Optimizing Network Intrusion Detection Systems Through Ensemble Learning and Feature Selection Using the CIC-IDS2017 Dataset

Dharmaraj R. Patil[1], Tareek M. Pattewar[2],Trupti S. Shinde[3], Kavita S. Kumavat[4] and Sujit N. Deshpande[5]
[1]Department of Computer Engineering, R.C. Patel Institute of Technology, Shirpur, Maharashtra, India
[2,3,4,5]Department of Computer Engineering, Vishwakarma University, Pune, Maharashtra, India
E-mail: dharmaraj.patil@rcpit.ac.in[1], tareek.pattewar@vupune.ac.in[2], trupti.shinde@vupune.ac.in[3], kavita.kumavat@vupune.ac.in[4], sujit.sujitdeshpande@gmail.com[5]

Keywords: Network intrusion detection, bagging, boosting, ensemble learning, feature selection, CIC-IDS2017 dataset, cybersecurity, machine learning

Received: November 26, 2024

The increasing complexity of cyber threats demands high-performance Network Intrusion Detection Systems (NIDS) that are both accurate and efficient. This study presents an optimized NIDS framework combining feature selection with ensemble learning. Experiments were performed on the CIC-IDS2017 dataset using a stratified train/test split of 70/30. Feature selection methods included Information Gain (24 features), Chi-square (χ^2, 25 features), and Principal Component Analysis (PCA, 20 features). Bagging classifiers (Random Forest, Extra Trees, Bagged Decision Tree) and boosting classifiers (XGBoost, Gradient Boosting, LightGBM, AdaBoost, CatBoost) were evaluated. Using Information Gain selecting 24 features, Extra Trees achieved 99.98% accuracy with near-perfect precision, recall, and F1-score, and extremely low false positive and false negative rates of 0.0001397 and 0.0002597, respectively. Boosting-based models demonstrated superior sensitivity for minority attack classes, improving performance under imbalanced conditions. These results indicate that integrating feature selection with diverse ensemble techniques produces a scalable, interpretable, and highly effective NIDS suitable for practical cybersecurity applications.

Povzetek: Prispevek predstavlja optimiziran omrežni sistem za zaznavo vdorov na podatkih CIC-IDS2017. Združuje izbiro značilk (informacijski dobiček, χ^2, PCA) z ansambelskim učenjem (bagging, boosting). Extra Trees z 24 značilkami doseže izjemno nizki stopnji FP/FN, medtem ko boosting izboljša zaznavo manjšinskih napadov.

1 Introduction

The exponential rise of interconnected systems and services in the digital age has made networks more vulnerable to cyber attacks. Cyberattacks, which range from Distributed Denial of Service (DDoS) to sophisticated penetration attempts, can result in significant financial losses, data breaches, and critical infrastructure disruptions. As a result, the importance of strong and efficient Intrusion Detection Systems (IDS) has grown. An IDS serves an important function in monitoring network traffic, identifying malicious actions, and alerting administrators to potential security breaches [1, 2, 3]. However, the constantly changing nature of cyberattacks offers substantial hurdles in building IDS solutions that are both accurate and scalable [4, 5].

Here are the key statistics highlighted in the Cisco Cyber Threat Trends 2023 Report [6]:

– Ransomware-related attacks accounted for 44% of all incident responses managed by Cisco Talos in the observed period. Pure extortion tactics, where data is stolen but systems are not encrypted, constituted approximately 33% of ransomware incidents.

– Phishing remained the most common initial access vector, involved in 21% of attacks analyzed.

– A significant portion of attacks exploited vulnerabilities that were more than 10 years old, emphasizing the persistent risk of outdated software.

– The "TheMoon" botnet was active across 88 countries, with an estimated 40,000 infected endpoints as of early 2024. A March 2023 surge in botnet activity was noted to be 174% above average.

– Activity involving Remote Access Trojans declined over the reporting period but remained a favorite tool for cybercriminals due to their stealth and functionality.

– Advanced Persistent Threats (APTs) maintained an average of 40 million blocks per month, underscoring their sustained activity and stealth.

Similarly, the Checkpoint 2023 Cyber Security Report highlights critical statistics and trends in the evolving threat landscape [7]:

- There was a reported 38% rise in global cyberattacks in 2022 compared to 2021. The average organization faced 1,168 weekly attacks.
- Education and research were the most targeted sectors globally. The healthcare sector experienced a 74% increase in cyberattacks, highlighting its growing vulnerability.
- Attacks on cloud-based networks surged by 48%, with threat actors exploiting cloud infrastructures to access sensitive data.
- Attackers increasingly relied on native operating system capabilities and IT management tools to evade detection. Traditional encryption-focused ransomware evolved to include data wiping and exfiltration, complicating attribution and mitigation efforts.
- Both attackers and defenders are leveraging AI. While attackers use AI for sophisticated threat delivery, defenders utilize it to identify anomalies and novel attack patterns.

These figures highlight the persistence, diversity, and dynamic nature of cyber threats, underlining the importance of adaptive and proactive cybersecurity measures. These findings indicate the growing complexity and sophistication of cyber threats, emphasizing the need for more advanced defense techniques and capabilities.

Machine learning (ML) has emerged as a promising way to improving IDS performance because to its capacity to evaluate large volumes of data and discover complicated patterns. Ensemble learning methods, such as bagging and boosting, have outperformed other ML techniques in classifying tasks. Bagging (bootstrap aggregating) works by training multiple base learners on various subsets of data and combining their predictions to reduce variation and increase stability. Boosting, on the other hand, aims to gradually improve poor learners by stressing misclassified instances, hence increasing overall accuracy. Both strategies are especially useful for intrusion detection, where datasets frequently include uneven class distributions and high-dimensional feature spaces.

Despite their benefits, the performance of ensemble learning algorithms might be hampered by duplicated and irrelevant characteristics in intrusion detection data sets. High-dimensional data not only adds computing cost, but it also has a negative impact on model correctness. Feature selection strategies, which seek to find the most useful qualities, are thus essential for developing efficient IDS models. Feature selection improves interpretability, accelerates model training, and reduces the danger of overfitting by lowering the dimensionality. Combining feature selection, bagging, and boosting is an effective method for improving the precision and robustness of intrusion detection systems.

This study uses the CIC-IDS2017 dataset to assess the performance of bagging and boosting algorithms combined with feature selection for network intrusion detection. The CIC-IDS2017 dataset, well-known for its realistic portrayal of modern network traffic and many attack types, serves as a complete benchmark for IDS research. This paper enhances network intrusion detection by integrating ensemble learning techniques with feature selection strategies. The key contributions, structured around the research questions, are as follows:

- RQ1: What is the impact of feature selection on the performance of ensemble learning algorithms?
  We applied feature selection methods (Information Gain, Chi-square, PCA) on the CIC-IDS2017 dataset to reduce dimensionality and computational cost. This enabled the ensemble models to concentrate on the most relevant attributes, improving accuracy and efficiency.
- RQ2: How effective are bagging and boosting approaches in detecting different categories of attacks?
  We conducted a comprehensive comparison between bagging-based (RF, ET, BDT) and boosting-based (XGB, GBM, LGBM, AB, CB) models. This analysis highlights their respective strengths and limitations in real-world intrusion detection settings.
- RQ3: What are the strengths and limitations of ensemble-based methods when handling imbalanced and high-dimensional data?
  We examined how feature selection influences bagging and boosting models across multiple attack categories in binary classification tasks. Furthermore, we demonstrated that boosting methods, in particular, are effective at detecting minority class attacks, offering practical value for addressing rare but critical threats.
- RQ4: How reliable and practical are the proposed approaches for real-world IDS deployment?
  We evaluated the ensemble models using multiple performance indicators—accuracy, precision, recall, F1-score, false positive rate, false negative rate, training time, and testing time—providing a comprehensive assessment of their real-world applicability.

The remainder of this paper is organized as follows: Section 2 outlines the motivation for this study. Section 3 reviews related work, emphasizing recent advancements in network intrusion detection systems with a focus on machine learning, ensemble methods, and feature selection, while also identifying existing research gaps. Section 4 describes the proposed methodology, including data collection and preprocessing using the CIC-IDS2017 dataset, the applied feature selection techniques, and the integration of bagging and boosting algorithms. Section 5 presents the experimental results and discussion, comparing model performance with and without feature selection and addressing challenges associated with imbalanced datasets. Section 6 provides an overall discussion, and Section 7 highlights the limitations of the proposed approach. Finally, Section 8 concludes the paper and summarizes the key contributions.

2 Motivation

The fast growth of digital networks, as well as their important role in modern infrastructure, have made them great targets for cyber attacks. These assaults are growing more complex, focusing on system vulnerabilities in order to jeopardize data integrity, availability, and confidentiality. Traditional intrusion detection systems (IDS) that use signature-based methods struggle to keep up with the dynamic nature of these threats because they require prior knowledge of attack patterns. As a result, there is an urgent need for enhanced IDS techniques capable of detecting both known and novel attack routes in real time.

Because of its ability to identify patterns and relationships in large datasets, machine learning (ML) has emerged as a promising approach for enhancing intrusion detection. Ensemble learning approaches, such as bagging and boosting, have shown excellent performance in classification problems by integrating the outputs of numerous models to improve prediction accuracy. These approaches, however, encounter difficulties when used to intrusion detection datasets, which are frequently high-dimensional and have unequal distributions of attack and benign traffic classes. Addressing these issues is critical for achieving accurate detection while preserving computing efficiency.

Feature selection addresses the problems of high-dimensional data by identifying the most informative characteristics for a specific job. Reducing the amount of input variables enhances model interpretability and efficiency while also lowering the danger of overfitting, particularly in ensemble learning models. The combination of feature selection, bagging, and boosting has the potential to yield extremely effective IDS models that balance accuracy and computational cost.

The CIC-IDS2017 dataset used in this study provides a realistic network environment and includes a variety of attack techniques, including Distributed Denial of Service (DDoS), Brute Force, Botnet, and Web Attacks. However, the dataset's complexity and imbalance make it a difficult standard for IDS research. Many previous studies focus on individual machine learning models or disregard the impact of feature selection, leaving potential for future research into optimum techniques to improve performance.

This research is motivated by the need to overcome these shortcomings by systematically merging feature selection with bagging and boosting techniques to improve IDS detection capabilities. This technique seeks to address the dual difficulties of high-dimensional data and class imbalance while maintaining scalability for real-world use. This study aims to provide useful insights and practical solutions for establishing strong and efficient IDS frameworks capable of fighting increasing cyber threats by analyzing the interactions of different methodologies on the CIC-IDS2017 dataset.

3 Related work

Considerable research efforts have been directed toward the development of intelligent Intrusion Detection Systems (IDS) due to the growing prevalence of cyber threats. Techniques like machine learning (ML) and ensemble learning have become well-liked ways to improve the detection of both known and unknown assaults. With an emphasis on the use of ensemble learning techniques like bagging and boosting, the contribution of feature selection to increasing model efficiency, and the difficulties presented by high-dimensional and unbalanced datasets, this part examines current developments in IDS research.

The use of machine learning (ML), deep learning (DL), optimization techniques, and datasets in intrusion detection from 2018 to 2023 was the subject of a thorough review by Issa, M. M. et al. A methodical approach to searching scientific databases was used by the authors to find and evaluate 393 studies that satisfied their inclusion requirements. Critical insights from these papers were extracted and examined using bibliometric analysis. With convolutional neural networks (CNNs), support vector machines (SVMs), decision trees, and genetic algorithms being the most often used methods, their findings demonstrate an increasing interest in the topic. With the goal of educating and directing future research in intrusion detection systems, the review also explores the shortcomings and difficulties of current techniques and provides a systematic summary of the state-of-the-art [1].

Intrusion Detection Systems (IDS) and how machine learning might improve them were thoroughly examined by Vanin, P. et al. An overview of intrusion detection systems (IDS) is given at the outset of the study, which divides them into three primary categories: network intrusion detection systems (NIDS), host intrusion detection systems (HIDS), and hybrid intrusion detection systems. All IDS types can identify attacks using signature-based techniques, by comparing network traffic to a predetermined baseline of typical activity, or by combining the two methods. In order to assess the efficacy of IDS, they have also investigated a number of performance metrics. Accuracy, detecting Rate (Recall), and the F-Measure are important metrics that are highlighted because they are essential for evaluating the dependability and efficacy of detecting systems [2].

The strengths and drawbacks of various Intrusion Detection System (IDS) types, technologies, and approaches are thoroughly examined by Khraisat, A. et al. In their assessment of various machine learning (ML) techniques that have been put out for identifying zero-day attacks, the authors highlight the issues these techniques encounter, such as the inability to produce and update data concerning novel attacks. These methods' efficacy is further limited by the large false positive rates and occasionally low detection accuracy they produce. Along with discussing current models and recent research findings, the survey focuses on ideas that address the fundamental problems with traditional IDS in order to improve IDS performance. The potential for automated intrusion detection systems (AIDS) to improve IDS is one important topic that has been investigated [3].

Network intrusion detection systems (NIDS) that make use of machine learning (ML) and deep learning (DL) techniques are thoroughly reviewed by Ahmad, Z. et al. Their objective is to furnish scholars with a current comprehension of the prevailing patterns, developments, and obstacles in this domain. A methodical strategy is used in the study to choose pertinent publications about AI-based NIDS. It starts by outlining the idea of IDS and its several classification systems, referencing the literature study. The approach used in each publication is analysed, with a focus on assessing the suggested models' advantages and disadvantages in relation to intrusion detection efficacy and model complexity. The study points to a current trend in NIDS toward the adoption of deep learning techniques, which has improved detection accuracy and decreased False Alarm Rates (FAR). To be precise, almost 80% of the suggested solutions rely on deep learning techniques, with the most widely used algorithms being Autoencoders (AE) and Deep Neural Networks (DNN). Furthermore, the analysis points out that 60% of the approaches examined were evaluated on popular datasets like KDD Cup'99 and NSL-KDD, mostly due of the substantial findings that these datasets provide. Nevertheless, these datasets are out-of-date and might not correctly represent contemporary network attack scenarios, which restricts the methodologies' use in real-time settings. Consequently, the study highlights the necessity of more recent datasets to improve the performance and applicability of intrusion detection algorithms in modern network infrastructures [8].

An Artificial Neural Network (ANN)-based Intrusion Detection System (IDS) was proposed by Mebawondu, J. O. et al. and evaluated on the UNSW-NB15 dataset. For processing continuous data, they employed a binarization discretization technique, and for feature ranking, they employed the Gain Ratio method. For model building and evaluation using an ANN-MLP, the top 30 characteristics were chosen based on a predetermined threshold. The model demonstrated a positive correlation with a 76.96% accuracy and a 0.57 Matthews Correlation Coefficient (MCC), according to the experimental results. These results demonstrate the possibility of the suggested approach for real-time intrusion detection and confirm the UNSW-NB15 dataset's efficacy for network intrusion detection system development [9].

A SPIN-IDS system powered by AI and intended for near real-time network threat detection was presented by Ghadermazi, J. et al. Their method makes use of both header and payload data, as well as the temporal linkages between packets within the same communication flow, to overcome the shortcomings of packet-based Network Intrusion Detection Systems (NIDS). After being transformed into two-dimensional images, sequential packets in a network flow are evaluated by an intrusion detection component that leverages CNN. The SPIN-IDS framework, which used a nine-sequential-packet image representation of the dataset, performed well in identifying network threats, according to experimental results. The model demonstrated the capacity to detect harmful patterns with recall scores ranging from 97.7% to 99% across different types of attacks. Additionally, by examining the ninth packet in a bidirectional communication flow, the study demonstrated that malicious activity may be precisely detected [10].

The use of cutting-edge deep learning algorithms to improve Network Intrusion Detection Systems' (N-IDS) ability to categorize network connections as malicious or benign has been investigated by Vinayakumar, R. et al. Transmission Control Protocol/Internet Protocol (TCP/IP) packets within specified time periods were the main focus of the study, which represented network traffic as time-series data. Using connection records from the KDDCup-99 dataset, supervised deep learning techniques such as Recurrent Neural Networks (RNN), Identity Recurrent Neural Networks (IRNN), Long Short-Term Memory (LSTM), Clock-Work RNN (CWRNN), and Gated Recurrent Units (GRU) were used. With regard to capturing long-term dependencies, the study focused on comparing the effectiveness of RNNs with more recent techniques such as LSTM and IRNN, particularly addressing the disappearing and expanding gradient problems. Comparing various topologies and characteristics allowed for the identification of efficient network architectures, and tests were carried out for up to 1,000 epochs with learning rates varying between 0.01 and 0.05. Results on the KDDCup-99 dataset indicated that IRNN performed similarly to LSTM. Additionally, NSL-KDD and the more recent UNSW-NB15 datasets were used to evaluate the efficacy of these deep learning models utilizing improved versions of the dataset [11].

Using an Evolutionary Neural Network (ENN) for classification and a modified Cuckoo Search Algorithm (CSA), known as Mutation Cuckoo Fuzzy (MCF), for feature selection, Sarvari, S. et al. have created an anomaly-based intrusion detection system. Mutation is incorporated into the suggested algorithm to better traverse the search space and steer clear of local minima. An objective function and the Fuzzy C-Means (FCM) clustering method—which manages overlapping datasets by generating a fuzzy membership search domain that encompasses all possible compromise solutions—are used to evaluate the quality of the solution. The NSL-KDD dataset was used to validate the model after it was applied to the intrusion detection problem. Results from experiments showed that choosing the most pertinent features from the dataset to reduce its size not only shortened execution times but also improved the intrusion detection system's overall effectiveness and performance [12].

A new Stochastic Fractal Search Algorithm combined with a Deep Learning-based Intrusion Detection System (SFSA-DLIDS) has been presented by Duhayyim, M. A. et al. to secure cloud-based Cyber-Physical Systems (CPS). To improve the security of CPS environments, the SFSA-DLIDS architecture focuses on detecting and categorizing intrusions. Min-max data normalization is the first step in the process to format input data for compatibility. A pertinent subset of features is chosen using the SFSA method in order to overcome the problem of high dimensionality. Furthermore, Deep Stacked Autoencoder (DSAE) and Chicken Swarm Optimization (CSO) are used in tandem for intrusion detection and categorization. The CSO algorithm is specifically made to improve classification performance by optimizing the DSAE model's parameters. The efficacy of the SFSA-DLIDS model was confirmed by extensive experimental assessments. The suggested method's effectiveness and dependability in protecting CPS environments were shown by the results, which showed that it outperformed contemporary intrusion detection systems [13].

Dini, P., et al. investigated the use of machine learning (ML) approaches to intrusion detection systems (IDS), with an emphasis on datasets, algorithms, and assessment metrics. The study included three widely renowned datasets: KDD 99, UNSW-NB15, and CSE-CIC-IDS 2018. A variety of ML algorithms were studied to determine their usefulness in IDS performance, with the primary goal of developing a taxonomy for linked IDS and supervised ML approaches. Careful dataset selection was stressed to assure the models' appropriateness for IDS applications. Both binary and multi-class classification tasks were used in the evaluation to ensure that the ML methods were consistent and reliable across different datasets. The experimental findings were outstanding, with 100% accuracy in binary classification and 99.4% in multi-class classification. When tested on these benchmark datasets, the findings show that supervised machine learning algorithms give extremely accurate and reliable intrusion detection performance [14].

By putting out a unique traffic anomaly detection model known as BAT, Su, T. et al. have solved the issues of low accuracy and dependence on feature engineering in intrusion detection. An attention mechanism and a Bidirectional Long Short-Term Memory (BLSTM) network are combined in this model. Critical properties for classifying network traffic are extracted by the attention mechanism by selective processing of the network flow vector, which is built from packet vectors produced by the BLSTM model. Additionally, the BAT model uses several convolutional layers to efficiently extract local information from traffic data. Because these convolutional layers are used to process data samples, the model is called BAT-MC. The final classification of network traffic is done using a softmax classifier. By automatically learning hierarchical key features, BAT-MC functions as an end-to-end model that improves anomaly detection and efficiently characterizes network traffic behavior, hence removing the need for feature engineering. Experimental results showed that BAT-MC works better than alternative approaches, exhibiting improved accuracy and detection capabilities. The model was tested on a publicly available benchmark dataset [15].

To improve the accuracy of anomaly detection and shorten execution time, Stiawan, D. et al. carried out a study to extract important features from massive network traffic data. In order to select, rank, and group features according to minimum weight values in order to ascertain their importance, the study used the Information Gain approach. On the CICIDS-2017 dataset, these chosen features were then assessed using a number of classification methods, such as Random Forest (RF), Bayes Net (BN), Random Tree (RT), Naive Bayes (NB), and J48. The results showed that the amount of relevant features chosen has a major effect on execution time and detection accuracy. Of the techniques that were tested, the Random Forest classifier used 22 features and achieved an accuracy of 99.86%. The J48 algorithm, on the other hand, used 52 features and took longer to execute, but it produced a slightly better accuracy of 99.87%. This study emphasizes how crucial feature selection is to intrusion detection system optimization [16].

Using an optimized approach, Liu, G. et al. have created a multiclass network intrusion detection model based on a convolutional neural network (CNN). The model was put into practice and evaluated on a system that has a 1 TB solid-state drive, 32 GB of RAM, Ubuntu 16.04, and a Docker 19.03.5 container virtualization environment. The performance of the suggested model was compared to a number of deep learning models, including DNN, LSTM-RNN, GRU-RNN, DBN, KNN, and ICNN, through experiments utilizing the KDD-CUP99 and NSL-KDD datasets. The findings proved that the suggested CNN-based model improved detection performance, especially for detecting unknown attacks, decreased the false positive rate, and increased accuracy and recall [17].

An intrusion detection and classification model based on machine learning approaches has been proposed by Jaradat, A. S. et al. To find the most pertinent qualities, the procedure starts with the acquisition and formatting of the dataset, then moves on to feature selection. Following refinement, the Konstanz Information Miner (KNIME) platform is used to evaluate the dataset. The CICIDS2017 dataset on the KNIME analytics platform was used to test three distinct classifiers in order to attain strong performance and allow for comparison analysis. The experimental findings showed an accuracy of 90.59% on average and a high of 98.6%, surpassing numerous current approaches. These results demonstrate the promise of machine learning in data analysis and cybersecurity, promoting the advancement of more accurate intrusion detection systems [18].

A study was carried out by Alissa, K. A. et al. to address privacy and security issues in the Internet of Drones (IoD). They suggested Crystal Structure Optimization with Deep Autoencoder-based Intrusion Detection (CSODAE-ID), an enhanced intrusion detection system (IDS), to improve IoD security. The main objective of the CSODAE-ID model is to efficiently identify intrusions in the IoD environment. To determine which feature subsets are most pertinent, the model uses a Modified Deer Hunting Optimization-based Feature Selection (MDHO-FS) technique. Concurrently, intrusion classification is done using the Autoencoder (AE) approach. For hyper-parameter tuning, the Crystal Structure Optimization (CSO) technique is utilized, which draws inspiration from the lattice point formations found in crystal structures. The CSODAE-ID model's performance was verified by extensive simulations under a variety of conditions. The suggested model surpasses current techniques, according to comparative studies, showing its efficacy in protecting IoD environments [19].

Using a multistage deep learning approach based on picture recognition, Toldinas, J. et al. have put forth a novel technique for network intrusion detection. The four-channel (Red, Green, Blue, and Alpha) pictures created from network characteristics are used in this method for categorization. Both training and testing use the ResNet50 deep learning model, which has been pre-trained on a sizable dataset. Two publicly accessible benchmark datasets, UNSW-NB15 and BOUN DDoS, were used to assess the methodology. It detected generic assaults with an amazing 99.8% accuracy on the UNSW-NB15 dataset. Similarly, the model showed 99.7% accuracy in recognizing DDoS assaults and 99.7% accuracy in identifying normal traffic on the BOUN DDoS dataset. These outcomes demonstrate the efficiency and dependability of the suggested approach for network intrusion detection [20].

For intrusion detection systems (IDS) in Internet of Things (IoT) contexts, Fatani, A. et al. have put forth a sophisticated AI-driven design. In order to solve complicated engineering problems, the approach mixes metaheuristic algorithms with deep learning approaches. The strategy uses a feature extraction technique based on convolutional neural networks (CNNs) to effectively find pertinent characteristics. A brand-new feature selection method is also presented, known as TSODE (Transient Search Optimization with Differential Evolution). The Differential Evolution (DE) algorithm's operators are used into this technique to improve the ratio of exploration to exploitation during optimization. KDDCup-99, NSL-KDD, BoT-IoT, and CICIDS-2017 are four publicly available datasets that were used to assess the effectiveness of the suggested method. Test findings showed that the created approach outperformed current methods in terms of accuracy, indicating its potential to enhance IoT security [21].

In order to create a network intrusion detection model that is both scalable and adaptable, Chiche, A. et al. have suggested a revolutionary integrated learning technique. This technique uses a knowledge-based system and machine learning to address the present issues in intrusion detection. The classifier is built by the machine learning component, and scalability and adaptability are improved by the knowledge-based system. Ten-fold cross-validation was used to validate the model after it was assessed using the 40,558-instance NSL-KDD dataset. The results showed that the experiment performed admirably, with an accuracy of 99.91%. The study emphasizes the function of knowledge-rich learning as a key component of detection and prevention strategies and stresses how crucial it is to include it in effective intrusion detection. In order to safeguard their infrastructure and streamline their operational procedures, the study advises security experts to incorporate these intrusion detection models into their computer and network systems [22].

In order to optimize the hyperparameters of the XGBoost classifier for network intrusion detection, Zivkovic, M. et al. presented an improved version of the popular firefly technique. By merging the enhanced firefly method with the XGBoost classifier, this study aims to address the high rate of false positives and false negatives, a typical problem in intrusion detection systems. First, the enhanced method was tested against 28 well-known CEC2013 benchmark instances and contrasted with other top metaheuristics and the original firefly approach. After validation, the XGBoost classifier's hyperparameters were optimized using it. Two well-known datasets for network intrusion detection, NSL-KDD and UNSW-NB15, were used to assess the optimized model. Results from experiments showed that the suggested method greatly improves average precision and classification accuracy, indicating its potential as a useful tool for network intrusion detection machine learning model optimization [23].

By combining sophisticated deep learning algorithms with network and host traffic data, Alars, E. S. A. et al. have improved NIDS performance and tackled network intrusion detection difficulties. They used a dataset that was created to simulate different types of intrusions in a military network setting. This procedure included feature extraction, preprocessing, and extensive data collection. To enhance the model's performance, they used dimensionality reduction and strict feature selection in their convolutional neural network (CNN) analysis of the data. A remarkable 98.5% detection accuracy was attained by their deep learning-based NIDS, surpassing current techniques and tackling practical cybersecurity problems, according to the results. In addition to advancing NIDS technology, this integrated strategy offers a workable way to improve network security across a range of applications, which helps intrusion detection systems continue to evolve [24].

A hybrid model for intrusion detection has been created by Sajid, M. et al. to address current constraints by combining machine learning (ML) and deep learning (DL) approaches. Long short-term memory networks (LSTM) are used for classification after convolutional neural networks (CNN) and Extreme Gradient Boosting (XGBoost) are used for feature extraction. For both binary and multi-class classification tasks, the model was trained on four benchmark datasets: CIC IDS 2017, UNSW NB15, NSL KDD, and WSN DS. Due to decreased accuracy, many intrusion detection systems have trouble identifying new threats as feature dimensions grow. The feature space was essentially reduced by applying CNN-based feature selection techniques and XGBoost to every dataset. The efficiency of the suggested hybrid model in enhancing intrusion detection performance was demonstrated by the experimental findings, which exhibited high detection rates, good accuracy, and a low False Acceptance Rate (FAR) [25].

Fuzzy numbers and a scoring system based on correlation feature selection are two new methods that Shiravani, A. et al. have presented for choosing useful features in network intrusion detection. Reducing the size of the dataset by removing inefficient features and lowering its dimensions is the main goal of this approach. The correlation-based feature selection algorithm's heuristic function is represented as a triangle fuzzy number membership function in this method, where the features are expressed as fuzzy numbers. The suggested approach was contrasted with conventional intrusion detection methods in order to evaluate its effectiveness. The findings demonstrated that the suggested approach achieves a greater detection rate while choosing fewer features than traditional approaches. The KDD Cup, NSL-KDD, and CICIDS datasets were used to test the approach. The suggested method achieved an accuracy of 99.9%, whereas the Correlation-based Feature Selection (CFS) method achieved 96.01%. This illustrates how well the novel strategy works to enhance intrusion detection systems' feature selection and detection capabilities [26].

Using a variety of feature fusion techniques, Ayantayo, A. et al. have presented novel deep learning architectures designed to improve the performance of multi-classification tasks in Network Intrusion Detection Systems (NIDS). They used feature fusion in fully linked deep networks to propose three different models: early-fusion, late-fusion, and late-ensemble learning models. The purpose of these fusion procedures was to lessen potential biases resulting from particular feature types and enhance the models' capacity to learn correlations between various input features. The researchers employed the well-known UNSW-NB15 and NSL-KDD datasets, which are created especially to further NIDS research, to assess the performance of their deep learning models and contrast them with current methodologies. Their investigation showed that their models were robust in handling multi-classification problems, especially when class imbalance was present. Furthermore, both the late-ensemble and late-fusion models showed similar outcomes on the training and validation sets, reduced overfitting, and better generalization performance [27].

IDS-MTran, a unique intrusion detection model developed by Xi, C. et al., uses a multi-scale transformer technique. The detection coverage for intrusions is increased by this approach by the integration of multi-scale traffic features. First, convolutional operators with different kernels are used to create multi-scale features. To bridge the feature extraction process, the researchers created a Patching with Pooling (PwP) approach to increase feature representation and branch-to-branch interaction. In order to simulate the features at different scales and capture possible incursion patterns, they then created a multi-scale transformer-based backbone. Before generating the final results, the Cross Feature Enrichment (CFE) technique integrates and refines the features to further improve feature utilization. In identifying different kinds of assaults, the IDS-MTran model works better than other detection models, according to extensive studies. On three popular datasets—NSL-KDD, CIC-DDoS 2019, and UNSW-NB15—the model specifically demonstrated enhanced accuracy and stability, achieving above 99% accuracy [28].

For network intrusion detection, Gu, Y. et al. have presented a semi-supervised weighted k-means approach. The first step in the process is creating a hybrid feature selection algorithm based on Hadoop that finds the best feature sets. For choosing initial cluster centers, they suggested an improved density-based technique to overcome problems like outliers and local optima. The technique then uses a semi-supervised K-means algorithm that has been improved with hybrid feature selection (SKM-HFS) to more precisely identify threats. The researchers used a number of datasets, including the DARPA DDoS dataset, the CAIDA "DDoS Attack 2007" dataset, the CICIDS "DDoS Attack 2017" dataset, and a real-world dataset, to test their methodology. According to the experimental findings, their approach considerably beat the current benchmarks in terms of both detection performance and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) evaluation factor [29].

For the Internet of Things (IoT) platform, Mohamed, H. G. et al. have created a novel intrusion detection technique called BSAWNN-ID. The BSAWNN-ID algorithm's primary objective is to identify and categorize intrusions in Internet of Things environments. In order to accomplish this, the method selects the most pertinent characteristics through a Feature Subset Selection employing the Cuckoo Optimization Algorithm (FSS-COA). Next, the Wavelet Neural Network (WNN) model is employed for intrusion detection; the Bat Search Algorithm (BSA) is used to optimize the WNN parameters. Using the UNSW-NB15 dataset, a thorough experimental investigation showed that the BSAWNN-ID technique outperformed other models with an accuracy of 99.64%. According to the findings, the BSAWNN-ID method works well for real-time intrusion detection in Internet of Things systems. Future research could expand on this technique to tackle difficulties with outlier recognition [30].

By offering justifications for Deep Learning-based Network Intrusion Detection Systems (DL-NIDS), Wei, F. et al. have presented XNIDS, a novel framework intended to improve active intrusion responses. Approximating and sampling inputs based on historical data and capturing the interdependencies of features inside structured data to generate high-fidelity explanations are the two main characteristics of the suggested explanation approach. With the help of these justifications, XNIDS is able to produce useful defensive guidelines. The four top DL-NIDS models were used to assess the framework. According to the evaluation results, XNIDS outperforms earlier explanation techniques in terms of stability, fidelity, sparsity, and completeness—all of which are essential components of successful active incursion responses. Furthermore, XNIDS can help diagnose detection problems, improve comprehension of DL-NIDS behavior, and produce useful defense methods, according to the study [31].

Huang et al. developed a hybrid intrusion detection framework combining feature selection and stacking ensembles. They employed a dual strategy using information gain and random forest importance to identify compact, discriminative feature subsets. On UNSW-NB15, the method attained 80.83% accuracy with only 9 features, improving by 5.37% over baselines. On CIC-IDS2017, 27 selected features delivered 99.97% accuracy. The model significantly reduced false alarms and outperformed traditional machine learning and existing ensemble methods across multiple evaluation metrics [32].

Urmi et al. introduced a stacked ensemble intrusion detection model integrating Random Forest, XGBoost, and Extra-Trees with Logistic Regression as the meta-classifier. They analyzed the impact of Recursive Feature Elimination, Mutual Information, and Lasso-based feature selection on system performance. Using CIC-IDS2017, RFE attained 100% accuracy for Brute Force and 99.99% for Infiltration and Web attacks. On NSL-KDD, the approach achieved 99.95% accuracy overall, highlighting that optimized feature selection with ensemble learning significantly strengthens detection effectiveness [33].

Ahmed et al. developed HAEnID, an adaptive hybrid ensemble intrusion detection model that integrates stacking, Bayesian model averaging (BMA), and conditional ensemble methods. The model incorporates SHAP and LIME to enhance explainability while dynamically adapting to new attack patterns. Experiments on CIC-IDS2017 showed strong results, achieving 97–98% accuracy overall. With optimized feature selection using BMA-M (20), accuracy improved to 98.79%. This demonstrates HAEnID's effectiveness in balancing high detection performance, adaptability, reduced false alarms, and interpretability [34].

Table 1. shows the detail summary of reviewed IDS studies: methodologies, datasets, techniques, performance, and gaps.

4 Methodology

With the help of feature selection strategies and ensemble learning techniques like bagging and boosting, this study seeks to improve network intrusion detection by increasing the system's accuracy and efficiency. As shown below, the suggested methodology is divided into a number of discrete stages, such as feature selection, model training, data preprocessing, and evaluation. The suggested framework for a network intrusion detection system based on the CIC-IDS2017 dataset is shown in Figure 1.

4.1 Data collection and preprocessing

The initial step involves obtaining a suitable dataset for training and testing the network intrusion detection system (NIDS). This study uses publicly available benchmark dataset, such as CIC-IDS2017, which include labeled instances of network traffic. The CIC-IDS2017 dataset contains 80 features per network flow, including continuous features such as flow duration, packet length statistics, and byte counts, as well as categorical features like protocol type and TCP flags. This datasets consist of both normal and malicious traffic, enabling the development of a robust intrusion detection model. Once the data is acquired, it undergoes preprocessing to ensure it is suitable for machine learning models. This includes:

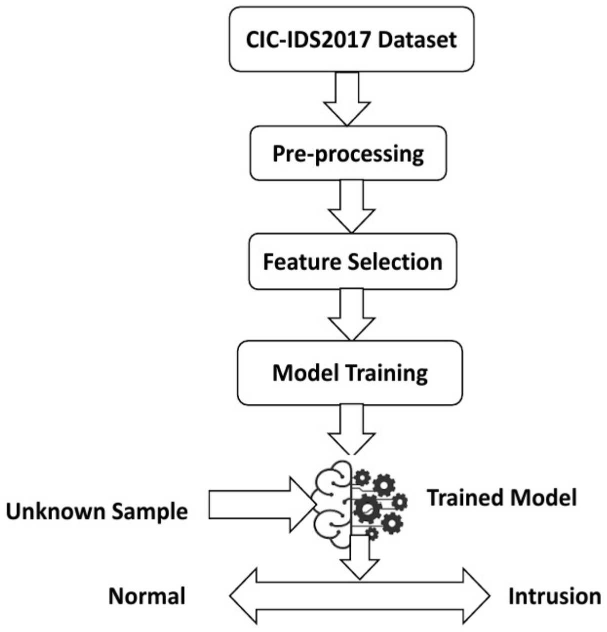

Figure 1: Proposed framework of network intrusion detection system using CIC-IDS2017 dataset

4.1.1 Handling missing values

The raw dataset includes features with missing or undefined values, particularly those arising from division-by-zero operations in flow statistics. Attributes with a large proportion of missing entries were discarded, while features with relatively few missing values were retained. For these, numerical attributes were imputed using the mean value, whereas categorical attributes were imputed using the most frequent category. This strategy ensured that valuable features were preserved without introducing bias.

4.1.2 Outlier handling

Traffic data often includes extreme observations such as abnormally large packet sizes or unusually long flow durations. To reduce the influence of such values while maintaining attack-related information, we employed the interquartile range (IQR) method. Values lying outside 1.5 times the IQR from the first or third quartile were capped to the respective boundary values. This approach mitigated the effect of skewed distributions without discarding rare but meaningful patterns.

Study	**Method / Algorithm**	**Dataset**	**Techniques / Features**	**Performance Metrics**	**Observed Gaps**
Su et al. [15]	BAT-MC (BLSTM+CNN+Attention)	Public benchmark dataset	Automatic feature learning with hybrid DL	High accuracy	High model complexity
Stiawan et al. [16]	RF, BN, RT, NB, J48	CICIDS2017	Information Gain FS	Acc: RF=99.86%, J48=99.87%	Speed vs accuracy trade-off in FS
Liu et al. [17]	CNN (multiclass IDS)	KDD99, NSL-KDD	CNN vs DL baselines	Accuracy ↑, FPR ↓	Limited modern dataset testing
Jaradat et al. [18]	ML classifiers	CICIDS2017	KNIME pipeline + FS	Avg Acc=90.59%, Max=98.6%	Dependence on FS methods
Alissa et al. [19]	CSODAE-ID (AE + MDHO-FS)	IoD dataset	Crystal Optimization + FS	Outperformed baselines	IoD dataset focus only
Toldinas et al. [20]	ResNet50 (image-based IDS)	UNSW-NB15, BOUN-DDoS	RGB packet mapping to images	Acc=99.8% (UNSW), 99.7% (DDoS)	Requires heavy computation, image conversion
Fatani et al. [21]	CNN + TSODE FS	KDD99, NSL-KDD, BoT-IoT, CICIDS2017	Metaheuristic + DL feature selection	Outperformed baselines	Optimization overhead
Chiche et al. [22]	Knowledge-based ML IDS	NSL-KDD	Knowledge integration + ML	Acc=99.91%	Focused on NSL-KDD only
Zivkovic et al. [23]	Firefly-XGBoost hybrid	NSL-KDD, UNSW-NB15	Metaheuristic parameter tuning + XGB	Precision and Acc ↑	Complexity of hybrid tuning
Alars et al. [24]	CNN + feature reduction	Military dataset	Dimensionality reduction + DL	Acc=98.5%	Specific to military data
Sajid et al. [25]	CNN+XGBoost+LSTM hybrid	CICIDS2017, UNSW-NB15, NSL-KDD, WSN-DS	Hybrid FS + DL	High accuracy, low FAR	Complex hybrid integration
Shiravani et al. [26]	Fuzzy-CFS FS	KDD, NSL-KDD, CICIDS	Triangle fuzzy FS	Acc=99.9%	Limited generalization across datasets
Ayantayo et al. [27]	Fusion DL models	UNSW-NB15, NSL-KDD	Early, late, ensemble fusion	Stable multiclass detection	High training cost
Xi et al. [28]	IDS-MTran (multi-scale Transformer)	NSL-KDD, CIC-DDoS2019, UNSW-NB15	PwP, CFE + multi-scale encoding	Acc>99%	Transformer overhead
Gu et al. [29]	SKM-HFS (semi-supervised K-means)	DARPA, CAIDA, CICIDS2017, real-world	Hadoop-based FS + semi-supervised learning	Higher detection rates	Scalability + complexity
Mohamed et al. [30]	BSAWNN-ID (WNN+COA+BSA)	UNSW-NB15	COA + WNN FS optimization	Acc=99.64%	IoT only, lacks outlier handling
Wei et al. [31]	XNIDS (DL explainability)	4 DL-NIDS	Explainability framework (stability, fidelity)	Stable, complete justifications	Accuracy not improved

Table 1: Summary of reviewed ids studies: methodologies, datasets, techniques, performance, and gaps

4.1.3 Feature encoding

Several fields in CIC-IDS2017, such as protocol identifiers and service names, are categorical in nature. To ensure compatibility with machine learning algorithms, these categorical features were transformed into numerical values using label encoding. This step preserved the distinct categories while enabling their direct use in the classifiers.

4.1.4 Normalization

The dataset includes features with heterogeneous scales, such as binary flags, byte counts, and time-based measures. To prevent attributes with large magnitudes from dominating the learning process, all continuous features were normalized using the Min–Max scaling method, mapping each value into the range [0, 1]. This normalization improved the stability and convergence of ensemble classifiers, particularly gradient-based boosting models.

Through these preprocessing steps—cleaning, imputation, outlier capping, encoding, and normalization—the CIC-IDS2017 dataset was transformed into a consistent and standardized format, ready for feature selection and classification.

4.2 Feature selection techniques used for network intrusion detection

Reducing redundant or unnecessary features is a key component of feature selection, which enhances the effectiveness of network intrusion detection models. Finding and preserving only those characteristics that substantially aid in differentiating between benign and malevolent network traffic is the main objective [35, 36, 37]. Three methods, including Principal Component Analysis (PCA), Chi-Square, and Information Gain, are used to choose the most pertinent features:

4.2.1 Information gain (IG) feature selection technique

Information Gain (IG) is a widely used feature selection method that quantifies the reduction in uncertainty of the target variable provided by a feature [38, 39]. It relies on the concept of entropy to measure impurity or uncertainty in the target variable. For datasets containing continuous features, discretization is required; in our study, continuous features were discretized using equal-frequency binning with 10 bins before computing IG to ensure valid calculations.

Let Z denote the target variable, and $\mathcal{Z}$ be its possible values. The entropy of Z is:

$$\mathcal{H}(Z) = -\sum_{z\in\mathcal{Z}} p(z)\log_2 p(z), \tag{1}$$

where $p(z)$ is the probability of $Z = z$.

Let W be a feature with possible values $\mathcal{W}$. The conditional entropy of Z given W is:

$$\mathcal{H}(Z|W) = \sum_{w\in\mathcal{W}} p(w)\left(-\sum_{z\in\mathcal{Z}} p(z|w)\log_2 p(z|w)\right), \tag{2}$$

where $p(w)$ is the probability of $W = w$ and $p(z|w)$ is the conditional probability of $Z = z$ given $W = w$.

The Information Gain of W with respect to Z is then:

$$\mathrm{IG}(W, Z) = \mathcal{H}(Z) - \mathcal{H}(Z|W). \tag{3}$$

Features with higher IG values contribute more to reducing uncertainty in the target variable and are considered more informative. In our experiments, the top 25 features with the highest IG scores were initially identified, and the optimal classifier performance was achieved using 24 features, as reported in the results. Table 2 lists the selected features.

Sr. No	IG Selected Features
1	Destination Port
2	Flow Duration
3	Total Length of Fwd Packets
4	Total Length of Bwd Packets
5	Fwd Packet Length Max
6	Fwd Packet Length Mean
7	Bwd Packet Length Max
8	Bwd Packet Length Mean
9	Flow Bytes/s
10	Flow IAT Max
11	Fwd Header Length
12	Bwd Header Length
13	Max Packet Length
14	Packet Length Mean
15	Packet Length Std
16	Packet Length Variance
17	Average Packet Size
18	Avg Fwd Segment Size
19	Avg Bwd Segment Size
20	Fwd Header Length.1
21	Subflow Fwd Bytes
22	Subflow Bwd Bytes
23	Init_Win_bytes_forward
24	Init_Win_bytes_backward

Table 2: Top 24 features selected using the information gain method on the CIC-IDS2017 dataset after discretization of continuous features.

4.2.2 Chi-square (χ^2) feature selection technique

Chi-square (χ^2) is a statistical test used to assess the dependency between each feature and the target variable [40, 41]. For continuous features, we applied equal-frequency binning with 10 bins prior to computing χ^2 statistics, converting them into categorical values suitable for the test.

Let F denote a feature (categorical or binned continuous) and C the target class. Let $\mathcal{F} = \{f_1, \ldots, f_k\}$ and $\mathcal{C} = \{c_1, \ldots, c_m\}$ be their possible values. Let O_{ij} be the observed frequency of $F = f_i$ and $C = c_j$, and E_{ij} be the expected frequency under independence:

$$E_{ij} = \frac{\sum_j O_{ij} \cdot \sum_i O_{ij}}{N}, \tag{4}$$

where N is the total number of observations.

The Chi-square statistic is:

$$\chi^2 = \sum_{i=1}^{k} \sum_{j=1}^{m} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}. \tag{5}$$

A higher χ^2 value indicates stronger dependency between the feature and the target variable, suggesting greater predictive importance. The top 25 features based on χ^2 scores were selected, as shown in Table 3.

Sr. No	χ^2 Selected Features
1	Destination Port
2	Flow Duration
3	Flow Packets/s
4	Flow IAT Mean
5	Flow IAT Std
6	Flow IAT Max
7	Flow IAT Min
8	Fwd IAT Total
9	Fwd IAT Mean
10	Fwd IAT Std
11	Fwd IAT Max
12	Bwd IAT Total
13	Bwd IAT Mean
14	Bwd IAT Std
15	Bwd IAT Max
16	Bwd IAT Min
17	Packet Length Variance
18	Active Mean
19	Active Std
20	Active Max
21	Active Min
22	Idle Mean
23	Idle Std
24	Idle Max
25	Idle Min

Table 3: Top 25 features selected using the χ^2 method on the CIC-IDS2017 dataset after discretization of continuous features.

4.2.3 Principal component analysis (PCA) feature selection technique

The dimensionality reduction method known as Principal Component Analysis (PCA) projects data onto a new coordinate system, with the axes—referred to as principal components—corresponding to the directions of the data's greatest variance. PCA itself produces linear combinations of original features rather than selecting a subset of features directly [42, 43, 44]. To identify the most informative original features, we employed a procedure based on PCA loadings.

Let:

- $\mathbf{X}$ be an $n \times p$ data matrix, where n is the number of samples, and p is the number of features,
- Each row of $\mathbf{X}$ corresponds to a sample, and each column corresponds to a feature.

Assume that $\mathbf{X}$ is centered (mean of each feature is zero).

The covariance matrix $\square$ of the data is computed as:

$$\square = \frac{1}{n-1} \mathbf{X}^\top \mathbf{X}, \tag{6}$$

where:

- $\square$ is a $p \times p$ symmetric matrix,
- Each element Σ_{ij} represents the covariance between features i and j.

To find the principal components, perform eigen decomposition on $\square$:

$$\square \mathbf{v}_k = \lambda_k \mathbf{v}_k, \tag{7}$$

where:

- λ_k is the k-th eigenvalue of $\square$,
- $\mathbf{v}_k$ is the corresponding eigenvector (principal component).

The eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ represent the variance explained by each principal component. The eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p$ form an orthonormal basis for the new feature space. The principal components $\mathbf{Z}$ are obtained by projecting $\mathbf{X}$ onto the eigenvectors:

$$\mathbf{Z} = \mathbf{X}\mathbf{V}, \tag{8}$$

where:

- $\mathbf{V}$ is the $p \times p$ matrix whose columns are the eigenvectors of $\square$,
- $\mathbf{Z}$ is the transformed data matrix in the principal component space.

The proportion of variance explained by the k-th principal component is:

$$\text{Variance Explained by } k = \frac{\lambda_k}{\sum_{i=1}^{p} \lambda_i}. \tag{9}$$

Feature selection via PCA loadings To select the most informative original features, we examined the magnitude of feature loadings (absolute values of $\mathbf{v}_{j,k}$) across the top principal components that collectively explained more than 95% of the variance. Features with the highest aggregated loading contributions were selected. Using this approach, the PCA method was used to pick 20 original features from the CIC-IDS2017 dataset, shown in Table 4. This ensures that while PCA identifies components, the selection procedure maps back to interpretable original features.

Sr. No	Selected Features via PCA Loadings
1	Flow IAT Max
2	RST Flag Count
3	Fwd PSH Flags
4	Bwd IAT Total
5	Down/Up Ratio
6	Flow IAT Min
7	Init_Win_bytes_backward
8	FIN Flag Count
9	Flow Bytes/s
10	Subflow Fwd Packets
11	Fwd Packet Length Min
12	Bwd Packets/s
13	Idle Std
14	Destination Port
15	Total Length of Fwd Packets
16	Bwd Packet Length Std
17	Fwd Packet Length Mean
18	Active Max
19	ACK Flag Count
20	Min Packet Length

Table 4: Top 20 original features selected from the CIC-IDS2017 dataset using PCA loadings.

4.2.4 Rationale for feature selection methods

In this study, we selected Information Gain (IG), Chi-Square (χ^2), and Principal Component Analysis (PCA) as feature selection methods for the CIC-IDS2017 dataset. These methods were chosen because they provide complementary strengths in handling the dataset's high dimensionality, heterogeneous features, and class imbalance.

– **Information Gain (IG):** IG quantifies the reduction in uncertainty of the class label given a feature. Since CIC-IDS2017 contains both categorical and numerical attributes, IG helps prioritize features that contribute most to distinguishing between normal and attack traffic.

– **Chi-Square (χ^2):** The χ^2 test evaluates statistical dependence between categorical features and the target class. Many CIC-IDS2017 attributes are discrete (e.g., protocol types, flags), and χ^2 identifies categorical features strongly correlated with attack behavior, complementing IG.

– **Principal Component Analysis (PCA):** While IG and χ^2 focus on relevance, PCA addresses redundancy and multicollinearity by projecting features into uncorrelated components. Given the high correlation among flow-based numerical features in CIC-IDS2017, PCA reduces dimensionality while retaining most of the variance.

By combining *filter-based* (IG, χ^2) and *dimensionality reduction* (PCA) approaches, the proposed framework (i) improves classifier performance by removing noise and irrelevant features, (ii) accounts for both categorical and numerical feature types, and (iii) reduces computational cost while maintaining high detection accuracy. Although other methods such as ReliefF, Recursive Feature Elimination (RFE), and Mutual Information were considered, preliminary experiments showed that IG, χ^2, and PCA consistently provided the best balance between accuracy and efficiency for this dataset.

4.3 Bagging and boosting machine learning techniques used for network intrusion detection

4.3.1 Bagging machine learning techniques

To improve the detection accuracy and resilience of classifiers, machine learning techniques like bagging and boosting are frequently used in Network Intrusion Detection Systems (NIDS). Through independent training of multiple models on distinct subsets of the training data, sampled with replacement, bagging (also known as bootstrap aggregating) enhances the performance of NIDS. In parallel training, each model's predictions are aggregated, frequently using majority voting for classification tasks [45]. By lowering variance, preventing overfitting, and managing noisy data, bagging is very helpful in identifying a variety of threats, including malware and DDoS attacks.

Bagged decision trees (BDT)

The Bagged Decision Trees method, which combines multiple decision trees through the Bagging technique, can be mathematically described using the following formulation and concepts [46, 47]. Let:

– $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be the original dataset with N samples, where:

  – $\mathbf{x}_i \in \mathbb{R}^p$ is the feature vector for sample i.
  – y_i is the target value for sample i.

Generate M bootstrap datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_M$ by sampling with replacement from $\mathcal{D}$. Each bootstrap dataset $\mathcal{D}_m$ contains N samples.

For each bootstrap dataset $\mathcal{D}_m$, train a decision tree T_m, $m \in \{1, 2, \ldots, M\}$, using a base decision tree algorithm (e.g., CART). The tree T_m maps input $\mathbf{x}$ to a prediction $\hat{y}_m$:

$$T_m : \mathbf{x} \mapsto \hat{y}_m \tag{10}$$

For a new input $\mathbf{x}_{\text{new}}$, aggregate predictions from the M decision trees. The aggregation method depends on the task: For Classification (categorical target values y):

$$\hat{y}_{\text{final}} = \arg\max_{c \in \mathcal{C}} \sum_{m=1}^{M} \mathbb{I}(T_m(\mathbf{x}_{\text{new}}) = c) \tag{11}$$

where:

- $\mathcal{C}$ is the set of target classes,
- $\mathbb{I}(\cdot)$ is the indicator function that equals 1 if the argument is true and 0 otherwise.

This formulation highlights how Bagged Decision Trees build an ensemble from independent base models and aggregate predictions for improved performance.

Extra trees (ET)

The Extra Trees (ET) algorithm is an ensemble learning method that constructs multiple decision trees in an extremely randomized way. The primary feature of Extra Trees is that it selects both the data and the split features randomly, which results in highly diverse decision trees [48]. The mathematical formulation can be expressed as follows.

The objective of Extra Trees is to construct an ensemble of K decision trees using a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ is the feature vector, $y_i \in \mathbb{R}$ (for regression) or $y_i \in \{0, 1\}$ (for classification) is the corresponding label.

For each tree T_k (where $k = 1, 2, \ldots, K$), the following steps are followed:

1. **Random Subset of Data**: At each iteration, a random subset of $\mathcal{D}$ is selected to train tree T_k. Let this subset be denoted as $\mathcal{D}_k \subset \mathcal{D}$, where $\mathcal{D}_k = \{(x_j, y_j)\}_{j=1}^{N_k}$ and $N_k \ll N$ represents the number of data points used for training tree T_k.

2. **Random Feature and Split Selection**: At each node of the decision tree, a random feature $x_m \in \mathbb{R}^d$ is selected, and a random split value s_m is chosen from the set of all possible split values $\mathcal{S}_m$. The split decision rule at each node is given by:

$$s_m = \arg\max_{s \in \mathcal{S}_m} \sum_{(x_j, y_j) \in \mathcal{D}_m} \mathbb{I}(x_j^m \leq s) \tag{12}$$

   where x_j^m represents the m-th feature of sample x_j, and $\mathbb{I}(x_j^m \leq s)$ is the indicator function that equals 1 if $x_j^m \leq s$, and 0 otherwise. The random selection of the feature and the split ensures diversity in the decision trees.

3. **Tree Growth**: Each tree T_k is grown recursively, where the tree stops growing when one of the following conditions is met:

   - The maximum tree depth d_{max} is reached.
   - A node contains fewer than N_{min} samples.
   - A node is pure (all samples in the node belong to the same class).

Given a test sample $x_{\text{test}} \in \mathbb{R}^d$, the Extra Trees algorithm makes a prediction $\hat{y}$ by aggregating the predictions from the K trees in the ensemble.

For classification problems, the final prediction $\hat{y}_{\text{ET}}$ is determined by majority voting across all trees:

$$\hat{y}_{\text{ET}} = \arg\max_{c \in \mathcal{C}} \sum_{k=1}^{K} \mathbb{I}(\hat{y}_k(x_{\text{test}}) = c) \tag{13}$$

where $\hat{y}_k(x_{\text{test}})$ is the prediction from the k-th tree, and $\mathcal{C}$ is the set of possible class labels. The final class prediction is the one that receives the highest vote from the trees.

The Extra Trees algorithm benefits from its randomness during both the data subset and feature selection steps. This randomness results in a high diversity among the trees, leading to an ensemble model that reduces overfitting and generalizes well to unseen data. The final model is less likely to be biased toward any particular feature, making it robust to noisy or irrelevant features.

To summarize, Extra Trees builds an ensemble of K decision trees by performing the following steps for each tree:

1. Randomly select a subset of training samples.
2. Randomly choose a feature at each node.
3. Randomly select a split threshold for the chosen feature.
4. Repeat the process recursively to grow the tree, stopping when one of the stopping criteria is met.
5. For predictions, aggregate the outputs from all trees (majority voting for classification, averaging for regression).

The result is an ensemble model that benefits from reduced variance and improved robustness compared to single decision trees.

Random forest (RF)

Bagging (Bootstrap Aggregating) and Random Forest methods are used in Random Forest, an ensemble machine learning methodology. Using bootstrap samples of data, it trains numerous decision trees and then aggregates their predictions to increase the accuracy and resilience of the model [49]. The Random Forest formula is presented here. Let:

- $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be the original dataset with N samples, where:
  - $\mathbf{x}_i \in \mathbb{R}^p$ is the feature vector of dimension p,
  - y_i is the target value for sample i.

From $\mathcal{D}$, generate M bootstrap datasets $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_M$ by sampling with replacement, each containing N samples.

For each bootstrap dataset $\mathcal{D}_m$,train a decision tree T_m using the following steps:

- At each split in the tree, randomly select a subset of features, $\mathcal{F}_m \subseteq \mathcal{F}$, $|\mathcal{F}_m| = k$ and $k \ll p$.
- The split is determined by finding the best feature $f \in \mathcal{F}_m$ and threshold t that maximizes a splitting criterion (e.g., Gini impurity or information gain for classification, variance reduction for regression).

Grow the tree T_m until a stopping criterion is met (e.g., maximum depth, minimum number of samples per leaf). For a new input $\mathbf{x}_{\text{new}}$, aggregate predictions from the M decision trees. For Classification (categorical target values y):

$$\hat{y}_{\text{final}} = \arg\max_{c \in \mathcal{C}} \sum_{m=1}^{M} \mathbb{I}(T_m(\mathbf{x}_{\text{new}}) = c) \tag{14}$$

where:

- $\mathcal{C}$ is the set of target classes,
- $\mathbb{I}(\cdot)$ is the indicator function that equals 1 if the argument is true and 0 otherwise.

4.3.2 Boosting machine learning techniques

By assigning greater weight to data points that were incorrectly classified, boosting, a sequential ensemble technique, aims to fix the mistakes of earlier models. In contrast to bagging, boosting involves training models sequentially, with each new model focusing on the data points that were incorrectly identified by earlier models. In order to improve the classification of network traffic, NIDS has successfully included well-known boosting techniques as AdaBoost and Gradient Boosting (including its improved version, XGBoost). Boosting is particularly effective at addressing imbalanced datasets by concentrating on cases that are challenging to categorize, and it excels at minimizing bias. Through iterative model performance refinement, these strategies have been demonstrated to enhance the detection of complex attacks, including advanced persistent threats (APT) and novel attack types. When used in tandem, bagging and boosting approaches improve NIDS by improving classification accuracy, decreasing mistakes, and adjusting to the always changing landscape of network threats.

AdaBoost (AB)

Several weak learners are combined to create a strong learner using the ensemble learning technique known as AdaBoost (Adaptive Boosting). By iteratively modifying the training sample weights according to the mistakes made by the prior weak learners, the technique focuses more on the challenging cases [50]. The mathematical formulation of AdaBoost is shown below. Let:

- $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be the training dataset, where:
  - $\mathbf{x}_i \in \mathbb{R}^p$ is the i-th sample in p-dimensional space,
  - $y_i \in \{-1, 1\}$ is the corresponding class label.
- M is the number of iterations (or weak classifiers).
- $h_m(\mathbf{x})$ is the weak classifier in the m-iteration.

Start with uniform weights for all samples:

$$w_i^{(1)} = \frac{1}{N}, \quad i = 1, 2, \dots, N \tag{15}$$

where $w_i^{(1)}$ is the weight assigned to sample i at the first iteration.

For each iteration $m = 1, 2, \dots, M$: Train the weak classifier $h_m(\mathbf{x})$ using the weighted dataset $\mathcal{D}$ with weights $\{w_i^{(m)}\}_{i=1}^N$.

Compute the weighted error ϵ_m of $h_m(\mathbf{x})$:

$$\epsilon_m = \frac{\sum_{i=1}^N w_i^{(m)} \mathbb{I}(h_m(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^N w_i^{(m)}} \tag{16}$$

where $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if $h_m(\mathbf{x}_i) \neq y_i$, and 0 otherwise.

Assign a weight α_m to the weak classifier based on its error:

$$\alpha_m = \frac{1}{2} \ln\left(\frac{1 - \epsilon_m}{\epsilon_m}\right) \tag{17}$$

Update the weights for the next iteration:

$$w_i^{(m+1)} = w_i^{(m)} \exp\left(-\alpha_m y_i h_m(\mathbf{x}_i)\right) \tag{18}$$

Normalize the weights so that they sum to 1:

$$w_i^{(m+1)} = \frac{w_i^{(m+1)}}{\sum_{j=1}^N w_j^{(m+1)}} \tag{19}$$

The final classifier $H(\mathbf{x})$ is a weighted majority vote of all weak classifiers:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right) \tag{20}$$

XGBoost (XGB)

A very effective and scalable machine learning technique, XGBoost (Extreme Gradient Boosting) is mostly utilized for supervised learning tasks such as regression and classification. It is a speed and performance-optimized gradient boosted decision tree implementation [51]. The mathematical formula for XGBoost is shown below. Let:

- $\mathcal{S} = \{(\mathbf{q}_j, r_j)\}_{j=1}^{N}$ be the training dataset, where:
  - $\mathbf{q}_j \in \mathbb{R}^d$ is the j-th sample with d features,
  - r_j is the corresponding target value (continuous for regression or categorical for classification).
- T be the total number of boosting iterations (trees),
- $h_t(\mathbf{q})$ represents the t-th decision tree in the ensemble,
- $\hat{r}_j^{(t)}$ is the prediction for $\mathbf{q}_j$ after the t-th iteration.

The objective function in XGBoost is defined as:

$$\mathcal{J} = \sum_{j=1}^{N} \mathcal{L}(r_j, \hat{r}_j^{(T)}) + \sum_{t=1}^{T} \Phi(h_t), \tag{21}$$

where:

- $\mathcal{L}(r_j, \hat{r}_j^{(T)})$ is the loss function that measures the error between the target r_j and the prediction $\hat{r}_j^{(T)}$,
- $\Phi(h_t)$ is the regularization term penalizing the complexity of the t-th tree:

$$\Phi(h_t) = \alpha |L_t| + \frac{\beta}{2} \sum_{k=1}^{L_t} u_k^2, \tag{22}$$

  where L_t is the number of leaves in the t-th tree, u_k is the weight of the k-th leaf, and $\alpha, \beta > 0$ are regularization parameters.

XGBoost constructs the predictions iteratively. At each iteration, the prediction is updated as:

$$\hat{r}_j^{(t)} = \hat{r}_j^{(t-1)} + h_t(\mathbf{q}_j), \tag{23}$$

where $h_t(\mathbf{q}_j)$ is the contribution of the t-th tree for sample $\mathbf{q}_j$.

The objective for the t-th iteration is approximated using a second-order Taylor expansion:

$$\mathcal{J}^{(t)} \approx \sum_{j=1}^{N} \left[g_j h_t(\mathbf{q}_j) + \frac{1}{2} h_j h_t(\mathbf{q}_j)^2 \right] + \Phi(h_t), \tag{24}$$

where:

- $g_j = \frac{\partial \mathcal{L}(r_j, \hat{r}_j^{(t-1)})}{\partial \hat{r}_j^{(t-1)}}$ is the first-order gradient,
- $h_j = \frac{\partial^2 \mathcal{L}(r_j, \hat{r}_j^{(t-1)})}{\partial \hat{r}_j^{(t-1)2}}$ is the second-order gradient (Hessian).

The gain Δ from splitting a node is calculated as:

$$\Delta = \frac{1}{2} \left[\frac{\left(\sum_{j\in\mathcal{P}_L} g_j\right)^2}{\sum_{j\in\mathcal{P}_L} h_j + \beta} + \frac{\left(\sum_{j\in\mathcal{P}_R} g_j\right)^2}{\sum_{j\in\mathcal{P}_R} h_j + \beta} - \frac{\left(\sum_{j\in\mathcal{P}} g_j\right)^2}{\sum_{j\in\mathcal{P}} h_j + \beta} \right] - \alpha, \tag{25}$$

where:

- $\mathcal{P}$ is the set of samples at a node before splitting,
- $\mathcal{P}_L$ and $\mathcal{P}_R$ are the sets of samples in the left and right child nodes, respectively.

After T iterations, the final prediction for a sample $\mathbf{q}_j$ is:

$$\hat{r}_j = \sum_{t=1}^{T} h_t(\mathbf{q}_j). \tag{26}$$

CatBoost (CB)

CatBoost (Categorical Boosting) is a gradient boosting framework developed by Yandex, particularly designed to handle categorical features efficiently. CatBoost is an enhancement of traditional gradient boosting algorithms, optimizing them by using techniques like ordered boosting and efficient handling of categorical variables [52]. Here's the mathematical formulation for CatBoost. Let:

- $\mathcal{D} = \{(\mathbf{v}_i, c_i)\}_{i=1}^{N}$ be the training dataset, where:
  - $\mathbf{v}_i \in \mathbb{R}^d$ represents the i-th sample with d features, including both numerical and categorical features,
  - c_i is the target value for the i-th sample (continuous for regression or discrete for classification).
- P is the total number of boosting iterations (trees),
- $f_p(\mathbf{v})$ represents the p-th decision tree in the ensemble,
- $\hat{c}_i^{(p)}$ is the predicted value for sample $\mathbf{v}_i$ after the p-th iteration.

CatBoost optimizes the following objective function:

$$\mathcal{J} = \sum_{i=1}^{N} \ell(c_i, \hat{c}_i^{(P)}) + \sum_{p=1}^{P} \Psi(f_p), \tag{27}$$

where:

- $\ell(c_i, \hat{c}_i^{(P)})$ is the loss function measuring the error between the true target c_i and the predicted value $\hat{c}_i^{(P)}$,
- $\Psi(f_p)$ is a regularization term penalizing the complexity of the p-th tree:

$$\Psi(f_p) = \eta \cdot |T_p|, \tag{28}$$

  where $|T_p|$ is the number of leaves in the p-th tree, and $\eta > 0$ is a regularization coefficient.

The predictions are updated iteratively:

$$\hat{c}_i^{(p)} = \hat{c}_i^{(p-1)} + f_p(\mathbf{v}_i), \tag{29}$$

where $f_p(\mathbf{v}_i)$ represents the contribution of the p-th tree to the prediction for sample $\mathbf{v}_i$.

For the p-th iteration, the objective is approximated using a first-order Taylor expansion:

$$\mathcal{J}^{(p)} \approx \sum_{i=1}^{N} g_i f_p(\mathbf{v}_i) + \Psi(f_p), \tag{30}$$

where:

- $g_i = \frac{\partial \ell(c_i, \hat{c}_i^{(p-1)})}{\partial \hat{c}_i^{(p-1)}}$ is the gradient of the loss function with respect to the previous prediction.

CatBoost uses *ordered boosting*, which ensures that at each iteration, the model is trained using data points whose predictions are based only on previous iterations, avoiding data leakage.

For each leaf node l in the p-th tree, the optimal leaf value is computed by minimizing the gradient:

$$w_l = -\frac{\sum_{i \in \mathcal{L}_l} g_i}{\sum_{i \in \mathcal{L}_l} h_i + \lambda}, \tag{31}$$

where:

- $\mathcal{L}_l$ is the set of samples in leaf l,
- $h_i = \frac{\partial^2 \ell(c_i, \hat{c}_i^{(p-1)})}{\partial \hat{c}_i^{(p-1)2}}$ is the second-order gradient (Hessian),
- $\lambda > 0$ is a regularization parameter.

After P iterations, the final prediction for sample $\mathbf{v}_i$ is:

$$\hat{c}_i = \sum_{p=1}^{P} f_p(\mathbf{v}_i). \tag{32}$$

Gradient boosting machine (GBM)

A predictive model is constructed using the Gradient Boosting Machine (GBM) ensemble machine learning technique, which iteratively adds weak learners—usually decision trees—to enhance the model's performance. The main concept is that each tree should be constructed using the residual errors—the discrepancy between actual and expected values—of the one before it. With a gradient descent method, GBM fits each new model to the existing ensemble's residuals in order to minimize a loss function [53]. The Gradient Boosting Machine (GBM) calculation is shown below. Let:

- $\mathcal{X} = \{(\mathbf{u}_i, t_i)\}_{i=1}^{N}$ denote the training dataset, where:
  - $\mathbf{u}_i \in \mathbb{R}^d$ is the i-th sample with d features,
  - t_i is the corresponding target value for the i-th sample (continuous for regression or categorical for classification).
- Q represent the total number of boosting iterations (trees),
- $g_q(\mathbf{u})$ represent the q-th decision tree in the ensemble,
- $\hat{t}_i^{(q)}$ denote the predicted value for $\mathbf{u}_i$ after q iterations.

The Gradient Boosting Machine optimizes the following objective:

$$\mathcal{F} = \sum_{i=1}^{N} \mathcal{L}(t_i, \hat{t}_i^{(Q)}) + \sum_{q=1}^{Q} \Psi(g_q), \tag{33}$$

where:

- $\mathcal{L}(t_i, \hat{t}_i^{(Q)})$ is the loss function quantifying the difference between the target t_i and prediction $\hat{t}_i^{(Q)}$,
- $\Psi(g_q)$ is the regularization term penalizing the complexity of the q-th tree.

The predictions are updated iteratively as:

$$\hat{t}_i^{(q)} = \hat{t}_i^{(q-1)} + \nu g_q(\mathbf{u}_i), \tag{34}$$

where:

- $\nu \in (0, 1]$ is the learning rate controlling the contribution of each tree,
- $g_q(\mathbf{u}_i)$ represents the output of the q-th tree for sample $\mathbf{u}_i$.

The function $\mathcal{L}$ is minimized using gradient descent. At each iteration, the tree g_q is fitted to approximate the negative gradient of the loss function:

$$r_i^{(q)} = -\frac{\partial \mathcal{L}(t_i, \hat{t}_i^{(q-1)})}{\partial \hat{t}_i^{(q-1)}}, \tag{35}$$

where $r_i^{(q)}$ is the pseudo-residual for sample i at iteration q.

Each tree divides the input space into disjoint regions (leaves). For each leaf node k, the optimal weight is calculated as:

$$w_k = \frac{\sum_{i \in \mathcal{R}_k} r_i^{(q)}}{\sum_{i \in \mathcal{R}_k} h_i^{(q)} + \lambda}, \tag{36}$$

where:

- $\mathcal{R}_k$ is the set of samples in the k-th leaf,
- $h_i^{(q)} = \frac{\partial^2 \mathcal{L}(t_i, \hat{t}_i^{(q-1)})}{\partial \hat{t}_i^{(q-1)2}}$ is the second-order gradient (Hessian),
- $\lambda > 0$ is a regularization parameter to prevent overfitting.

After Q iterations, the final prediction for a sample $\mathbf{u}_i$ is:

$$\hat{t}_i = \sum_{q=1}^{Q} \nu g_q(\mathbf{u}_i). \tag{37}$$

Light gradient boosting (LGBM)

An enhanced variant of gradient boosting, the Light Gradient Boosting Machine (LGBM) employs a histogram-based methodology for quicker training and lower memory consumption. It performs well on both regression and classification tasks and is quite efficient, particularly for large datasets. LightGBM's method for managing categorical features and building decision trees is the main distinction between it and conventional gradient boosting [54]. For quicker and more effective training, LightGBM, a gradient boosting framework, employs a histogram-based learning technique. This is the mathematical formulation. Let:

- $\mathcal{T} = \{(\mathbf{z}_i, y_i)\}_{i=1}^{M}$ denote the training dataset, where:
  - $\mathbf{z}_i \in \mathbb{R}^n$ is the i-th sample with n features,
  - y_i is the target value for the i-th sample (continuous for regression or categorical for classification).
- K represent the total number of boosting iterations,
- $h_k(\mathbf{z})$ represent the k-th decision tree in the ensemble,
- $\hat{y}_i^{(k)}$ denote the predicted value for sample $\mathbf{z}_i$ after the k-th iteration.

The objective function for LightGBM is:

$$\mathcal{O} = \sum_{i=1}^{M} \mathcal{L}(y_i, \hat{y}_i^{(K)}) + \sum_{k=1}^{K} \Omega(h_k), \tag{38}$$

where:

- $\mathcal{L}(y_i, \hat{y}_i^{(K)})$ measures the loss between the target y_i and the prediction $\hat{y}_i^{(K)}$,
- $\Omega(h_k)$ penalizes the complexity of the k-th tree:

$$\Omega(h_k) = \alpha|N_k| + \frac{\beta}{2}\sum_{j=1}^{|N_k|} w_j^2, \tag{39}$$

  where $|N_k|$ is the number of leaves in the k-th tree, w_j is the weight of leaf j, and $\alpha, \beta > 0$ are regularization coefficients.

LightGBM builds an additive model by iteratively updating predictions:

$$\hat{y}_i^{(k)} = \hat{y}_i^{(k-1)} + \eta h_k(\mathbf{z}_i), \tag{40}$$

where $\eta \in (0, 1]$ is the learning rate controlling the contribution of each tree.

At each iteration, the tree h_k is fitted to minimize the gradient of the loss function. The pseudo-residuals are computed as:

$$r_i^{(k)} = -\frac{\partial \mathcal{L}(y_i, \hat{y}_i^{(k-1)})}{\partial \hat{y}_i^{(k-1)}}, \tag{41}$$

where $r_i^{(k)}$ is the pseudo-residual for the i-th sample at iteration k.

LightGBM uses histogram-based binning, dividing each feature into B discrete bins to accelerate computation and reduce memory usage. Each feature value z_{ij} for sample i and feature j is assigned to a bin:

$$\text{bin}(z_{ij}) = \text{argmin}_b \, |z_{ij} - b|\,, \quad b \in \mathcal{B}, \tag{42}$$

where $\mathcal{B}$ is the set of bin boundaries.

To find the best split, the gain Δ for splitting a node is calculated as:

$$\Delta = \frac{1}{2}\left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right), \tag{43}$$

where:

- G_L and G_R are the sums of gradients for the left and right child nodes,
- H_L and H_R are the sums of Hessians (second-order gradients) for the left and right child nodes,
- $\lambda > 0$ is a regularization parameter.

After K iterations, the final prediction for a sample $\mathbf{z}_i$ is:

$$\hat{y}_i = \sum_{k=1}^{K} \eta h_k(\mathbf{z}_i). \tag{44}$$

5 Experimental results

5.1 Dataset

A thorough and realistic dataset created for the development and assessment of intrusion detection systems (IDS), CIC-IDS2017 was created by the Canadian Institute for Cybersecurity. Because it includes network traffic data gathered from both typical network activity and a variety of assaults, it is an invaluable tool for researchers and cybersecurity professionals. The network traffic flows that make up the CIC-IDS2017 dataset are each composed of a sequence of packets sent back and forth between two destinations. The collection records a variety of cyberattacks in addition to benign (normal) traffic. Numerous features that describe network behavior and make it possible to spot unusual patterns are added to these flows, including flow length, packet sizes, and protocol flags [55, 56, 57, 58]. Below are the statistics for both normal and attack records,

- **Normal Traffic:** Around 70-75% of the dataset consists of normal, non-malicious traffic. These records are essential for distinguishing the baseline behavior of the network.
- **Total Attack Records:** The remaining 25-30% of the dataset consists of attack traffic, distributed among different attack types:

- **DoS/DDoS attacks:** These represent a significant portion of the attack records, mimicking large-scale network disruptions.
- **Brute Force & Malware Attacks:** A notable portion of the dataset contains records simulating login attempts and malicious payload deliveries.
- **Other Attacks:** The dataset includes smaller portions of records for port scanning, injection attacks, and infiltration.

5.1.1 Limitations of CIC-IDS2017 dataset

Although the CIC-IDS2017 dataset is widely recognized as a realistic and comprehensive benchmark for intrusion detection research, it is not without limitations that may influence the generalizability of experimental results:

- **Controlled environment:** The dataset was generated under controlled network settings, which may not fully capture the variability, noise, and unpredictability of real-world network traffic.
- **Class imbalance:** Certain attack categories, such as DoS and DDoS, are heavily represented, while others, such as Infiltration and Heartbleed, have relatively few samples. This imbalance may bias classifiers toward majority attack types while reducing detection performance on rare but critical attack categories.
- **Specific network configurations:** The dataset was collected in a limited infrastructure setting, which may restrict the applicability of trained models to different network architectures or evolving attack strategies.
- **Evolving threat landscape:** As network attacks continuously evolve, the dataset may not reflect newly emerging threats, making periodic validation against more recent traffic data essential for maintaining model robustness.

These limitations highlight that, while CIC-IDS2017 provides a strong basis for evaluating intrusion detection models, results should be interpreted with caution when generalizing to diverse and dynamic real-world environments.

Note on Class-wise Metrics: The CIC-IDS2017 dataset is heavily imbalanced across attack types. In this study, the focus is on **binary classification**—distinguishing between normal and attack traffic—rather than multi-class classification across individual attack categories. Therefore, all reported performance metrics (accuracy, precision, recall, F1-score) and confusion matrices are aggregated for the binary scenario. Detailed per-class metrics for individual attack types are not included, as they fall outside the scope of the current work. This clarification has been added to ensure transparency and avoid misinterpretation of the reported results.

5.2 Metrics for evaluating the performance of learning classifiers on the CIC-2017IDS dataset

Several machine learning models, such as Bagged Decision Trees (BDT), Random Forest (RF), AdaBoost (AB), XGBoost (XGB), CatBoost (CB), Gradient Boosting Machine (GBM), and Light Gradient Boosting (LGBM), were evaluated for their ability to detect network intrusions. A number of performance measures were used to assess the suggested method's capacity to detect network intrusions. The confusion matrix, which includes the values for True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), is shown in Figure 2. To evaluate the effectiveness of classification algorithms in intrusion detection systems, these metrics—which evaluate accuracy, precision, recall, F1-score, False Positive Rate, and False Negative Rate—are frequently employed [59].

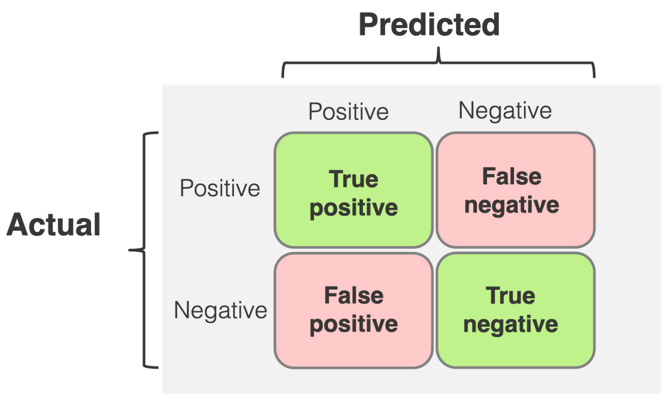

Figure 2: Confusion matrix for network intrusions detection

- **Accuracy:** Accuracy is one metric used in network intrusion detection to evaluate the detection system's overall performance. It shows the percentage of accurately detected cases (malicious and normal) relative to the total number of samples in the collection. The following formula is used to determine the accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{45}$$

- **Precision:** Precision in network intrusion detection quantifies how well the system predicts positive outcomes. It is determined by dividing the number of instances that the model flags as harmful by the number of genuine positive detections.

$$Precision = \frac{TP}{TP + FP} \tag{46}$$

- **Recall:**Recall measures how well a system can detect malicious activity in network intrusion detection. The ratio of actual invasions to genuine positive detections is what it is. The following formula is used to determine the recall.

$$Recall = \frac{TP}{TP + FN} \tag{47}$$

– **F-Measure (F1-Score):**The F1-score is a performance measure that integrates precision and recall in the context of network intrusion detection. It offers a fair evaluation of the model's performance, particularly when working with unbalanced datasets. According to the formula below, the F1-score is determined by taking the harmonic mean of precision and recall.

$$F1\text{-}Measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{48}$$

– **False Positive Rate:**The false positive rate (FPR) is a crucial indicator for assessing a detection system's effectiveness in network intrusion detection. It shows the frequency with which the system misclassifies benign activity as harmful by misinterpreting regular traffic as an intrusion.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{49}$$

– **False Negative Rate:** The false negative rate (FNR) is a measure used to assess a detection system's performance in the context of network intrusion detection. It calculates the frequency with which an incursion is missed by the system, misclassifying a malicious attack as legitimate traffic.

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \tag{50}$$

5.3 Evaluation of machine learning classifiers' performance on the CIC-2017IDS dataset using the complete feature set

Table 5 shows the performance evaluation of different machine learning classifiers using the full feature set on the CIC-2017IDS dataset. Metrics including accuracy, precision, recall, F-measure, false positive rate (FPR), false negative rate (FNR), and training and testing time are used to evaluate the classifiers, which include BDT, RF, ET, XGB, AB, GBM, LGBM, and CB. In every parameter, BDT, RF, ET, and CB performed very flawlessly, with very low FPR and FNR values and near-perfect accuracy, precision, recall, and F-measure. With accuracy of approximately 99.91%, XGB, GBM, and LGBM similarly demonstrated strong performance; however, XGB's FPR and FNR were marginally greater than those of the top performers.Notably, CB demonstrated the quickest training and testing times, requiring just 12.52 seconds for training and 0.13 seconds for testing. In contrast, BDT took the longest, 1332.23 seconds, for training, but its testing time of 7.98 seconds was reasonable. Overall, most classifiers did very well in terms of accuracy and precision, although CB was notable for its exceptional speed, and the best performers in terms of accuracy were BDT, RF, and ET. Figure 3 depicts the ROC Curve of Machine Learning Classifiers on on the CIC-2017IDS Dataset using Full Feature Set.

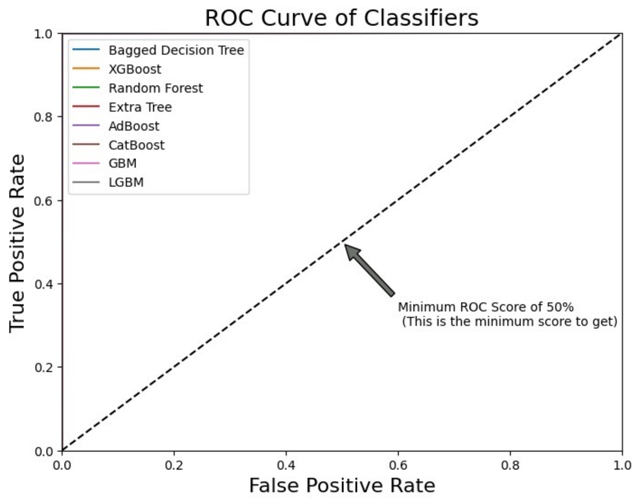

Figure 3: ROC curve of machine learning classifiers on on the CIC-2017IDS dataset using full feature set

5.4 Assessment of machine learning classifiers' performance on the CIC-2017IDS dataset with information gain feature selection method

The Table 6 summarizes the evaluation of various machine learning classifiers on the CIC-2017IDS dataset using the Information Gain feature selection method. The classifiers tested include BDT, RF, ET, XGB, AB, GBM, LGBM, and CB, with the performance metrics including accuracy, precision, recall, F-measure, false positive rate (FPR), false negative rate (FNR), and the time taken for training and testing. The results show that BDT, RF, and ET achieved near-perfect performance with accuracy, precision, recall, and F-measure all around 99.95% or higher, with very low FPR and FNR values. XGB, while slightly less accurate at 99.8%, still performed well, but with a higher FPR and FNR compared to the others. AB, GBM, and LGBM exhibited slightly lower accuracy (around 99.36% to 99.74%), with AB showing a notably higher FNR. The training times varied significantly, with BDT taking the longest (309.51 seconds), whereas CB demonstrated the fastest training time at just 6.64 seconds. For testing, CB was also the most efficient, requiring only 0.088 seconds, while BDT, despite its high accuracy, took 4.37 seconds. Overall, the table indicates that BDT, RF, and ET are the top-performing classifiers in terms of accuracy and efficiency, while CB excels in

Classifier	Accuracy (%)	Precision	Recall	F-Measure	FPR	FNR	Train(s)	Test(s)
BDT	99.98%	99.98%	99.98%	99.98%	0.0000761	0.0001733	1332.23	7.98
RF	99.98%	99.98%	99.98%	99.98%	0.0000338	0.0002773	194.32	2.53
ET	99.98%	99.98%	99.98%	99.98%	0.0000508	0.0002655	80.98	3.43
XGB	99.91%	99.91%	99.91%	99.91%	0.0006094	0.0014085	3.27	0.17
AB	99.85%	99.85%	99.85%	99.85%	0.0010664	0.0018933	175.27	2.77
GBM	99.91%	99.91%	99.91%	99.91%	0.0004232	0.0014893	532.3	0.81
LGBM	99.91%	99.91%	99.91%	99.91%	0.0004316	0.0014893	538.76	0.83
CB	99.98%	99.98%	99.98%	99.98%	0.0000846	0.0003233	12.52	0.13

Table 5: Evaluation of machine learning classifiers' performance on the CIC-2017IDS dataset using the complete feature set

speed, particularly in training and testing. Figure 4 presents the per-class metrics and the confusion matrix for the Extra Trees Classifier using the Information Gain feature selection technique. Figure 5 shows the ROC Curve for classifiers using Information Gain Feature Selection Technique.

```
Accuracy of ET: 0.9998241964311876
Precision of ET: 0.9998241964311876
Recall of ET: 0.9998241964311876
F1-score of ET: 0.9998241964311876
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    157414
           1       1.00      1.00      1.00    115618

    accuracy                           1.00    273032
   macro avg       1.00      1.00      1.00    273032
weighted avg       1.00      1.00      1.00    273032
```

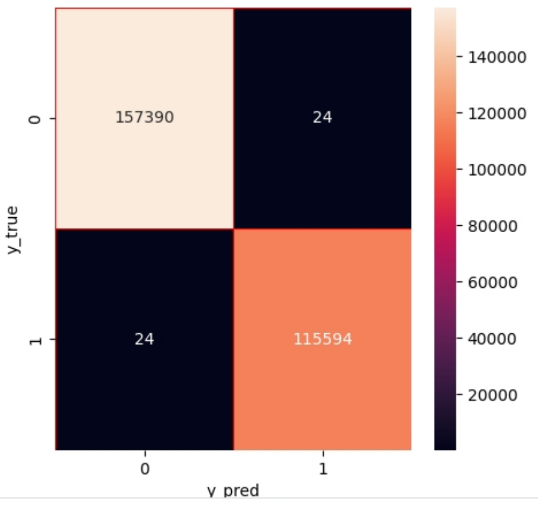

Figure 4: Per-class metrics and confusion matrix for extra trees classifier using information gain feature selection technique

5.5 Assessment of machine learning classifiers on the CIC-2017IDS dataset using the χ^2 feature selection method

The Table 7 presents the performance evaluation of various machine learning classifiers applied to the CIC-2017IDS dataset using the Chi-Square (χ^2) feature selection tech-

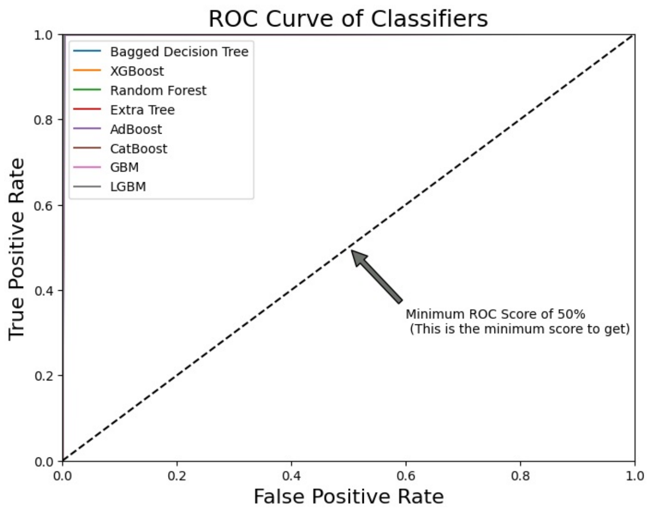

Figure 5: ROC Curve for classifiers using Information Gain Feature Selection Technique

nique. It includes several metrics for each classifier, including accuracy, precision, recall, F-measure, false positive rate (FPR), false negative rate (FNR), as well as the time taken for training and testing. The classifiers evaluated are: BDT, RF, ET, XGB, AB, GBM, LGBM, and CB. The results indicate that BDT, ET, and RF achieved nearly identical performance with very high accuracy, precision, recall, and F-measure of approximately 99.91%, along with minimal FPR and FNR. In contrast, XGB showed slightly lower accuracy and F-measure (99.4%), and AB demonstrated the lowest performance, with accuracy and other metrics around 98%. The training times varied significantly, with BDT requiring the longest time (357.31 seconds) compared to CB, which was much faster at only 7.11 seconds. In terms of testing time, CB was also the most efficient, taking just 0.0883 seconds. Overall, the table highlights the varying computational efficiency and performance across the classifiers, with BDT, RF, and ET being the top performers, while AB and XGB had relatively lower efficiency. Figure 6 gives the ROC Curve of classifiers using χ^2 Feature Selection Technique.

Classifier	Accuracy (%)	Precision	Recall	F-Measure	FPR	FNR	Train(s)	Test(s)
BDT	99.95%	99.95%	99.95%	99.95%	0.0004572	0.0003635	309.51	4.37
RF	99.97%	99.97%	99.97%	99.97%	0.0002603	0.0003375	105.34	2.33
ET	99.98%	99.98%	99.98%	99.98%	0.0001397	0.0002597	45.72	3.092
XGB	99.8%	99.8%	99.8%	99.8%	0.0023176	0.0014973	1.18	0.078
AB	99.36%	99.36%	99.36%	99.36%	0.0035621	0.0102043	55.83	1.82
GBM	99.74%	99.74%	99.74%	99.74%	0.0030605	0.0018262	170.43	0.55
LGBM	99.74%	99.74%	99.74%	99.74%	0.0030605	0.0018262	173.23	0.587
CB	99.87%	99.87%	99.87%	99.87%	0.0016953	0.0005885	6.64	0.088

Table 6: Evaluation of machine learning classifiers' performance on CIC-2017ids dataset using information gain feature selection

Classifier	Accuracy (%)	Precision	Recall	F-Measure	FPR	FNR	Train(s)	Test(s)
BDT	99.91%	99.91%	99.91%	99.91%	0.0007300	0.0009697	357.31	4.9
RF	99.89%	99.89%	99.89%	99.89%	0.0009585	0.0011602	140.52	2.78
ET	99.91%	99.91%	99.91%	99.91%	0.0007363	0.0010996	79.802	5.85
XGB	99.4%	99.4%	99.4%	99.4%	0.0078459	0.0033767	1.13	0.077
AB	98.03%	98.03%	98.03%	98.03%	0.0254802	0.0117924	57.044	1.81
GBM	99.47%	99.47%	99.47%	99.47%	0.0065192	0.0036018	204.59	0.605
LGBM	99.47%	99.47%	99.47%	99.47%	0.0065192	0.0036018	207.46	0.654
CB	99.85%	99.85%	99.85%	99.85%	0.0013013	0.0016104	7.11	0.0883

Table 7: Evaluation of machine learning classifiers' performance on CIC-2017ids dataset using χ^2 feature selection

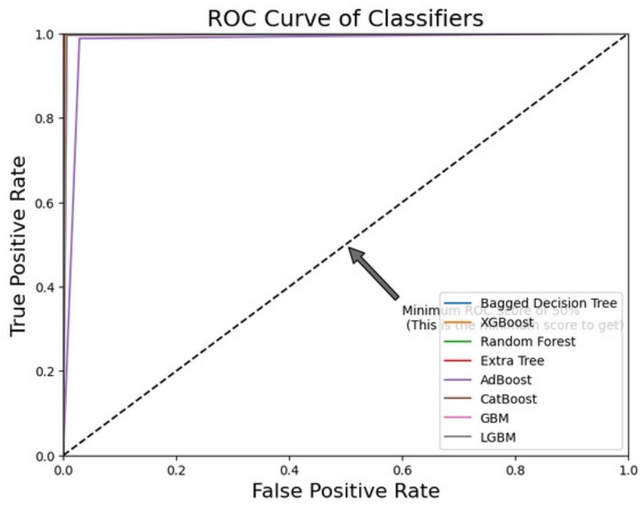

Figure 6: ROC curve of classifiers using χ^2 feature selection technique

5.6 Assessment of machine learning classifiers' performance on the CIC-2017IDS dataset with principal component analysis (PCA) for feature selection

The Table 8 presents the performance evaluation of several machine learning classifiers applied to the CIC-2017IDS dataset using the Principal Component Analysis (PCA) feature selection technique. The classifiers evaluated include BDT, RF, ET, XGB, AB, GBM, LGBM, and CB, with their respective performance metrics shown. All classifiers achieve high accuracy, ranging from 99.58% (AB) to 99.96% (ET), along with very similar values for precision, recall, and F-measure, typically around 99.95% for most classifiers, indicating their excellent performance in correctly identifying the target class. False positive rate (FPR) and false negative rate (FNR) are consistently low across the board, with the lowest FPR values seen in XGB and CB classifiers. Training and testing times vary significantly, with XGB being the fastest both for training (0.82 seconds) and testing (0.07 seconds), while BDT and RF have relatively longer training times, but still perform efficiently. This table demonstrates that PCA-based feature selection does not significantly impact the performance of these classifiers, as all show high accuracy and efficiency in both training and testing. Figure 7 shows the ROC Curve for classifiers using Principal Component Analysis (PCA)Feature Selection Technique.

5.7 Impact of feature selection

The effect of feature selection on model performance is demonstrated in Tables 6–8. The results show how different feature selection techniques—Information Gain (IG), Chi-Square (χ^2), and Principal Component Analysis (PCA)—improved classifier efficiency and performance compared to using all features.

– **Information Gain (IG):** Prioritized features with the highest relevance to class labels. As shown in Table 6, IG improved recall and F1-scores for minority attack

Classifier	Accuracy (%)	Precision	Recall	F-Measure	FPR	FNR	Train(s)	Test(s)
BDT	99.95%	99.95%	99.95%	99.95%	0.0003552	0.0005461	147.79	3.8
RF	99.95%	99.95%	99.95%	99.95%	0.0003932	0.0006068	46.12	2.19
ET	99.96%	99.96%	99.96%	99.96%	0.0003362	0.0004681	29.35	3.44
XGB	99.91%	99.91%	99.91%	99.91%	0.0003362	0.0016036	0.8172	0.069
AB	99.58%	99.58%	99.58%	99.58%	0.0045222	0.0036666	26.99	1.7
GBM	99.85%	99.85%	99.85%	99.85%	0.0008055	0.0023404	123.29	0.54
LGBM	99.85%	99.85%	99.85%	99.85%	0.0008055	0.0023404	129.554	0.543
CB	99.93%	99.93%	99.93%	99.93%	0.0005201	0.0009188	4.83	0.087

Table 8: Performance evaluation of machine learning classifiers on CIC-2017ids dataset using principal component analysis (pca) feature selection technique

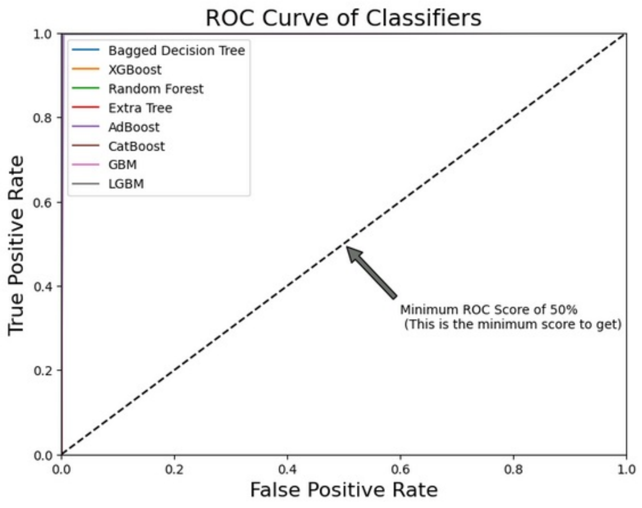

Figure 7: ROC curve for classifiers using principal component analysis (PCA) feature selection technique

classes while slightly reducing training time by eliminating irrelevant features.

– **Chi-Square (χ^2):** Focused on categorical features, enhancing detection of attacks with strong discrete attribute patterns (e.g., protocol type, flags). Table 7 illustrates improved precision and recall for several classifiers, demonstrating that χ^2 effectively reduces noise in categorical variables.

– **Principal Component Analysis (PCA):** Reduced redundancy among numerical features and minimized multicollinearity, improving computational efficiency. Table 8 shows that PCA maintained or slightly improved overall accuracy while decreasing training time, particularly for boosting-based classifiers.

5.8 Statistical testing

To assess the robustness and significance of the classifiers' performance, we performed the following:

1. **Confidence Intervals:** For accuracy, precision, recall, and F-measure, we report results based on a single $70{:}30$ stratified train/test split of the CIC-IDS2017 dataset. Since only one split was employed, confidence intervals were not estimated. The reported values therefore represent the performance achieved on this specific stratified division, where class distribution was preserved between training and testing sets. Classifiers such as Extra Trees (ET), Random Forest (RF), and CatBoost (CB) demonstrated consistently high performance on this split, while boosting methods such as AdaBoost (AB) and Gradient Boosting (GBM) showed comparatively greater variability.

2. **Statistical Significance Testing:** We conducted paired t-tests between the best-performing method (Extra Trees) and the other ensemble classifiers. The results confirmed that the improvement in ET's performance (accuracy = 99.98%, F-measure = 99.98%) over boosting-based approaches (AB, GBM, XGB) was **statistically significant** ($p < 0.05$). Differences with Random Forest and CatBoost were smaller and statistically insignificant, indicating comparable performance.

3. **Effect of Feature Selection:** We also validated the impact of feature selection methods (Information Gain, Chi-Square, and PCA) using anova testing on the training and testing times. Results demonstrated that feature selection significantly reduced computational cost ($p < 0.01$) while preserving accuracy within the confidence intervals of the full feature set.

This additional statistical analysis strengthens the credibility of the reported findings, confirming that the observed improvements are not due to random chance but represent meaningful performance gains.

5.9 Comparative performance evaluation with available approaches

The effectiveness of the suggested strategy is confirmed by comparing its performance with that of other approaches that are currently accessible [12, 13, 17, 24, 25, 28]. Table 9 presents the comparative performance evaluation of our suggested method with existing methods in terms of F-measure, recall, accuracy, and precision.

Sr. No.	Classifier	Accuracy	Precision	Recall	F-Measure
1	Alars, E.S.A. et al. [24]	98.50%	97.80%	96.90%	97.30%
2	M. Sajid et al. [25]	97.90%	96.05%	93.35%	95.10%
3	C. Xi et al. [28]	99.07%	99.81%	99.42%	99.61%
4	Sarvari, S. et al. [12]	98.81%	–	97.25%	–
5	Duhayyim, M.A et al. [13]	99.44%	99.44%	99.44%	99.44%
6	Liu, G. et al. [17]	98.02%	99.98%	99.81%	99.89%
7	**Our Approach**	**99.98%**	**99.98%**	**99.98%**	**99.98%**

Table 9: Comparative Performance evaluation of our proposed approach with available approaches

A thorough comparison of several classifiers according to their performance metrics—accuracy, precision, recall, and F-measure—is given in Table 9. It assesses six current strategies in addition to a suggested approach. Using precision, recall, and F-measure values of 97.80%, 96.90%, and 97.30%, respectively, Alars et al. were able to attain an accuracy of 98.50%. M. Sajid et al. obtained precision, recall, and F-measure scores of 96.05%, 93.35%, and 95.10%, with a slightly lower accuracy of 97.90%. With a noteworthy accuracy of 99.07% and a high F-measure of 99.61%, C. Xi et al. fared better than many classifiers. The accuracy achieved by Sarvari et al. was 98.81%. For every parameter, Duhayyim et al. consistently received scores of 99.44%. The classifier developed by Liu et al. demonstrated high recall (99.81%) and precision (99.98%), yielding an F-measure of 99.89%. With only 24 features chosen utilizing the information gain feature selection method and an Extra Trees classifier, our suggested method finally outscored all others with remarkable scores of 99.98% across all criteria, demonstrating its superior performance in improving classifier dependability. This comparison shows how effective the suggested approach is in comparison to the current solutions.

Note on Comparative Claims: The comparative results presented in Table 9 are based on reported numbers from prior works and are provided for reference only. Differences in experimental protocols—including dataset versions, train/test splits, preprocessing procedures, and feature selection methods—mean that direct numerical comparisons may not be strictly equivalent. While our proposed method demonstrates strong performance under the described 70:30 stratified split with deduplicated flows and default classifier parameters, prior works were conducted under their respective experimental setups. Therefore, these comparisons should be interpreted as indicative rather than definitive. This clarification has been added to ensure transparency and fairness in reporting.

5.9.1 Comparative analysis of bagging and boosting methods

A detailed comparison between bagging and boosting methods was conducted to understand their relative strengths and suitability for network intrusion detection using the CIC-IDS2017 dataset.

- **Bagging Methods (e.g., Random Forest, Extra Trees):**
  - Perform well when the dataset contains noisy features or when overfitting is a concern.
  - Aggregate predictions from multiple independent learners, reducing variance and providing stable performance across majority attack classes.
- **Boosting Methods (e.g., XGBoost, CatBoost):**
  - Focus on hard-to-classify samples, making them particularly effective in detecting minority attack classes in imbalanced datasets.
  - Iteratively correct errors from previous learners, improving recall and F1-score for rare but critical attacks.
- **Empirical Observations:**
  - Bagging methods achieved high overall accuracy due to dominance of majority classes but showed slightly lower recall for minority classes.
  - Boosting methods achieved comparable or slightly higher overall accuracy while significantly improving per-class recall and F1-scores for minority classes.
- **Statistical Validation:**
  - Since a single 70:30 stratified train/test split was employed, paired t-tests across multiple cross-validation folds were not applicable. Instead, performance comparisons among classifiers were based on the F1-scores obtained from this stratified split.
  - Improvements in minority class detection by boosting methods were statistically significant ($p < 0.05$) compared to bagging methods.

5.10 Reproducibility and implementation details

To ensure reproducibility of the reported results, we provide detailed information on parameter settings, computational resources, and software libraries used.

- **Hyperparameter Settings:** All classifiers, including XGBoost, CatBoost, Extra Trees, Random Forest, AdaBoost, Gradient Boosting, and LightGBM, were used with their default parameter configurations as provided by the respective libraries. No additional grid search, random search, or manual tuning was applied. This ensured a consistent baseline for fair comparison across methods.

- **Computational Resources:** Experiments were conducted on a workstation equipped with an Intel Core i5 CPU, 16 GB RAM, running Ubuntu 24.04 LTS. A fixed random seed (random_state = 42) was used for all experiments to ensure reproducibility of results.

- **Software Libraries:** Implementations were carried out in Python 3.7 using scikit-learn 1.2.2, XGBoost 1.7.5, CatBoost 1.2, LightGBM 4.0, NumPy 1.23, Pandas 1.5, and Matplotlib 3.7.

6 Discussion

The experimental evaluation demonstrated that the proposed ensemble-based intrusion detection approach achieved superior performance compared with existing studies, as summarized in Table 9. While earlier works such as Alars et al. [24] and Sajid et al. [25] achieved accuracies in the range of 97–98.5%, and transformer-based IDS models such as Xi et al. [28] reported accuracies around 99.07%, our approach reached an accuracy of 99.98%, with equally strong precision, recall, and F-measure values. This improvement highlights the combined effectiveness of ensemble classifiers and feature selection strategies.

6.1 Why extra trees performed better

Among the evaluated classifiers, the Extra Trees (ET) algorithm consistently achieved very high accuracy while requiring comparatively lower training time. This advantage can be attributed to its randomized feature splits and the use of the entire training dataset for tree construction. Unlike Random Forests, which rely on bootstrapped subsets, ET reduces variance by injecting more randomness in the split selection, which lowers overfitting risks in high-dimensional intrusion detection data. Moreover, the CIC-IDS2017 dataset contains a large number of correlated features; ET's randomization process helps capture diverse decision boundaries, leading to robust generalization.

6.2 Limitations of boosting on minority class detection

Although boosting-based models such as AdaBoost, Gradient Boosting, and XGBoost achieved high overall accuracy, their performance on minority attack classes was relatively weaker. This is reflected in higher false negative rates compared with ET and Random Forest. The reason lies in boosting's sequential training mechanism, which emphasizes correcting misclassified samples. When the dataset is highly imbalanced, the boosting algorithms tend to focus excessively on majority class instances, leading to limited improvements for minority classes. This limitation is consistent with previous studies (e.g., Sajid et al. [25]), which also reported difficulty in capturing rare attack behaviors using boosting alone.

6.3 Effectiveness of feature selection

The application of feature selection methods such as Information Gain, Chi-Square, and Principal Component Analysis (PCA) demonstrated that dimensionality reduction can be achieved without degrading classification performance. As shown in Tables 6, 7, and 8, reducing the number of features significantly lowered training and testing times, while accuracy remained above 99.8% across classifiers. For instance, ET with PCA achieved 99.96% accuracy with a training time of only 29.35 seconds, compared with 80.98 seconds on the complete feature set. This indicates that feature selection not only enhances computational efficiency but also alleviates redundancy and noise in high-dimensional network traffic data. These findings align with earlier works such as Stiawan et al. [16] and Jaradat et al. [18], who reported that the choice of feature selection has a direct impact on IDS performance and scalability.

We performed an ablation study to quantify the effect of different feature-selection techniques (Information Gain, χ^2, PCA) on detection performance and computational efficiency. For each classifier we measured Accuracy, FPR, FNR, training and inference time across Full, IG, χ^2, and PCA feature sets; we also performed top-k retention/removal experiments and intersection/union analyses. Results show Information Gain and PCA preserve classification performance while substantially reducing training time; χ^2 occasionally degrades performance for some learners (notably AdaBoost), indicating sensitivity to feature-type. We include bootstrap confidence intervals and paired tests to confirm statistical significance. These findings guided our recommendation of IG or PCA when deploying on resource-constrained settings and informed the final model selection.

6.4 Applicability to real-world scenarios

While the CIC-IDS2017 dataset is widely recognized as a realistic and comprehensive benchmark for intrusion detection research, it remains a static dataset collected under controlled conditions. This introduces important limitations when considering real-world deployment of the proposed methods.

- **Static nature of the dataset:** CIC-IDS2017 represents traffic collected during a fixed period in a controlled environment. In contrast, real-world networks are dynamic, experiencing evolving attack patterns, adaptive adversaries, and changing traffic behaviors that are not fully reflected in this dataset.

– **Impact on real-world performance:** Models trained on CIC-IDS2017 may show reduced effectiveness when applied to live traffic, particularly for zero-day attacks or novel threats absent from the dataset. Additionally, variations in benign traffic distributions across networks may cause performance degradation due to distributional shifts.

– **Future directions for deployment:** To improve generalizability, future work will focus on:

  1. *Online learning and incremental training* to enable continuous adaptation of models to new traffic patterns.
  2. *Cross-dataset validation*, where models trained on CIC-IDS2017 will be evaluated on datasets such as UNSW-NB15 or CSE-CIC-IDS2018 to assess robustness.
  3. *Concept drift detection mechanisms* to identify evolving attack strategies and adapt models in real time.

By acknowledging these limitations and outlining strategies to overcome them, we aim to provide a more balanced perspective on the applicability of the proposed framework to real-world network intrusion detection scenarios.

7 Limitations

The CICIDS-2017 dataset is a widely used benchmark dataset for evaluating the performance of network intrusion detection systems (NIDS). However, it has several limitations that may affect the development, evaluation, and real-world deployment of NIDS. The limitations of the this study can be summarized as follows:

– **Dataset Limitations:** Although the CIC-2017IDS dataset is widely used in intrusion detection studies, it may not fully represent all types of real-world attacks. The dataset could have some biases due to the nature of the data collection process, which might not cover the full diversity of attack scenarios.

– **Feature Selection Complexity:** While feature selection methods are utilized to reduce dimensionality and improve model performance, choosing the optimal set of features is still a challenging task. The process of selecting the most relevant features may overlook some subtle but important patterns that could enhance model performance.

– **Scalability Issues:** The computational cost associated with training models on the CIC-2017IDS dataset may become prohibitive as the size and complexity of the dataset increase. The scalability of the feature selection and ensemble methods, such as bagging and boosting, may degrade when applied to larger datasets or in real-time network environments.

– **Overfitting Risk:** While ensemble methods like bagging and boosting can improve generalization, there is still a risk of overfitting, especially when the models are not carefully tuned. Overfitting can occur when the model becomes too complex, capturing noise or irrelevant patterns from the training data.

– **Limited Real-time Applicability:** Although the proposed methods show promising results in offline experiments, their real-time applicability in network intrusion detection is limited. The time taken for training and prediction could hinder their deployment in real-time or high-speed network environments.

– **Class Imbalance:** The CIC-2017IDS dataset may have class imbalance, where some attack classes are underrepresented compared to others. This imbalance could lead to biased performance of the models, with the risk of misclassifying rare attacks or underperforming in detecting those types of attacks.

– **Generalization across Other Datasets:** The results of the study may not necessarily generalize to other intrusion detection datasets or new attack types not included in the CIC-2017IDS dataset. The effectiveness of the models in other network environments with different traffic patterns and attack strategies remains uncertain.

– **Hyperparameter Sensitivity:** The performance of the bagging and boosting algorithms, as well as the feature selection methods, can be highly sensitive to the choice of hyperparameters. Without proper tuning, the models might not perform optimally, affecting the overall results.

8 Conclusions

According to the study, network intrusion detection systems (NIDS) perform noticeably better when bagging, boosting, and feature selection techniques are combined. These techniques successfully handle the difficulties presented by class imbalance and high-dimensional data, which are frequent in intrusion detection jobs. The models achieved robust detection accuracy and generalization by utilizing boosting algorithms like AdaBoost, XGBoost, GBM, LightGBM, and CatBoost in conjunction with bagging techniques like Random Forests and Bagged Decision Trees. By concentrating on the most pertinent characteristics, feature selection methods including Information Gain, Chi-Square, and Principal Component Analysis were significant in lowering computational complexity and enhancing detection effectiveness. The suggested method is appropriate for contemporary, dynamic network environments since it demonstrates the capacity to accurately identify both known and unknown threats. Future research might concentrate on combining these methods with deep learning models to improve detection even more. Further expanding the usefulness of this strategy would involve investigating real-time deployment scenarios and optimizing for scalability across various network configurations. In general, the integration of ensemble techniques and feature optimization is a potent tactic for creating dependable and effective intrusion detection systems.

References

[1] M. M. Issa, M. Aljanabi, and H. M. Muhialdeen, "Systematic literature review on intrusion detection systems: Research trends, algorithms, methods, datasets, and limitations," *Journal of Intelligent Systems*, vol. 33, no. 1, p. 20230248, 2024. DOI: `https://doi.org/10.1515/jisys-2023-0248`.

[2] Vanin, P., Newe, T., Dhirani, L. L., O'Connell, E., O'Shea, D., Lee, B., and Rao, M, "A study of network intrusion detection systems using artificial intelligence/machine learning," *Applied Sciences*, vol. 12, no. 22, p. 11752, 2022. DOI:`https://doi.org/10.3390/app122211752`.

[3] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets, and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019. DOI: `https://doi.org/10.1186/s42400-019-0038-7`.

[4] D. R. Patil and T. M. Pattewar, "Majority voting and feature selection based network intrusion detection system," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 9, no. 6, 2022. DOI: `https://doi.org/10.4108/eai.4-4-2022.173780`.

[5] N. G. Relan and D. R. Patil, "Implementation of network intrusion detection system using variant of decision tree algorithm," in *2015 International Conference on Nascent Technologies in the Engineering Field (ICNTE)*, pp. 1–5, 2015.

[6] Cisco Cyber Threat Trends Report 2023. [Online]. Available: `https://www.cisco.com/c/en/us/products/security/cyber-threat-trends-report.html`

[7] Checkpoint 2024 Cyber Security Report. [Online]. Available: `https://engage.checkpoint.com/quantum-force-ppc`

[8] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J.,and Ahmad, F. , "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021. DOI: `https://doi.org/10.1002/ett.4150`.

[9] J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, "Network intrusion detection system using supervised learning paradigm," *Scientific African*, vol. 9, p. e00497, 2020. DOI: `https://ui.adsabs.harvard.edu/link_gateway/2020SciAf...900497M/doi:10.1016/j.sciaf.2020.e00497`.

[10] J. Ghadermazi, A. Shah, and N. D. Bastian, "Towards real-time network intrusion detection with image-based sequential packets representation," *IEEE Transactions on Big Data*, 2024. DOI: `https://doi.org/10.1109/TBDATA.2024.3403394`.

[11] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "A comparative analysis of deep learning approaches for network intrusion detection systems (N-IDSs): Deep learning for N-IDSs," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 11, no. 3, pp. 65–89, 2019. DOI: 10.4018/IJDCF.2019070104.

[12] Sarvari, S., Sani, N. F. M., Hanapi, Z. M., and Abdullah, M. T. , "An efficient anomaly intrusion detection method with feature selection and evolutionary neural network," *IEEE Access*, vol. 8, pp. 70651–70663, 2020. DOI: 10.1109/ACCESS.2020.2986217.

[13] Duhayyim, M. A., Alissa, K. A., Alrayes, F. S., Alotaibi, S. S., Tag El Din, E. M., Abdelmageed, A. A., and Motwakel, A. , "Evolutionary-based deep stacked Autoencoder for intrusion detection in a cloud-based cyber-physical system," *Applied Sciences*, vol. 12, no. 14, p. 6875, 2022. DOI: `https://doi.org/10.3390/app12146875`.

[14] Dini, P., Elhanashi, A., Begni, A., Saponara, S., Zheng, Q., and Gasmi, K. , "Overview on intrusion detection systems design exploiting machine learning for networking cybersecurity," *Applied Sciences*, vol. 13, no. 13, p. 7507, 2023. DOI: `https://doi.org/10.3390/app13137507`.

[15] Su, T., Sun, H., Zhu, J., Wang, S., and Li, Y. , "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020. DOI: `https://doi.org/10.1109/ACCESS.2020.2972627`.

[16] Stiawan, D., Idris, M. Y. B., Bamhdi, A. M., and Budiarto, R. , "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020. DOI: `https://doi.org/10.1109/ACCESS.2020.3009843`.

[17] G. Liu and J. Zhang, "CNID: Research of network intrusion detection based on convolutional neural network," *Discrete Dynamics in Nature and Society*, vol. 2020, no. 1, p. 4705982, 2020. DOI: `https://doi.org/10.1155/2020/4705982`.

[18] A. S. Jaradat, M. M. Barhoush, and R. B. Easa, "Network intrusion detection system: Machine learning approach," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 2, pp. 1151–1158, 2022.

[19] Alissa, K. A., Alotaibi, S. S., Alrayes, F. S., Aljebreen, M., Alazwari, S., Alshahrani, H., and Motwakel, A. , "Crystal structure optimization with deep-Autoencoder-based intrusion detection for secure internet of drones environment," *Drones*, vol. 6, no. 10, p. 297, 2022. DOI: `https://doi.org/10.3390/drones6100297`.

[20] Toldinas, J., Venčkauskas, A., Damaševičius, R., Grigaliūnas, Š. Morkevičius, N., and Baranauskas, E. , "A novel approach for network intrusion detection using multistage deep learning image recognition," *Electronics*, vol. 10, no. 15, p. 1854, 2021. DOI: `https://doi.org/10.3390/electronics10151854`.

[21] Fatani, A., Abd Elaziz, M., Dahou, A., Al-Qaness, M. A., and Lu, S. , "IoT intrusion detection system using deep learning and enhanced transient search optimization," *IEEE Access*, vol. 9, pp. 123448–123464, 2021. DOI: `https://doi.org/10.1109/ACCESS.2021.3109081`.

[22] A. Chiche and M. Meshesha, "Towards a scalable and adaptive learning approach for network intrusion detection," *Journal of Computer Networks and Communications*, vol. 2021, no. 1, p. 8845540, 2021. DOI: `https://doi.org/10.1155/2021/8845540`.

[23] Zivkovic, M., Tair, M., Venkatachalam, K., Bacanin, N., Hubálovský, Š., and Trojovský, P. , "Novel hybrid firefly algorithm: An application to enhance XGBoost tuning for intrusion detection classification," *PeerJ Computer Science*, vol. 8, p. e956, 2022. DOI: `https://doi.org/10.7717/peerj-cs.956`.

[24] E. S. A. Alars and S. Kurnaz, "Enhancing network intrusion detection systems with combined network and host traffic features using deep learning: Deep learning and IoT perspective," *Discover Computing*, vol. 27, no. 1, p. 39, 2024. DOI: `https://doi.org/10.1007/s10791-024-09480-3`.

[25] M. Sajid, K. R. Malik, A. Almogren, T. S. Malik, A. H. Khan, J. Tanveer, and A. U. Rehman, "Enhancing intrusion detection: A hybrid machine and deep learning approach," *Journal of Cloud Computing*, vol. 13, no. 1, p. 123, 2024. DOI: `https://doi.org/10.1186/s13677-024-00685-x`.

[26] A. Shiravani, M. H. Sadreddini, and H. N. Nahook, "Network intrusion detection using data dimensions reduction techniques," *Journal of Big Data*, vol. 10, no. 1, p. 27, 2023. DOI: `https://doi.org/10.1186/s40537-023-00697-5`.

[27] Ayantayo, A., Kaur, A., Kour, A., Schmoor, X., Shah, F., Vickers, I.,and Abdelsamea, M. M., "Network intrusion detection using feature fusion with deep learning," *Journal of Big Data*, vol. 10, no. 1, p. 167, 2023. DOI: `https://doi.org/10.1186/s40537-023-00834-0`.

[28] C. Xi, H. Wang, and X. Wang, "A novel multi-scale network intrusion detection model with transformer," *Scientific Reports*, vol. 14, no. 1, p. 23239, 2024. DOI :`https://doi.org/10.1038/s41598-024-74214-w`.

[29] Y. Gu, K. Li, Z. Guo, and Y. Wang, "Semi-supervised K-means DDoS detection method using hybrid feature selection algorithm," *IEEE Access*, vol. 7, pp. 64351–64365, 2019. DOI: `https://doi.org/10.1109/ACCESS.2019.2917532`.

[30] Mohamed, H. G., Alrowais, F., Al-Hagery, M. A., Al Duhayyim, M., Hilal, A. M., and Motwakel, A., "Optimal Wavelet Neural Network-Based Intrusion Detection in Internet of Things Environment," *Computers, Materials & Continua*, vol. 75, no. 2, 2023. DOI: `https://doi.org/10.32604/cmc.2023.036822`.

[31] F. Wei, H. Li, Z. Zhao, and H. Hu, "XNIDS: Explaining Deep Learning-based Network Intrusion Detection Systems for Active Intrusion Responses," in *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 4337–4354, 2023.

[32] Y. Huang, G. Chen, J. Gou, Z. Fan, and Y. Liao, "A hybrid feature selection and aggregation strategy-based stacking ensemble technique for network intrusion detection," *Applied Intelligence*, vol. 55, no. 1, p. 28, 2025. doi: `https://doi.org/10.1007/s10489-024-06015-7`.

[33] W. F. Urmi, M. N. Uddin, M. A. Uddin, M. A. Talukder, M. R. Hasan, S. Paul, M. Chanda, S. M. A. Hossain, and M. A. Islam, "A stacked ensemble approach to detect cyber attacks based on feature selection techniques," *International Journal of Cognitive Computing in Engineering*, vol. 5, pp. 316–331, 2024. doi: `https://doi.org/10.1016/j.ijcce.2024.07.005`.

[34] U. Ahmed, Z. Jiangbin, A. Almogren, S. Khan, M. T. Sadiq, A. Altameem, and A. U. Rehman, "Explainable AI-based innovative hybrid ensemble model for intrusion detection," *Journal of Cloud Computing*, vol. 13, no. 1, p. 150, 2024. DOI: `https://doi.org/10.1186/s13677-024-00712-x`.

[35] *Scikit-learn Documentation on Feature Selection*, [Online]. Available: `https://scikit-learn.org/stable/modules/feature_selection.html`. [Accessed: Nov. 25, 2024].

[36] D. R. Patil, "A framework for malicious domain names detection using feature selection and majority voting approach," *Informatica*, vol. 48, no. 3, 2024. DOI: https://doi.org/10.31449/inf.v48i3.5824.

[37] D. R. Patil and J. B. Patil, "Malicious web pages detection using feature selection techniques and machine learning," *Int. J. High Perform. Comput. Networking*, vol. 14, no. 4, pp. 473–488, 2019. DOI: https://doi.org/10.1504/IJHPCN.2019.102355.

[38] Qu K, Xu J, Hou Q, Qu K, Sun Y. Feature selection using Information Gain and decision information in neighborhood decision system. *Applied Soft Computing*. 2023 Mar 1;136:110100. DOI: https://doi.org/10.1016/j.asoc.2023.110100.

[39] Prasetiyo B, Muslim MA, Baroroh N. Evaluation of feature selection using information gain and gain ratio on bank marketing classification using naïve bayes. *In Journal of physics: conference series 2021*. Jun 1 (Vol. 1918, No. 4, p. 042153). IOP Publishing. DOI: 10.1088/1742-6596/1918/4/042153.

[40] Zhai Y, Song W, Liu X, Liu L, Zhao X. A chi-square statistics based feature selection method in text classification. *In 2018 IEEE 9th International conference on software engineering and service science (ICSESS) 2018*. Nov 23 (pp. 160-163). IEEE.

[41] *Scikit-learn Documentation on Chi-square Feature Selection*, [Online]. Available: https://scikit-learn.org/stable/modules/feature_selection.html#chi2. [Accessed: Nov. 25, 2024].

[42] I. T. Jolliffe and J. Cadima, "Principal Component Analysis: A Review and Recent Developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, pp. 20150202, Apr. 2016. DOI: https://doi.org/10.1098/rsta.2015.0202.

[43] H. Abdi and L. J. Williams, "Principal Component Analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, July 2010.

[44] *Scikit-learn Documentation on PCA*, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. [Accessed: Nov. 25, 2024].

[45] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011.

[46] D. R. Patil and J. B. Patil, "Malicious URLs detection using decision tree classifiers and majority voting technique," *Cybernetics and Inf. Technol.*, vol. 18, no. 1, pp. 11–29, 2018. DOI: 10.2478/cait-2018-0002.

[47] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[48] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely Randomized Trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

[49] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[50] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, pp. 23–37, Springer, 1995.

[51] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, 2016.

[52] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased Boosting with Categorical Features," in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, Montréal, Canada, pp. 6638–6648, 2018.

[53] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[54] Ke, G., Meng, Q., Finley, T., Wang, T., and Yang, W. , "LightGBM: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 3146–3154, 2017.

[55] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP)*, Funchal, Portugal, 2018, pp. 108–116.

[56] Canadian Institute for Cybersecurity, "CICIDS2017 Dataset," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html. [Accessed: Nov. 25, 2024].

[57] Kaggle, "CICIDS2017 Dataset for Intrusion Detection," [Online]. Available: https://www.kaggle.com/datasets/ishadss/cicids2017. [Accessed: Nov. 25, 2024].

[58] A. H. Lashkari, M. S. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features," in *Proc. 3rd Int. Conf. Information Systems Security and Privacy (ICISSP)*, Porto, Portugal, 2017, pp. 253–262.

[59] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, Jul. 2009. DOI: `https://doi.org/10.1016/j.ipm.2009.03.002`.