International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Design and Implementation of Machine Learning-Based Network Intrusion Detection

Dr. Srinivas Ambala[1], Dr. Anirudh Krushna Mangore[2], Dr. Mubin Tamboli[3], Dr. Satpalsing Devising Rajput[4], Dr. Shwetambari Chiwhane[5], Amol Dhumane[6]

Submitted: 19/07/2023 **Revised**: 09/09/2023 **Accepted**: 26/09/2023

Abstract: Systems for detecting intrusions are vital to network security and are necessary for maintaining network integrity. To improve the effectiveness of machine learning techniques, ensemble learning has been frequently used. Additionally, the quality of training data has a significant impact on detecting abilities. Marginal density ratios have consistently outperformed the powerful unilabiate classifiers. In this paper, we recommend a system for detecting intrusions that is based on groups of SVMs and has been functionally enhanced. Our method involves transforming the original characteristics to marginal density logarithmic thresholds in order to produce new, enhanced, and modified training data. The creation of an intrusion detection model then takes place using an SVM-Set. The system employs a number of machine learning techniques, including ensemble learning, to enhance detection performance. Raising the caliber of training data also involves the use of feature augmentation. Using an ensemble of Support Vector Machine (SVM) models, the suggested approach creates an effective intrusion detection framework. The effectiveness of the proposed design was assessed using simulations and the data base CICIDS2017, which simulates network traffic in the real world. The results of the experiment were compared to earlier studies, and it was found that the precision of binary and multiclass categorization had increased. Another illustration of the efficiency of the model was the high level of precision of the restored transportation system.

Keywords*: Network intrusion detection, Cyber security, Support vector machine, ensemble learning*

1. Introduction

In today's highly interconnected world, network security is an extremely important topic of discussion due to the heavy reliance that businesses place on computer networks to both store and transmit critical information. However, because of the ever-increasing complexity of online dangers, conventional security measures are becoming increasingly ineffective. The monitoring of system traffic and the detection of unusual or malicious activity are two essential functions played by intrusion detection systems (IDS) in the detection and prevention of system attacks. Another key role played by IDS is the identification of potential vulnerabilities in the system. The automatic learning techniques have recently attracted attention in the field of intrusion detection due to their ability to recognize complex undiscovered attacks. In particular, the combination of several classifiers to increase the detection's robustness and precision has demonstrated promising results. The quality of training data affects how well intrusion detection systems perform. High-quality training data that accurately captures the underlying patterns and characteristics of these attacks is necessary for the classification of network attacks.

The system for detecting network intrusions proposed in this paper uses ensemble learning and feature augmentation approaches. The system's overall performance and detecting skills are to be improved. We particularly concentrate on using Support Vector Machine (SVM) ensemble models, which are well-known for their efficiency in dealing with high-dimensional data and nonlinear classification issues. The approach of feature augmentation, which uses logarithm marginal density ratios to change the original features, is also something we introduce. With the help of this transformation, it will be easier to distinguish between legitimate network traffic and criminal activity by producing new and enhanced training data. Through comprehensive testing and evaluation, we show the effectiveness and market advantages of our suggested strategy.

[1]Associate Professor, Department of Computer Engineering, Pimpri Chinchwad College of Engineering, Pune, Maharashtra, India
[2]Associate Professor and HOD, Computer Engineering Department, Gharda Institute of Technology, Level, Ratnagiri, Maharashtra, India
[3]Associate Professor, Pimpri Chinchwad College of Engineering, Pune, Maharashtra, India
[4]Assistant Professor, Pimpri Chinchwad College of Engineering Nigdi, Pune, Maharashtra, India
[5]Assistant Professor, Computer Science and Engineering, Symbiosis Institute of Technology, Pune, Maharashtra, India
[6]Associate Professor, Computer Science and Engineering, Symbiosis Institute of Technology, Pune, India
ambala.srinivas@pccoepune.org[1], anirudhmangore@gmail.com[2], mubin.tamboli@pccoepune.org[3], rajputsatpal@gmail.com[4], shwetambari.chiwhane@sitpune.edu.in5, amol.dhumane@sitpune.edu.in[6]

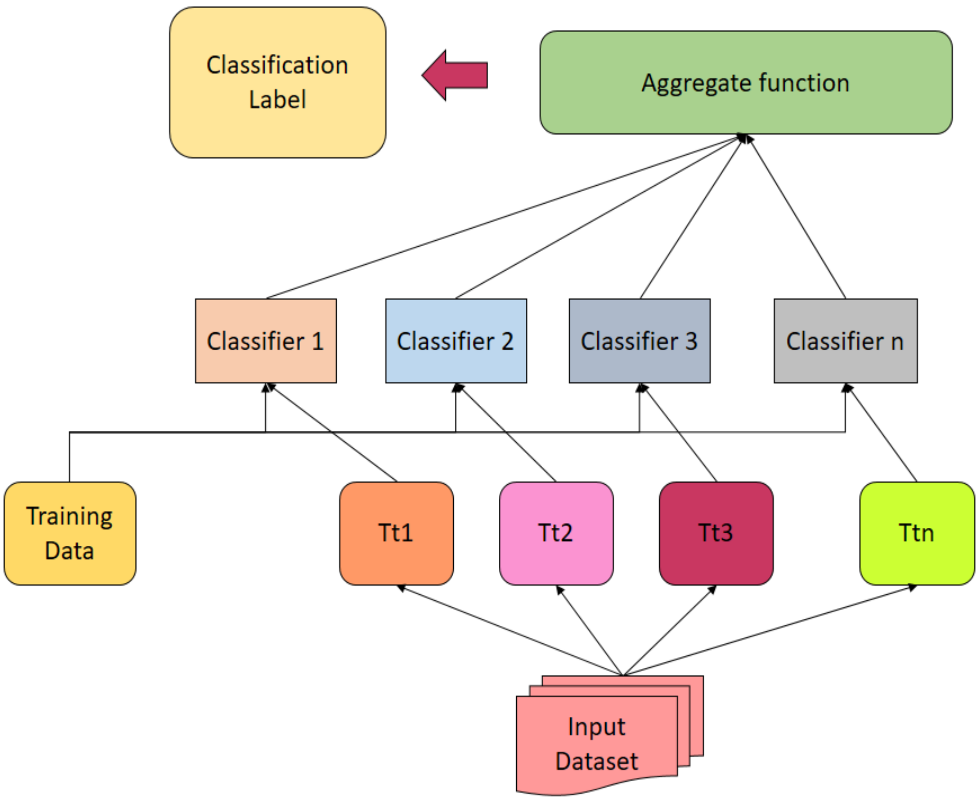

Fig 1: Structural representation of detailed classifier

Studies already conducted in the intrusion detection field emphasize the importance of learning algorithms and data quality in deciding how well intrusion detection approaches perform. The DT-EnSVM intrusion detection framework, which combines ensemble learning and data transformation techniques, is presented in this research. Our method starts with ratio transformations to produce high-quality training data, then base learners are trained using the highly efficient Support Vector Machine (SVM). An ensemble-based detection of intrusion model is created by combining these SVM classifiers using a nonlinear combination method. Our suggested method outperforms current techniques in terms of precision, detecting rate, false fright rate, and learning speediness, as demonstrated by empirical findings on the NSL-KDD dataset plus two additional detection datasets.

Compared to previous work, this paper introduces four important additions to the field of intrusion detection systems (IDS). First, we combine a strategy for data quality enhancement with ensemble learning to create an IDS framework that is accurate and robust. Second, we thoroughly compare the detection accuracy of classifier ensembles and individual base classifiers. Thirdly, we optimize the differences between training datasets for the base learners by using fuzzy c-means clustering to improve the ensemble's performance. Finally, the adaptability of our suggested methodology enables it to be used for a variety of real-world classification-related applications, including spam detection and credit scoring.

2. Review of Literature

Numerous studies have demonstrated that the performance of classifier ensembles or numerous classifier systems is superior to that of a single classifier. In recent years, there has been a rise in interest in the utilisation of group learning strategies as a means of improving the overall performance of autonomous learning models. Collective education, as opposed to traditional techniques of automatic learning, which try to derive a single hypothesis from training data, creates a number of hypotheses and integrates them to get to a conclusion. Traditional methods of automated learning have been around for a long time. The solo method to education is substantially less well-suited to generalisation than the team-based approach, which is one of the reasons why the latter is particularly exciting. Team-based education typically involves a sizeable number of interns. The combination of a number of different approaches has proved that it is advantageous in a range of settings, such as the detection of intrusions. In light of this, it is possible that the use of group learning as an effective intrusion detection strategy could be considered.

Particularly when working with big and complicated data sets, intrusion detection systems must place a significant emphasis on the quality of the data that is being received. In order to obtain formation data of a high quality, it is essential to either change the raw data or recreate them from scratch. This is due to the intrinsic complexity of the formation. The results of the [15] study offered evidence for the idea that the accuracy of the data may be enhanced by making adjustments to the proportions. Their research that was based on NSL-KDD shown a significant increase in detection accuracy. In light of these findings, adjustments to the ratios involved need to be made in order to make our detection method more efficient.

Only [17] are the only ones that we are aware of who have attempted to combine ratio transformations into a single SVM model in the past. However, when compared to the technique that we have described, they fall short in a number of important respects. To begin, the intrusion detection model is constructed fundamentally differently than other models. In contrast to our suggested approach, which is based on an ensemble model that incorporates a number of different detection models, Wang's method only makes use of a single detection model to make their determinations. There is a high probability of finding heterogeneities in data samples, particularly in data streams, which present the opportunity for idea drift. Deep neural networks, also known as DNNs, have demonstrated that they are capable of producing more accurate classifications when compared to more traditional forms of supervised machine learning. The effectiveness of many deep learning algorithms is, however, constrained by the high time complexity of the techniques.

We based our research on the autoencoder (AE) model and applied it in real-world intrusion detection system (IDS) environments. The AE model served as our primary source of motivation. To do this, we will first use AE to reconstruct the input features, and then we will convert those features into a hyperspace representation that faithfully reflects the essential components of the input data. By utilising this strategy, the difficulty of training and the far-reaching effects of acquiring extra abilities can both be mitigated. In addition, we integrated supervised machine learning with AE, which resulted in a considerable improvement to the overall performance of the classification task. The data are given the appropriate level of preparation by the preparation module, which uses the appropriate methodologies. After being preprocessed, the information is passed to an autoencoder (AE) module, which compresses it using a stacked autoencoder (SAE) model. This results in lower-dimensional reconstruction features. After that, the classification module will employ these attributes in order to produce classification results.

It is essential to note that the database module, which is sometimes referred to as the feature library, saves the compressed characteristics of each network traffic. This feature library can be put to a variety of different uses. To begin, it simplifies the process of testing and retraining the categorization module by making use of the features that were previously saved. Furthermore, it makes it easy for these features to be returned to their previous traffic, which simplifies the process of post-event analysis and forensics.

Contribution of Paper:

- To maintain the high accuracy of the classification method, the classifier received ongoing training through the reference and base methods.
- Through the use of effective representations and the reduction of dimensionality, our strategy maximizes the results of classification for the standard automatic learning algorithms in single- and multiple-classification.
- We reconstruct the flow after compression, which may be utilized for further evaluation and forensics, making full advantage of the SAE model's capabilities as well as the feature library in the database module.

3. Cicids2017 Dataset

When defending against sophisticated attacks on a network, the implementation of systems for the detection and prevention of intrusions (IDSs and IPSs) is absolutely necessary. However, the efficiency of anomaly-based intrusion detection methods is typically undermined due to a lack of trustworthy test and validation datasets. Our examination of eleven pre-existing datasets from 1998 reveals both the datasets' flaws and their lack of dependability. Many of these datasets do not have sufficient variety or network traffic volume to account for all of the many forms of attacks that are known. In addition, some datasets hide the payload information of packets, which misrepresents the kind of assaults that are happening right now. In addition, certain datasets are missing important features and metadata, which are prerequisites for conducting an in-depth study [30].

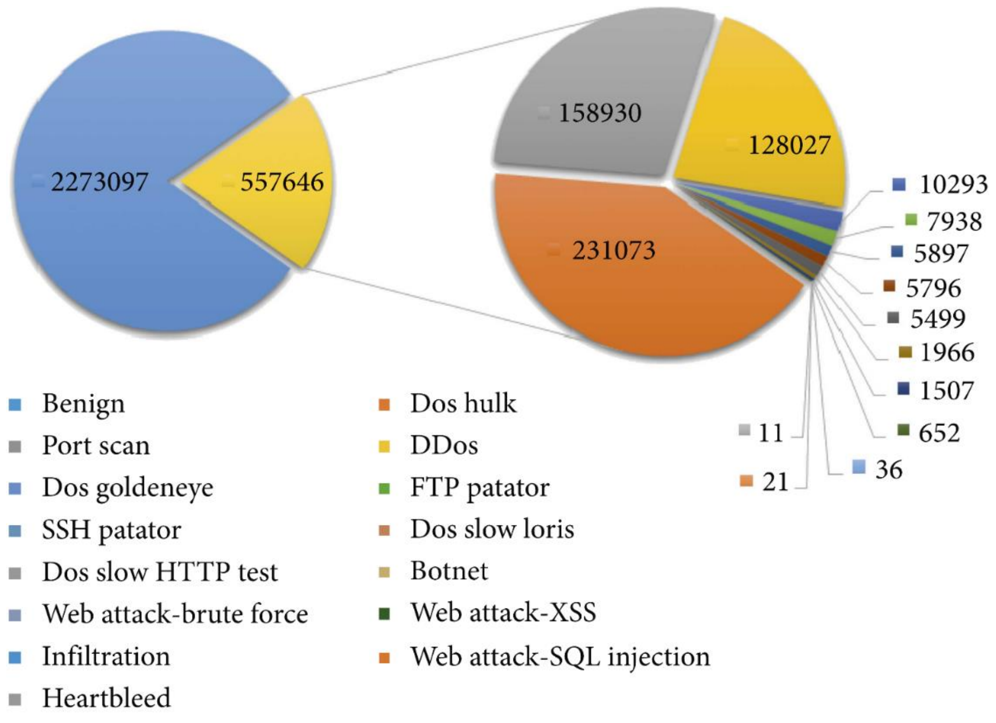

Fig 2: Representation of Dataset

In the most recent version of our approach for assessing datasets, we have identified eleven essential criteria that must be addressed in order to construct a reliable benchmark dataset. The older IDS datasets did not fulfil any of these conditions at any point in time. The condensed list of these prerequisites includes the following items:

Detailed Network Topology The dataset includes a comprehensive network topology that consists of a variety of network devices, including firewalls, modems, switches that are needed, and routers, as well as several different operating systems.

Complete Traffic: The 12 individual computers that make up the Victim-Network, along with the user profiling agents and actual attacks that come from the Attack-Network, produce a vast range of traffic.

Labelled Dataset: The dataset has labels for both excellent traffic and different kinds of attack types. A considerable deal of information is also provided regarding the times of the strikes.

Complete Interaction: The dataset logs connections made over the Internet as well as local area network (LAN) communications that take place within the network.

Complete accumulate: A mirror port or tapping system is utilised on the storage server in order to collect and record all network traffic. This helps to ensure that complete data capture has taken place.

Diversity in Attacks: The dataset, which is based on research conducted by McAfee in 2016, includes a wide range of popular attack types, including Web-based, Brute force, DoS, DDoS, Infiltration, Heart-bleed, Bot, and Scan attacks.

Because the network activity from the most important switch, memory dumps, and calls to the system from all of the victims are all recorded as the attacks are being carried out, the dataset is guaranteed to be diverse. This is because of the recording of all of these events simultaneously.

CICFlowMeter is used to extract more than 80 network flow features from the generated network data. These features may be found in the Feature Set. The dataset is provided in the form of a CSV file, and a PCAP analyzer and CSV generator are both available for use.

Metadata: The dataset is detailed in great length in the work that has been published. This includes information on time, assaults, flows, and labels.

4. Proposed System

We outline the workflow and elements of our suggested imposition finding framework in this section. The preprocessing component, automatic encoder module, data module, categorization module, and feedback module are the five main components of the system. Together, these modules form a strong intrusion detection framework with excellent accuracy and minimal training requirements. Figure 1 depicts the proposed framework, where different functions are represented by different colour lines. The primary detection process is represented by the black line, while the retraining process is shown by the orange line. The processes that interact with other functions are depicted by the blue two-way arrows, while the restoration function is represented by the green line.

1. Detection of Data function:

The preprocessing module processes the raw network traffic that the data collector has collected. Following feature extraction from the preprocessed data, the autoencoder module reconstructs a low-dimensional feature representation. The categorization module receives this reconstructed feature as an input. The reconstructed feature is simultaneously saved in the database module. This feature is used by the trained classifier to generate the final output results and make predictions within the classification module. Our platform efficiently gathers and analyses network traffic using this workflow, extracts useful characteristics using autoencoder methods, and uses a trained classifier to categorize and identify probable intrusions. The database module also acts as a repository for the reconstructed features, allowing for additional analysis and research.

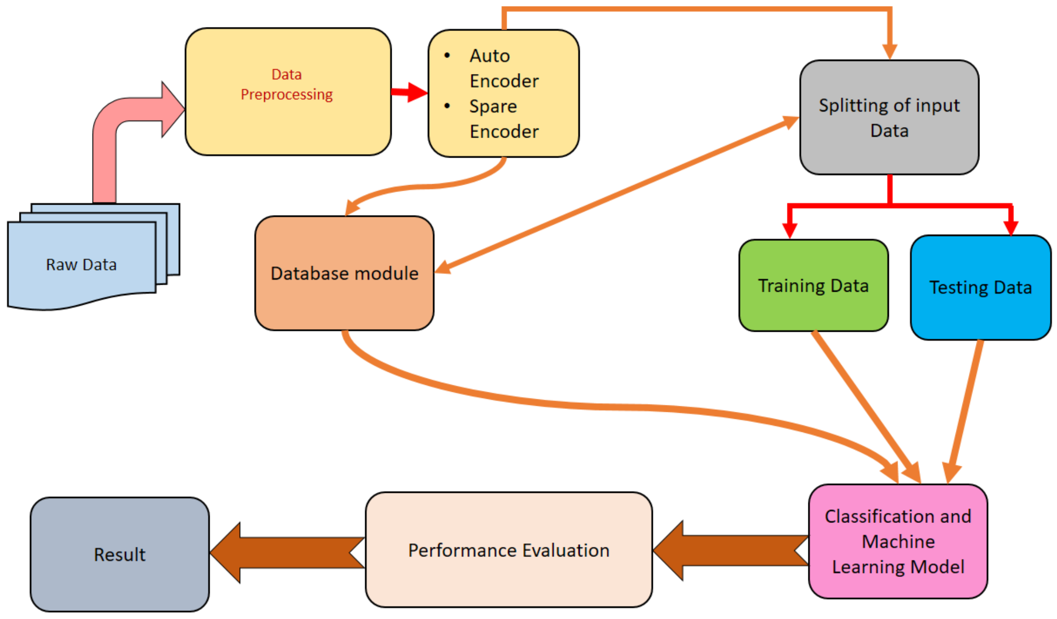

Fig 3: Proposed system for intrusion detection

2. Sparse Autoencoder (SAE):

A Stacked Autoencoder (SAE)'s fundamental training diagram is shown in Figure 2. Let's take into account N input nodes, N output nodes, and M hidden layer nodes. Reconstructing the input (x) to make it equal to the output (y), or x y, is the aim of the SAE. The SAE achieves this by condensing the input (x) into a representation made up of one or more lower-dimensional hidden layers (a). The reconstructed output (y) is then mapped back to this concealed representation.

Applying the appropriate weights (W), biases (b), and an activation function (f) to the input values (x) and propagating them through the hidden layers can calculate the output of the neurons in each layer. We minimize the loss function to determine the ideal values for the weight matrix (W) and bias vector (b) in a Stacked Autoencoder (SAE) using the backpropagation algorithm. Typically, the loss function is shown as follows:

$$L(Wt, bt) = \frac{1}{Nt} * \Sigma \left||xt - yt|\right|^2$$

In this equation, the terms x and y stand for the input and output data, respectively, while N stands for the number of training samples and || ||2 stands for the squared Euclidean distance between x and y. By changing the weights and biases in the SAE, the objective is to minimize this loss function. Using the iterative backpropagation technique, which includes computing the gradients of the loss functions with respect to the weights and biases, we can alter the values of W and b to reduce the overall loss. The SAE learns the best representations thanks to this repeated optimization process, which also increases output reconstruction accuracy.

```
$Input: training\ dataset$
$Output: trained\ SAE\ model$
$Initialization:$
$Step\ 1: perform\ forward\ propagation\ on\ all\ input\ samples$
$Step\ 2: calculate\ the\ output\ of\ each\ node\ a\ in\ the\ hidden\ layer$
$Step\ 3: calculate\ the\ output\ error\ of\ the\ cost\ function$
$Step\ 4: updating\ the\ weights\ and\ biases\ of\ each\ layer\ using\ the$
$Step\ 5: repeat\ Step\ 2, 3, and\ 4\ until\ the\ reconstruction\ error\ is$
```

End

5. Classification Techniques

The Random Forest (RF) algorithm was the primary classification technique in your experiment's classification module. An ensemble technique called Random Forest uses several different decision trees. Every decision tree in the forest is driven by a subpopulation that is randomly distributed in terms of formation and characteristic data. There are several benefits to using forests that are used haphazardly as opposed to those that are based on individual decisions. The model's variance decreases as the number of trees in the forest increases, while the bias stays constant. This indicates that random forests are less prone to overfitting and have a stronger ability to generalize to new data. In comparison to other models, random forests require less parameter tweaking, making them simpler to operate. On the other hand, a decision tree (DT) resembles an method since each node without parameter indicates a test of an attribute and each nut displays the results of that test for a given range of values. The categories or values connected to the decision tree are contained in the nodules of the parameter. You evaluate the relevant feature attribute for the object to be classified starting at the root node, and then you choose the relevant branch up until the leaf node based on its value. This process is known as employing a decision tree. The final choice is then based on the category or value kept in the leaf node.

A.CART Algorithm:

The method of binary segmentation that is utilised in CART (Classification and Regression Tree) splits the data into two halves, which then serve as the basis for the left and right subtrees. Due to the fact that each non-leaf node has two children, the tree has one more leaf node than it does non-leaf nodes. In CART classification, selecting the most useful characteristics for data partitioning is done with the help of a statistic called the Gini index. The Gini index is a statistical tool that determines the degree of impurity or lack of homogeneity in a given collection of samples. A lower Gini index is indicative of more purity and better characteristics for classification when compared to higher indices. After the data have been partitioned, the Gini index is lowered to its lowest possible value, and the attribute that achieves this lowest Gini index is selected as the most desirable subattribute.

•

In CART, an evaluation of the impureness of a particular decision tree node is performed by computing the Gini index. In order to compute the Gini index for a given node within the context of the classification, the following equation is utilised.

$$\text{Gini Index} = 1 - \Sigma((\text{pk})^2)$$

Where:

- Σ stands for the total of all classes.
- The likelihood that an item in the node belongs to class k is expressed as p_k.

The Gini index expression for sample set Dt can be stated as follows under the condition of feature A if the sample set Dt is divided into n parts |Dt1|, |Dt2|,..., |Dtn| depending on a particular value of feature A:

$$\text{Gini}(\text{Dt},\text{At}) = \left(\frac{|\text{Dt1}|}{|\text{Dt}|}\right) * \text{Gini}(\text{Dt1}) + \left(\frac{|\text{D2}|}{|\text{Dt}|}\right) * \text{Gini}(\text{Dt2}) + \dots + \left(\frac{|\text{Dtn}|}{|\text{Dt}|}\right) * \text{Gini}(\text{Dtn})$$

Where:

- Gini (Dt, At) represents the Gini index of sample set D under the condition of feature A.
- |Dti| represents the number of samples in the ith part of the division.
- Gini (Dti) represents the Gini index of the ith part of the division.

Algorithm 1: CART algorithm:

```
$\textbf{Input}: training\ dataset\ D$
$\textbf{Output}: CART$
$N: threshold\ of\ the\ number\ of\ samples\ in\ node. n: number\ of\ samples\ in\ the\ node$
$G: Gini\ index\ threshold\ for\ D$
$Gini\ (D): Gini\ index\ of\ D$
$Based\ on\ D, starting\ from\ the\ root\ node$
$if\ n < N\ or\ Gini\ (D) < G\ or\ no\ more\ features$
$recursively\ perform\ the\ following\ operations\ on\ each\ node\ to\ construct\ a\ binary\ tree$
```

```
$For\ each\ feature\ A, for\ each\ of\ its\ possible\ values\ a$
$The\ split\ will\ be\ into\ D1\ and\ D2\ based\ on\ whether\ the\ test$
$for\ A = a, and\ use\ equation\ (6)\ to\ calculate\ Gini\ (D, A)$
$Among\ all\ possible\ features\ A\ and\ all\ its\ possible\ segmentation\ points\ a,$
$The\ feature\ with\ the\ smallest\ Gini\ index$
$and\ its\ corresponding\ segmentation\ point\ are\ selected\ as\ the\ optimal\ feature$
$The\ optimal\ segmentation\ point$
$Generate\ two\ subnodes\ from\ the\ current\ node\ and\ assign$
$The\ training\ dataset\ to\ the\ two\ subnodes\ according\ to\ the\ features$
$\textbf{Return}\ CART$
```

Algorithm 2: Tree Classifier Generation

```
$\textbf{Input}: training\ dateset$
$\textbf{Output}: tree\ structured\ classifier$
$S: number\ of\ training\ samples$
$M: number\ of\ features. m: number\ of\ features\ input\ (m << M)$
$N: number\ of\ trees\ generated$
$If\ the\ tree\ to\ be\ generated\ is\ less\ than\ N,$
    $Step\ 1: from\ the\ S\ training\ samples, take\ samples\ S\ times\ in\ a\ way\ with\ a\ put$
            $-\ back\ sampling\ to\ form\ a\ training\ set$
    $Step\ 2: use\ unselected\ samples\ to\ make\ predictions\ and\ evaluate\ their\ errors$
    $Step\ 3: for\ each\ node, m\ features\ are\ randomly\ selected$
    $Step\ 4: according\ to\ these\ m\ features, calculate\ the\ best\ split\ method$
    $Step\ 5: grow\ to\ be\ largest\ extent\ possible\ without\ pruning$
$\textbf{Return}\ tree\ structured\ classifier$
```

B. Evaluation Metrics:

The accuracy (ACC) is calculated as the percentage of correctly classified instances, whether they are normal or attacks, and is determined by the following formula:

$$ACC = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

The formula for calculating precision (P), which is the proportion of pertinent instances among the identified instances:

$$P = \frac{TP}{(TP + FP)}$$

Recall (R) is calculated as the ratio of the number of relevant instances over the total number of relevant instances discovered:

$$\mathrm{R} = \frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})}$$

The F1-Score is a metric that combines recall and precision into one number. It can be calculated using the formula below as the weighted average of recall and precision:

$$\mathrm{F1Score} = \frac{(2 * \mathrm{P} * \mathrm{R})}{(\mathrm{P} + \mathrm{R})}$$

In particular, when $\alpha = 1$, the formula for the F1-Score simplifies. Overall, these formulas allow us to calculate accuracy, precision, recall, and the F1-Score, which are commonly used metrics for evaluating classification performance.

6. Result and Discussion

In order to assess the effectiveness of our proposed benchmark in binary and multiclass classification and to evaluate the effectiveness of the small-scale characteristics identified by our methodology, our experiment was conducted using the CICIDS2017 data source. In order to assess the validity of our model, we also compute the testing and capacity times. The techniques work in binary and multiclass classification when several parameters are used (here, we take DT to be a model of RF). The development and testing times for binary and multiclass classification algorithms with various setups are presented in Tables 4 and 5. Except for DT, several parameterization techniques operate in disparate ways in these graphs and tables.

The suggested approach yields encouraging results in terms of precision, recall, accuracy, and F1-score. Table 2 provides a summary of the full results for binary classifications of traffic into benign and pathological. Figure 5 also shows the corresponding confusion matrix, which shows how well the classification worked. 681,564 of the total samples analysed were accurately classified as benign traffic, while 167,012 of the total samples evaluated were correctly classified as aberrant traffic.

Table 1: Comparison of Training and Testing time for algorithms

Algorithms	Training time (sec)	Testing time (sec)
RF-20 trees	256.789	15.96
DT	430.519	12.64
RF-30 trees	512.858	226.34
RF-60 trees	1276.363	63.65
RF-110 trees	2556.573	22.09

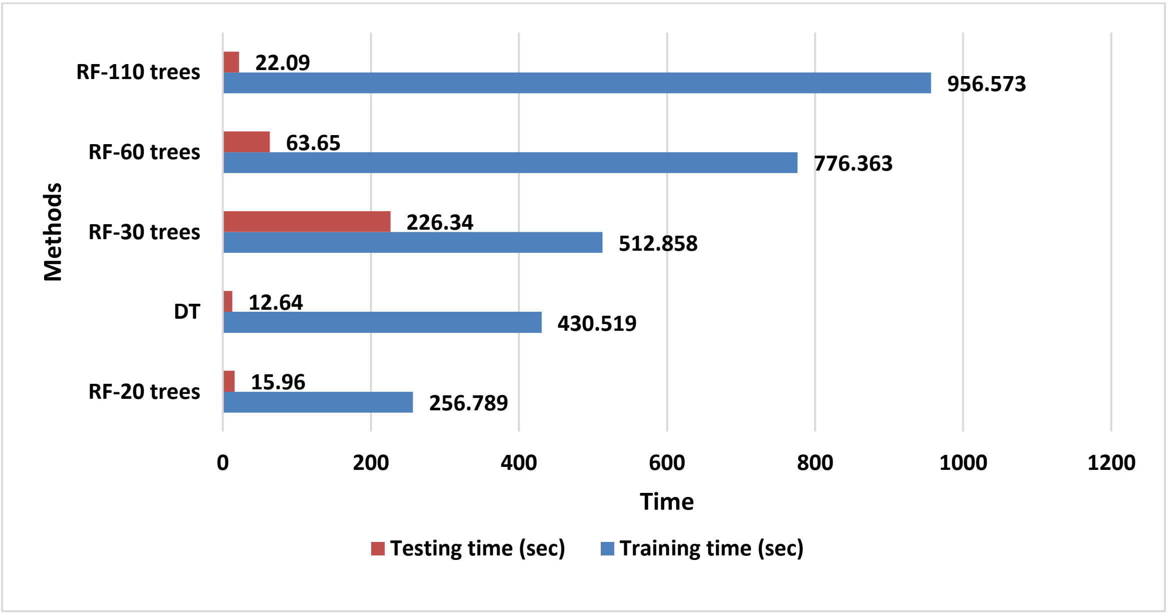

Fig 4: Representation of Training and Testing time for algorithms on Dataset

The evaluation results are shown in Tables 1 and 2, which indicate the framework's capacity to recognize 15 different types of attacks. Even though Infiltration, Heartbleed, and SQL Injection have a smaller representation in the dataset than other attack samples, the method correctly recognizes these attack types. The confusion matrix in Figures 4 and 5 illustrates how the framework occasionally misclassifies samples. This is a crucial point to keep in mind. The methodology, however, shows a stronger propensity to classify attack-related traffic as opposed to benign traffic, showing a priority for identifying possible threats.

Table 2: Comparison of Performance metrics for different methods

Algorithms	**Accuracy in (%)**	**Precision in (%)**	**Recall in (%)**	**F1-Score in (%)**
RF-20 trees	98.12	97.23	98.87	98.88
DT	98.87	98.01	99.1	98.72
RF-30 trees	99.11	98.01	98.66	98.81
RF-60 trees	99.65	97.33	98.44	98.76
RF-110 trees	99.77	99.43	99.78	99.65

The RF-20 trees algorithm accurately identified 98.12% of the samples, according to its accuracy score of 98.12%. Additionally, it showed a precision of 97.23%, indicating that 97.23% of the samples it correctly identified as positive actually were positive. The recall, or true positive rate, was 98.87%, meaning that 98.87% of the positive samples were correctly identified by the algorithm. The precision and recall-combining F1-score was determined to be 98.88%. The accuracy of the DT algorithm was 98.87%, which was slightly superior performance. It demonstrated a precision of 98.01%, correctly categorizing 98.01% of the positive samples. The system successfully detected 99.1% of the true positive samples, as seen by the 99.1% recall. The DT algorithm's F1-score was calculated to be 98.72%.

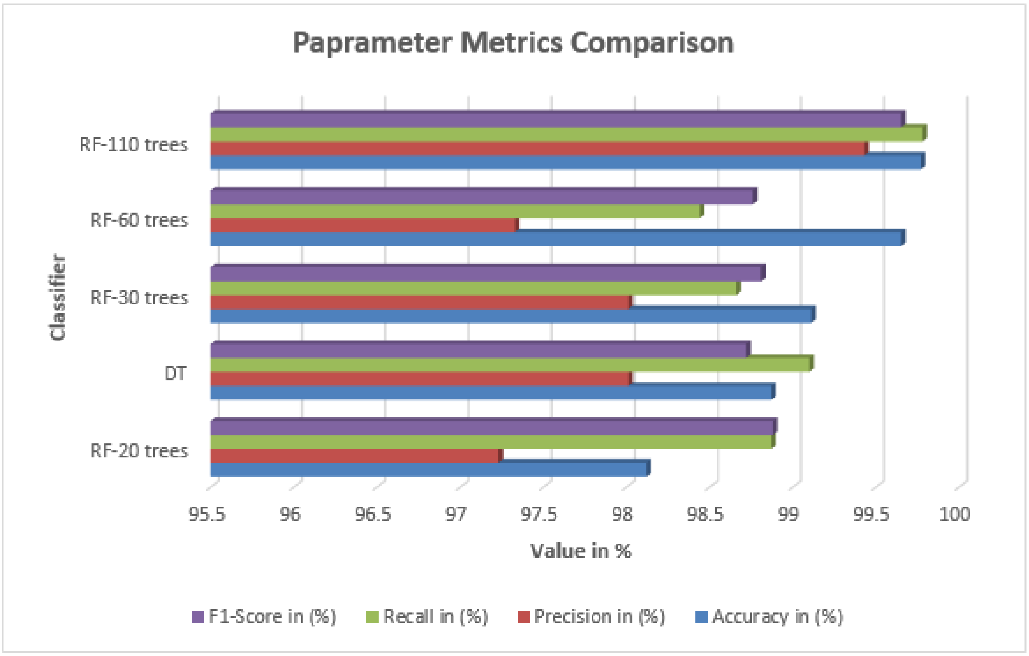

Fig 5: graphical representation of Comparison of Performance metrics for different methods

With a score of 99.11%, the RF-30 trees algorithm had even higher accuracy. A precision of 98.01% meant that 98.01% of the positive samples were accurately identified. The algorithm was able to correctly identify 98.66% of the true positive samples, as evidenced by the recall of 98.66%. This algorithm's F1-score was calculated to be 98.81%. The performance of the RF-60 trees algorithm was superior to the earlier ones, with an astounding accuracy of 99.65%. It showed a 97.33% precision, which means that it correctly identified 97.33% of the positive samples. The algorithm was able to recognize 98.44% of the true positive samples, as evidenced by the recall, which was 98.44%. This algorithm's F1-score was calculated to be 98.76%.

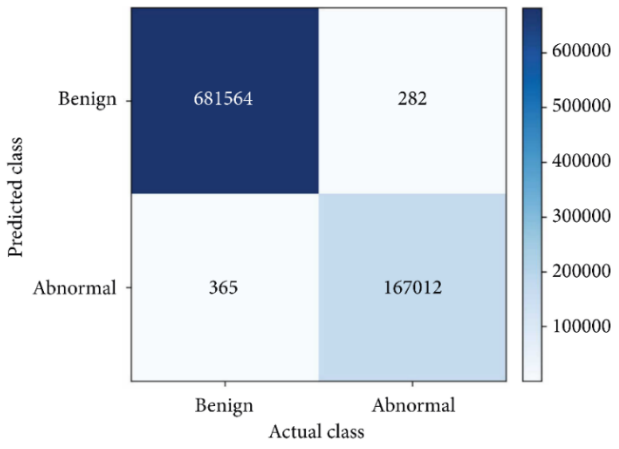

Fig 5: Confusion matrix of classification method

With a score of 99.77%, the RF-110 trees algorithm had the most accurate results of any algorithm. The excellent precision of 99.43% showed that 99.43% of the positive samples were accurately identified. The method was able to correctly classify 99.78% of the genuine positive samples based on the recall, which was 99.78%. This algorithm's F1-score was calculated to be 99.65%. These outcomes demonstrate how well the suggested algorithms categorize various types of traffic with accuracy. The algorithms usually shown improved performance across all assessment measures as the number of trees in the random forest ensemble rose, with the RF-110 trees method receiving the highest ratings.

7. Conclusion

In order to improve the security of computer networks, machine learning-based network intrusion detection has been designed and put into use. Accurate classification and detection of network intrusions have been accomplished by using algorithms like Random Forest and Decision Trees. The findings show that the suggested framework offers good precision, recall, accuracy, and F1-score, showing its capacity to correctly recognize and categorize various sorts of traffic, including normal and aberrant patterns, as well as different types of attacks. The Random Forest ensemble method consistently produced better results, especially when using more trees, demonstrating its capacity to lower overfitting and raise generalization. The framework strengthens network security and lowers the risk of data breaches by utilizing the power of machine learning to identify possible threats and promote quick reaction. By offering strong intrusion detection capabilities, the installation of this framework can greatly improve the overall security posture of computer networks. The growth of machine learning algorithms throughout time, as well as the incorporation of cutting-edge methods like deep learning and anomaly detection, can be the focus of future research and development in this area. Network intrusion detection systems can efficiently respond to changing threats and safeguard crucial systems and data in an increasingly linked world by staying on the cutting edge of technical development.

References:

[1] A. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2017.

[2] N. Farnaaz and M. A. Jabbar, “Random forest modeling for network intrusion detection system,” *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[3] H. Wang, J. Gu, and S. Wang, “An effective intrusion detection framework based on SVM with feature augmentation,” *Knowledge-Based Systems*, vol. 136, pp. 130–139, 2017.

[4] P. S. Bhattacharjee, A. K. M. Fujail, and S. A. Begum, “A comparison of intrusion detection by K-means and fuzzy C-means clustering algorithm over the NSL-KDD dataset,” in *Proceedings of the 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, IEEE, Chennai, India, 2017.

[5] I. M. Akashdeep, I. Manzoor, and N. Kumar, “A feature reduced intrusion detection system using ann classifier,” *Expert Systems with Applications*, vol. 88, pp. 249–257, 2017.

[6] Y. Chuan-Long, Z. Yue-Fei, F. Jin-Long et al., “A deep learning approach for intrusion detection using recurrent neural networks,” *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[7] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, “Application of deep reinforcement learning to intrusion detection for supervised

problems," *Expert Systems with Applications*, vol. 141, Article ID 112963, 2019.

[8] H. He, X. Sun, H. He, G. Zhao, L. He, and J. Ren, "A novel multimodal-sequential approach based on multi-view features for network intrusion detection," *IEEE Access*, vol. 7, pp. 183207–183221, 2019.

[9] P. Sun, P. Liu, Q. Li et al., "DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security and Communication Networks*, vol. 2020, Article ID 8890306, 11 pages, 2020.

[10] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Engineering Journal*, vol. 4, no. 4, pp. 753–762, 2013.

[11] W. Wang, T. Guyet, R. Quiniou, M.-O. Cordier, F. Masseglia, and X. Zhang, "Autonomic intrusion detection: adaptively detecting anomalies over unlabeled audit data streams in computer networks," *Knowledge-Based Systems*, vol. 70, pp. 103–117, 2014.

[12] H. Yao, Q. Wang, L. Wang, P. Zhang, M. Li, and Y. Liu, "An intrusion detection framework based on hybrid multi-level data mining," *International Journal of Parallel Programming*, vol. 47, no. 4, pp. 740–758, 2019.

[13] H. Zhang, Y. Li, Z. Lv, A. K. Sangaiah, and T. Huang, "A real-time and ubiquitous network attack detection based on deep belief network and support vector machine," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 790–799, 2020.

[14] E. Min, J. Long, Q. Liu et al., "Su-IDS: a semi-supervised and unsupervised framework for network intrusion detection," in *Proceedings of the International Conference on Cloud Computing and Security*, pp. 322–334, Springer, Cham, Switzerland, 2018.

[15] M. R. Karbir, R. Onik, and T. Samad, "A network intrusion detection framework based on bayesian network using wrapper approach," *International Journal of Computer Applications*, vol. 166, no. 4, pp. 975–8887, 2017.

[16] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Computers & Security*, vol. 86, pp. 53–62, 2019.

[17] A. Ahmim, L. Maglaras, M. A. Ferrag et al., "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 228–223, IEEE, Santorini, Greece, 2019.

[18] V. Kumar, V. Choudhary, V. Sahrawat et al., "Detecting intrusions and attacks in the network traffic using anomaly based techniques," in *Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 554–560, IEEE, Coimbatore, India, 2020.

[19] A. S. Qureshi, A. Khan, N. Shamim, and M. H. Durad, "Intrusion detection using deep sparse auto-encoder and self-taught learning," *Neural Computing and Applications*, vol. 32, no. 8, pp. 3135–3147, 2020.

[20] S. Nathan, T. N. Ngoc, V. D. Phai et al., "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[21] A. Y. Javaid, Q. Niyaz, W. Sun et al., "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th Eai International Conference on Bio-inspired Information & Communications Technologies, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)*, pp. 21–26, Nairobi, Kenya, 2016.

[22] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[23] R. Abdulhammed, H. Musafer, A. Alessa et al., "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, 2019.

[24] G. C. Fernández and S. Xu, "A case study on using deep learning for network intrusion detection," in *Proceedings of the MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pp. 1–6, IEEE, Norfolk, VA, USA, 2019.

[25] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "Enhancing network intrusion detection classifiers using supervised adversarial training," *The Journal of Supercomputing*, vol. 76, no. 9, pp. 6690–6719, 2020.

[26] P. Madani and N. Vlajic, "Robustness of deep autoencoder in intrusion detection under adversarial

contamination," in *Proceedings of the 5th Annual Symposium and Bootcamp*, pp. 1–8, ACM, New York, NY, USA, 2018.

[27] E. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units(elus)," 2016

[28] S. Ustebay, Z. Turgut, and M. A. Aydin, "Cyber attack detection by using neural network approaches: shallow neural network, deep neural network and autoencoder," in *Proceedings of the 2019 International Conference on Computer Networks*, pp. 144–155, Spinger, Cham, Switzerland, 2019.

[29] L. Breiman, J. Friedman, J. Charles et al., *Classification and Regression Trees*, Chapman and Hall/CRC, London, UK, 1984.

[30] Dataset CICIDS2017: https://www.kaggle.com/datasets/cicdataset/cicids2017/code

[31] M. Tavallaee, E. Bagheri, W. Lu et al., "A detailed Analysis of the KDD CUP 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, Ottawa, Canada, 2009.

[32] I. Sharafaldin and A. Ali, "Ghorbani toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 108–116, Funchal, Portugal, 2018.

[33] M. Patel, B. Naughton, C. Chan et al., "Mobile-edge computing introductory technical white paper: mobile-edge computing(MEC) industry initiative," 2014.

[34] M. Eskandari, Z. H. Janjua, M. Vecchio et al., "Passban IDS: an intelligent anomaly-based intrusion detection system for IoT Edge devices," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6882–6897, 2020.

[35] Y. Zhang, W. Jia, and K. Wang, "An edge IDS based on biological immune principles for dynamic threat detection," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8811035, 15 pages, 2020.

[36] P. A. A. Resende and A. C. Drummond, "A survey of random forest based Methods for intrusion detection systems," *ACM Computing Surveys*, vol. 51, no. 3, pp. 1–36, 2018.

[37] Khetani, V. ., Gandhi, Y. ., Bhattacharya, S. ., Ajani, S. N. ., & Limkar, S. . (2023). Cross-Domain Analysis of ML and DL: Evaluating their Impact in Diverse Domains. International Journal of Intelligent Systems and Applications in Engineering, 11(7s), 253–262.

[38] S. Raponi, M. Caprolu, and R. Di Pietro, "Intrusion detection at the network edge: solutions, limitations, and future directions," in *Proceedings of the International Conference on Edge Computing*, Springer, Rome, Italy, 2019.

[39] Vijayalakshmi, V., & Sharmila, K. (2023). Secure Data Transactions based on Hash Coded Starvation Blockchain Security using Padded Ring Signature-ECC for Network of Things. International Journal on Recent and Innovation Trends in Computing and Communication, 11(1), 53–61. https://doi.org/10.17762/ijritcc.v11i1.5986

[40] Jones, D., Taylor, M., García, L., Rodriguez, A., & Fernández, C. Using Machine Learning to Improve Student Performance in Engineering Programs. Kuwait Journal of Machine Learning, 1(1). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/101

[41] Sherje, N.P., Agrawal, S.A., Umbarkar, A.M., Kharche, P.P., Dhabliya, D. Machinability study and optimization of CNC drilling process parameters for HSLA steel with coated and uncoated drill bit (2021) Materials Today: Proceedings,.