A Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree Model

Lukman Adebayo Ogundele[1], Femi Emmanuel Ayo[2], Adekunle Mathew Adeleye[3], Samuel Oluwatosin Hassan[4]
[1,2,4]Department of Computer Sciences, Olabisi Onabanjo University, Ago-Iwoye, 120107, Ogun State, Nigeria
[3]Department of Computer Science, National Open University of Nigeria, Ibadan Nigeria

Article Info

Article history:

Received: 28, 12, 2024
Revised: 10, 03, 2025
Accepted: 20, 03, 2025
Published: 30,03, 2025

Keywords:

Intrusion detection,
Feature selection,
Decision tree,
Neural network,
NSL-KDD dataset

ABSTRACT

Network Intrusion Detection System (NIDS) is a mechanism for detecting anomaly in computer networks. Several NIDS techniques have been developed in the past, but these techniques are still limited in detection accuracy, error rate and in detecting new attacks. In this study, a Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree model for NIDS was developed. The developed model is divided into four modules: Data collection, data preprocessing, feature selection and detection. The data collection module adapted the NSL-KDD dataset for implementation due to its modern attack representation. The data preprocessing module used the random undersampling technique to reduce data imbalance problem. The feature selection module consists of a hybrid feature selection method to select the most important features from the adapted intrusion dataset. The detection module involves a neural network-based decision tree classifier for the automatic generation of rules for intrusion detection. The results showed that the developed model based on the full dataset is better than the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.7, 1.3, 98.42%, 98.54%, 98.56% and 98.56 respectively. Similarly, the results showed that the developed method based on the reduced dataset is better than the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.9, 1.2, 99.42%, 99.54%, 99.56%, and 99.56% respectively.

Corresponding Author:

Femi Emmanuel Ayo
Department of Computer Sciences,
Olabisi Onabanjo University,
Ago-Iwoye, 120107,
Ogun State,
Nigeria.
Email:ayo.femi@oouagoiwoye.edu.ng

1. INTRODUCTION

The internet has developed into a natural phenomenon that is inextricably linked to people's everyday lives, and a significant amount of data needs to be secured from various illicit actions [1]. Concentrated network attacks are possible because attackers are always drawn to valuable information. According to [2], the main goal of data security is to develop and apply protective computer models that are invulnerable to usage, disclosure, interruption, alteration, and damage, as well as security breaches. Moreover, information security reduces actions that could compromise the three main security objectives of integrity, availability, and confidentiality [3]. Ensuring that the information sent is only accessible to those who should be in possession of it is the goal of confidentiality. To guard against unwanted access, the information is encrypted. Making sure that the data is not intercepted or manipulated is the focus of integrity. It guarantees that the sender's intended message is received exactly by the recipient. Availability guarantees that a network or system resource can be used and reached when needed by a system that has been given permission. A system's security is jeopardized when there is a breach. The numerous repercussions of these attacks have led researchers to create systems that can identify intrusions. The primary goal of creating intrusion detection systems (IDSs) is to distinguish between malicious and benign attacks. Intrusion detection systems can be categorized based on their environments [4] and their detection mechanisms [5]. Intrusion detection systems are further divided into two categories based on the environments in which they operate: host-based IDS (HIDS) and network-based

IDS (NIDS). The detection mechanism is further categorized into Anomaly-based Intrusion Detection Systems (AIDS) and Signature-based Intrusion Detection Systems (SIDS). Researchers are particularly focused on anomaly-based IDS since it has the advantage of being able to detect new patterns, whereas signature-based IDS has limitations in this regard [6, 7, 8, 9]. Notwithstanding their creation, a variety of intrusion detection systems remain open to attack [10]. In addition, despite the application of artificial intelligence and traditional machine learning algorithms to increase detection accuracy of intrusion detection systems, current intrusion detection systems continue to achieve poor performance [11, 12]. According to researchers, feature selection should be used in the pre-processing stage of machine learning algorithms to exclude irrelevant features, increase detection accuracy and reduce computational complexity. Feature selection can be categorized into filter, wrapper, and embedded methods. Akyol and Atila [13] suggested that feature selection strategy works well for developing and implementing intrusion detection systems with increased accuracy. The majority of anomaly detection systems used feature selection methods on the intrusion datasets to achieve higher accuracy and a lower false alarm rate [14]. The dataset is preprocessed to remove redundant features and noise, leaving a smaller set of features for the development of a high-performance model for the prediction of attack types with the base classifier.

This study however developed a Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree (HNIDF-NN-DT) model. The developed model is a neural network-based decision tree model that combine the high-level interpretability of decision tree with high decision-making accuracy of the neural network. The developed model is divided into data collection, data preprocessing, feature selection, and detection. The data collection module adapted the NSL-KDD dataset for implementation due to its modern attack representation. The data preprocessing module convert the collected dataset into the format suitable for machine learning algorithms. The data preprocessing module also used random undersampling method to reduce data class imbalance problem. The feature selection module consists of a hybrid feature selection method to select the most important features from the preprocessed dataset. The detection module involves a neural network-based decision tree classifier for the automatic generation of rules for intrusion detection. Each node of the decision tree represents a neural network processing unit. The node partitioning entropy function of the decision tree was replaced by the sigmoid function of the neural network. The selected features from the feature selection module serve as input into a neural network-based decision tree classifier to generate the rules for the automatic design of NIDS and accurate prediction of attacks. The decision tree provides the detection rules for the interpretability of detection decisions and the neural network provide the accuracy needed for attack detection. The contribution of this study include: (1) The use of a hybrid feature selection method for optimal feature selection. (2) The use of random under sampling method in the preprocessing phase to reduce dataset imbalance problem. (3) The use of a neural network-based decision tree model for network intrusion detection system to automatically generate the rules for intrusion detection. The neural network-based decision tree is for the decision tree to provide the detection rules for the interpretability of detection decisions and the neural network to provide the accuracy needed for attack detection. The rest of this study is organized as follows: Section 2 includes related work. The materials and method are presented in section 3. The results and discussion are presented in section 4. Section 5 conclude the work with a summary of the findings.

2. Literature review

2.1 Motivation of the work

The need to detect intrusions accurately grows daily as attackers keep getting sophisticated. The motivation of this study is focused on developing an intrusion detection system capable enough to detect new attacks and classify attacks accurately. Most of the past techniques for network intrusion detection cannot detect new attacks and are limited in terms of detection accuracy and false alarm rates. The research and implementation of advanced network intrusion detection techniques based on machine learning will contribute to the understanding of network security and anomaly detection.

2.2 Feature selection

One of the problems machine learning algorithms confront is high dimensionality. These days, it's typical to have datasets with a large number of fields or columns. Large datasets are typically employed in intrusion detection systems, and employing machine learning techniques to train the system on datasets with lots of features can take a lot of time, increase learning complexity, and have a negative effect on the system if irrelevant characteristics are used. Thus, feature extraction or selection aids in eliminating extraneous features or noise from the data, increasing the classification rate. The main distinction between feature selection and extraction is that the former creates new features or columns based on preexisting features, sometimes misinterpreting the data, while the latter selects features without altering the original data or creating new features or columns to be used. Researchers have confirmed that feature selection is a critical and significant phase in machine learning. Machine learning feature selection techniques are broadly divided into three categories:

2.2.1 Filter method

Using the filter method, features are subjected to statistic measurements. A feature is then either chosen or rejected based on the specified threshold. The Filter method's inability to take into account the base classifier—that is, its independence from the classifier—and propensity to choose sizable feature subsets are drawbacks. One advantage is its speed, as it doesn't necessitate communication with the classifier; another is its generality, as it assesses the characteristics of the input. Some of the filter techniques employed to the dataset for feature selection are Information Gain, Correlation-based, and Chi squared test.

2.2.2 Wrapper method

Wrapper method accounts for the classifier. Because many feature combinations are examined, it depends on the classifier and is typically computationally demanding. It assesses the accuracy of feature subsets. Because there is interaction between the wrapper and the classifier, it has the benefit of reaching a high accuracy. The lack of generality of the wrapper method is an additional drawback, in addition to its computational complexity and sluggish execution speed. The subset of features will only be unique to the classifier that is being evaluated.

2.2.3 Embedded method

Embedded method incorporates the advantages of both the wrapper and filter approaches by adding feature associations and drastically reducing computational costs. Each time the training process is repeated using iterative approaches, the features that contribute the most to the training for that iteration are carefully extracted. Many subsets are produced from the dataset by the embedded approach. It selects features for the model at random and tries to execute every possible combination. A subset of features will be selected and added to the dataset for training based on the subset with the highest accuracy.

2.3 Neural network

A neural network is a highly connected and adaptive deep learning model. The network is a hierarchical network structure made up of several vertices and interconnected edges. The vertices represent the processing unit and the interconnection of edges between vertices denote the association between connected vertices. At each iteration, the vertices broadcast faults backward as part of iterative learning, which increases the accuracy of the network. The network employed weight adjustment to lessen the learning error by constructing the vertices from attached weights. To control the network's output, a number of activation functions are also utilized.

2.4 Decision tree (DT)

Alpaydin [15] defines a decision tree as a “hierarchical data structure implementing the divide-and-conquer strategy”. It is a nonparametric technique that is suitable for both classification and regression problems. The model is tree-like in architecture which can easily be interpreted. It learns by performing feature selection, generating, and pruning trees. During training, the algorithm can select the most suitable features and build child nodes from the root node. Decision tree classifiers that have been used are ID4, C4.5, and CART, they have better accuracy for known intrusions but are not good at detecting unknown intrusions [16]. Ingre et al. [17] proposed a decision tree-based IDS for the NSL-KDD dataset. In the study, fourteen features were selected from the dataset using the correlated feature selection (CSF) method. The overall accuracy was 83.7% and 90.3%. Azad and Jha [18] also proposed an intrusion detection system based on a C4.5 decision tree on a KDD Cup 99 dataset and a high accuracy of 99.89% was achieved.

2.5 Related studies

Various researchers have proposed various methods for the detection of an intrusion based on data mining techniques. Khan et al. [19] developed a hybrid deep learning ID framework that combines a recurrent neural network (CRNN) and a convolutional neural network (CNN) to classify hostile attacks within the network. In the HCRNNIDS, temporal features were caught by the recurrent neural network (RNN) to improve the ID system's classification and accuracy, while native attributes were captured by the convolutional neural network (CNN). The CIC-DS 2018 ID data collection was used in studies. The hybrid intrusion detection system (HCRNNIDS) outperformed the ID approaches assessed by the researchers, with a detection rate of 97.7%, according to the results of its investigation. Thaseen et al. [20] developed an integrated ID system that prioritizes features based only on the strongest correlation between the class outcome and the features combined with Artificial Neural Network (ANN). This is achieved through the use of correlation-based feature selection. We conducted an experimental analysis on the UNSW-NB ID and NSL-KDD datasets. The model performed better in terms of sensitivity, accuracy, and specificity than several state-of-the-art methods. However, there were several disadvantages to the model's use of ANN. It took a long time and a lot of training data to complete the IDS training. Adding more layers improved the findings, but the model's memory and computational performance remained poor. Artificial neural network characteristics such as the number of hidden layers and neurons per layer enhanced system performance at the expense of increased temporal complexity. Salih and Abdulazeez [21] examined 17 current methods that have been put out by different scholars between 2018 and 2020. Selecting an algorithm has proven to be a challenging issue; researchers felt that evaluating the performance of several classifiers was the best approach, which prompted a review of the systems that were already in place. The review's conclusion is that feature selection improves the model's performance by reducing training and testing times through the removal of superfluous features. Hybrid classifiers may also offer the best option for attack detection. In the end, Practical Swarm Optimization (PSO) produced the best results for feature selection, whereas Random Forest obtained the best accuracy. Choudhury and Bhowal [22] examined the application of machine learning algorithms and a variety of categorization approaches to analyze network traffic. It was determined that J48, Random Tree, BayesNet, PART, Logistic, Random Forest, REPTree, IBK, and JRip were suitable for the classification method. More attention was paid to the machine learning techniques of bagging, boosting, and blending (or stacking), and analyses of their accuracy were conducted. These algorithms were compared using the WEKA tool, and the outcomes were displayed according to performance standards. Using the NSL-KDD dataset, cross validation at tenfolds was used to simulate the classification models. The researchers discovered that Random Forest and BayesNet worked well. In comparison to Bagging, Boosting, and Blending, Boosting performed significantly better. The researchers went on to say that the suggested algorithms might be utilized to create security-related network intrusion detection devices that operate well.

Mahfouz et al. [23] examined classifiers by taking into account three distinct dimensions: feature selection, hyper-parameter selection, and class imbalances. They only examined supervised machine learning techniques in their analysis. The SMO (SVM), Naive Bayes, J48 (DT), Multilayer Perceptron (NN), Logistic Regression, and IBK (KNN) methods were taken into consideration by the researchers. The models were constructed for various stages. Phase one involved using the chosen classifiers with their default settings and using the dataset without any preprocessing. By applying Stratified Cross-Validation with 10 folds on the training set of data from the NSL-KDD dataset, the classifiers were trained. Over-fitting is evident in all classifiers based on the accuracy of the training and test sets. At the second phase, data preprocessing was done. The selection of features was done using the InfoGainAttributeEval technique, and the hyperparameter optimization for each classifier was done using CVParameterSelection. Overfitting was found even though the second phase's accuracy was higher than the first phases. By undersampling the dominant classes and oversampling the minority classes, the third phase attempted to reduce the class imbalance in the dataset. In contrast to the first and second phases, the mitigation of imbalance classes helped lessen the barriers to the identification of R2L and U2R attacks.

Kajal and Nandal [4] used an artificial bee colony (ABC) in conjunction with a genetic algorithm and the Discrete Wavelet Transform (DWT) for feature selection. Their study's objective was to develop an intrusion detection system that could reliably identify attacks that were restricted to Denial of Service (DDOS) attacks by utilizing the KDD dataset. The hybridized Artificial Neural Network and Support Vector Machine (ANN-SVM) was used by the system as the foundation classifier. The suggested method was found to have outperformed the other two existing systems in terms of effectiveness when it came to detecting intrusions using the KDD dataset against denial-of-service attacks. In their research, [1] suggested using the random forest classifier and principal component analysis (PCA) to create a model for an effective intrusion detection system. Specifically, the study provided a technique for creating an effective IDS. Principal component analysis helped to extract features from the data by reducing its dimensions, while random forest helped to classify attacks using the Knowledge Discovery Dataset (KDD) into normal and intrusion categories. First, they used PCA to extract characteristics, followed by random forest to classify attacks, and lastly, model evaluation. The study by Kasongo and Sun [24] used a Wrapper based Feature Extraction Unit (WFEU) to implement a Feed-forward Deep Neural Network (FFDNN) wireless Intrusion Detection System. The Extra Trees technique was employed by the WFEU approach to extract optimal features. The UNSW and AWID datasets were used to evaluate the model's efficacy and efficiency. The system was compared to several different machine learning algorithms that are currently in use, such as Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Naive Bayes (NB), and Decision Tree (DT). Following feature extraction with the WFEU, input was utilized for various algorithms for binary and multi-class classification, including the FFDNN classifier. The WFEU-FFDNN technique outperformed the other approaches in terms of accuracy, and the results show that the recommended system performed very well when compared to previous efforts.

Zhou et al. [25] suggested the heuristic Correlation-based Feature Selection - Bat Algorithm (CFS-BA) in order to reduce the dimensionality of the dataset owing to the several redundant features in datasets. Additionally, the researchers presented a voting method that integrated the Random Forest's AO (average of probabilities) rule, Forest by Penalizing, and the C4.5 probability distributions. identifies methods as ensemble because a classifier might not be effective in identifying every possible combination of threats. The objective of the research is to develop an objective model that minimizes the amount of time and computational complexity needed to enhance the stability and dependability of the intrusion detection system. Evaluation and cross-validation were conducted on the NSL-KDD, AWID, and CIC-IDS datasets. The CFS-BA algorithm was implemented, and relevant features were obtained. These features demonstrate that the approach was successful in achieving the intended goal and drastically reducing dimensionality. For example, the NSL-KDD dataset saw a reduction of 41 features to 10, the AWID dataset saw a reduction of 155 features to 8, and the CIC-IDS saw a reduction of 78 features to 13 features. According to the analyses, comparisons, and findings, the CFS-BA-Ensemble approach has significant advantages for intrusion detection systems.

Halimaa and Sundarakantham [26] used 19,000 records from the NSL-KDD dataset to compare the prediction accuracy and misclassification rate of two prediction models, Naive Bayes and SVM, in three distinct scenarios. The models in the first scenario made predictions based on a dataset that was not pre-processed. In the second case, the models made predictions based on the characteristics obtained and the features were selected using CfsSubsetEval. In the third case, the researchers fed the models with the normalized data after normalizing the dataset for prediction. There were notable variations in each scenario's accuracy and misclassification rates. SVM outperformed Naive Bayes in every situation in terms of accuracy and misclassification rate.

Bhati and Rai [27] used the effectiveness of support vector machine (SVM) variants on the NSL-KDD dataset, including Fine Gaussian (98.7%), Medium Gaussian (98.5%), Quadratic (96.1%), and Linear (98.6%). IDS implementation consisted of four main steps: gathering data, preprocessing, splitting the data into train and test sets, and lastly evaluating the model using metrics for confusion matrix, accuracy, and Receiver Operating Characteristic (ROC). Based on the results, it was found that the Fine Gaussian SVM variation had the best intrusion detection accuracy and lowest error rate. Bindra and Sood [28] analyzed Random Forest ML, Linear SVM, K-Nearest Neighbor (KNN), Discriminant Analysis, Gaussian Naive Bayes, and Logistic Regression algorithms in order to identify DDoS attacks in networks. The CIC-IDS-2017 dataset was used to train and evaluate the methods that are compared in this research. With a 96.2% accuracy rating, Random Forest was the most accurate model. The study credits the algorithm's usage of cross-validation for the successful outcome. The tests also highlight the need to reduce the dataset's dimensionality; the Select Percentile technique was used to reduce the features from 85 to 12, and the median of an attribute was used in place of nan, which is not a number.

Chu et al. [29] examined the outcomes of machine learning (ML) methods on the NSL-KDD dataset. These methods included the use of SVM, Nave Bayes, Decision Trees, and Artificial Neural Networks (ANN) with MLP to identify Probe, DoS, Remote to Local, and User to Root assaults. The precision of SVM was 97.72 percent; that of ANN was 97.82 percent; that of Nave Bayes was 90 percent; and that of the J48 was 59.3 percent. The researchers found that adjusting the gamma and c parameters allowed SVM to achieve high accuracy, but adding four layers to the Artificial Neural Network produced the highest accuracy. Although the two algorithms that performed better did not show a significant difference in their results, the researchers found that when they applied principal component analysis to minimize the feature space of the data on the NSL-KDD dataset, the speed of classification increased significantly. Kim et al. [30] built an Intrusion Detection System model using a convolutional neural network (CNN) and assessed it by contrasting it with a model created using a recurrent neural network (RNN) using the CSE-CIC-IDS 2018 dataset. The data underwent preprocessing, and features were selected. The dataset must be converted into pictures for a CNN. A model consists of a fully linked layer, max-pooling layers, and convolutional layers. The model was implemented by deploying maxpooling behind each convolutional layer. The max pooling layer was necessary even though a CNN model does not need it since there is very little chance of losing significant features from the max pooling because the modified images solely include quantitative data and do not contain any unseen signatures. Moreover, the activation function'relu' was used for every convolutional layer. Every stage of the max pooling process involves lead dropout in an attempt to reduce overfitting. Lastly, a fully connected layer is positioned underneath the final max-pooling layer. Applying the CNN model to the CIC-2018 dataset showed that it performed better in label classification than the RNN model. Additionally, one method suggested to enhance the model's performance was to prepare the dataset with a proportion of data categorized as benign and data labeled as attacks. In their study, Patgiri et al. [31] used Random Forest and Support Vector Machine to examine the use of machine learning for intrusion detection. Recursive feature elimination (RFE) was used to filter relevant features for both classifiers in an effort to reduce computing time. The models created were evaluated using the NSL-KDD dataset. The researchers found that the substantial experimentation they conducted was both time-consuming and negatively impacted performance. Before the Recursive Feature Elimination approach was used, Random Forest's performance was superior to SVM's. Conversely, after applying Recursive feature Elimination Cross Validation (RFECV) with the classifiers, SVM outperformed Random Forest in terms of performance.

Taher et al. [32] used the NSL-KDD dataset to assess the effectiveness of SVM and artificial neural networks. Their goal was to identify the classifier that had the highest accuracy and success rate. A wrapper method was used to choose the features for the proposed models, resulting in a reduction of 35 features to 17. The study also sought to determine the ideal learning rate and number of hidden layers; the findings indicate that three hidden layers and a learning rate of 0.1 were optimal. The two classifiers were compared both before and after feature selection was used in the evaluation process. When feature selection was used in either scenario, the detection accuracy was higher; and artificial neural network performed better in both instances. Xu et al. [33] presented an anomaly-based intrusion detection system, an optimization technique, a weighted KNN, and the elimination of feature selection in an effort to enhance network intrusion detection performance. The KDDCup intrusion detection dataset was used to evaluate the suggested IDS, and the results show that weighted KNN increased efficiency at the expense of a slight accuracy loss. The researchers found that Naïve Bayes and SVM can also be utilized for this purpose, even though only KNN was tested, and they stated that they expect to apply SVM and Naive Bayes parameter optimization in the future.

Yulianto et al. [34] addressed the imbalance of training data (CIC-IDS 2017 Monday-Working-Hours-DDoS-Attack dataset) by improving the performance of an AdaBoost-based Intrusion Detection System that was already in place [35]. Significant attributes were selected using Principal Component Analysis (PCA), Ensemble Feature Selection (glm, gbm, treebag, ridge and lasso), and Synthetic Minority Oversampling Technique (SMOTE). The AdaBoost classifier was used for the label classification. First, the 225,745 records of data were cleaned and scaled in accordance with the suggested methodology. The dataset was then divided into two subsets: training, which comprised 70 percent of the data (158,022 records), and testing, which made up the remaining 30 percent (67,723 records). SMOTE was employed to oversample the unbalanced classes, and 25 features were selected using the Ensemble technique. Five iterations of cross-validation were carried out after the model was built using AdaBoost. Ultimately, the precision, recall, and accuracy criteria were used to evaluate the efficiency of model. The results showed that the suggested technique performed better than [35]. Haripriya and Jabba [36] examined a number of machine learning algorithms that have been proposed by prior research to find the best methods for intrusion detection system. Single, hybrid, and ensemble classifiers were among the methods employed in the review of previous research. As a result, no specific way could be chosen for the development of the Intrusion Detection System, according to the researchers, who discovered that every algorithm has unique significance and contributions when compared to other approaches. Additionally, when a particular quantity of traffic data is unavailable, the researchers found that it can be challenging to train algorithms.

The work conducted by Ashraf et al. [37] employed J48, Random Forest, and Naïve Bayes classifiers to calculate the detection rate and accuracy of IDSs. The experiments made use of the NSL KDD dataset. The study examined the classification efficacy of J48, Random Forest, and Naïve Bayes on the 20% NSL KDD dataset. The results showed that Random Forest performed better than Naïve Bayes in terms of accuracy and detection rate. Given that all three classifiers were able to reach 90% recall and precision, a hybrid model that combines the three might be suggested in the future. Chiba et al. [38] proposed an excellent method for creating a Back Propagation Neural Network (BPNN)-based NIDS using KDD CUP' 99 datasets. Models totaling forty-eight were developed by combining implementation parameters in different ways. After evaluation, two IDSs were determined to be the best based on false positive rate, detection rate, F-score, and AUC. the count of attributes, normalization, the architecture of the neural network—particularly the nodes to be used in the layers—the momentum term, learning rate, and transfer function were the parameters that were found to be the most important to be used in the construction of the classifier. In the third stage, the combinations of various parameters were generated. In stage four, IDS was put into practice. In the fifth and final stage, the study selected two models for comparison based on their respective levels of efficiency. Future research will employ an optimized algorithm that looks for the best arguments to affect the model's performance.

Hajisalem and Babaie [39] proposed a novel hybrid classification technique based on artificial fish swarms (AFS) and artificial bee colonies (ABC). The implemented methodology had the following structure: dividing up the training datasets, choosing features, creating rules, and using hybrid classification. Based on the planned ABC-AFS, the framework was created. Fuzzy C-Means Clustering (FCM) and Correlation-based Feature Selection (CFS) methods were applied to divide the training data and remove redundant features. In addition to the CART technique, If-Then rules built based on the chosen attributes were used in attempts to differentiate between normal and abnormality records. The hybrid approach that was described was trained using the rules that were developed. According to performance measures, the suggested approach obtained a 99 percent detection rate and a 0.01 percent false positive rate using the UNSW-NB15 and NSL-KDD datasets. Furthermore, a time and computational cost comparison showed that the model's overhead is equal to that of competing options.

Al-yaseen et al. [40] presented a system framework built on a multi-level hybrid support vector machine and extreme learning. The modified K-means was used to reduce the dataset size and obtain smaller samples of the dataset, or 10% of the KDD data, which results in tiny quality data, according to the researchers, who claimed that SVM takes a lot of training time. There was one ELM classifier and four SVM classifiers used. The findings demonstrated that they performed better than cutting-edge techniques and that there were no significant variations in the detection performance. Additionally, the results show that the smaller dataset contributed to improved accuracy and faster training times. However, using many classifiers led to an extended testing period. Lin et al. [41] proposed a KNN model to reduce the feature characterisation of the dataset to one dimension, based on Cluster Center and Nearest Neighbors (CANN). The recently obtained dataset was employed to assess the foundational classifier. The first dimension of the dataset is where the CANN classifier performs better than the KNN and SVM classifiers, according to the results. The frequency of false alarms was reduced and the accuracy of identification was higher with the CANN. KNN and SVM were shown to be computationally costly when tested on two datasets, however CANN was found to be computationally light. Regarding the limitations of the study, CANN failed to identify R2L and U2R attacks since the one-dimensional representation could not accurately reflect them. Moreover, CANN required extra computations to eliminate the separation-based features.

Maharaj and Khanna [42] worked on Voting Feature Interval (VFI) and Algorithm Reptree. In light of the fact that certain classification algorithms forbid the prediction of more than two classes, the researchers decided to focus their study's conclusions on developing a classifier capable of multiclass classification. Most current systems, they claim, only identify whether an attack has occurred or not and do not provide specific information about the type of attack. The KDDCUP dataset was evaluated utilizing the Receiving Operating Characteristic (ROC) curve, which provided information on the Area Under Curve (AUC), False Positive Rate (FPR), and True Positive Rate (TPR). The analysis that followed led to the conclusion that the REPTree learning algorithm is superior and more effective for intrusion detection systems. The goal of the Ghosh et al. [43] study was to enhance the performance of the classifier for detecting intrusion by utilizing hybridized K-Nearest Neighbors and Neural Networks (KNN-NN) and multilevel feature selection. Before the classification, four steps were required: first, the NSL-KDD data set used for the experimental analysis was preprocessed;

second, the Rough Set Theory (RST) wrapper method was chosen for the first feature selection; and third, normalization and discretization were completed before using the RST. In step three, the second level feature selection procedure was carried out using the Information Gain (IG) filter approach. Following completion of all of these, the classification process began. It was carried out in two stages, with KNN being applied initially and the output being used as an input for neural networks.

Jongsuebsuk et al. [44] developed an IDS for classifying network attack data using a fuzzy rule algorithm and a genetic algorithm on the KDD99 dataset in addition to the researchers' own dataset. The results showed that the developed IDS can accurately identify network threats in real time with low false alarm rate and high detection speed. Patel et al. [3] conducted a review of data mining approaches, including K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), Support Vector Machine (SVM), Decision Tree (DT), and Naive Bayes (NB). The researchers went on to discuss the benefits and drawbacks of the algorithms. Although the dataset statistics for this study were not disclosed, all of the implemented algorithms performed poorly. However, the evaluation results led the researchers to the conclusion that combining multiple algorithms could potentially mitigate their disadvantages because different algorithms have varying perspectives on the situation and improve system performance**.** Vinayakumar et al. [45] proposed a robust intelligent malware detection using deep learning. First, the study uses several datasets to compare the effectiveness of deep learning and standard machine learning architectures for malware detection. Second, the datasets used to train and evaluate the machine learning model are free of any bias thanks to the study. Thirdly, the research suggests a brand-new method of picture processing that provides the best settings for deep learning architectures and machine learning algorithms to produce an efficient zero-day malware detection model. The suggested deep learning architectures outperform traditional machine learning methods, according to a thorough comparison analysis of their model. In a large data setting, the study's primary contribution is the integration of deep learning and visualization architectures enabling hybrid approaches based on image processing, static, and dynamic processes.

Mahindru and Sangal [46] proposes a framework for android malware detection using machine learning techniques. The proposed framework uses its dynamic analysis to find viruses in Android apps. The study trained the suggested framework by choosing features obtained by applying feature selection algorithms in order to detect malware from real-world apps. A model was constructed using the chosen features while taking various machine learning algorithms into account. More than 500,000 Android apps were subjected to experimental investigation in this study. The outcome showed that the suggested model outperformed similar techniques for malware detection on practical applications. Baek et al. [47] proposed a two-stage hybrid malware detection using deep learning in internet of things environment. In the first step, opcode is retrieved via static analysis, and harmful files are identified using a bidirectional long short-term memory model on the information that has been extracted. A dynamic analysis of files categorized as innocuous in a nested virtual environment is part of the second stage. Convolutional neural networks were utilized to detect malwars after information on behavior and process memory was extracted from the behavior log based on system modifications.

Bakır and Bakır [48] proposes a malware detection using auto-encoder based feature extractor and machine learning algorithms. The study suggests using a brand-new autoencoder-based model called DroidEncoder to categorize Android malware apps. An image-based Android app dataset with 3000 malicious and 3000 benign apps was employed in the study. Additionally, three distinct auto-encoders—ANN-based, CNN-based, and VGG19-based—were used in the study to extract features from the malware dataset that was displayed. Three separate feature extraction experiments were carried out in the study in order to train several machines learning algorithms, including support vector machines, k-nearest neighbors, decision trees, additional trees, LightGBM, XGBoost, Random Forest, and linear regression. The outcomes demonstrated that the suggested technique for malware identification performed better than other comparable machine learning techniques. Khan et al. [49] proposes a Digital DNA Sequencing engine for ransomware detection using machine learning. The study evaluated the suggested approach's effectiveness using 582 ransomware and 942 benign cases, evaluating performance according to the terms of accuracy, recall, f-measure, and precision. The findings demonstrated that the suggested strategy outperformed alternative techniques for ransomware detection.

Ciaramella et al. [50] proposed an explainable ransomware detection with deep learning techniques. An openly accessible dataset was used to evaluate the proposed deep learning model for ransomware detection. When the suggested model was compared to other comparable techniques, the findings demonstrated improved accuracy for malware identification during the training and test phases across a dataset of more than 15,000 components. Additionally, the research took advantage of the Gradient-weighted Class Activation Mapping in the created technique to raise the malware detection classification accuracy. Almazroi and Ayub [51] proposes a deep learning hybridization for improved malware detection in smart Internet of Things (IoT) environment. The study presented a unique Feed Forward Neural Network Framework (BEFNet) for Internet of Things scenarios, based on BERT. Eight datasets, each representing a distinct malware type, were used to evaluate the proposed system using different modules. Additionally, the Spotted Hyena Optimizer (SO) was used to optimize the suggested technique, demonstrating its flexibility in handling various malware data shapes. In comparison to similar machine learning techniques, the comparative evaluation findings highlight the superior performance of the suggested strategy for malware detection in IoT environments.

Brown et al. [52] proposes an automated machine learning (AutoML) for deep learning-based malware detection. The research offered a thorough examination and practical advice on applying AutoML to the identification of online and static malware. In the case of static, the suggested approach was examined using two popular malware datasets: SOREL-20M, which was used to show the method's effectiveness on big datasets, and EMBER-2018, a smaller dataset that was deliberately chosen to impede the performance of machine learning models. The study also demonstrated how adjusting the parameters of the neural architecture search (NAS) method can lead to the identification of a malware detection model that is more ideal for use with these static analysis datasets. Additionally, the study showed that AutoML performs well when used with Convolutional Neural Networks (CNNs) for cloud services in cases including online virus detection. The study used a newly produced online malware dataset, with and without other programs running in the background during malware execution, to compare the proposed AutoML technique to six current state-of-the-art CNNs. The outcomes shown that the suggested AutoML technique outperformed the most advanced CNNs with minimal overhead in terms of architecture discovery. Overall, the test findings demonstrated that AutoML-based static and online malware detection models perform as well as or better than cutting-edge malware detection models.

Talukder et al. [53] suggested a hybrid ML model for NIDS. The goal of the suggested approach is based on how current IDS fall short in addressing difficulties with low detection accuracy and processing enormous amounts of data for attack detection. As a result, the study suggested a hybrid model that combines ML and DL techniques in order to create a trustworthy and accurate IDS. Five stages make up the suggested paradigm. The first stage handled missing values and encoded attribute values as part of data preprocessing. SMOTE was used in the second step to solve the problem of data imbalance in the datasets. The most pertinent features from the datasets were extracted in the third stage using the XGBoost algorithm. The processed dataset is split into a training and testing set in the fourth stage using K-fold cross validation. After choosing the best attributes, ML and DL algorithms were used to create the models. The effectiveness of other ML algorithms, including RF, DT, KNN, MLP, Convolution Neural Network (CNN), and Artificial Neural Network (ANN), for detecting network intrusion was compared with the performance of the created ML and DL techniques. When compared to existing models, the suggested strategy provides excellent detection accuracy on two datasets without any overfitting, according to the results. The proposed technique was not tested on modern datasets and does not provide a structured taxonomy for intrusion classification. Table 1 show the highlights of related studies.

Table 1. Highlights of related studies

S/N	Author/Year	Technique	Advantage	Disadvantage
1.	Khan et al. [19]	Recurrent Neural Network and a Convolutional Neural Network	Provides excellent detection accuracy	Long training time
2.	Thaseen et al. [20]	Correlation-based Artificial Neural Network	Better detection accuracy	Long training time and require a lot of training data
3.	Salih and Abdulazeez [21]	Particle Swarm Optimization and Random Forest	Detection speed and accurate	Slow and not suitable for real-time detections
4.	Choudhury and Bhowal [22]	Random Forest and BayesNet	Better classification performances	Long training time
5.	Mahfouz et al. [23]	Machine learning techniques	Reduce class imbalance and increase classification rate	Overfitting problem
6.	Kajal and Nandal [4]	Artificial Neural Network and Support Vector Machine (ANN-SVM)	High precision value	Long training time
7.	Waskle et al. [1]	Random forest classifier and principal component analysis	Reduced feature dimension and accurate classification	It is not robust against data imbalance problem
8.	Kasongo and Sun [24]	Wrapper-based Feed-forward Deep Neural Network	Better detection accuracy	High training time and overfitting problem
9.	Zhou et al. [25]	Correlation-based Feature Selection- Bat Algorithm (CFS-BA) and Ensemble method	Minimizes the amount of processing time and data imbalance problem	Inability to detect rare network attacks
10.	Halimaa and Sundarakantham [26]	Naive Bayes and Support Vector Machine (SVM)	Better detection accuracy	It is not suitable for large datasets and inability to detect rare network attacks
11.	Bhati and Rai [27]	Fine Gaussian SVM variation	Better detection accuracy and low error rate	Long training time
12.	Bindra and Sood [28]	Random Forest	Provides high detection accuracy	Slow and ineffective for real-time predictions
13.	Chu et al. [29]	SVM	High detection speed	Slight accuracy loss based on their feature selection method
14.	Kim et al. [30]	Convolutional Neural Network (CNN)	Better classification rate	Data imbalance problem

15.	Patgiri et al. [31]	Random Forest and Support Vector Machine	Provides high detection accuracy	High time and computational cost
16.	Taher et al. [32]	SVM and artificial neural networks	Provides high detection accuracy	High time and computational cost
17.	Xu et al. [33]	Weighted KNN	Increased efficiency	Slight accuracy loss
18.	Yulianto et al. [34]	AdaBoost classifier	Provides better detection accuracy	It is not suitable for noisy data and sensitive to outliers
19.	Haripriya and Jabba [36]	Machine learning algorithms	Provide a review contribution to the field of intrusion detection systems	It is very difficult to train the algorithms when certain amount of traffic data is not available
20.	Ashraf et al. [37]	Random Forest Algorithm	Provides good detection accuracy	Slow and ineffective for real-time predictions
21.	Chiba et al. [38]	Back Propagation Neural Network	Provides excellent detection accuracy	High time and computational cost
22.	Hajisalem and Babaie [39]	Artificial fish swarms (AFS) and artificial bee colonies (ABC)	High detection rate and low false positive rate	High time and computational cost
23.	Al-yaseen et al. [40]	Support vector machine and Extreme learning	Improved accuracy and faster training times	Extended testing period
24.	Lin et al. [41]	Cluster center and nearest neighbor (CANN) approach	Effective detection rate and false alarm	Required extra computations. It failed to identify R2L and U2R attacks
25.	Maharaj and Khanna [42]	Data mining approach	Effective performance and low false alarm rates	High training time
26.	Ghosh et al. [43]	K-Nearest Neighbors and Neural Networks (KNN-NN)	Good classification result	High computation cost and requires high memory storage
27.	Jongsuebsuk et al. [44]	Fuzzy Genetic Algorithm	Low false alarm rate and high detection	Time-consuming
28.	Patel et al. [3]	Data mining approach	Provide review of data mining approaches	Poor performances of implemented methods and dataset statistics were not disclosed
29.	Vinayakumar et al. [45]	Deep learning	High detection accuracy	Lack of Interpretability
30.	Mahindru and Sangal [46]	Machine learning techniques	High detection rate	Model stability issue
31.	Baek et al. [47]	Hybrid deep learning	Provides excellent detection accuracy	Require a lot of training data for effective learning
32.	Bakır and Bakır [48]	Auto-encoder based feature extractor	Provides excellent detection accuracy	Prone to overfitting
33.	Khan et al. [49]	Active Learning Algorithm	Good detection performance	It cannot deal with limited annotated data
34.	Ciaramella et al. [50]	Deep learning techniques	Good computational efficiency	Lack of feature selection method
35.	Almazroi and Ayub [51]	BERT-based Feed Forward Neural Network	High detection accuracy	High computational time
36.	Brown et al. [52]	Convolutional Neural-Networks	High detection accuracy	High computational time
37.	Talukder et al. [53]	Hybrid ML method	Provides excellent detection accuracy	It was not tested on modern datasets

3. Materials and method

In this study, a Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree (HNIDF-NN-DT) model was developed. The developed model is divided into four modules: Data collection, data preprocessing, feature selection and detection. The data collection module adapted the NSL-KDD dataset for implementation due to its modern attack representation. The data preprocessing module convert the adapted dataset into the format suitable for machine learning. The data preprocessing module also used the random undersampling technique to reduce data imbalance problem. The feature selection module consists of a hybrid feature selection method. A hybrid feature selection method was used to select the most important features from the adapted intrusion dataset. The first step of the hybrid feature selection method utilized forward selection to choose feature subsets. The forward selection start from an empty set, and sequentially add features from the full feature set with an already selected features that result in the highest accuracy of the classifier. The second step used backward elimination method to select feature subsets. The backward elimination starts from the full set, and sequentially remove the feature that results in the smallest decrease in the accuracy of the classifier. Two intermediate feature subsets are formed from the first and second steps with least number of features. The third step combines the two intermediate feature subsets into a pool and used Genetic search method to select the best features from the pool of feature subset. The genetic search algorithm evaluates the merits of each attribute and returns selected features with highest fitness value. The rule evaluation phase of the third step check if two feature subsets have the same fitness value, the rule-based engine returns the feature subset with least number of subset features. The detection module involves a neural network-based decision tree classifier for the automatic generation of rules for intrusion detection. The selected features from the feature selection module serve as input into a neural network-based decision tree classifier to generate the rules for the automatic design of NIDS. Each node of the decision tree represents a neural network processing unit. The node partitioning entropy function of the decision tree was replaced by the sigmoid function of the neural network. The selected features from the feature selection module serve as input into a neural network-based decision tree classifier to generate the rules for the automatic design of NIDS and accurate prediction of attacks. The decision tree provides the detection rules for the interpretability of detection decisions and the neural network provide the accuracy needed for attack detection. Figure 1 shows the architecture of the developed Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree model.

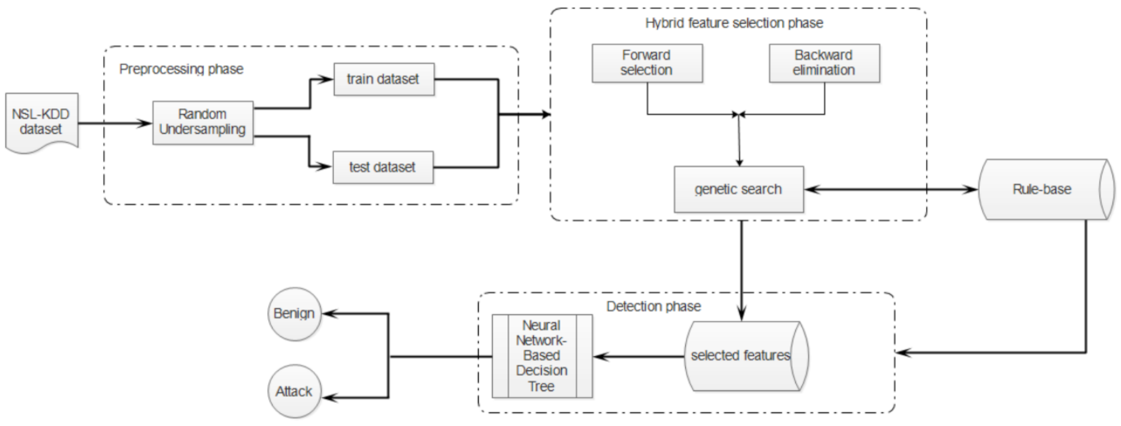


Figure 1: Architecture of the developed architecture of the developed Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree mode

3.1. Dataset collection module

The NSL-KDD dataset was adopted for implementation due to its effectiveness for modern intrusion detection system [54]. The dataset consists of 41 attributes that are classified as either normal or anomalous. Table 2 shows the attack categories in the dataset.

Table 2. List of Attack Category

Attack type	Category
Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Apache2, Processtable, Udpstorm, Worm	DoS
Ipsweep, Nmap, Portsweep, Satan, Mscan , Saint	Probing
Ftp_Write, Guess_Passwd, Imap, Multihop, Phf, Spy, Warezclient, Warezmaster, Warezmaster, Sendmail, Named, Snmpgetattack, Snmpguess, Xlock, Xsnoop, Httptunnel	R2L
Buffer_Overflow, Loadmodule, Perl, Rootkit, Ps, Sqlattack, Xterm	U2R

3.2. Data preprocessing module

The data preprocessing module also used the random undersampling technique to reduce data imbalance problem. Random Under-sampling was used to shift the class distribution of the dataset in order to prevent model bias. Using the random undersampling technique, instances from the majority class are chosen at random and then removed from the training dataset. Additionally, datasets with an uneven distribution of observations for the target class—that is, one class label having a very high number of observations while the other has a very low number of observations—are referred to as imbalanced data. Random under-sampling technique is used regularize the minority or majority class. In order to match the majority class with the minority class, rows from the majority class can be randomly removed using this technique. A balanced dataset for both majority and minority classes can be obtained after sampling the data. Given that both classes have an equivalent number of records in the dataset, it follows that the classifier will assign equal weight to each class.

3.3 Feature selection

In order to choose the most important characteristics from the preprocessed dataset, the feature selection module employed a hybrid feature selection technique:

- Forward selection**:** Beginning with an empty set (Equation 1), sequentially adds features from the whole feature set X^+ that results in the greatest accuracy J(Y_k+ X^+) when combined with the features Y_k that have already been chosen (Equation 2).

$$Y_o = \{\emptyset\} \tag{1}$$

$$X^{+} = \underset{X \notin Y_k}{\operatorname{argmax}}[J(Y_k + X)] \tag{2}$$

- Backward elimination: Beginning with the full set (Equation 3), sequentially eliminates the characteristic X^{-} that causes the accuracy classifier to decrease the least J$(Y_k - X)$ as in Equation 4.

$$Y_o = X \tag{3}$$

$$X^{-} = \underset{X \notin Y_k}{\operatorname{argmax}}[J(Y_k - X)] \tag{4}$$

- Genetic search**:** Genetic search is a search motivated by natural evolution. This genetic search employs a fitness function, which is a linear combination of an accuracy term and a simplicity term.

$$Fitness(X) = \frac{3}{4}A + \frac{1}{4}\left(1 - \frac{S+F}{2}\right) \tag{5}$$

  where X is a feature subset, A is the average cross-validation accuracy of the classifier, S is the number of instances or training samples, and F is the number of subset features.

- Rule evaluation: The rule-based engine returns a feature subset (V_i) with fewer features (X_F) if there are many feature subsets $(F_{>})$ with equivalent fitness values; otherwise, it returns the feature subset with the greatest fitness value (F_{hi}) to the basic classifier as in (6).

$$R = \begin{cases} V_i, if\ V_i \in F_{>} \cap X_F \\ V_i, if F_{hi} \cap \emptyset \end{cases} \tag{6}$$

3.4 Detection

The detection module involves a neural network-based decision tree classifier for the automatic generation of rules for intrusion detection. The selected features from the feature selection module serve as input into a neural network-based decision tree classifier to generate the rules for the automatic design of NIDS. Each node of the decision tree represents a neural network processing unit. The node partitioning entropy function of the decision tree was replaced by the sigmoid function of the neural network. The selected features from the feature selection module serve as input into a neural network-based decision tree classifier to generate the rules for the automatic design of NIDS and accurate prediction of attacks. The neural network-based decision tree provides the detection rules for the interpretability of detection decisions and the neural network provide the accuracy needed for attack detection. The neural network-based decision tree was applied to the chosen features in order to determine the rules for the automatic creation of network intrusion detection system. The neural network-based decision tree technique was used to extract the rules that separate normal traffic from intrusive traffic using the adapted dataset. A neural network-based decision tree can be expressed as a recursive partition of the instance space. In the developed neural network-based decision tree, each internal node represents a neural network processing unit. The node partitioning entropy function of the decision tree was replaced by the sigmoid function of the neural network to split the instance space into two or more sub-spaces according to the sigmoid function of the input attributes values. The measure for selecting the best split based on the degree to which instances belong more to one class than the others is given by Equation 7.

$$Entropy\ (t) = -\sum_{i=0}^{c-1} p(i|t)\ log_2\, p(i|t) \tag{7}$$

where:

c is the number of classes while $p(i|t)$ denotes the fraction of instances belonging to class *i* at a given node *t.*

In order to represent each node of the decision tree as a neural network processing unit, the following procedures are defined: First, there may be several inputs x_i, i = 1, …, m in the nodes of the decision tree. Each input x_i is multiplied by the corresponding weight w_{ki} where k is the index of a given neuron in a neural network. The weighted sum of products $x_i w_{ki}$, for i = 1, …, m is usually denoted as *net* in the neural network literature:

$$net_k = x_i w_{ki} + b_k \tag{8}$$

Finally, a neural network computes the output y_k as a certain function of net_k value as in (9) and the modified entropy function for selecting the best split based on the degree to which instances belong more to one class than the others using the sigmoid function is given by Equation 10.

$$y_k = \frac{1}{(1+e^{-net})} \tag{9}$$

$$Entropy\ (t) = -\sum_{i=1}^{c} 1/(1+e^{-net})\ log_2[1/(1+e^{-net})] \tag{10}$$

where: $1/(1+e^{-net})$ denotes the probability of randomly selecting an instance in class i. The splitting criterion used by the J48 decision tree algorithm is known as gain ratio to determine the goodness of a split. The criterion is defined as in Equation 11.

$$Gain\ ratio\ (D, A) = Entropy\ (D)\ - \sum_{v\,\in\,Values(A)} \frac{|D_v|}{|D|}\, Entropy\ (D_v) \tag{11}$$

where:$Values(A)$ denote the set of all possible values for attribute A and D_v denote the subset of dataset D having value v for attribute A.

3.5 Algorithms

Algorithm 1 depict the hybrid feature selection method. The hybrid feature selection technique uses Forward Selection, Backward Elimination, Genetic Search, and Rule Evaluation techniques. A hybrid feature selection technique was applied to the preprocessed dataset to determine which features were most crucial. The first step of the hybrid feature selection technique employed forward selection to choose feature subsets, while the second step used backward elimination. The forward selection process begins with an empty set and sequentially adds features from the whole feature set with features that have already been chosen to provide the classifier's greatest accuracy. Backward elimination starts with the entire set and sequentially eliminates the characteristic that causes the accuracy classifier to decrease the least. The first and the second steps produced two intermediate feature subsets with least number of features. In the third step, the two intermediate feature subsets are combined into a pool, and the best features are chosen from the pool of feature subsets using the genetic search approach. The genetic search algorithm evaluates the performance of each attribute and returns selected features with highest performance. The rule evaluation phase was used to return the feature subset with least number of subset features if there exist two feature subsets that have the same fitness value. Genetic search method was used to search the final best features from the pool of feature subsets.

ALGORITHM 1: Hybrid feature selection

```
INPUT: Training Dataset X{$x_1, x_2, \ldots, x_k$ | xi ∈ C}
OUTPUT: $X_r$ //reduced features
Process:
   1.  BEGIN
   2.  $Y \leftarrow X(x_1, x_2, \ldots, x_k)$
   3.  $Y_o = \{\emptyset\}$ //forward selection
   4.  $X^+ = \underset{X \notin Y_k}{\text{argmax}}[J(Y_k + X)]$
   5.  $Y_{k+1} = Y_k + X^+, k = k + 1$ //update
   6.  $Y_o = X$ // Backward elimination
   7.  $X^- = \underset{X \notin Y_k}{\text{argmax}}[J(Y_k - X)]$
   8.  $Y_{k+1} = Y_k - X^-; k = k + 1$ //update
   9.  P = $X^+ + X^-$ // feature pool
   10. BEGIN
   11.     Generate random population n from P
   12.     Np ← m x n
   13.     WHILE Np is not empty do
   14.         P1 ← TournamentSelection (Np)
   15.         P2 ← TournamentSelection (Np - P1)
   16.         Select a random number r ∃ $0 \geq r < 1$
   17.         IF ( r < p) do // if r is less than the crossover rate
   18.             crossover(P1, P2)
   19.          ELSE return P1 , P2
   20.             mutation(P1 , P2)
   21.             k ← P1 , P2
   22.             Return k ∈ P for which f(x) is highest.
   23.         END-IF
   24.      END WHILE
   25.      IF two feature subsets have the same fitness value
   26.         //return the feature subset with least number of subset features
   27.         return $X_r$
   28.      END-IF
   29. END
```

Algorithm 2 depict the algorithm for finding the best split attribute of the decision tree using the sigmoid based entropy function. The algorithm started by initializing weights for all the decision tree nodes. The algorithm checks if all instances in the dataset X have same class c and label the decision tree accordingly. The algorithm then checks if no attribute has positive entropy based on the sigmoid function, and assign the decision tree the most common class in the dataset. The algorithm checks for attribute with the highest entropy based on the sigmoid function and label the decision tree with such attribute and return the resultant tree.

ALGORITHM 2: find_best_split

```
INPUT: Dataset X, Feature F
OUTPUT: Tree T
Process
    1.  BEGIN
    2.  $Entropy\ (t) = -\sum_{i=0}^{c-1} p(i|t)\ log_2\, p(i|t)$
    3.  $net_k = x_i w_{ki} + b_k$
    4.  $y_k = \frac{1}{(1+e^{-net})}$
    5.  $Entropy\ (t) = -\sum_{i=1}^{c} 1/(1+e^{-net})\ log_2[1/(1+e^{-net})]$
    6.  $Gain\ ratio\ (D,A) = Entropy\ (D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy\ (D_v)$
    7.  $W = w_0$
    8.  For $F = 0$ to $F - 2$ do
    9.      if all instances in X have same class c
    10.         Label(T) =c; Return T
    11.     If F=∅ or no attribute has positive entropy
    12.         Label (T) = most common class in X; return T
    13.     F←attribute with highest entropy
    14.     Label(T) = F
    15. End For
    16. For each f of F
    17.     $X_f$ ←instances in X with F=f
    18.     If $X_f$ is empty
    19.         Let $T_f$ be a new tree
    20.         Label ($T_f$)=most common class in X
    21.     Else
    22.         $T_f$=($X_f$, F-{F})
    23.     Add a branch from T to $T_f$ with label f
    24.     Return T
    25. End For each
End
```

Algorithm 3 depict the Neural Network-Based Decision Tree (NN-DT). This algorithm builds the J48 decision tree based on neural network. The J48 decision tree builds a *tree of rules* during the training phase to appropriately separate the network traffics into normal or attack classes. During testing phase, the *tree of rules* is generated for the automatic construction of rules to classify the input features as normal or attack. The algorithm works by recursively selecting the best attribute to split the data (Step 9) and expanding the leaf nodes of the tree (Step 13 and 14) until the stopping criteria is met (Step 3). The *createNode*() function extends the decision tree by creating a new node. A node in the decision tree has either a test condition, denoted as *node.test_cond*, or a class label, denoted as *node.label*. The *Classify* () function (Step 5) determines the class label to be assigned to the leaf node. For each leaf node *t*, let *p(i|t)* denote the fraction of training data from class *i* associated with the node *t*. the leaf node is assigned to the class that has the majority number of training data. The *find_best_split*() function determines which attribute should be selected as the test condition for splitting the training records. The *stopping_cond*() function is used to terminate the tree-growing process by testing whether all the instances have either the same class label or the same attribute values.

ALGORITHM 3: Neural Network-Based Decision Tree (NN-DT)

```
INPUT: Attribute_set F{ $f_1,f_2,\ldots,f_n$ }, Training Dataset $X_r\{x_1,x_2,\ldots,x_n \mid x_i \in C\}$, SplitRatio S
OUTPUT: Pruned Decision Tree
Process:
    1.  BEGIN
    2.  // BuildTreeClassifier
    3.  IF stopping_cond(X,F) = true THEN
    4.      leaf = createNode()
    5.      leaf.label = Classify(X)
    6.      return leaf
    7.  ELSE
    8.      root = createNode()
    9.      root.test_cond = find_best_split(X,F)
    10.     let V = {v|v is a possible outcome of root.test_cond}
```

```
11.       FOR EACH v ∈ V DO
12.            Xv = {x |root.test_cond(x) = v and x ∈ X}
13.            child = buildTreeClassifier(X,F)
14.            add child as descendant of root and label the edge as v
15.       END FOR
16.  END IF
17.  Return root
18.  // Prunning Algorithm
19.  GrowingSet G ← set of training data used to build the tree
20.  PruningSet P ← set of training data for validating the tree
21.  P,G  = splitExamples(S,X,F) //split training data
22.  tree  = TreeClassifier(G,F)
23.  WHILE(true)
24.        prunedTree =  bestSimplification(tree,P) //pruning process
25.        IF (accuracy(prunedTree,P) < accuracy(tree,P))
26.           Break;
27.           tree = prunedTree
28.        END IF
29.  END WHILE
30.  Return Pruned Decision Tree

END
```

4. Results

4.1 Implementation

The implementation of the developed Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree (HNIDF-NN-DT) model was carried out with Java programming language due to its support for pure object-oriented design, flexibility, portability and the rich graphical interface. NetBean was used as the Integrated Development Environment (IDE). Notepad++ was used to edit the dataset. WEKA API was used as the java plug-in for the used algorithms. Microsoft Excel was used for chart development.

4.2 Feature extraction

The developed hybrid feature selection method selected 6 features as shown in the Table 3 below. The essence of the feature selection is to extract the most important features from the full dataset that will enhance detection accuracy and reduce computational time. The table show the number of features selected, the serial number of features selected and their feature names.

4.3 Neural Network-Based Decision Tree

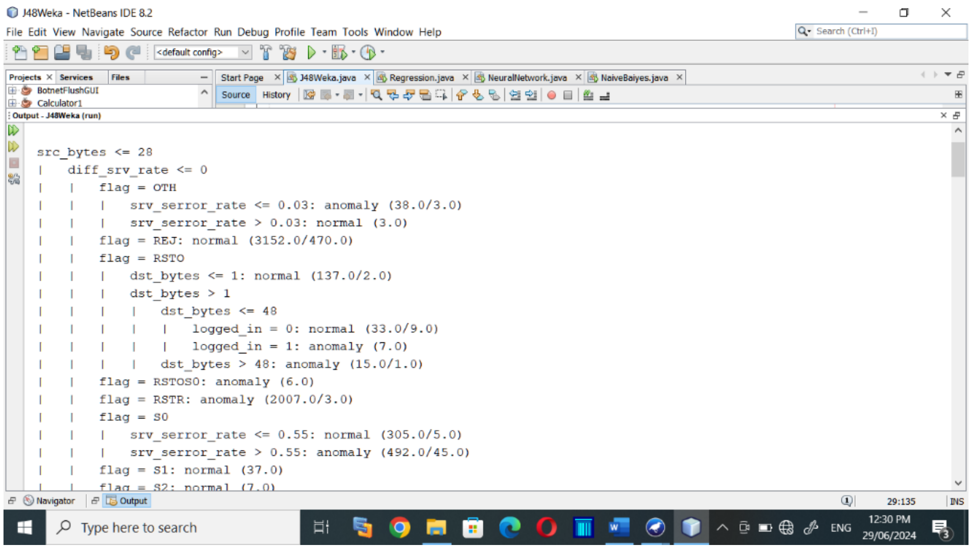
The result of this neural network-based decision tree can be described as set of rules, encapsulating the attributes in the selected features of the adapted dataset. Figure 2 show the rules representation of the neural network-based decision tree. The developed neural network-based decision tree model generated 101 leaves of rules for the classification of network intrusions. The neural network-based decision tree algorithm is a predictive data mining technique used in the prediction of network intrusions. It uses a predictive model to transform from item of observations (represented by branches) to rules about the target value of the item (represented in the leaves). Each node of the decision tree represents a neural network processing unit for accurate prediction and for interpretability of classification decisions.

The justification for the neural network-based decision tree is to provide automatic rules generation for the construction of NIDS. Another justification for the use of the neural network-based decision tree is for the decision tree to provide the detection rules for the interpretability of detection decisions and the neural network to provide the accuracy needed for attack detection. The developed neural network-based decision tree enables the resultant classification tree to be easier to understand and interpret. A predictive neural network-based decision tree was developed using J48 and neural network weka API in Java program. A 5-fold cross-validation was used to avoid overfitting and further prevent dataset imbalance problem.

Table 3: Selected features

No of features selected	Feature selected	Feature name
6	4,5,6,12,26,30	flag, src_bytes, dst_bytes, logged_in, srv_serror_rate, diff_srv_rate

Figure 2: A snippet of the rules for the neural network-based decision tree.



```
src_bytes <= 28
|   diff_srv_rate <= 0
|   |   flag = OTH
|   |   |   srv_serror_rate <= 0.03: anomaly (38.0/3.0)
|   |   |   srv_serror_rate > 0.03: normal (3.0)
|   |   flag = REJ: normal (3152.0/470.0)
|   |   flag = RSTO
|   |   |   dst_bytes <= 1: normal (137.0/2.0)
|   |   |   dst_bytes > 1
|   |   |   |   dst_bytes <= 48
|   |   |   |   |   logged_in = 0: normal (33.0/9.0)
|   |   |   |   |   logged_in = 1: anomaly (7.0)
|   |   |   |   dst_bytes > 48: anomaly (15.0/1.0)
|   |   flag = RSTOS0: anomaly (6.0)
|   |   flag = RSTR: anomaly (2007.0/3.0)
|   |   flag = S0
|   |   |   srv_serror_rate <= 0.55: normal (305.0/5.0)
|   |   |   srv_serror_rate > 0.55: anomaly (492.0/45.0)
|   |   flag = S1: normal (37.0)
|   |   flag = S2: normal (7.0)
```

Table 4 show the rules extracted from the leaves of the neural network-based decision tree generated in Figure 2. A total of 101 rules were extracted from the leaves of the generated neural network-based decision tree model for the prediction of anomaly or normal. For instance, rule 1 in Table 4 can be interpreted as “If src_bytes <= 28 && diff_srv_rate <= 0 && flag = OTH && srv_serror_rate <= 0.03 THEN put class in anomaly”. This means if src_bytes attribute <= 28 and diff_srv_rate attribute <= 0 and flag attribute = OTH and srv_serror_rate attribute <= 0.03, then classify network traffic as an anomaly class for the development of network intrusion detection system. Similarly, rule 2 in Table 4 can be interpreted as “If srv_serror_rate > 0.03 THEN put class in normal”. This means that if the srv_serror_rate attribute > 0.03 then classify network traffic as normal class. Table 4 shows 24 rules out of the 101 rules generated due to space limitation.

4.4 Performance metrics

The developed Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree model for network intrusion detection system was evaluated and compared with other related machine learning methods using the evaluation metrics listed below.

- **True positive** rate (TP): The number of cases that were appropriately categorized in the typical class is shown here. This is indicated in (12).

$$True\ positive\ rate = \frac{TP}{TP + FN} \tag{12}$$

- **False positive rate (FP):** These are the instances that were wrongly assigned to the typical class. It is indicated in (13).

$$False\ positive\ rate = \frac{FP}{FP + TN} \tag{13}$$

- **Precision:** This serves as a measure of the accuracy, assuming that a particular class that was predicted to be positive truly is positive. It is indicated in (14).

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

- **Recall:** This is an indicator of how many labelled occurrences a prediction model successfully recognizes, as shown in (15).

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

- **F-measure:** This represents the harmonic mean of the precision and recall based on a particular threshold. It is employed to evaluate the quality of the classification, as suggested in (16).

$$F - measure = \frac{2(precision * recall)}{precision + recall} \tag{16}$$

- **Accuracy:** This is the proportion of instances that were correctly categorized over all instances. It is indicated in (17).

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{17}$$

Table 4: Sample Leaves of the neural network-based decision tree

#No	Rule
1	If src_bytes <= 28 && diff_srv_rate <= 0 && flag = OTH && srv_serror_rate <= 0.03 THEN put class in anomaly
2	If srv_serror_rate > 0.03 THEN put class in normal
3	If flag = REJ THEN put class in normal
4	If flag = RSTO && dst_bytes <= 1 THEN put class in normal
5	If dst_bytes > 1 && dst_bytes <= 48 && logged_in = 0 THEN put class in normal
6	If logged_in = 1 THEN put class in anomaly
7	If dst_bytes > 48 THEN put class in anomaly
8	If flag = RSTOS0 THEN put class in anomaly
9	If flag = RSTR THEN put class in anomaly
10	If flag = S0 && srv_serror_rate <= 0.55 THEN put class in normal
11	If srv_serror_rate > 0.55 THEN put class in anomaly
12	If flag = S1 THEN put class in normal
13	If flag = S2 THEN put class in normal
14	If flag = S3 THEN put class in normal
15	If flag = SF && dst_bytes <= 3 && src_bytes <= 8 && src_bytes <= 7 && src_bytes <= 5 THEN put class in anomaly
16	If src_bytes > 5 THEN put class in normal
17	If src_bytes > 7 THEN put class in anomaly
18	If src_bytes > 8 && src_bytes <= 17 THEN put class in normal
19	If src_bytes > 17 && src_bytes <= 27 && src_bytes <= 18 THEN put class in anomaly
20	If src_bytes > 18 && src_bytes <= 20 && src_bytes <= 19 THEN put class in normal
21	If src_bytes > 19 THEN put class in anomaly
22	If src_bytes > 20 THEN put class in normal
23	If src_bytes > 27 THEN put class in anomaly
24	If dst_bytes > 3 && dst_bytes <= 29200 && src_bytes <= 4 && logged_in = 0 THEN put class in normal

4.5 Results

Table 5 show the comparison of the developed Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree (HNIDF-NN-DT) model for network intrusion detection system with other existing Machine Learning (ML) algorithms using the full dataset. Most of the ML algorithms on intrusion detection using the full dataset obtained F1-score of at least 73.03% classification rates. The F1-score of HNIDF-NN-DT on the full dataset is better with 98.56% compared to Multilayer Perceptron with the closest score of 94.79%. The results suggest Random Forest as the least ML algorithm for the classification of intrusions with F1-score of 73.03%. The outcomes demonstrated that the optimum method for network intrusion detection is HNIDF-NN-DT while the worst network intrusion detection method is Random Forest across the evaluation metrics using the full dataset. The results showed that the developed HNIDF-NN-DT is an improvement over the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.7, 1.3, 98.42%, 98.54%, 98.56% and 98.56 respectively compared to the other related methods. Overall, the results demonstrated that the various ML methods on the full dataset for network intrusion detection performed well.

Table 5: Performance comparison on full dataset

Algorithm	TP	FP	Accuracy	Precision	Recall	F1 score
C4.5 tree	97.5	2.5	95.78	94.43	92.63	93.23
Decision Table	94.2	5.8	89.33	88.54	88.34	88.24
Bagging	97.8	2.2	93.76	93.51	89.61	92.36
KNN	97.7	2.3	89.57	86.53	85.76	86.34
Logistic	93.9	6.1	94.32	94.53	89.53	93.33
Naïve Bayes	90.3	9.7	77.36	69.53	75.24	75.34
Bayesian Logistic Regression	91.6	8.4	83.45	93.52	88.62	91.63
Naïve Bayes Multinomial	90.7	9.3	80.06	75.34	74.34	75.35
Multilayer Perceptron	94.2	5.8	92.53	94.63	94.36	94.79
Random forest	97.9	2.1	73.04	73.03	73.02	73.03
HNIDF-NN-DT	98.7	1.3	98.42	98.54	98.56	98.56

Table 6 show the comparison of the developed Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree model for network intrusion detection system with other existing machine learning algorithms using the reduced dataset. Most of the ML algorithms on intrusion detection using the reduced dataset obtained F1-score of at least 75.04% classification rates. The F1-score of HNIDF-NN-DT on the reduced dataset is better with 99.56% compared to Logistic with the closest score of 95.23%. The results suggest Random Forest as the least ML algorithm for the classification of intrusions with F1-score of 75.04%. The outcomes demonstrated that the optimum method for network intrusion detection is HNIDF-NN-DT while the worst network intrusion detection method is Random Forest across the evaluation metrics using the reduced dataset. The results showed that the developed HNIDF-NN-DT is an improvement over the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.9, 1.2, 99.42%, 99.54%, 99.56%, and 99.56% respectively compared to the other related methods. Overall, the results demonstrated that the various ML methods on the reduced dataset for network intrusion detection performed better when compared to their performances on the full dataset.

Table 6: Performance comparison on reduced dataset

Algorithm	TP	FP	Accuracy	Precision	Recall	F1 score
C4.5 tree	98.4	1.6	96.68	95.43	93.63	94.23
Decision Table	95.1	4.9	90.23	90.44	90.24	90.14
Bagging	98.7	1.3	94.66	94.41	91.51	93.26
KNN	98.6	1.4	91.47	88.43	87.66	88.24
Logistic	94.8	5.2	95.22	95.43	91.43	95.23
Naïve Bayes	92.2	7.8	79.26	77.43	77.24	77.24
Bayesian Logistic Regression	93.5	6.5	85.35	95.42	90.52	93.53
Naïve Bayes Multinomial	90.7	9.3	80.06	75.34	74.34	75.35
Multilayer Perceptron	95.2	4.8	94.43	94.53	94.26	94.69
Random forest	98.8	1.2	75.04	75.03	75.02	75.04
HNIDF-NN-DT	98.9	1.2	99.42	99.54	99.56	99.56

Figure 3 shows the graphical representation of the performance rate of each of the methods under comparison using the full dataset. The graph showed that the F1-score of the developed HNIDF-NN-DT on the full dataset is better with 98.56% compared to Multilayer Perceptron with the closest score of 94.79%. The graph also suggests Random Forest as the least ML algorithm for the classification of intrusions with F1-score of 73.03%. The outcomes demonstrated that the optimum method for network intrusion detection is HNIDF-NN-DT while the worst network intrusion detection method is Random Forest across the evaluation metrics using the full dataset. The results showed that the developed HNIDF-NN-DT is an improvement over the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.7, 1.3, 98.42%, 98.54%, 98.56% and 98.56 respectively. The graph showed that the developed model exhibited highest value of performances compared to the other models.

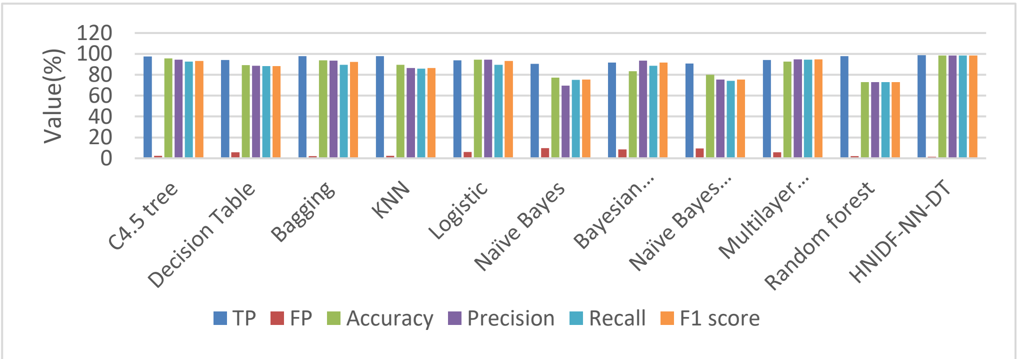


Figure 3: Graph of performance comparison on full dataset

Figure 4 shows the graphical representation of the performance rate of each of the methods under comparison using the reduced dataset. The graph showed that the F1-score of HNIDF-NN-DT on the reduced dataset is better with 99.56% compared to Logistic with the closest score of 95.23%. The graph also suggests Random Forest as the least ML algorithm for the classification of intrusions with F1-score of 75.04%. The graph showed that the optimum method for network intrusion detection is HNIDF-NN-DT while the worst network intrusion detection method is Random Forest across the evaluation metrics using the reduced dataset. The graph also showed that the developed HNIDF-NN-DT is an improvement over the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.9, 1.2, 99.42%, 99.54%, 99.56%, and 99.56% respectively. Overall, the graph demonstrated that the various ML methods on the reduced dataset for network intrusion detection performed better when compared to their performances on the full dataset.

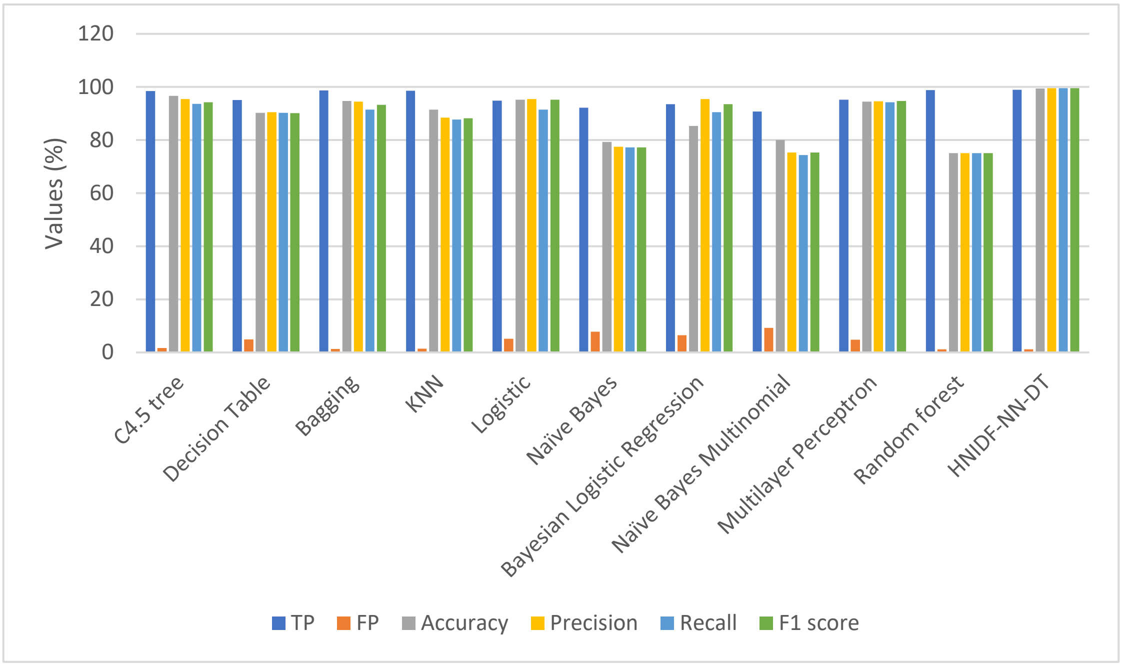


Figure 4: Graph of performance comparison on reduced dataset

5. Conclusion

The internet has presented new security challenges which deserve sophisticated mechanisms to avoid compromising confidentiality and integrity of data. The need to secure Internet applications on the global networks has become an important task due to the ever-increasing cybercrimes. A common technique for identifying intrusions in computer networks is the Network Intrusion Detection System (NIDS). Several NIDS techniques have been developed in the past, but these techniques are still limited in detection accuracy and error rate. Research has shown that one of the most considered strategies to achieve these sophisticated techniques is through the application of machine learning algorithms to the problem. Several IDSs have been proved to lack performance efficiency in detecting new attacks. Intruders find sophisticated and new means to intrude the privacy of a system. In this study, a Hybrid Network Intrusion Detection Framework using Neural Network-Based Decision Tree model for network intrusion detection system using the full and reduced datasets was developed. The study adapted a hybrid feature selection approach to choose the best attributes for NIDS. In order to automatically construct the rules for intrusion detection, the chosen attributes were trained with a Neural Network-Based Decision Tree model. The NSL-KDD dataset was used for experimentation and accuracy, precision, recall and F1-Score were used as metrics for the evaluation of the developed model. The results showed that the developed HNIDF-NN-DT based on the full dataset is an improvement over the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.7, 1.3, 98.42%, 98.54%, 98.56% and 98.56 respectively. Similarly, the results showed that the developed HNIDF-NN-DT based on the reduced dataset is an improvement over the other related methods with TP, FP, accuracy, precision, recall, and F1-score of 98.9, 1.2, 99.42%, 99.54%, 99.56%, and 99.56% respectively.

References

[1] S. Waskle, L. Parashar and U. Singh, " Intrusion detection system using PCA with random forest approach," in In 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC) , 2020.

[2] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: a survey.," EURASIP Journal on Wireless Communications and Networking, pp. 1-16, 2013.

[3] R. Patel, A. Thakkar and A. Ganatra, " A survey and comparative analysis of data mining techniques for network intrusion detection systems," International Journal of Soft Computing and Engineering (IJSCE), vol. 2, no. 1, pp. 265-260, 2012.

[4] A. Kajal and S. K. Nandal, "A hybrid approach for cyber security: improved intrusion detection system using Ann-Svm.," Indian Journal of Computer Science and Engineering, , vol. 11, no. 4, pp. 325-412, 2020.

[5] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems.," Journal of network and computer applications, vol. 34, no. 4, pp. 1184-1199, 2011.

[6] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández and E. Vázquez, " Anomaly-based network intrusion detection: Techniques, systems and challenges," computers & security, vol. 28, no. 1-2, pp. 18-28., 2009.

[7] I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes and J. Gama, "Host-based IDS: A review and open issues of an anomaly detection system in IoT," Future Generation Computer Systems, vol. 133, pp. 95-113., 2022.

[8] A. Heidari and M. A. Jabraeil Jamali, " Internet of Things intrusion detection systems: a comprehensive review and future directions," Cluster Computing, vol. 26, no. 6, pp. 3753-3780, 2023.

[9] O. H. Abdulganiyu, T. A. Tchakoucht and Y. K. Saheed, " Towards an efficient model for network intrusion detection system (IDS): systematic literature review," Wireless Networks, vol. 30, no. 1, pp. 453-482, 2024.

[10] F. E. Ayo, J. B. Awotunde, S. O. Folorunso, R. Panigrahi, A. Garg and A. K. Bhoi, " Bot-FFX: A Robust and Efficient Framework for Fast Flux Botnet (FFB) Detection," Wireless Personal Communications, pp. 1-24, 2024.

[11] M. Saied, S. Guirguis and M. Madbouly, " Review of artificial intelligence for enhancing intrusion detection in the internet of things," Engineering Applications of Artificial Intelligence , vol. 127, p. 107231, 2024.

[12] F. E. Ayo, L. A. Ogundele, S. Olakunle, J. B. Awotunde and F. A. Kasali, " A hybrid correlation-based deep learning model for email spam classification using fuzzy inference system," Decision Analytics Journal, vol. 10, p. 100390., 2024.

[13] K. Akyol and Ü. Atila, "A study on performance improvement of heart disease prediction by attribute selection methods," Academic Platform-Journal of Engineering and Science, vol. 7, no. 2, pp. 174-179, 2019.

[14] F. E. Ayo, S. O. Folorunso, A. A. Abayomi-Alli, A. O. Adekunle and J. B. Awotunde, " Network intrusion detection based on deep learning model optimized with rule-based hybrid feature selection.," Information Security Journal: A Global Perspective, vol. 29, no. 6, pp. 267-283, 2020.

[15] E. Alpaydin, Introduction to machine learning., MIT press., 2020.

[16] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu and C. Wang, "Machine learning and deep learning methods for cybersecurity.," Ieee access, vol. 6, pp. 35365-35381, 2018.

[17] B. Ingre, A. Yadav and A. K. Soni, "Decision tree-based intrusion detection system for NSL-KDD dataset," in In Information and Communication Technology for Intelligent Systems (ICTIS 2017), ., 2018.

[18] C. Azad and V. K. Jha, "Genetic algorithm to solve the problem of small disjunct in the decision tree-based intrusion detection system," International Journal of Computer Network and Information Security, vol. 7, no. 8, pp. 56-71, 2015.

[19] M. A. Khan, M. A. Khan, S. U. Jan, J. Ahmad, S. S. Jamal, A. A. Shah and W. J. Buchanan, " A deep learning-based intrusion detection system for MQTT enabled IoT," Sensors, vol. 21, no. 21, p. 7016, 2021.

[20] I. S. Thaseen, B. Poorva and P. S. Ushasree, "Network intrusion detection using machine learning techniques," in In 2020 International conference on emerging trends in information technology and engineering (IC-ETITE) , 2020.

[21] A. A. Salih and A. M. Abdulazeez, " Evaluation of classification algorithms for intrusion detection system: A review," Journal of Soft Computing and Data Mining, vol. 2, no. 1, pp. 31-40, 2021.

[22] S. &. B. A. Choudhury, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection (ICSTM).," in In 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, 2015.

[23] A. M. Mahfouz, D. Venugopal and S. G. Shiva, " Comparative analysis of ML classifiers for network intrusion detection," in In Fourth International Congress on Information and Communication Technology: ICICT 2019, London, 2020.

[24] S. M. Kasongo and Y. Sun, " A deep learning method with wrapper-based feature extraction for wireless intrusion detection system," Computers & Security, vol. 92, p. 101752, 2020.

[25] Y. Zhou, G. Cheng, S. Jiang and M. Dai, " Building an efficient intrusion detection system based on feature selection and ensemble classifier," Computer networks, vol. 174, p. 107247, 2020.

[26] A. Halimaa and K. Sundarakantham, " Machine learning based intrusion detection system," in In 2019 3rd International conference on trends in electronics and informatics (ICOEI) , 2019.

[27] B. S. Bhati and C. S. Rai, "Analysis of support vector machine-based intrusion detection techniques," Arabian Journal for Science and Engineering, vol. 45, no. 4, pp. 2371-2383, 2020.

[28] N. Bindra and M. Sood, " Detecting DDoS attacks using machine learning techniques and contemporary intrusion detection dataset," Automatic Control and Computer Sciences, vol. 53, no. 5, pp. 419-428, 2019.

[29] W. L. Chu, C. J. Lin and K. N. Chang, " Detection and classification of advanced persistent threats and attacks using the support vector machine," Applied Sciences, vol. 9, no. 21, p. 4579, 2019.

[30] J. Kim, Y. Shin and E. Choi, " An intrusion detection model based on a convolutional neural network," Journal of Multimedia Information System, vol. 6, no. 4, pp. 165-172, 2019.

[31] R. Patgiri, U. Varshney, T. Akutota and R. Kunde, "An investigation on intrusion detection system using machine learning," in In 2018 IEEE Symposium Series on Computational Intelligence (SSCI) , 2018.

[32] K. A. Taher, B. M. Y. Jisan and M. M. Rahman, "Network intrusion detection using supervised machine learning technique with feature selection," in In 2019 International conference on robotics, electrical and signal processing techniques (ICREST) , 2019.

[33] H. Xu, K. Przystupa, C. Fang, A. Marciniak, O. Kochan and M. Beshley, " A combination strategy of feature selection based on an integrated optimization algorithm and weighted k-nearest neighbor to improve the performance of network intrusion detection," Electronics, vol. 9, no. 8, p. 1206, 2020.

[34] A. Yulianto, P. Sukarno and N. A. Suwastika, "Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," in In Journal of Physics: Conference Series, 2019.

[35] A. A. Aburomman and M. B. I. Reaz, " A novel SVM-kNN-PSO ensemble method for intrusion detection system.," Applied Soft Computing, vol. 38, pp. 360-372, 2016.

[36] L. Haripriya and M. A. Jabbar, " Role of machine learning in intrusion detection system," in In 2018 second international conference on electronics, communication and aerospace technology (ICECA), 2018.

[37] N. Ashraf, W. Ahmad and R. Ashraf, " A comparative study of data mining algorithms for high detection rate in intrusion detection system.," Annals of Emerging Technologies in Computing (AETiC), pp. 2516-0281, 2018.

[38] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri and M. Rida, " A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection," Computers & Security, vol. 75, pp. 36-58, 2018.

[39] V. Hajisalem and S. Babaie, " A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," Computer Networks, vol. 136, pp. 37-50, 2018.

[40] W. L. Al-Yaseen, Z. A. Othman and M. Z. A. Nazri, " Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system," Expert Systems with Applications, vol. 67, pp. 296-303, 2017.

[41] W. C. Lin, S. W. Ke and C. F. Tsai, " CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," Knowledge-based systems, vol. 78, pp. 13-21, 2015.

[42] N. Maharaj and P. Khanna, " A comparative analysis of different classification techniques for intrusion detection system," International Journal of Computer Applications, vol. 95, no. 17, 2014.

[43] P. Ghosh, C. Debnath, D. Metia and R. Dutta, " An efficient hybrid multilevel intrusion detection system in cloud environment," IOSR Journal of Computer Engineering, vol. 16, no. 4, pp. 16-26, 2014.

[44] P. Jongsuebsuk, N. Wattanapongsakorn and C. Charnsripinyo, " Network intrusion detection with fuzzy genetic algorithm for unknown attacks.," in In The International Conference on Information Networking 2013 (ICOIN), 2013.

[45] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, " Deep learning approach for intelligent intrusion detection system," Ieee Access,, vol. 7, pp. 41525-41550, 2019.

[46] A. Mahindru and A. L. Sangal, "MLDroid—framework for Android malware detection using machine learning techniques," Neural Computing and Applications, vol. 33, no. 10, pp. 5183-5240, 2021.

[47] S. Baek, J. Jeon, B. Jeong and Y. S. Jeong, "Two-stage hybrid malware detection using deep learning," Human-centric Computing and Information Sciences, vol. 11, no. 27, pp. 10-22967, 2021.

[48] H. Bakır and R. Bakır, " DroidEncoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms," Computers and Electrical Engineering, vol. 110, p. 108804, 2023.

[49] F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry and Y. Nam, "A digital DNA sequencing engine for ransomware detection using machine learning," IEEE Access, vol. 8, pp. 119710-119719, 2020.

[50] G. Ciaramella, G. Iadarola, F. Martinelli, F. Mercaldo and A. Santone, "Explainable ransomware detection with deep learning techniques," Journal of Computer Virology and Hacking Techniques, vol. 20, no. 2, pp. 317-330, 2024.

[51] A. A. Almazroi and N. Ayub, " Deep learning hybridization for improved malware detection in smart Internet of Things," Scientific Reports, vol. 14, no. 1, p. 7838, 2024.

[52] A. Brown, M. Gupta and M. Abdelsalam, " Automated machine learning for deep learning-based malware detection," Computers & Security, vol. 137, p. 103582, 2024.

[53] M. A. Talukder, K. F. Hasan, M. M. Islam, M. A. Uddin, A. Akhter, M. A. Yousuf and M. A. Moni, "A dependable hybrid machine learning model for network intrusion detection," Journal of Information Security and Applications, vol. 72, p. 103405, 2023.

[54] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, " A detailed analysis of the KDD CUP 99 data set," in In 2009 IEEE symposium on computational intelligence for security and defense applications, 2009.