Babylonian Journal of Networking
Vol.2025, **pp**. 25–42
DOI: https://doi.org/10.58496/BJN/2025/003; ISSN: 3006-5372
https://mesopotamian.press/journals/index.php/BJN

Research Article

Evaluation of Information Security through Networks Traffic Traces for Machine Learning Classification

Tamara saad Mohamed[1,2,*], Saad mohammed khalifah[3], Ridwan Marqas [4,5], Saman M. Almufti [6,7], Renas R Asaad[8]

[1] *Department of Computer Engineering Technology, Kut university,kut,Iraq.*
2 Imam Ja'afar Al-Sadiq University
3 Head of Department of Engineering of Medical devices techniques department Communication security, kut university college, kut ,Iraq.
4 Computer Science, Knowledge University, Erbil, Iraq.
5 Software engineering, Firat university, Elazig, Turkey.
6 Department of Computer Science, College of Science, Knowledge University, Erbil, Iraq.
7 Information Technology Department, Technical College of Informatics-Akre, Akre University for Applied Sciences, Duhok, Iraq.
8 Department of Computer Science, College of Science, Knowledge University, Erbil, Iraq

ARTICLE INFO

Article History

Received 05 Jan 2025
Revised 08 Feb 2025
Accepted 24 Feb 2025
Published 25 Mar 2025

Keywords

Networks traffic

Information security
Machine learning

Random forest algorithm

NSL-KDD test and training dataset

ABSTRACT

The classification's traffic is regarded as a significant study domain because to the rising demand among network users. In addition to improving the identification of network services and addressing difficulties related to the security of traffic networks, it also simplifies and improves the accuracy of a broad variety of Internet application modes and activities. Over the course of the last several years, a multitude of traffic classification algorithms have been developed alongside their effective implementation. This paper proposed evaluate the classification issues of information security which happened through networks traffic by using machine learning (ML) classification algorithm the Random Forst by using NSL- KDD test and NSL-KDD train dataset to show the performance of testing data and training data.

1. INTRODUCTION

Traffic classification is the first process that facilitates the identification of various protocols and applications present inside the network. Various activities, including monitoring and optimization, may be executed on the detected traffic to enhance network performance. Traffic classification is a crucial method for identifying and categorizing unknown network types, as it addresses several network issues and offers diverse solutions for Internet service providers and their technological devices.[1]. Traffic analysis encompasses the whole process of intercepting traffic data to identify correlations, relations, trends, abnormalities, and configuration errors inside networks. One of the subsets of techniques that fall under this category is known as traffic classification. Its primary objective is to classify network traffic into predetermined categories, such as regular traffic and dangerous traffic..[2]. The analysis of network traffic sheds light on how new networks should be designed with user preferences and the basic concepts of Internet service operation in mind by providing comprehensive data on all services. Because of these factors, the categorization of network traffic is still an open subject, and research teams are always proposing new techniques that could function well in the future[3]. The attention of the Internet community was immediately drawn to the categorization of traffic from the very beginning. For the goal of improving Quality of Service (QoS) and managing network security, a number of different classification strategies have been proposed as potential approaches to network traffic classification. Standard categorization approaches

*Corresponding author. Email: tamara.mohhh@gmail.com

which require modifications to Transmission Control Protocol/Internet Protocol (TCP/IP) architecture have not gained acceptance because they necessitate substantial administrative work. In addition, port-based approaches and deep packet inspection have limits when it comes to coping with new traffic characteristics[4].machine learning(ML) methods might alleviate the shortage of qualified individuals competent to handle these specialized cybercrime detection systems. In addition, modern cyber-attacks, which are both automated and evolving, need robust mechanisms for detection and response. Machine learning (ML), which is able to learn from errors made in the past and immediately counteract more recent ones, is one possible reaction that might be used to combat these kinds of attacks. These links facilitate productivity development, process efficiency, and the creation of opportunities for new enterprises, however, network security is gaining significance as the growth of Internet of Things (IoT) devices escalates [5]. Because of the rising number of devices that are linked to the internet and the related rise in danger, the relevance of security precautions is expanding at an alarming rate. [6]. Training of traffic classification using machine learning has been of concern to the researchers as has been regarded as a promising approach to achieving higher performance. The incorporation of intelligence into network operations is made possible by machine learning, which thereby improves network management. In [7], authors consider machine learning approaches for a rather practical aim of predicting the classification of the network traffic..

2. LITERATURE REVIEW

The analysis employs four variants of Neural Network estimators to perform traffic application classification. This study evaluates the proposed method through four evaluation conditions consisting of feed forward and Multilayer Perceptron (MLP) and NARX (LevenbergMarquardt) and NARX (Naïve Bayes). The four examined scenarios produced accuracy measurements of 95.6%, 97% and 97.6%, 97%.[9]. In order to train their model, the authors use a large dataset that includes many different types of network traffic. Notable accuracy rates were achieved by decision trees (93%), random forests (97.89%), support vector machines (91%), and recurring neural network models (89.49%), demonstrating the usefulness of their technique.[7] This research use machine learning methods to forecast network traffic classification. They use four supervised learning algorithms: random forest, support vector machine, , k-nearest neighbors, and decision tree. They also use a port-based approach to traffic categorization predicated on the commonly allocated port numbers of apps. Subsequently, we contrast the outcomes of this strategy with those derived from the machine learning algorithms. [8] A novel framework model is presented in this research. An innovative selection of features measurement approach called CorrAUC is proposed, followed by the development and design of a new feature selection algorithm named Corrauc. This algorithm employs an additional technique to accurately filter features and select effective ones for the chosen machine learning algorithm using the AUC metric. This work proposed an investigation to study what enables this technique succeed in traffic classification problems. A systematic review process is developed following the steps required to classify traffic through Machine Learning techniques.[9]this paper The suggested classifiers demonstrate superior performance when network traffic streams are created with varying time parameters (timeout). The findings indicate that ensemble methods, namely Gradient Boosting, Random Forest and , surpass individual machine learning classifiers.

3. METHODOLOGY

An abstract representation of our suggested system is shown in the following Figure1:

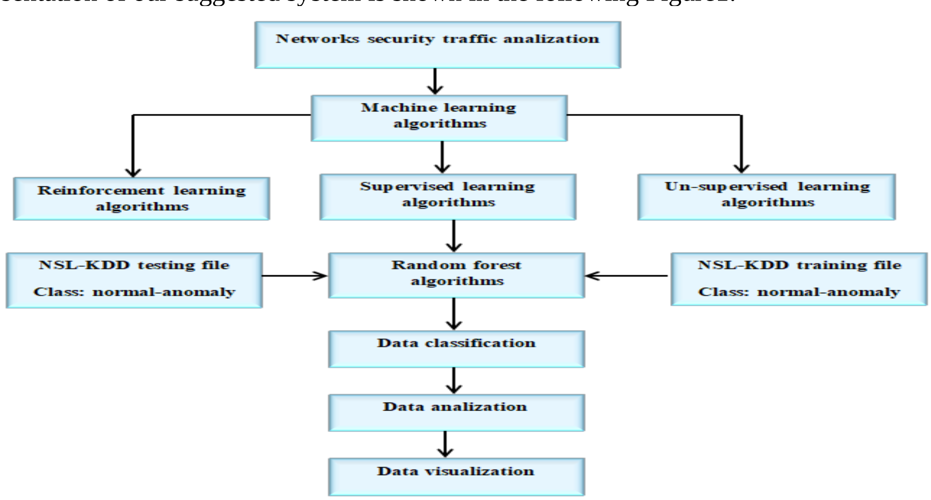

Fig. 1. flow chart of the proposed system.

3.1 Description of Dataset

NSL-KDD is a dataset proposed to address several essential issues much better than the KDD'99 dataset . Since there are currently no publicly available data sets for network-based intrusion detection systems, we think this updated KDD dataset can still serve as a useful benchmark for researchers to compare various intrusion detection algorithms, despite the fact that it has some of the issues highlighted by McHugh and might not be an exact representation of current real networks. In addition, the NSL-KDD training and test sets include sufficient amounts of data. This advantage makes it cost-effective to perform tests on the whole dataset without necessitating the random selection of a smaller sample. Consequently, the evaluation results of different academic pursuits will be consistent and similar.The whole NSL-KDD training and testing dataset with binary labels in ARFF format.The NSL-KDD dataset offers many advantages over the original KDD dataset: it addresses some intrinsic problems of KDD, and the training set omits duplicate information, hence avoiding classifiers from being biased towards more frequent entries. The proposed test sets have no duplicate entries; hence, the learners' performance is not biased by methods that provide higher identification rates for common records. The number of selected records from each challenge level group correlates directly with the percentage of records in the original KDD dataset. Various machine learning algorithms exhibit notable variance in their classification rates, allowing for a more precise assessment of various learning approaches. It is not necessary to randomly choose a smaller subset of the dataset in order to do experiments since there are enough records in both the training and testing sets..As a consequence, the assessment outcomes of many research studies will be uniform and comparable. Every entry in the NSL-KDD dataset comprises 42 attributes.41 attributes represent the characteristic attributes of the data, and the rest one attribute called class represents the if the data is normal or anomaly.[10][11][12].
In the current research, NSL-KDDTest is adopted for experimenting on intrusion detection systems in network security which is common dataset. The dataset includes 42 attributes, which characterize various characteristics of the network connections and was created on a basis of normal and anomalous traffic as shown in Table 1.
Dataset Overview:

1. Dataset Name: NSL-KDDTest
2. Number of Attributes: 42
3. Target Classes: (Normal, Anomaly)
4. Purpose: Network Intrusion Detection

TABLE I. DATA SET'S FEATURES DESCRIPTION

#	Attribute Name	Data Type	Description
1	duration	Real	Length of the connection.
2	protocol_type	Categorical	Type of transport protocol (TCP, UDP, ICMP).
3	service	Categorical	Network service used (e.g., HTTP, FTP, SMTP, etc.).
4	flag	Categorical	Status flag of the connection (e.g., SF, RSTO, S0, etc.).
5	src_bytes	Real	Bytes transferred from source to destination.
6	dst_bytes	Real	Bytes transferred from destination to source.
7	land	Binary (0,1)	Indicates if source and destination addresses are the same.
8	wrong_fragment	Real	Number of incorrect fragments.
9	urgent	Real	Number of urgent packets.
10	hot	Real	Number of key indicators for suspicious activity.
11	num_failed_logins	Real	Number of failed login attempts.
12	logged_in	Binary (0,1)	Whether the connection was successful.
13	num_compromised	Real	Number of compromised conditions.
14	root_shell	Real	Indicates if a root shell was obtained.
15	su_attempted	Real	Indicates if an *su* command was attempted.
16	num_root	Real	Number of root-level accesses.
17	num_file_creations	Real	Number of file creation operations.
18	num_shells	Real	Number of shell prompts invoked.
19	num_access_files	Real	Number of operations accessing control files.
20	num_outbound_cmds	Real	Number of outbound commands in an FTP session.
21	is_host_login	Binary (0,1)	Indicates if the login was to a host account.
22	is_guest_login	Binary (0,1)	Indicates if the login was to a guest account.
23	count	Real	Number of connections to the same host.
24	srv_count	Real	Number of connections to the same service.

25	serror_rate	Real	Percentage of connections with SYN errors.
26	srv_serror_rate	Real	Percentage of connections to the same service with SYN errors.
27	rerror_rate	Real	Percentage of connections with REJ errors.
28	srv_rerror_rate	Real	Percentage of connections to the same service with REJ errors.
29	same_srv_rate	Real	Percentage of connections to the same service.
30	diff_srv_rate	Real	Percentage of connections to different services.
31	srv_diff_host_rate	Real	Percentage of connections to different hosts.
32	dst_host_count	Real	Number of connections to the same destination host.
33	dst_host_srv_count	Real	Number of connections to the same service on the destination host.
34	dst_host_same_srv_rate	Real	Percentage of connections to the same service on the destination host.
35	dst_host_diff_srv_rate	Real	Percentage of connections to different services on the destination host.
36	dst_host_same_src_port_rate	Real	Percentage of connections using the same source port.
37	dst_host_srv_diff_host_rate	Real	Percentage of connections to different hosts using the same service.
38	dst_host_serror_rate	Real	Percentage of destination host connections with SYN errors.
39	dst_host_srv_serror_rate	Real	Percentage of destination host connections to the same service with SYN errors.
40	dst_host_rerror_rate	Real	Percentage of destination host connections with REJ errors.
41	dst_host_srv_rerror_rate	Real	Percentage of destination host connections to the same service with REJ errors.
42	class	Categorical	Label indicating whether the connection is *normal* or *anomalous*.

3.2 Description Of Analytical Data by Algorithms

3.2.1 Machine Learning Algorithms

It is worth mentioning that according to machine learning which is a concept that means sets of raw instructions, a computer is capable of learning from data and improving its ability of building a model without the interference of human being. Some common categories for algorithms used in machine learning are as follows:

1. In the first kind of learning, known as supervised learning, the relationship between the input and the output is already established, and computers learn from data that has been labeled.
2. When provided with unlabeled data, algorithms engage in unsupervised learning in order to detect clusters or patterns. This is done in order to identify patterns.
3. Reinforcement Learning: This kind of learning enables an algorithm to learn on her own through use of feedback from the environment which may be in form of reward signals or punishments. This is in addition to the fact that supervised algorithms are necessary for the analysis of supervised datasets.
   [13].

3.2.1.1 Supervised Learning Algorithms

As a response or target variable, a label is paired with each example in the datasets used to train supervised learning algorithms. Finding a process that reliably assigns corresponding labels to incoming data is the main objective. Because of this, the model will have an easier time making correct predictions when given fresh data. There are mainly two types of supervised learning tasks: classification and regression. A few examples of supervised learning algorithms that are often used include:

1. Linear Regression

Linear regression figures out the best straight line that shows the relationship between the inputs of the independent variable and the inputs of the dependent variable in order to predict a continuous value.

- Reduces the mismatch between actual and expected values by a technique known as least squares to optimally match the data.
- Prediction of obesity based on height, or predicting the rate of homes based on certain measurements.

2. **Logistic Regression**
   - lobistic regression predicts probability and sort data items into two classes for example; data items that are likely to be spam and those that are not.
   - It uses a logistic function (S-shaped curve) to map the input characteristics with the class probability.
   - TestUtils is used for classification tasks like binary or multiclass. Produces probability for categorizing data.
   - Example: Forecasting a customer's ability to purchase a product online (positive/negative) or determining an individual's health status (ill/not ill).

3. **Decision Trees**

This decision tree algorithm is excellent for both regression and classification tasks. It uses a tree-like structure to split data into branches based on feature values. The leaf nodes provide the final prediction, while each decision node represents a feature. If you're looking for additional decision tree algorithms, you can explore:

- Iterative Dichotomiser 3 (ID3) Algorithms. C5. Algorithms. Classification and Regression Trees Algorithms.

4. **Support Vector Machines (SVM)**

Support Vector Machines (SVM) identify the optimal border, known as a hyperplane, that distinguishes data points into distinct categories. The system utilizes support vectors, which are essential data points, to delineate the hyperplane. Because of the utilization of kernel functions for linear and non-linear problems and also due to the fact that it focuses on the separation of classes, it is applicable when dealing with high dimensional data or structures.

5. k-Nearest Neighbors (k-NN)

Easy to implement, the concept of KNN involves the determination of the future values a new data point basing on the Friedman of its weights similar data points in the training set. This gives the benefit of being able to handle either classification problems as well as regression ones in case there is a difficulty with either one of the two. Never underestimates the distance of a given point with another point in the training set using a distance measures such as Minkowski, Manhattan or Euclidean distance. Based on these calculated distances, the k near neighbor of the new data point is fitted. During the classification stage, the label is determined by the frequency of such label being present among the K-NN's. Under regression, the method uses the average of the k closest neighbors and returns the value of the function.

6. **Naive Bayes**

Grounded on Bayes' theorem, it implies that every feature is mutually independent (hence termed "naive").

- confirms the function that calculates probabilities for every class and assign the highest probability for a data point to a peculiar class. I guess the degree of independence of the features used might not hold all the time since they are not normally valid in the real world.
- Effectively handles high-dimensional data.
- Applied particularly in the text categorization tasks for instance in the identification of junk email: Naive Bayes' algorithm.

7. Random Forest

A random forest combines many decision trees into one ensemble method.

- Uses feature selection and random sampling to guarantee that trees are different. For classification, the final prediction is based on a majority vote, and for regression, it's based on an average.
- Benefits: mitigates overfitting relative to standalone decision trees.
- Accommodates extensive datasets with elevated dimensionality.

8. Gradient Boosting (e.g., XGBoost, LightGBM, CatBoost)

Each iteration of a model is improved upon by these algorithms, which work in an incremental fashion to build models. Builds a strong prediction model by combining weak learners, such decision trees. Perfect for jobs requiring categorization and regression analysis. Machine Learning using Gradient Boosting...

- XGBoost, short for "Extreme Gradient Boosting," is a more sophisticated version of GB that uses regularization to lessen the likelihood of overfitting. For big datasets, it's faster than regular Gradient Boosting.
- Light Gradient Boosting Machine (LightGBM) incorporates categorical data naturally and uses a histogram-based technique to improve computational efficiency.
- CatBoost: Using integrated encoding methods, it is specifically tailored for category data. Improves generalizability and speeds up training with symmetric trees. Look at AdaBoost and stacking-ensemble learning if you want to learn more about ensemble learning and gradient boosting.

9. Neural Networks (Including Multilayer Perceptron)

Supervised machine learning algorithms include Neural Networks like Multilayer Perceptrons (MLPs) that minimize error by adjusting weights during training using the back propagation algorithm. These networks need labeled data for training and to determine the relationship between inputs and desired outputs. One kind of neural network is the multilayer perceptron (MLP), which has multiple layers of nodes and can be used for both regression and classification tasks. It is often used for such things as image recognition, filter spam, numerical value predictions, stock values or the value of properties [14][15][16][17][18].

3.2.1.2 Random Forest

Classification and prediction are two applications of the machine learning model known as a random forest. For efficient data collection, a large quantity of high-quality data is required to train AI models and machine learning algorithms. Improving algorithms, software and hardware efficiency, user behavior evaluation, pattern detection, decision-making, predictive modeling, and problem-solving are all made possible by system performance data, which is crucial for achieving these goals.[19]. Random Forest (RF) is a widely used machine learning methodology in data mining. It functions under the supervision of a collective and has achieved substantial recognition.[20]

1. Further definition: Random Forest, is a classifier that comprises different tree-based models h(x, Θk) k=1, 2, …wherein the Θk is an independently, identically distributed random vector and every tree plus a unit vote to the majority class at point x[21].
2. Random forests may be used for in two ways; a response that is categorical variable, termed 'classification' or a continuous response, known as "regression." Expectation variables may be classified as either categorical or continuous. From a computational perspective, random forests are appealing due to their :
    - Inherent ability to handle both regression and classification tasks (multilayer).
    - Comparatively fast in training and prediction.
    - Rely only on one or two adjustable factors.
    - You possess an inherent estimation of the generalization mistake.
    - The subsequent methods may be used directly for high-dimensional issues.
    - They may be easily executed concurrently.
    - Statistically, random forests are appealing because of the supplementary features they provide, including:
        - ✓ Metrics of varying significance.
        - ✓ Assigning weight to the differential layer.
        - ✓ Determination of the absent value.
        - ✓ The cognition

3.2.1.3 Random Forest Working

A network of decision trees is used in the Random Forest method of machine learning, which is designed to reduce the degree of correlation that exists between features. When working with enormous datasets, Random Forest is famously sluggish; its computational complexity is O(n), where n is the number of samples. Random Forest is a very slow algorithm. This integration makes it possible for several processes to execute in parallel, which results in a gain in performance. The relationship between decision trees is removed by the use of Random Forest, which does this by selecting samples and characteristics at random. The decision tree is constructed by randomly selecting a group of features and then selecting a matched volume of data from the main training dataset. Both of these components are picked at random. By decreasing the degree to which decision trees are dependent on one another, these two randomization strategies increase the accuracy of the model while simultaneously lowering the likelihood of overfitting. A graph that is enclosed inside another graph is referred to as a "inset," not a "insert." This is the proper form of the term. When contrasted with the phrase "alternatively," the former is the more desirable option.
.[19]. As explained in image below as shown in Figure2:

1. The process starts with a dataset including rows and their associated class labels (columns).
2. Therefore, from the training data set, many decision trees occur as follows. Every tree is created from a random selection of the given data with substitution and a random selection of features.
3. This procedure is referred to as bagging or bootstrap aggregating.
4. Each Decision Tree inside the ensemble individually learns the ability to make predictions.
5. Upon encountering a novel, unobserved instance, each Decision Tree inside the ensemble generates predictions.
6. The ultimate forecast is derived from the aggregation of the predictions of all the Decision Trees.[22]

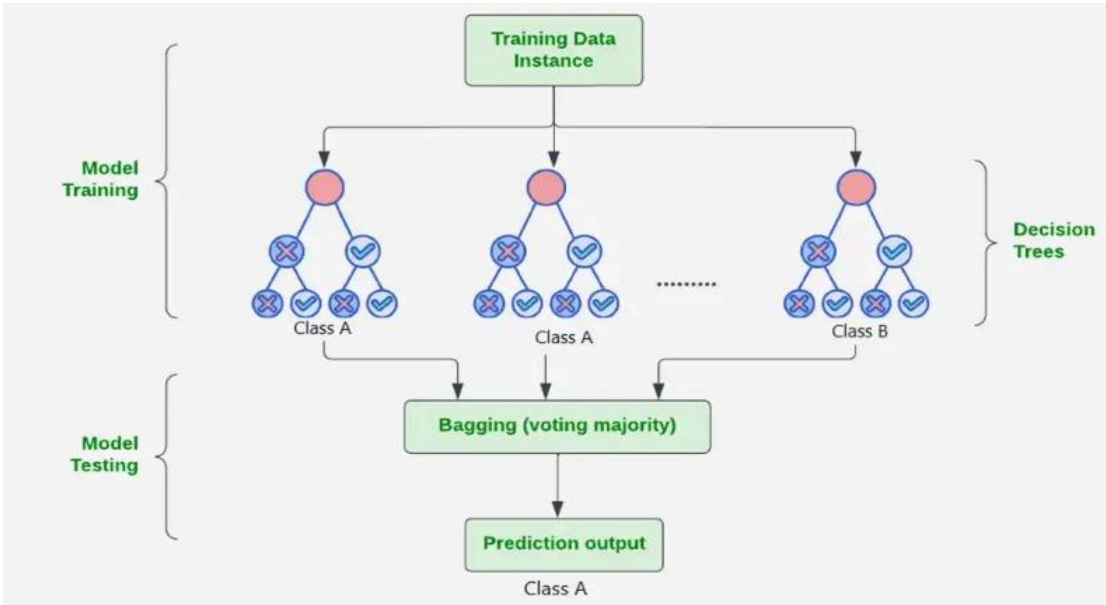

Fig. 2. Random Forest Algorithm in Machine Learning.[22]

4. RESULTS AND DISCUSSION

The analyzation has been done according to cross-validation of 10 folds. The result's of both files of testing and training files(the total of instances are 148517:which includes 125973 instances of training file and 22544 instances of testing file,the details listed below:

1. Duration: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instance classifies as normal or anomaly related to time.
2. Protocol_type: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instance classifies as normal or anomaly and how much instances belongs to the types of protocol as TCP,UDP or ICMP.
3. Service: We categorize this attribute in terms of the class type with normal as blow color and anomaly as red color, and the number of instances that is classified under the types of services. (aol,auth,bgp,courier,csnet_ns,ctf,daytime,discard,domain,domain_u,echo,eco_i,ecr_i,efs,exec,finger,ftp,ftp_data,gopher,harvest,hostnames,http,http_2784,http_443,http_8001, imap4,IRC,iso_tsap,klogin,kshell,ldap,link,login,mtp,name,netbios_dgm,netbios_ns, netbios_ssn, netstat, nnsp, nntp, ntp_u, other, pm_dump, pop_2, pop_3, printer, private, red_i, remote_job, rje, shell,smtp,sql_net, ssh,sunrpc,supdup,systat,telnet,tftp_u,tim_i,time,urh_i,urp_i,uucp, uucp_path, vmnet,whois,X11,Z39_50).
4. Flag: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Flag(OTH,REJ,RSTO,RSTOS0,RSTR,S0,S1,S2,S3,SF',SH).
5. src_bytes: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of src_bytes (real).
6. Dst_bytes. : weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of dst_bytes (real).
7. Land: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_bytes.Land (0,1).

8. Wrong_fragment: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Wrong_fragment (real).
9. Urgent: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of urgent (real).
10. Hot: : weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of hot (real).
11. Num_failed_logins: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of dst_bytes (real).
12. Logged_in: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of dst_bytes (0,1).
13. Num_compromised: : weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Num_compromised (real).
14. Root_shell: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Root_shell (real).
15. Su_attempted: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Su_attempted (real).
16. Num_root: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Num_root (real).
17. Num_file_creations: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Num_file_creations (real).
18. Num_shells: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Num_shells (real).
19. Num_access_files: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Num_access_files (real).
20. Num_outbound_cmds: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Num_outbound_cmds (real).
21. Is_host_login: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Is_host_login(0,1).
22. Is_guest_login: : weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Is_host_login(0,1).
23. Count,srv_count: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Count,srv_count(real).

24. Serror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Serror_rate (real).
25. Srv_serror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Srv_serror_rate (real).
26. Rerror_rate: : weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Rerror_rate (real).
27. Srv_rerror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Srv_rerror_rate (real).
28. Srv_rerror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Srv_rerror_rate (real).
29. Same_srv_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Same_srv_rate (real).
30. Diff_srv_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Diff_srv_rate (real).
31. Srv_diff_host_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Srv_diff_host_rate (real).
32. Dst_host_count: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_count (real).
33. Dst_host_srv_count: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_srv_count (real).
34. Dst_host_same_srv_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_same_srv_rate (real).
35. Dst_host_diff_srv_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_diff_srv_rate (real).
36. Dst_host_same_src_port_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_same_src_port_rate (real).
37. Dst_host_srv_diff_host_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_srv_diff_host_rate (real).
38. Dst_host_serror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_serror_rate (real).
39. Dst_host_srv_serror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_srv_serror_rate (real).

40. Dst_host_rerror_rate,dst_host_srv_rerror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_rerror_rate,dst_host_srv_rerror_rate (real).
41. Dst_host_srv_serror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_srv_serror_rate (real).
42. Dst_host_rerror_rate,dst_host_srv_rerror_rate: weka classify this attribute according to the class type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of Dst_host_rerror_rate,dst_host_srv_rerror_rate (real).
43. Class: weka classify this attribute type(normal as blow color, anomaly as red color) and shows how much instances classifies as normal or anomaly and how much instances belongs to the types of normal as blow color, anomaly as red color .

4.1 Analytical Data with RandomForest by NSL-KDDTest

Firstly, we analyzed the NSL-KDD dataset file of KDDTest.arff by using Weka data mining tool to classify this data using the classification algorithm RandomForest; the visualizing of all attributes shown in Figures 3,4,5,6: the analyzation has been done according to cross-validation of 10 folds. The details explained in section 4. , the following is the execution of algorithm:

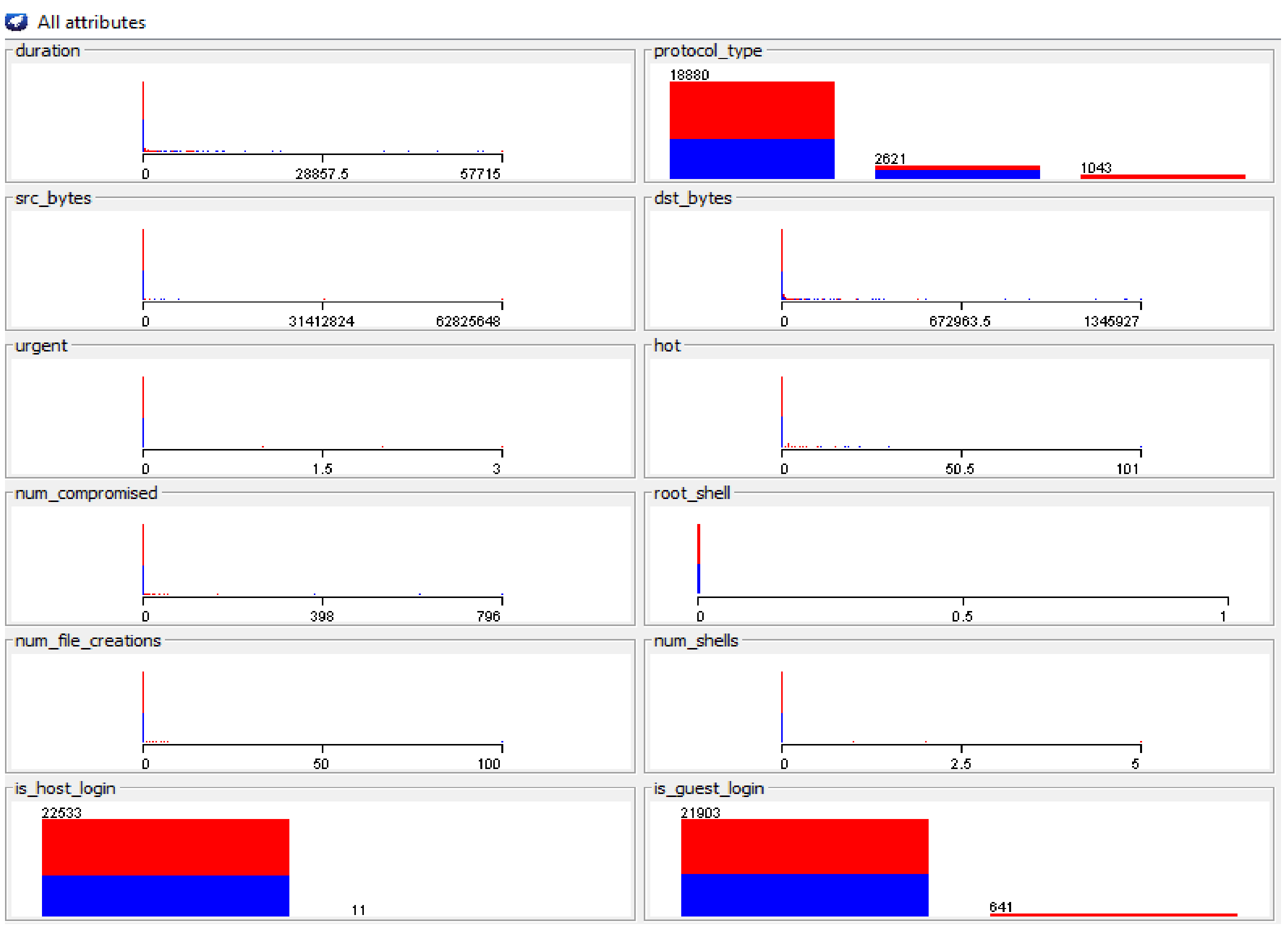

Fig. 3. Visualizing of attributes

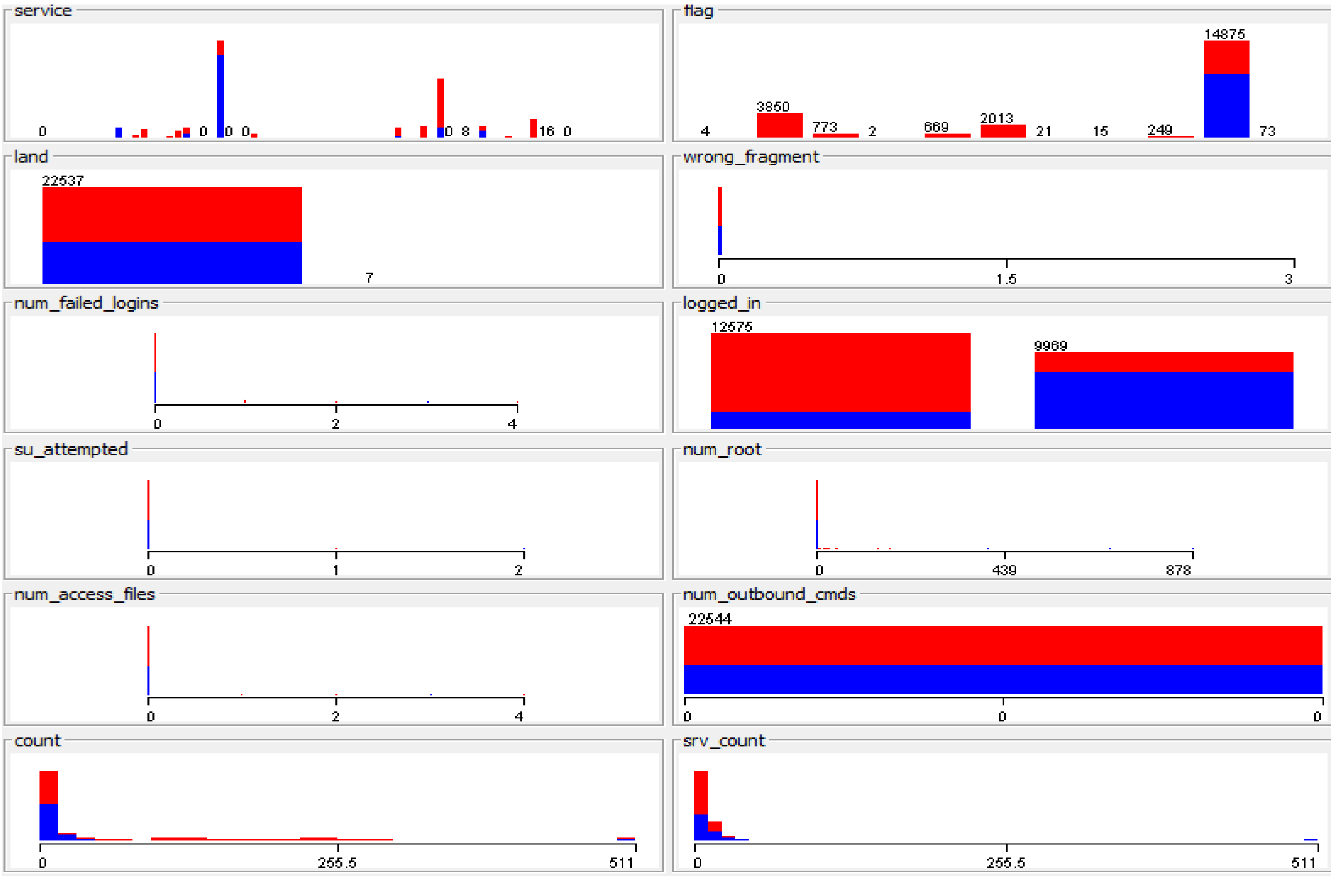

Fig.4. Visualizing of attributes

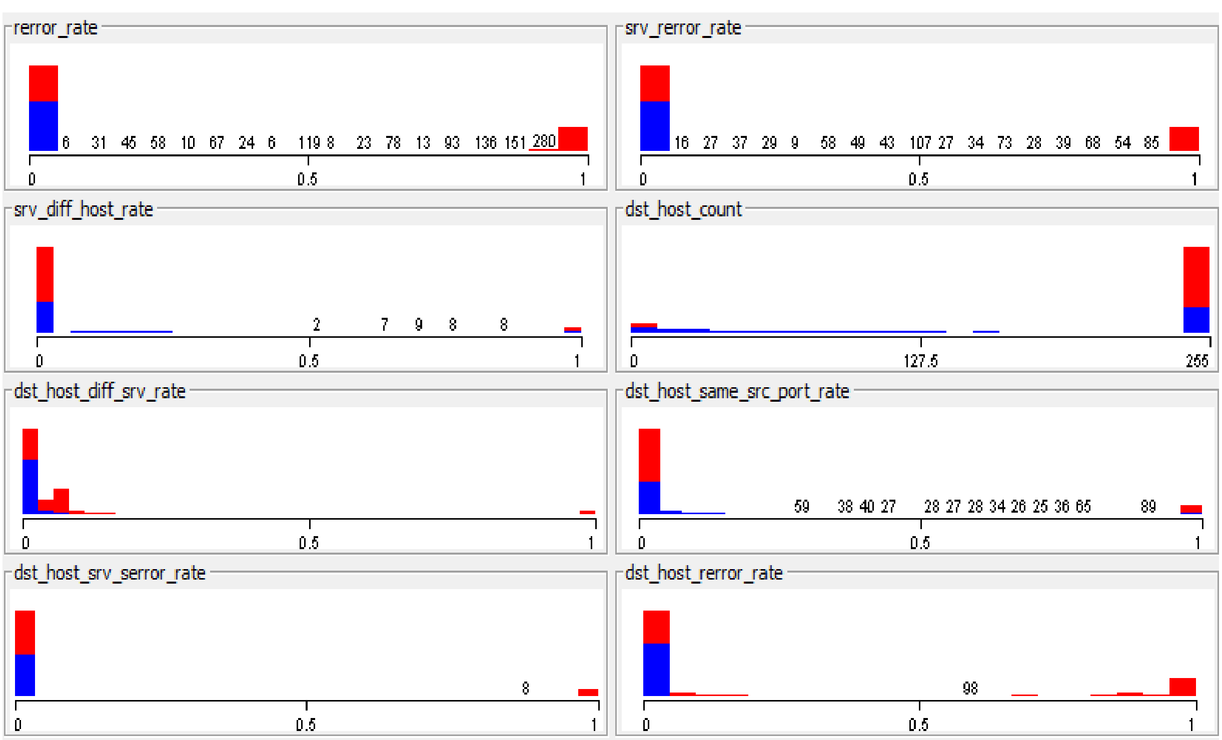

Fig. 5. Visualizing of attributes

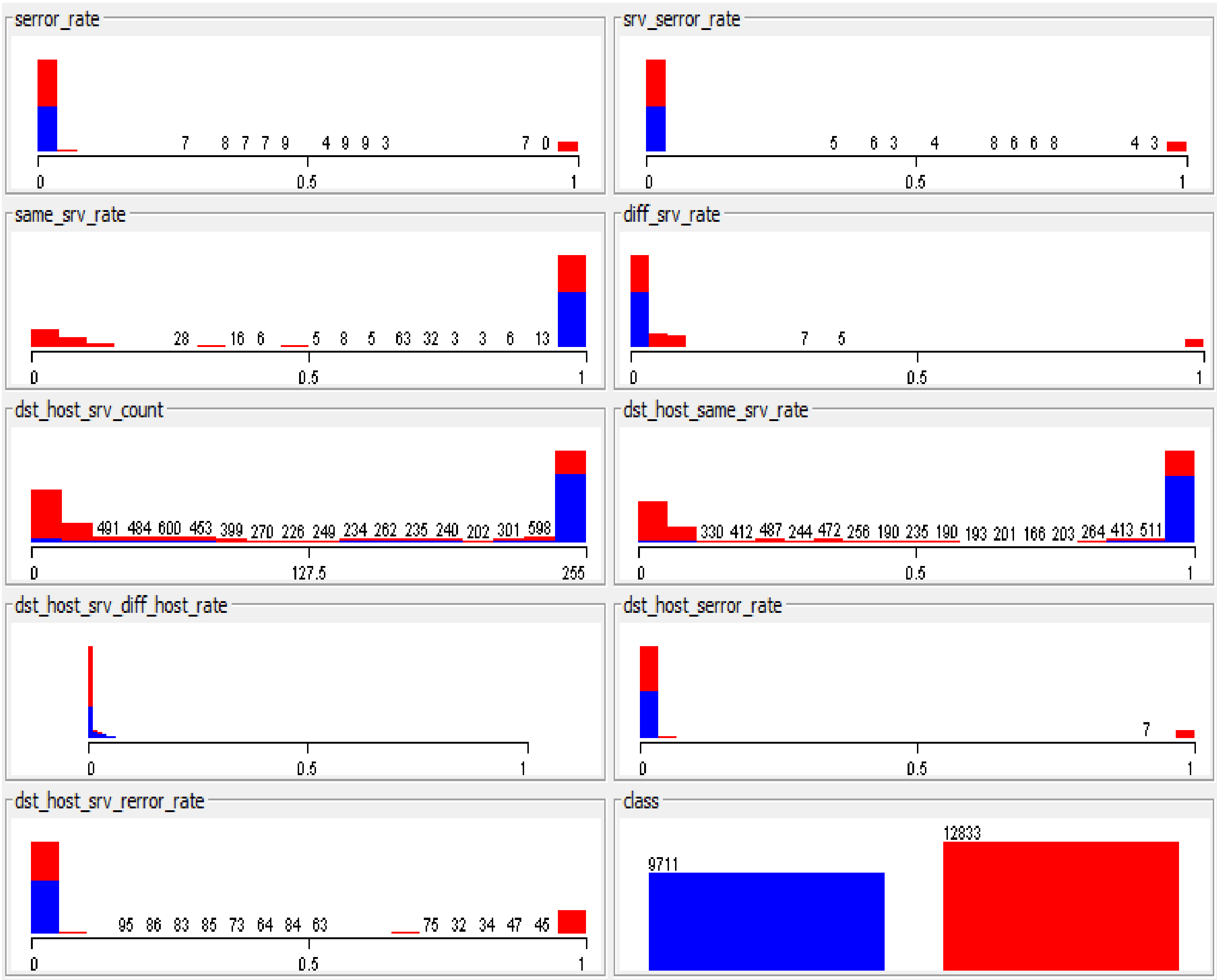

Fig. 6. Visualizing of attributes

4.1.1 Run information

1. Scheme: weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
2. Relation: KDDTest
3. Instances: 22544
4. Attributes: 42:

   Attributes:42:

   “Duration,protocol_type,service,flag,src_bytes,dst_bytes.Land,wrong_fragment,urgent,hot,num_failed_logins,logged_in,num_compromised,root_shell,su_attempted,num_root,num_file_creations,num_shells,num_access_files,num_outbound_cmds,is_host_login,is_guest_login,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,same_srv_rate,diff_srv_rate,srv_diff_host_rate,dst_host_count,dst_host_srv_count,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate, class”.
5. Test mode involves10-fold cross-validation.
6. Classifier model (full training set).
7. RandomForest bagging with number of trees = 100 base learner .
8. Time taken to build model: 6.44 seconds.
9. Stratified cross-validation .
10. Summary

This is the performance evaluation of a classification model. Here's what the metrics signify:

- **Correctly-classified-instances**: Out of 22,544 instances, 22,252 were classified correctly, achieving **98.7048% accuracy**, which is outstanding.
- **Incorrectly-classified-instances**: Only 292 instances were misclassified, representing **1.2952%**, which is very low.
- **Kappa statistic**: A high value of **0.9736** indicates excellent agreement between predicted and actual classifications, beyond random chance.
- **Mean Absolute Error (MAE)**: **0.0201** signifies the average error in predictions is quite small.
- **Root Mean Squared Error (RMSE)**: **0.0986** reveals an overall low error magnitude in predictions.
- **Relative Absolute Error (RAE)**: **4.0972%**, showing the prediction errors are relatively very low compared to a naive baseline.
- **Root Relative Squared Error (RRSE)**: At **19.9075%**, it shows the variance in prediction errors compared to the baseline.

Overall, these metrics indicate that the model performs impressively well with minimal errors and excellent consistency.

11. Detailed Accuracy By Class : the accuracy of classification in random forest algorithm shown in the Table 2.

TABLE II. ACCURACY OF CLASSIFICATION IN RANDOM FOREST ALGORITHM

	TP_Rate	FP_Rate	Precision	Recall	F_Measure	MCC	ROC_Area	PRC_Area	Class
	0.984	0.011	0.985	0.984	0.985	0.974	0.999	0.999	normal
	0.989	0.016	0.988	0.989	0.989	0.974	0.999	0.999	anomaly
Weighted Avg.	0.987	0.014	0.987	0.987	0.987	0.974	0.999	0.999	

12. Confusion Matrix : confusion matrix shown in the Table 3.

TABLE III. CONFUSION MATRIX

a	b	Classified as
9560	151	a=normal
141	12692	b=anomaly

13. Performance parameters of confusion matrix:
    - TP-Rate: true positive alarm rate, representing cases accurately identified as a certain class.
    - FP-Rate: false positive alarm rate ,(instances incorrectly categorized as belonging to a certain class).
    - Precision: the ratio of true occurrences of a class to the total instances categorized as that class. Recall is the ratio of occurrences identified as a certain class to the actual total inside that class, synonymous with the true positive rate.
    - F-Measure: A composite metric for accuracy and recall, computed as 2 * accuracy * Recall / (Precision + Recall).
    - MCC serves as an indicator of the accuracy of binary (two-class) classifications in machine learning. It considers true and false positives and negatives and is often seen as a balanced metric applicable even when the classes vary significantly in size.
    - ROC-Area (Receiver Operating Characteristic) measurement: A crucial metric produced by Weka. They provide an overview of the classifiers' overall performance.
    - Precision-Recall Curve Area (PRC-Area): The Precision-Recall plot is more effective than the ROC plot in the assessment of binary classifiers for imbalanced datasets [23].

4.2 Analytical Data with RandomForest by NSL-KDDTrain

Secondly we annualized the NSL-KDD dataset file of KDDTrain by using Weka data mining tool to classify this data using the classification algorithm tree.RandomForest ;the visualizing of all attributes shown in Figures 7,8,9,10 : the analyzation has been done according to cross-validation of 10 folds. The details explained in section 4. ,the following is the execution of algorithm:

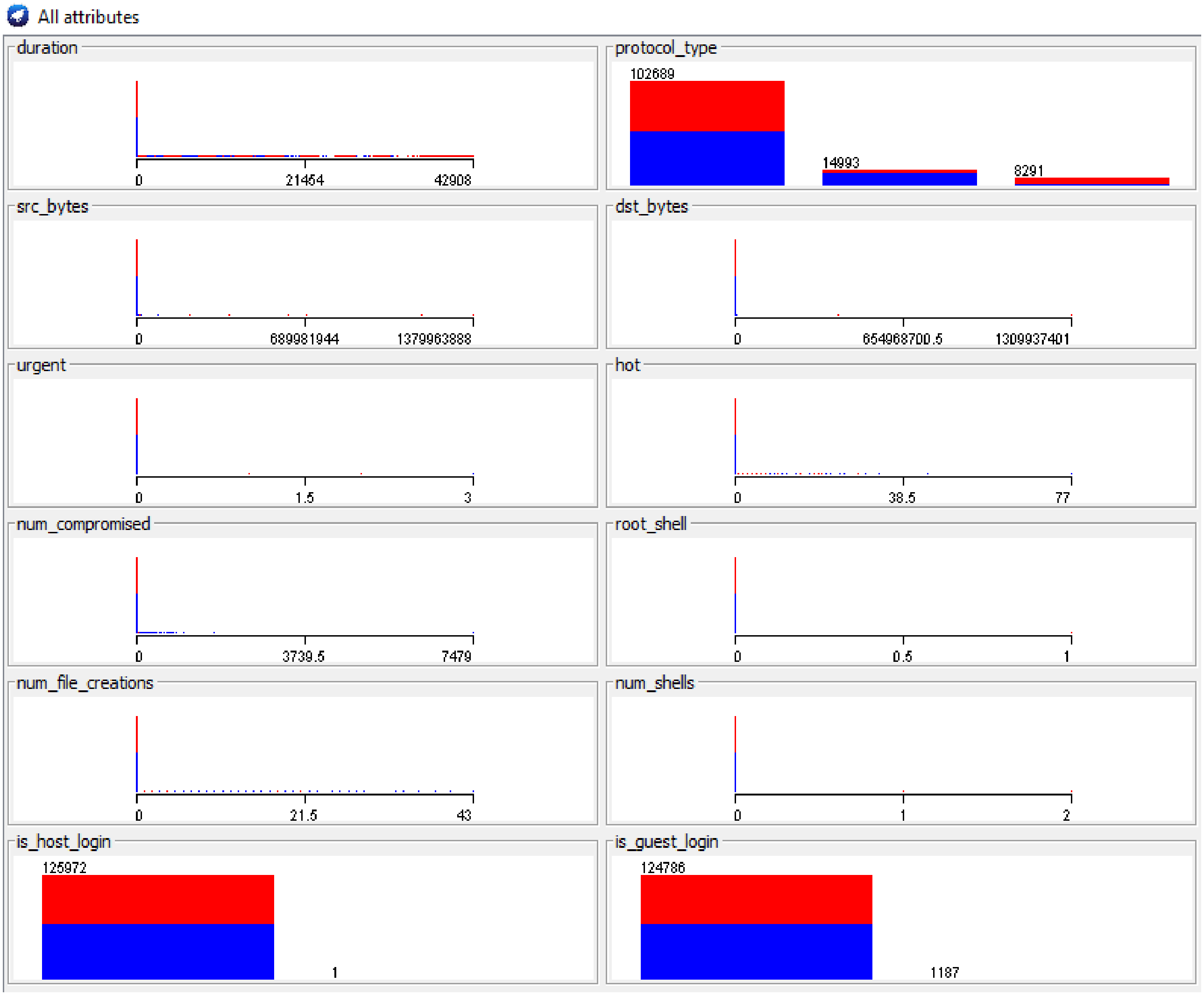

Fig.7. Visualizing of attributes

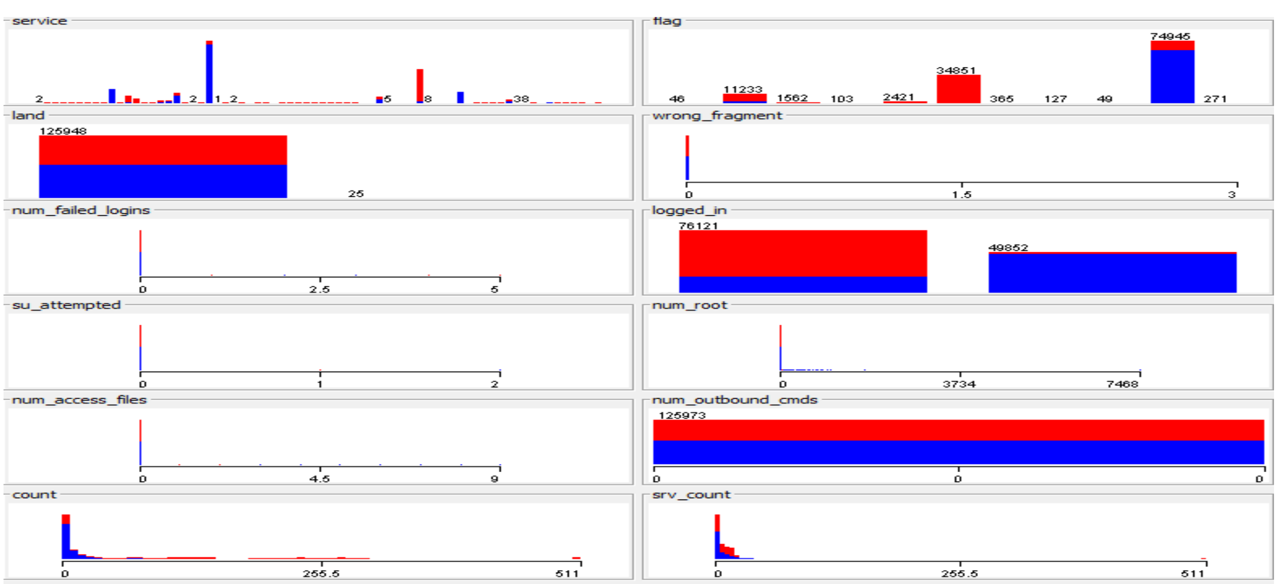

Fig. 8. Visualizing of attributes

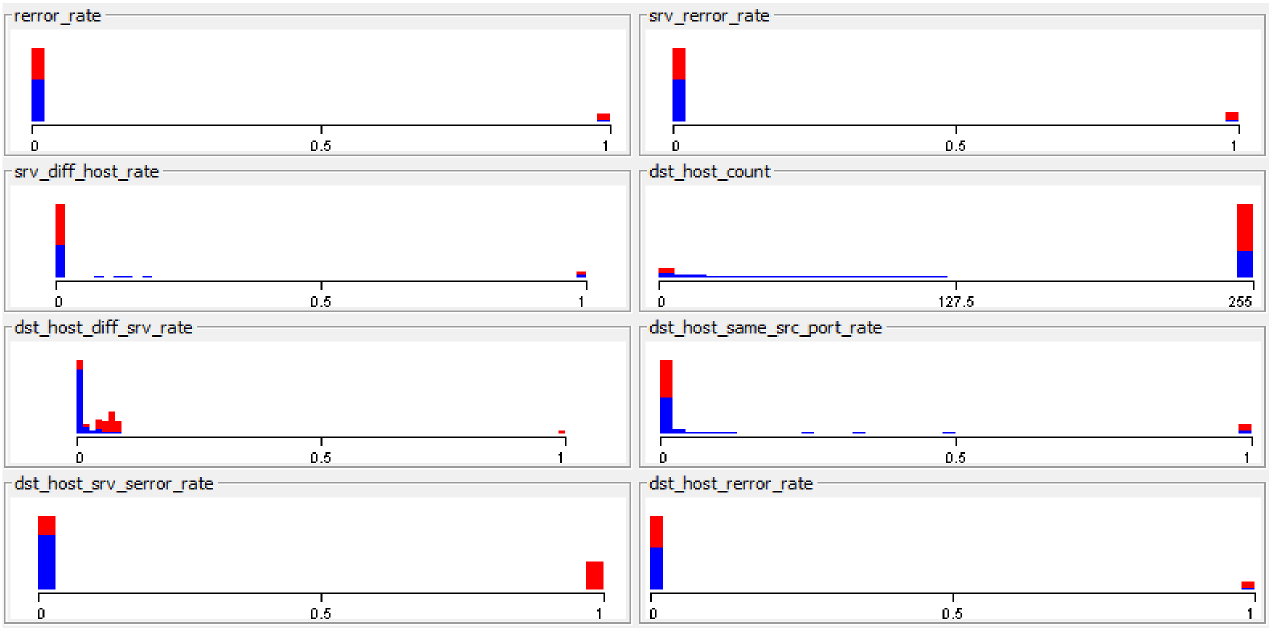

Fig. 9. Visualizing of attributes

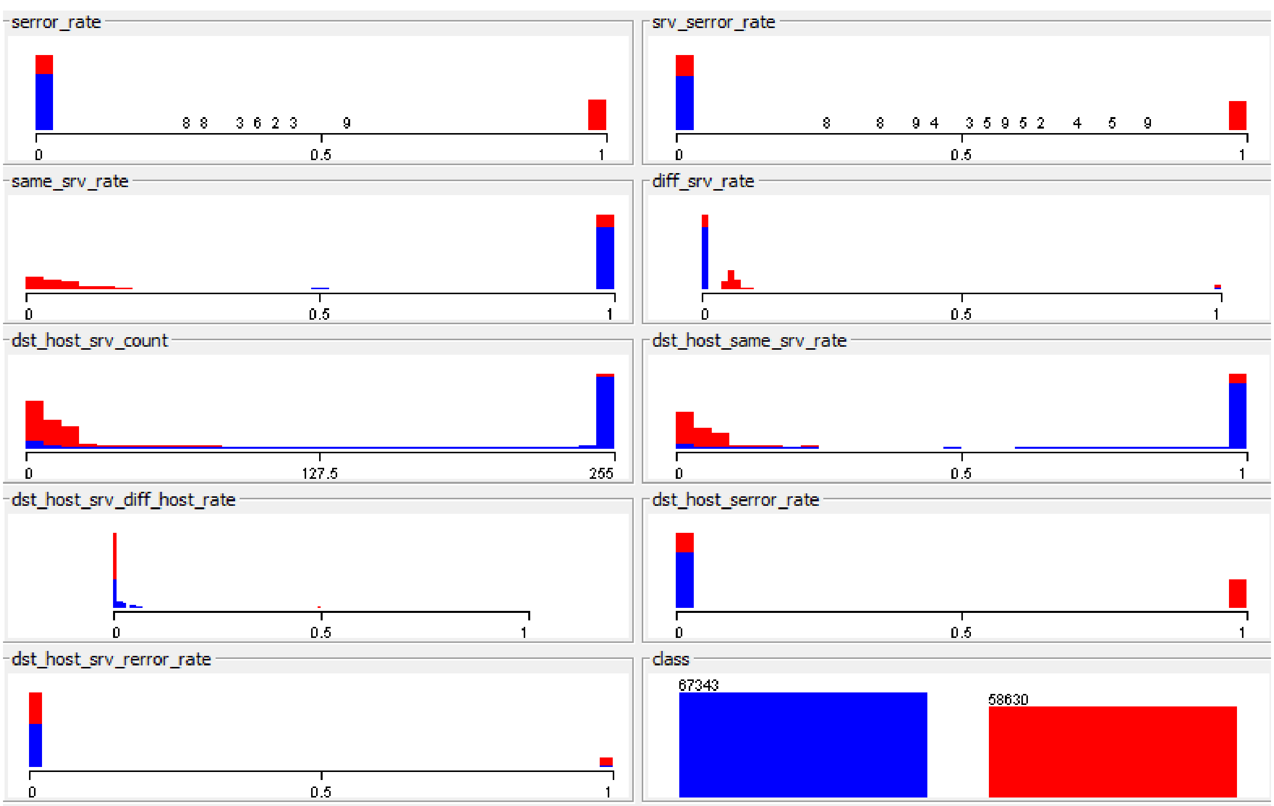

Fig. 10. Visualizing of attributes

4.2.1 Run Information

1. Scheme: weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
2. Relation: KDDTrain
3. Instances: 125973

Attributes:42:
"Duration,protocol_type,service,flag,src_bytes,dst_bytes.Land,wrong_fragment,urgent,hot,num_failed_logins,logged_in,num_compromised,root_shell,su_attempted,num_root,num_file_creations,num_shells,num_access_files,num_outbound_cmds,is_host_login,is_guest_login,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,same_srv_rate,diff_srv_rate,srv_diff_host_rate,dst_host_count,dst_host_srv_count,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate, class".

4. Test mode involves 10-fold cross-validation.
5. Classifier model (full training set) .
6. Random-Forest bagging with 100 iterations and base learner.
7. Time taken to build model: 90.31 seconds.
8. Stratified cross-validation.
9. Summary :

This is the summary of classification model evaluation metrics. Here's a quick breakdown of what these metrics typically indicate:

- **Correctly-classified-instances**: Out of 125,973 instances, 125,869 (99.9174%) were classified correctly. This reflects a high accuracy rate.
- **Incorrectly-classified-instances**: Only 104 instances (0.0826%) were classified incorrectly—an impressively low error rate.
- **Kappa-statistic**: At 0.9983, this indicates almost perfect agreement between the predicted and actual classifications.
- **Mean-absolute-error**: A value of 0.0028 suggests a very low average difference between predictions and actual values.
- **Root-mean-squared-error**: This metric is at 0.0285, showing that prediction errors are minimal on average.
- **Relative-absolute-error**: At 0.5704%, this indicates the mean error relative to the observed data range—a very small percentage.
- **Root-relative-squared-error**: At 5.7071%, this reflects the root mean squared error relative to the variance in the observed data.
- **Total Number-of-Instances**: There were 125,973 instances in total for evaluation.

These values indicate the model performs exceptionally well! Let me know if you'd like me to elaborate on any of these terms or discuss potential use cases.

10. Detailed accuracy by class : the accuracy of classification in random forest algorithm shown in the Table 4.

TABLE IV. ACCURACY OF CLASSIFICATION

	TP_Rate	FP_Rate	Precision	Recall	F_Measure	MCC	ROC_Area	PRC_Area	Class
	1.000	0.001	0.999	1.000	0.999	0.998	1.000	1.000	normal
	0.999	0.000	1.000	0.999	0.999	0.998	1.000	1.000	anomaly
Weighted Avg.	0.999	0.001	0.999	0.999	0.999	0.998	1.000	1.000	

11. Confusion Matrix : confusion matrix shown in the Table 5.

TABLE V. CONFUSION MATRIX

a	b	Classified as :
67319	24	a = normal
80	58550	b = anomaly

5. CONCLUSION AND FUTURE WORKS

It is evident that several applications in future 5G and SDN networks for which timely and accurate classification and characterization of traffic and applications are crucial, are becoming more critical. Through the use of payload-independent traffic data, methodologies that are based on machine learning are able to overcome some of the restrictions that are associated with classic classification methods in the context of network traffic characterization. This work

employs "machine learning" techniques using the "random forest" algorithm to categorize Internet traffic into general application types based on flow metrics derived from the NSL-KDD test and training files, which comprise 42 attributes. Feature choices were conducted prior to traffic classification. Results reveal that the "random forest" approach offers superior accuracy and demonstrates more computing efficiency. Analysis of classification findings using ten-fold cross-validation indicates a general classification accuracy of correctly classified instances are22252 means 98.7048 % while the Incorrectly Classified Instances are 292 means 1.2952 % for testing file and the time taken to build model is 8.64 seconds .The correctly classified instances are125869 means 99.9174 % while the Incorrectly Classified Instances are 104 means 0.0826 % for training file and the time taken to build model is 90.31 seconds. Within all traffic classifications, a properly enough level is attained for the diagnosis of Internet traffic. The robustness of the "random forest" algorithm model is assessed using a novel test dataset. Despite the deterioration of recall, precision, and F1-measure to certain extent, the median accuracy of classification for the primary application subcategories is still quite satisfactory. Experimental findings demonstrate that the classification model using "random forest" achieves superior precision for the predominant class (normal, anomalous), confirming that methodologies grounded on supervised learning algorithms often provide enhanced classification accuracy.
The future studies will focus on applying the classification technique described in this work on other traffic datasets that may have attributed that are different from the one used in this study. The application of traffic and usage recognition through machine learning techniques in wireless SDN networks to enhance the traffic optimization, the transferring rules of SDN, segmentation of wireless SDN network and flow QoS control is one of the promising research areas.

Conflicts of Interest

The paper states that there are no personal, financial, or professional conflicts of interest.

Funding

The absence of any funding statements or disclosures in the paper suggests that the author had no institutional or sponsor backing.

Acknowledgment

The authors extend appreciation to the kut university college for their unwavering support and encouragement during the course of this research.

References

[1] M. S. Sheikh and Y. Peng, “Procedures, Criteria, and Machine Learning Techniques for Network Traffic Classification: A Survey,” IEEE Access, vol. 10, pp. 61135–61158, 2022, doi: 10.1109/ACCESS.2022.3181135.
[2] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, “Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey,” IEEE Commun. Surv. Tutorials, vol. 21, no. 2, pp. 1988–2014, 2019, doi: 10.1109/COMST.2018.2883147.
[3] Z. S. Younus and M. Alanezi, “A Survey on Network Security Monitoring: Tools and Functionalities,” Mustansiriyah J. Pure Appl. Sci., vol. 1, no. 2, pp. 55–86, 2023, [Online]. Available: https://mjpas.uomustansiriyah.edu.iq/index.php/mjpas/article/view/33.
[4] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, “A review on machine learning – based approaches for Internet traffic classification,” pp. 673–710, 2020.
[5] T. S. Mohamed and S. Aydin, “IoT-Based Intrusion Detection Systems: A Review,” Smart Sci., vol. 10, no. 4, pp. 265–282, 2022, doi: 10.1080/23080477.2021.1972914.
[6] N. Abid, “Enhanced IoT Network Security with Machine Learning Techniques for Anomaly Detection and Classification Enhanced IoT Network Security with Machine Learning Techniques for Anomaly Detection and Classification,” no. December 2023, 2024, doi: 10.14741/ijcet/v.13.6.5.
[7] R. M. Alzoman and M. J. F. Alenazi, “Smart City Networks,” pp. 1–17, 2020.
[8] X. Du and M. Guizani, “CorrAUC : a Malicious Bot-IoT Traffic Detection Method in IoT Network Using Machine Learning Techniques,” vol. 4662, no. c, pp. 1–12, 2020, doi: 10.1109/JIOT.2020.3002255.
[9] A. A. Afuwape, Y. Xu, J. H. Anajemba, and G. Srivastava, “Performance evaluation of secured network traffic classification using a machine learning approach,” Comput. Stand. Interfaces, vol. 78, p. 103545, 2021, doi: https://doi.org/10.1016/j.csi.2021.103545.
[10] RUIZHEZHAO, “NSL-KDD,” IEEE DataPort, 2022.
[11] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[12] R. D. Ravipati, M. Abualkibash, and A. Computing, “A S URVEY ON D IFFERENT M ACHINE L EARNING A LGORITHMS AND W EAK C LASSIFIERS B ASED ON,” vol. 10, no. 3, pp. 1–11, 2019.
[13] B. Mahesh, “Machine Learning Algorithms - A Review,” no. October, 2020, doi: 10.21275/ART20203995.
[14] F. Y. Osisanwo, J. E. T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi, and J. Akinjobi, “Supervised Machine Learning Algorithms : Classification and Comparison,” no. June, 2017, doi: 10.14445/22312803/IJCTT-V48P126.
[15] D. B. Editors, Emerging Technology in Modelling and Graphics. 2018.
[16] C. Singh, “Supervised Machine Learning: Algorithms and Applications,” no. January, 2023, doi: 10.1002/9781119821908.ch1.
[17] A. R. Khan, M. Kashif, R. H. Jhaveri, R. Raut, T. Saba, and S. A. Bahaj, “Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions,” Secur. Commun. Networks, vol. 2022, 2022, doi: 10.1155/2022/4016073.
[18] M. Suyal and P. Goyal, “A Review on Analysis of K-Nearest Neighbor Classification Machine Learning Algorithms based on Supervised Learning,” no. November, 2023, doi: 10.14445/22315381/IJETT-V70I7P205.
[19] H. A. Salman, A. Kalakech, and A. Steiti, “Random Forest Algorithm Overview,” vol. 2024, pp. 69–79, 2024.
[20] V. Y. Kulkarni, “Random Forest Classifiers : A Survey and Future Research Directions,” vol. 36, no. 1, pp. 1144–1153, 2013.
[21] L. E. O. Breiman, “Random Forests,” pp. 5–32, 2001.
[22] geeksforgeeks, “Random Forest Algorithm in Machine Learning,” geeksforgeeks, 2025. https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/.
[23] T. S. Mohamed, S. Aydin, A. Alkhayyat, and R. Q. Malik, “Kalman and Cauchy clustering for anomaly detection based authentication of IoMTs using extreme learning machine,” no. April, pp. 1–14, 2022, doi: 10.1049/cmu2.12467.