264	0.99122
50 folds	1.00417	0.98907

Fig. 11 shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for U2R as per the reading of Table 27. The maximum accuracy of 1.00417 is achieved at 50 folds and the minimum accuracy of 0.98907 is also achieved at 50 folds.

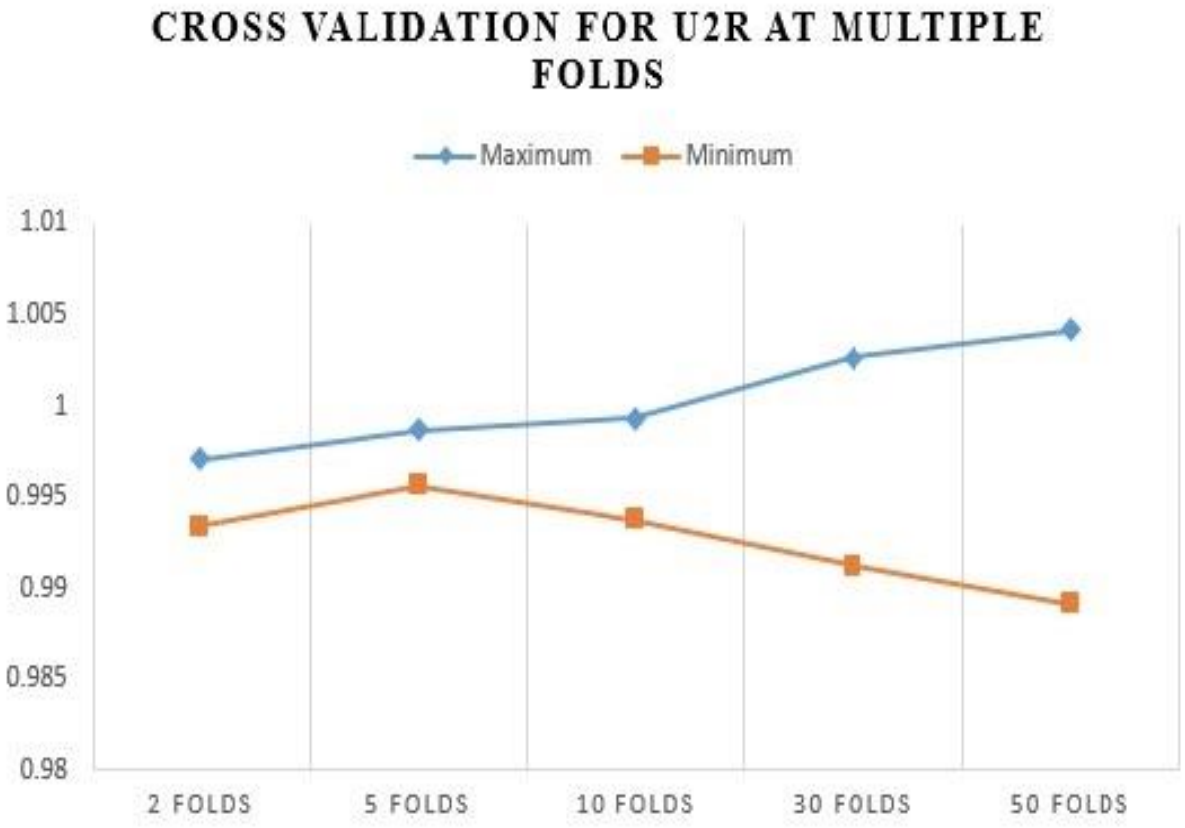

Fig. 11 Figure shows the graphical representation of the maximum and minimum Cross-Validation Accuracy at multiple folds for U2R

Table 28 shows the comparative evaluation of maximum Accuracy achieved at different folds for the four attacks under discussion.

Table 28. Table shows the comparative evaluation of maximum Accuracy achieved at different folds

Accuracy	DoS	Probe	R2L	U2R

2 folds	0.99778	0.99225	0.97261	0.99703
5 folds	0.99773	0.99326	0.98012	0.99867
10 folds	1.00005	0.99644	0.98369	0.9993
30 folds	1.00156	0.9986	0.99111	1.00264
50 folds	1.00325	1.00207	0.99318	1.00417

Fig. 12 shows the graphical representation of the maximum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for the four attacks. The maximum accuracy has been achieved at 50 folds for each of the attacks which implies the more is the number of folds, the greater is the maximum Cross-Validation Accuracy achieved.

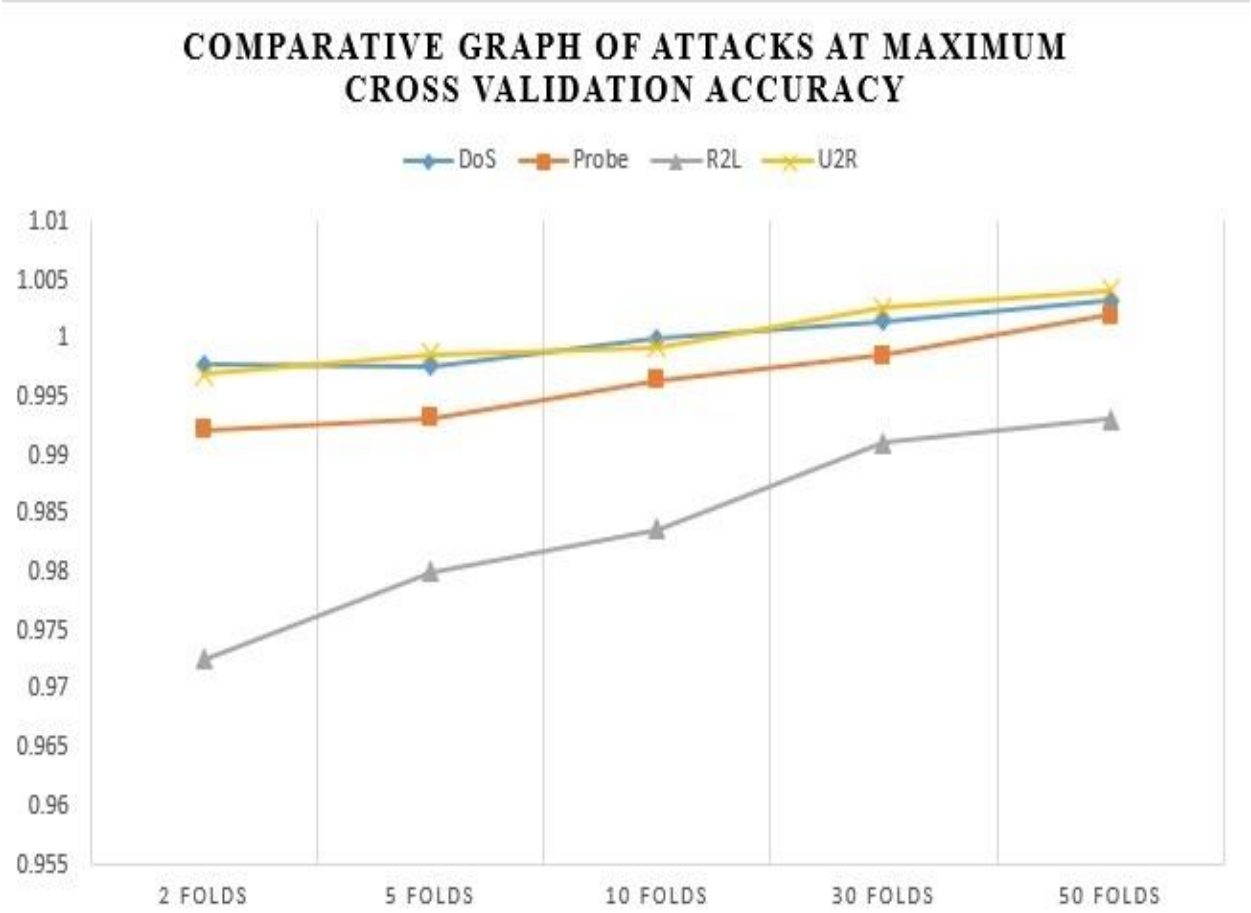

Fig. 12 Figure shows the graphical representation of the maximum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for the four attacks

Table 29 shows the comparative evaluation of minimum Accuracy achieved at different folds for the four attacks under discussion.

Table 29. Table shows the comparative evaluation of minimum Accuracy achieved at different folds

Accuracy	**DoS**	**Probe**	**R2L**	**U2R**
2 folds	0.99546	0.98895	0.96975	0.99335
5 folds	0.99645	0.9886	0.96764	0.99561
10 folds	0.99471	0.98526	0.96549	0.99374

30 folds	0.99296	0.98376	0.95823	0.99122
50 folds	0.99081	0.97963	0.95728	0.98907

Fig. 13 shows the graphical representation of the minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for the four attacks. The minimum accuracy has been achieved at 50 folds for each of the attacks which implies the more is the number of folds, the lesser is the minimum Cross-Validation Accuracy achieved.

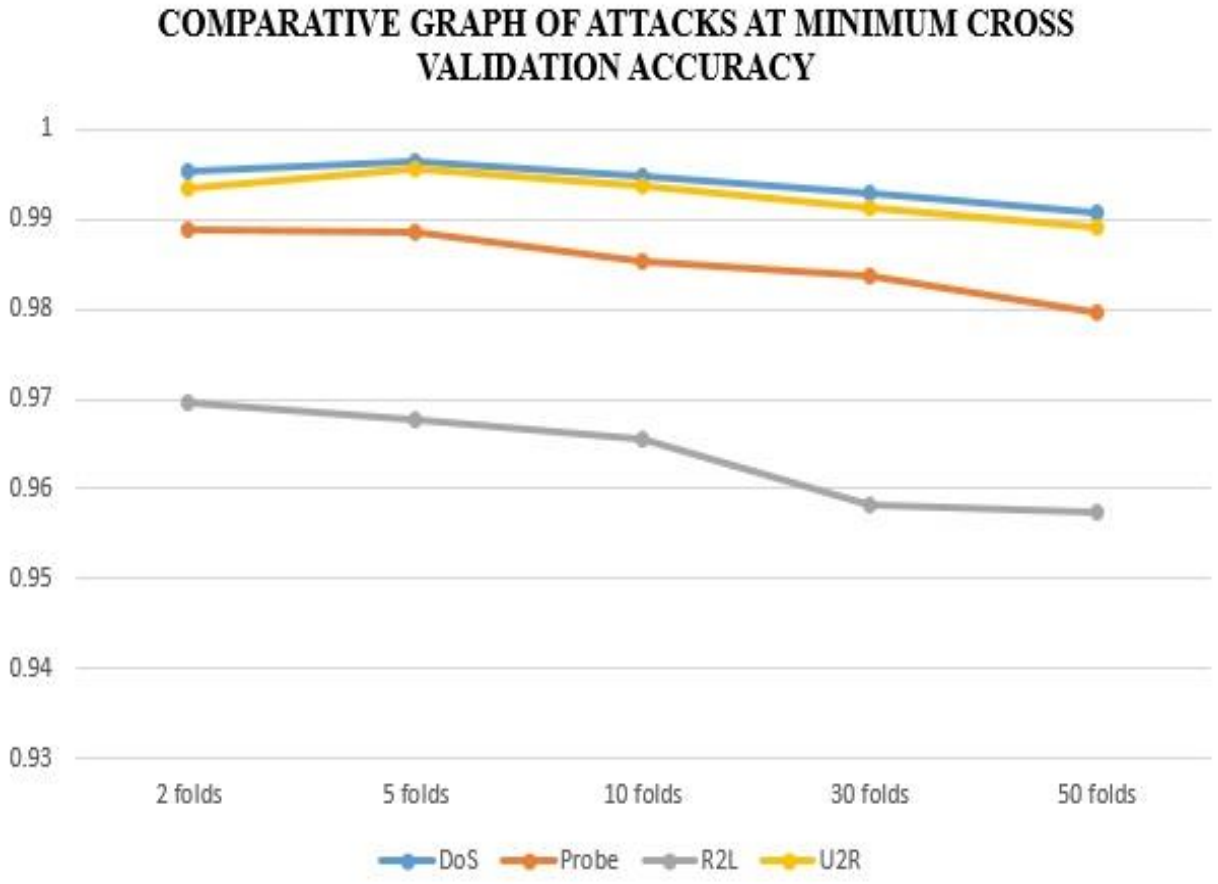

Fig. 13 Figure shows the graphical representation of the minimum Cross-Validation Accuracy at 2, 5, 10, 30, 50 folds for the four attacks

The results have been obtained at different percentages (ratios) for the seven prominent classifiers, LR (Logistic Regression), NB (Naïve Bayes), DT (Decision Trees), AB (AdaBoost), RF (Random Forest), kNN (k Nearest Neighbors), and SVM (Support Vector Machine) for calculating Accuracy, Precision, Recall, and F1Score as performance evaluation parameters.

Table 30 shows the ontained readings performance evaluation parameters for the seven classifiers at 60-40%.

Table 30. Table shows the readings of performance evaluation parameters of seven different classifiers at 60-40%

Parameters / Classifiers	**LR**	**NB**	**DT**	**AB**	**RF**	**kNN**	**SVM**
Accuracy	0.848	0.929	0.93	0.925	0.927	0.929	0.929
Precision	0.989	0.988	0.999	0.995	0.999	0.998	0.998
Recall	0.821	0.923	0.914	0.911	0.91	0.913	0.913

F1Score	0.897	0.955	0.955	0.951	0.952	0.954	0.954

Fig. 14 demonstrates the readings obtained in Table 30 in graphical form. The blue-colored bars denote Accuracy, orange-colored bars represent Precision, grey-colored bars represent Recall, and yellow-colored bars are for F1Score.

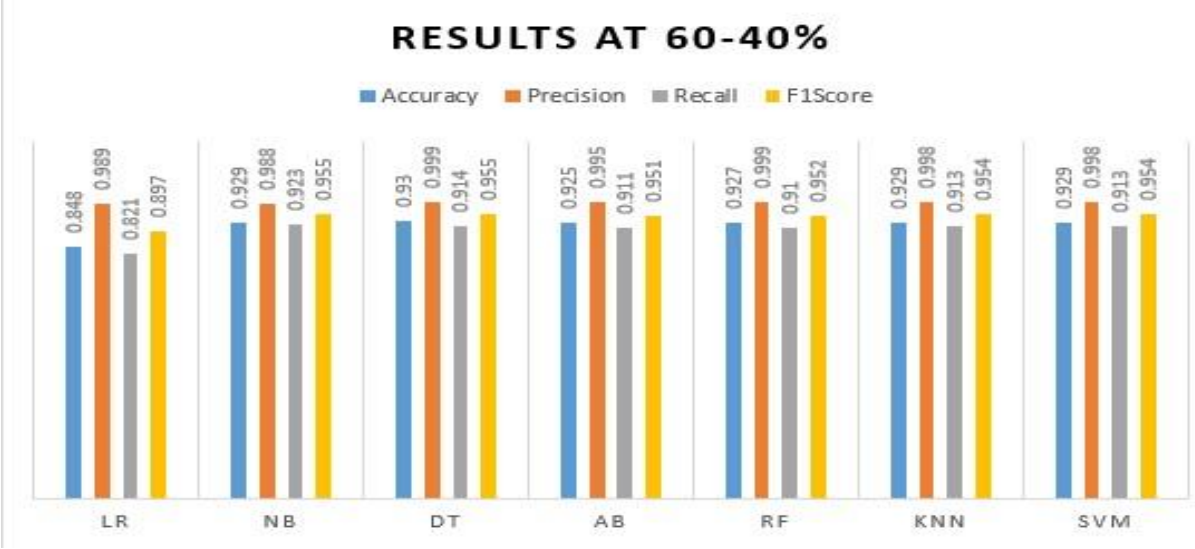

Fig. 14 Figure shows the readings of performance evaluation parameters of seven different classifiers at 60-40%

Table 31 shows the obtained readings performance evaluation parameters for the seven classifiers at 70-30%.

Table 31. Table shows the readings of performance evaluation parameters of seven different classifiers at 70-30%

Parameters / Classifiers	**LR**	**NB**	**DT**	**AB**	**RF**	**kNN**	**SVM**
Accuracy	0.848	0.929	0.929	0.925	0.927	0.927	0.928
Precision	0.989	0.988	0.999	0.995	0.999	0.998	0.998
Recall	0.821	0.923	0.913	0.911	0.91	0.912	0.913
F1Score	0.897	0.955	0.954	0.951	0.952	0.955	0.954

Fig. 15 demonstrates the readings obtained in Table 31 in graphical form. The blue-colored bars denote Accuracy, orange-colored bars represent Precision, grey-colored bars represent Recall, and yellow-colored bars are for F1Score.

Fig. 15 Figure shows the readings of performance evaluation parameters of seven different classifiers at 70-30%

Table 32 shows the obtained readings performance evaluation parameters for the seven classifiers at 90-10%.

Table 32. Table shows the readings of performance evaluation parameters of seven different classifiers at 90-10%

Parameters / Classifiers	**LR**	**NB**	**DT**	**AB**	**RF**	**kNN**	**SVM**
Accuracy	0.848	0.929	0.93	0.925	0.927	0.928	0.928
Precision	0.989	0.988	0.999	0.995	0.999	0.999	0.998
Recall	0.821	0.923	0.915	0.911	0.911	0.914	0.915
F1Score	0.897	0.955	0.955	0.951	0.953	0.956	0.956

Fig. 16 demonstrates the readings obtained in Table 32 in graphical form. The blue-colored bars denote Accuracy, orange-colored bars represent Precision, grey-colored bars represent Recall, and yellow-colored bars are for F1Score.

Fig. 16 Figure shows the readings of performance evaluation parameters of seven different classifiers at 90-10%

Table 33 shows the obtained readings performance evaluation parameters for the seven classifiers at 50-50%.

Table 33. Table shows the readings of performance evaluation parameters of seven different classifiers at 50-50%

Parameters / Classifiers	**LR**	**NB**	**DT**	**AB**	**RF**	**kNN**	**SVM**
Accuracy	0.848	0.929	0.929	0.925	0.926	0.927	0.928
Precision	0.989	0.988	0.999	0.995	0.999	0.998	0.998
Recall	0.821	0.923	0.913	0.911	0.91	0.913	0.914
F1Score	0.897	0.955	0.954	0.951	0.952	0.955	0.956

Fig. 17 demonstrates the readings obtained in Table 33 in graphical form. The blue-colored bars denote Accuracy, orange-colored bars represent Precision, grey-colored bars represent Recall, and yellow-colored bars are for F1Score.

Fig. 17 Figure shows the readings of performance evaluation parameters of seven different classifiers at 50-50%

Table 34 shows the obtained readings performance evaluation parameters for the seven classifiers at 80-20%.

Table 34. Table shows the readings of performance evaluation parameters of seven different classifiers at 80-20%

Parameters / Classifiers	LR	NB	DT	AB	RF	kNN	SVM
Accuracy	0.848	0.929	0.928	0.925	0.926	0.926	0.925
Precision	0.989	0.988	0.999	0.995	0.999	0.999	0.998
Recall	0.821	0.923	0.912	0.911	0.91	0.924	0.925
F1Score	0.897	0.955	0.954	0.951	0.952	0.953	0.953

Fig. 18 demonstrates the readings obtained in Table 34 in graphical form. The blue-colored bars denote Accuracy, orange-colored bars represent Precision, grey-colored bars represent Recall, and yellow-colored bars are for F1Score.

Fig. 18 Figure shows the readings of performance evaluation parameters of seven different classifiers at 80-20%

VI. CONCLUSION

The research paper made use of XGBoost for performing feature selection on the NSL KDD dataset. XGBoost can be put into numerous use cases such as classification, regression, ranking, and user-defined prediction problems. Once the boosted trees are constructed, the process of retrieving importance scores for different attributes gets easier. The importance allows determining the usefulness of each feature in the making of boosted decision trees by providing appropriate scores. The greater is the use of attributes in making key decisions, the higher will be its comparative importance. The significance is assessed unambiguously for every attribute in the dataset, permitting attributes to be categorized and matched to each other. The high flexibility, parallel processing, regularization, and ability to handle missing data via its in-build features made XGBoost the best choice to be adopted for conducting feature selection in the adopted research methodology in the research paper.

The research paper elaborated on different types of IDS. Seven prominent classifiers (LR, NB, DT, AB, RF, kNN, and SVM) have been discussed with their pros and cons. The NSL KDD dataset has been used for training and testing. Feature selection has been used as depicted in the detailed flowchart and the algorithm to enhance the overall performance of the classifiers. The results obtained at different ratios are summarized below.

At 60-40%:

DT achieves the highest Accuracy of 0.93. DT and RF achieved the best Precision of 0.999. NB scores 0.923 as best achieved Recall. F1Score is best for NB and DT at 0.955. The obtained results are shown in Table 35.

Table 35. Table shows the classifiers which performed best against different performance evaluation parameters with numerical values at 60-40%

Performance Evaluation Parameters	Classifiers	Numerical Values
Accuracy	DT	0.93
Precision	DT / RF	0.999
Recall	NB	0.923
F1Score	NB / DT	0.955

At 70-30%:

NB and DT achieve the highest Accuracy of 0.929. DT and RF achieved the best Precision of 0.999. NB scores 0.923 as best achieved Recall. F1Score is best for NB and kNN at 0.955. The obtained results are shown in Table 36.

Table 36. Table shows the classifiers which performed best against different performance evaluation parameters with numerical values at 70-30%

Performance Evaluation Parameters	Classifiers	Numerical Values
Accuracy	NB / DT	0.929
Precision	DT / RF	0.999
Recall	NB	0.923
F1Score	NB / kNN	0.955

At 90-10%:

NB achieve the highest Accuracy of 0.929. DT, RF, and kNN achieved the best Precision of 0.999. NB scores 0.923 as best achieved Recall. F1Score is best for kNN and SVM at 0.956. The obtained results are shown in Table 37.

Table 37. Table shows the classifiers which performed best against different performance evaluation parameters with numerical values at 90-10%

Performance Evaluation Parameters	Classifiers	Numerical Values
Accuracy	NB	0.929
Precision	DT / RF / kNN	0.999
Recall	NB	0.923
F1Score	kNN / SVM	0.956

At 50-50%:

NB and DT achieve the highest Accuracy of 0.929. DT and RF achieved the best Precision of 0.999. NB scores 0.923 as best achieved Recall. F1Score is best for SVM at 0.956. The obtained results are shown in Table 38.

Table 38. Table shows the classifiers which performed best against different performance evaluation parameters with numerical values at 50-50%

Performance Evaluation Parameters	Classifiers	Numerical Values
Accuracy	NB / DT	0.929
Precision	DT / RF	0.999
Recall	NB	0.923
F1Score	SVM	0.956

At 80-20%:

NB achieve the highest Accuracy of 0.929. DT, RF, and kNN achieved the best Precision of 0.999. SVM scores 0.925 as best achieved Recall. F1Score is best for NB at 0.955. The obtained results are shown in Table 39.

Table 39. Table shows the classifiers which performed best against different performance evaluation parameters with numerical values at 80-20%

Performance Evaluation Parameters	Classifiers	Numerical Values
Accuracy	NB	0.929
Precision	DT / RF / kNN	0.999
Recall	SVM	0.925
F1Score	NB	0.955

So, it can be concluded that in most of the cases the NB scored with the best Accuracy and Recall values. DT and RF consistently performed with high accuracy. NB, SVM, and kNN achieved good F1Score.

DECLARATIONS

Conflict of Interest - On behalf of all authors, the corresponding author states that there is no conflict of interest.
Availability of data and material – Data used in the research is available at NSL-KDD data repositories.

REFERENCES

Aburomman, A. A. and Reaz, M. B. I. (2017) 'A survey of intrusion detection systems based on ensemble and hybrid classifiers', *Computers and Security*, 65, pp. 135–152. doi: 10.1016/j.cose.2016.11.004.

Agrawal, S. and Agrawal, J. (2015) 'Survey on anomaly detection using data mining techniques', *Procedia Computer Science*, 60(1), pp. 708–713. doi: 10.1016/j.procs.2015.08.220.

Ahmim, A., Derdour, M. and Ferrag, M. A. (2018) 'An intrusion detection system based on combining probability predictions of a tree of classifiers', *International Journal of Communication Systems*, 31(9), pp. 1–17. doi: 10.1002/dac.3547.

Alaparthy, V. and Morgera, S. D. (2019) 'Modeling an Intrusion Detection System Based on Adaptive Immunology', *International Journal of Interdisciplinary Telecommunications and Networking*, 11(2), pp. 42–55. doi: 10.4018/ijitn.2019040104.

Alaparthy, V. T. and Morgera, S. D. (2018) 'A Multi-Level Intrusion Detection System for Wireless Sensor Networks Based on Immune Theory', *IEEE Access*, 6, pp. 47364–47373. doi: 10.1109/ACCESS.2018.2866962.

Amouri, A., Alaparthy, V. T. and Morgera, S. D. (2018) 'Cross layer-based intrusion detection based on network behavior for IoT', *2018 IEEE 19th Wireless and Microwave Technology Conference, WAMICON 2018*, pp. 1–4. doi: 10.1109/WAMICON.2018.8363921.

Andresini, G. *et al.* (2020) 'Multi-Channel Deep Feature Learning for Intrusion Detection', *IEEE Access*, 8, pp. 53346–53359. doi: 10.1109/ACCESS.2020.2980937.

Belavagi, M. C. and Muniyal, B. (2016) 'Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection', *Procedia Computer Science*, 89, pp. 117–123. doi: 10.1016/j.procs.2016.06.016.

Buczak, A. L. and Guven, E. (2016) 'A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection', *IEEE Communications Surveys and Tutorials*, 18(2), pp. 1153–1176. doi: 10.1109/COMST.2015.2494502.

da Costa, K. A. P. *et al.* (2019) 'Internet of Things: A survey on machine learning-based intrusion detection approaches', *Computer Networks*, 151,

pp. 147–157. doi: 10.1016/j.comnet.2019.01.023.

Derhab, A. (2019) 'A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection', 7.

Dhaliwal, S. S., Nahid, A. Al and Abbas, R. (2018) 'Effective intrusion detection system using XGBoost', *Information (Switzerland)*, 9(7). doi: 10.3390/info9070149.

Dhanabal, L. and Shantharajah, S. P. (2015) 'A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms', *International Journal of Advanced Research in Computer and Communication Engineering*, 4(6), pp. 446–452. doi: 10.17148/IJARCCE.2015.4696.

Farnaaz, N. and Jabbar, M. A. (2016) 'Random Forest Modeling for Network Intrusion Detection System', *Procedia Computer Science*, 89, pp. 213–217. doi: 10.1016/j.procs.2016.06.047.

Gao, X. *et al.* (2019) 'An Adaptive Ensemble Machine Learning Model for Intrusion Detection', *IEEE Access*, 7, pp. 82512–82521. doi: 10.1109/ACCESS.2019.2923640.

García-Teodoro, P. *et al.* (2009) 'Anomaly-based network intrusion detection: Techniques, systems and challenges', *Computers and Security*, 28(1–2), pp. 18–28. doi: 10.1016/j.cose.2008.08.003.

Gu, J. *et al.* (2019) 'A novel approach to intrusion detection using SVM ensemble with feature augmentation', *Computers and Security*, 86, pp. 53–62. doi: 10.1016/j.cose.2019.05.022.

Hassan, M. M. *et al.* (2020) 'Increasing the Trustworthiness in the Industrial IoT Networks through a Reliable Cyberattack Detection Model', *IEEE Transactions on Industrial Informatics*, 16(9), pp. 6154–6162. doi: 10.1109/TII.2020.2970074.

Hussain, J. and Lalmuanawma, S. (2016) 'Feature Analysis, Evaluation and Comparisons of Classification Algorithms Based on Noisy Intrusion Dataset', *Procedia Computer Science*, 92, pp. 188–198. doi: 10.1016/j.procs.2016.07.345.

Karatas, G., Demir, O. and Sahingoz, O. K. (2020) 'Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset', *IEEE Access*, 8, pp. 32150–32162. doi: 10.1109/ACCESS.2020.2973219.

Kasongo, S. M. and Sun, Y. (2020) 'Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset', *Journal of Big Data*, 7(1). doi: 10.1186/s40537-020-00379-6.

Khan, F. A. and Gumaei, A. (2019) *A Comparative Study of Machine Learning Classifiers for Network Intrusion Detection*, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer International Publishing. doi: 10.1007/978-3-030-24265-7_7.

Khraisat, A. *et al.* (2019) 'Survey of intrusion detection systems: techniques, datasets and challenges', *Cybersecurity*, 2(1). doi: 10.1186/s42400-019-0038-7.

Kwon, D. *et al.* (2019) 'A survey of deep learning-based network anomaly detection', *Cluster Computing*, 22, pp. 949–961. doi: 10.1007/s10586-017-1117-8.

Liu, H. and Lang, B. (2019) 'Machine learning and deep learning methods for intrusion detection systems: A survey', *Applied Sciences (Switzerland)*, 9(20). doi: 10.3390/app9204396.

Liu, J. and Xu, L. (2019) 'Improvement of SOM classification algorithm and application effect analysis in intrusion detection', in *Advances in Intelligent Systems and Computing*. Springer Verlag, pp. 559–565. doi: 10.1007/978-981-10-8944-2_65.

Naseer, S. *et al.* (2018) 'Enhanced network anomaly detection based on deep neural networks', *IEEE Access*, 6, pp. 48231–48246. doi: 10.1109/ACCESS.2018.2863036.

Networks, A. H. (2003) 'SVM-based Intrusion Detection System for Wireless', pp. 2147–2151.

Panigrahi, R. and Borah, S. (2018) 'A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems', *International Journal of Engineering and Technology(UAE)*, 7(3.24 Special Issue 24), pp. 479–482.

Riahi Sfar, A. *et al.* (2018) 'A roadmap for security challenges in the Internet of Things', *Digital Communications and Networks*, 4(2), pp. 118–137. doi: 10.1016/j.dcan.2017.04.003.

Roshan, S. *et al.* (2018) 'Adaptive and online network intrusion detection system using clustering and Extreme Learning Machines', *Journal of the Franklin Institute*, 355(4), pp. 1752–1779. doi: 10.1016/j.jfranklin.2017.06.006.

Sharafaldin, I. *et al.* (2017) 'Towards a Reliable Intrusion Detection Benchmark Dataset', *Software Networking*, 2017(1), pp. 177–200. doi: 10.13052/jsn2445-9739.2017.009.

Sheikhan, M., Jadidi, Z. and Farrokhi, A. (2012) 'Intrusion detection using reduced-size RNN based on feature grouping', *Neural Computing and Applications*, 21(6), pp. 1185–1190. doi: 10.1007/s00521-010-0487-0.

Surendar, M. and Umamakeswari, A. (2016) 'InDReS: An Intrusion Detection and response system for Internet of Things with 6LoWPAN', *Proceedings of the 2016 IEEE International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2016*, pp.

1903–1908. doi: 10.1109/WiSPNET.2016.7566473.

Teng, S. *et al.* (2018) 'SVM-DT-based adaptive and collaborative intrusion detection', *IEEE/CAA Journal of Automatica Sinica*, 5(1), pp. 108–118. doi: 10.1109/JAS.2017.7510730.

Thamilarasu, G. and Chawla, S. (2019) 'Towards deep-learning-driven intrusion detection for the internet of things', *Sensors (Switzerland)*, 19(9). doi: 10.3390/s19091977.

Vasilomanolakis, E. *et al.* (2015) 'Taxonomy and survey of collaborative intrusion detection', *ACM Computing Surveys*, 47(4). doi: 10.1145/2716260.

Wang, P. *et al.* (2010) 'Survey on HMM based anomaly intrusion detection using system calls', *ICCSE 2010 - 5th International Conference on Computer Science and Education, Final Program and Book of Abstracts*, (2008), pp. 102–105. doi: 10.1109/ICCSE.2010.5593839.

Wang, Y. *et al.* (2018) 'A fog-based privacy-preserving approach for distributed signature-based intrusion detection', *Journal of Parallel and Distributed Computing*, 122, pp. 26–35. doi: 10.1016/j.jpdc.2018.07.013.

Wei, P. *et al.* (2019) 'An optimization method for intrusion detection classification model based on deep belief network', *IEEE Access*, 7, pp. 87593–87605. doi: 10.1109/ACCESS.2019.2925828.

Xiao, Y. *et al.* (2019) 'An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks', *IEEE Access*, 7(c), pp. 42210–42219. doi: 10.1109/ACCESS.2019.2904620.

Yan, B. and Han, G. (2018) 'Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System', *IEEE Access*, 6, pp. 41238–41248. doi: 10.1109/ACCESS.2018.2858277.

Yao, H. *et al.* (2019) 'MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system', *IEEE Internet of Things Journal*, 6(2), pp. 1949–1959. doi: 10.1109/JIOT.2018.2873125.

Yin, C. *et al.* (2017) 'A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks', *IEEE Access*, 5, pp. 21954–21961. doi: 10.1109/ACCESS.2017.2762418.

Yu, Y. and Bian, N. (2020) 'An Intrusion Detection Method Using Few-Shot Learning', *IEEE Access*, 8(1), pp. 49730–49740. doi: 10.1109/ACCESS.2020.2980136.

Zhang, X. *et al.* (2019) 'A Multiple-Layer Representation Learning Model for Network-Based Attack Detection', *IEEE Access*, 7, pp. 91992–92008. doi: 10.1109/ACCESS.2019.2927465.

Zhang, Y. *et al.* (2019) 'A novel ensemble method for k-nearest neighbor', *Pattern Recognition*, 85, pp. 13–25. doi: 10.1016/j.patcog.2018.08.003.