electronics

MDPI

Article

Addressing Class Imbalance in Intrusion Detection: A Comprehensive Evaluation of Machine Learning Approaches

Vaishnavi Shanmugam [1], Roozbeh Razavi-Far [1,*] and Ehsan Hallaji [2]

[1] Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada; v.s@unb.ca
[2] Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada; hallaji@uwindsor.ca
* Correspondence: roozbeh.razavi-far@unb.ca

Abstract: The ever-growing number of cyber attacks in today's digitally interconnected world requires highly efficient intrusion detection systems (IDSs), which accurately identify both frequent and rare network intrusions. One of the most important challenges in IDSs is the class imbalance problem of network traffic flow data, where benign traffic flow significantly outweighs attack instances. This directly affects the ability of machine learning models to identify minority class threats. This paper is intended to evaluate various machine learning algorithms under different levels of class imbalances, using resampling as a strategy for this problem. The paper will provide an experimental comparison by combining various algorithms for classification and class imbalance learning, assessing the performance through the F1-score and geometric mean (G-mean). The work will contribute to creating robust and adaptive IDS through the judicious integration of resampling with machine learning models, thus helping the domain of cybersecurity to become resilient.

Keywords: intrusion detection; machine learning; class imbalance; classification; network security

Academic Editors: Li Pan, Ning Liu and Conghui Zheng

Received: 10 November 2024
Revised: 20 December 2024
Accepted: 24 December 2024
Published: 27 December 2024

Citation: Shanmugam, V.; Razavi-Far, R.; Hallaji, E. Addressing Class Imbalance in Intrusion Detection: A Comprehensive Evaluation of Machine Learning Approaches. *Electronics* **2025**, *14*, 69. https://doi.org/10.3390/electronics14010069

1. Introduction

In today's technology-driven world, where industries heavily rely on computers and the Internet, securing information is a priority. One crucial aspect of information security is enhancing the ability to identify cyber threats effectively, which calls for the development of IDS. These systems are built to analyze network traffic data and raise an alarm whenever an intrusion is detected. However, a major hurdle is the imbalance between the volume of network traffic data and the relatively few instances of potential attacks. This imbalance also affects established benchmark datasets, making it challenging for the machine learning and deep learning algorithms used in IDS.

When dealing with imbalanced training data, models often excel in identifying traffic (i.e., majority class) but struggle with spotting instances of potential attacks (i.e., minority class). This problem becomes more pronounced in scenarios involving classes where rare attack types might be misclassified as other common attack types. In the realm of cyber security and network traffic analysis, pinpointing minority attack cases holds more importance than simply identifying regular data.

Researchers have put forward strategies such as resampling techniques to tackle performance issues stemming from imbalanced datasets. Various methods such as Random Under Sampling (RUS), Random Over Sampling (ROS), the Adaptive Synthetic Sampling Method (ADASYN) [1], and a combination of both Synthetic Minority Oversampling

Techniques (SMOTEs) [2] are used to balance the distribution of classes. They achieve this by increasing the instances of the minority class or reducing those of the majority class.

The rapid advancement in communication technologies such as 5G and the Internet of Things (IoT) has resulted in a surge in network traffic, leading to an increased risk of network breaches. IDSs are a component of organizational cybersecurity infrastructures. They monitor network activities and identify behavior that may indicate security breaches. IDSs can be classified into Network IDSs (NIDSs) and Host IDSs (HIDSs), with NIDSs focusing on analyzing data from the network.

IDSs play a role in establishing defense measures against potential threats to ensure the security, privacy, and availability of network resources. They are broadly categorized into signature-based detection, which detects known threat patterns, and anomaly-based detection, which identifies network behaviors as threats by establishing a baseline normal behavior pattern. However, one significant challenge faced in implementing IDS is dealing with imbalanced training datasets.

The evolving landscape of cybersecurity threats underscores the need for intelligent IDS that can outsmart potential attackers. While traditional signature-based IDS excel in spotting known threats, they struggle with emerging attacks. This challenge highlights the significance of anomaly-based detection methods, which can identify deviations from network behavior patterns. However, these approaches often face challenges, with a number of identifications making it even more complex to uphold a secure network environment. By utilizing machine learning and artificial intelligence, IDSs can enhance their ability to detect threats by learning from data to recognize patterns and anticipate future risks. This strategy does not only boost the precision of threat identification but also aids in minimizing the occurrence of false alarms that commonly affect anomaly-based systems.

In response to the changing cybersecurity landscape marked by emerging risks and vulnerabilities, IDSs must evolve by integrating detection techniques and drawing insights from previous security incidents. Machine learning and artificial intelligence serve as assets in bolstering IDSs' capabilities by empowering them to learn from information, identify patterns, and forecast potential cyberattacks. This research contributes to this progression through an evaluation of diverse machine learning methods in varying data imbalance scenarios.

This study focuses on overcoming class imbalance in IDS datasets and assessing the effectiveness of machine learning methods. The goal is to enhance intrusion detection systems' efficiency and resilience, offering insights for cybersecurity professionals and researchers. To do so, this study thoroughly evaluates the effectiveness of IDS techniques under various ratios of class imbalance (e.g., 1:10, 1:100, 1:500, and 1:1000). The machine learning methods investigated include Random Forest, Decision Tree, KNN, MLP, Naive Bayes, Weighted Decision Tree, Easy Ensemble, AdaBoostCost, Under OverBagging, and RUSBoost. Performance measures such as F1-score and G-mean are utilized to offer an evaluation of each technique's ability to detect intrusions. Beyond tackling the class imbalance issue, this study delves into how various resampling techniques impact the performance of IDS. By combining resampling approaches with machine learning models, this research aims to build resilient and dependable intrusion detection systems that are capable of effectively handling diverse and dynamic network environments.

The remainder of this paper is organized as follows. Section 2 reviews preliminaries and related works. Section 3 presents the problem statement and introduces the case study used in this study. Section 4 overviews the utilized methods and other relevant techniques for mitigating class imbalance. Section 5 explains the experimental setup and then reports and analyzes the experimental results. Finally, the paper is concluded in Section 6.

2. Background

This section presents the literature review and explains the preliminaries required for this study.

2.1. Intrusion Detection Systems

In the realm of network security, the IDS serves a function by pinpointing unauthorized or malicious activities and facilitating swift responses. As shown in Figure 1, an IDS can be classified based on factors such as data origin, system structure, and detection techniques.

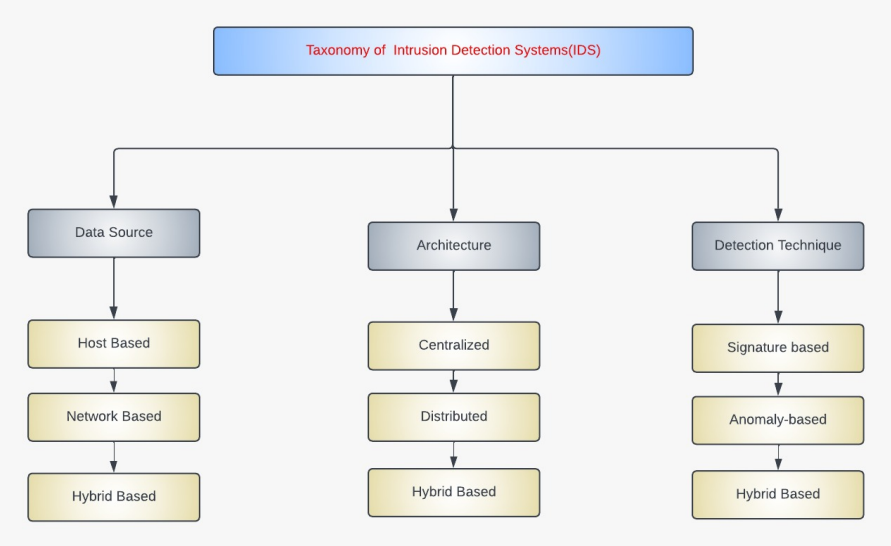

Figure 1. Taxonomy of different IDS categories.

Familiarity with these groups is vital for choosing and implementing a suitable solution to meet specific network security needs.

2.1.1. Data Source and Implementation

NIDSs are highly effective for identifying threats such as Distributed Denial of Service (DDoS) attacks, attempts by malicious actors to intercept communication, and scanning of network ports by analyzing patterns in data traffic. These systems are strategically positioned to provide coverage, although they may face challenges when dealing with encrypted data and generating a large volume of information that needs advanced processing capabilities. Despite these obstacles, NIDSs [3] play a role in monitoring large network environments in real-time.

Hybrid IDSs combine the features of both HIDSs and NIDSs [4] by merging insights from individual hosts with a comprehensive view [5] of network activities. This integrated approach boosts the detection rate by linking events across hosts and networks, making them effective against multi-stage cyber attacks. However, deploying and managing an IDS requires meticulous integration and coordination due to their complexity.

2.1.2. Architecture and Placement Strategy

The effectiveness of IDS is significantly influenced by their architecture and placement within the network. IDSs can be centralized, distributed, or hybrid in nature. Centralized IDSs consolidate monitoring data into a location for analysis, streamlining management tasks and offering a broad perspective on network security. Nevertheless, they may face scalability issues, but they serve as a potential single point of failure in larger networks. Centralized systems are particularly suitable for environments requiring data correlation to detect advanced threats. Distributed IDSs operate on independent systems spread

across different network segments. Each system monitors traffic and alerts about any suspicious activities. This decentralized approach improves scalability and eliminates the risk of a single point of failure found in centralized systems. Distributed IDSs work well in segmented networks, providing local insights and quicker detection, though they require more intricate management and coordination.

Hybrid IDS architectures strike a balance between distributed systems by combining their advantages. In this setup, detection tasks are divided among nodes, while a central system handles data aggregation and analysis. This configuration offers scalability along with analysis, making it effective against complex threats. Hybrid architectures are suitable for environments with security needs due to their adaptable deployment options.

2.1.3. Detection Methods

Signature-based detection relies on comparing data with a database of known threat patterns to identify potential security risks. This approach is particularly good at recognizing attacks [6]. Popular tools such as SNORT and Suricata, which use pattern matching algorithms, are favored for their accuracy and low false alarm rates. However, while signature-based methods excel at spotting known threats, they may struggle with detecting emerging dangers, underscoring the importance of regularly updating the signature database.

Hybrid detection techniques blend the advantages of both anomaly-based and signature-based strategies. These systems pinpoint recognized threats through signature matching while also uncovering unfamiliar attacks through anomaly analysis, providing a comprehensive shield against cyber threats [7]. Albeit effective, these solutions demand careful integration and continuous oversight to maximize their effectiveness.

2.2. Data-Driven IDS

The advancement of technology has led to a growing dependence on data-driven and AI-based techniques for analyzing datasets and identifying patterns that could signal potential cyber threats. AI-based IDSs can be categorized into several groups, as depicted in Figure 2. Each of these approaches contributes uniquely to improving the effectiveness and efficiency of IDSs.

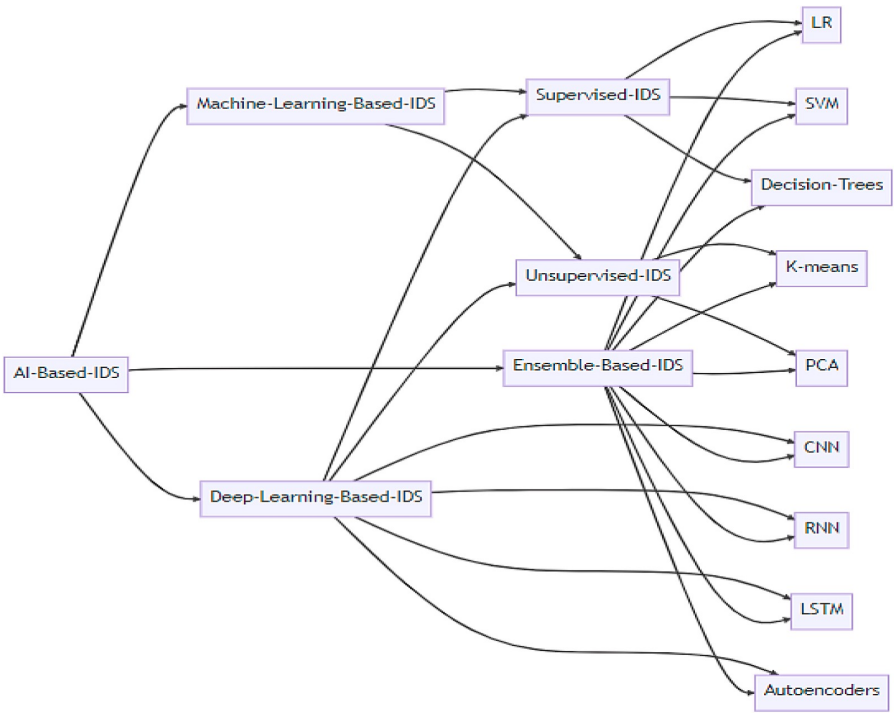

Figure 2. Categorization of AI-Based IDSs [8].

2.2.1. Unsupervised Learning

Unsupervised learning is used to automatically find patterns and associations in data without standalone labeling. In such methods, data points of a similar nature are clustered together so that the outliers in the system can be identified. This approach is particularly suitable for unknown threat detection. For instance, K-means clustering is used to analyze data based on its similarity. By doing so, individual anomalies may not fit into any cluster and be flagged as outliers. This ability becomes essential when dealing with new threats that do not exist in the training data distribution. Neural network-based unsupervised learners such as Autoencoders can also be used for anomaly detection by compressing the input representation into a lower-dimensional space and reconstructing it. The reconstruction error is analyzed to identify deviations from normal patterns of logs, indicating potential intrusions [9]. Despite the potency of unsupervised learning, it poses several challenges such as identifying the correct number of clusters and handling high-dimensional data. Regardless, it enables the IDS to detect threats that have not been observed before and so is a crucial part of a modern IDS.

2.2.2. Supervised Learning

On the other hand, supervised learning relies on labeled datasets to train models for recognizing the correlation between input features and their corresponding outputs. This approach effectively identifies attack patterns similar to the prior knowledge at the time of the training. Approaches such as Support Vector Machines (SVMs), Decision Trees, and Neural Networks are examples of this category that are commonly used in IDS.

2.2.3. Deep Learning

Deep learning uses neural networks with a large number of layers to learn features automatically and recognize complicated patterns from a huge volume of data. For instance, CNNs are appropriate for image and spatial data, while RNNs are applied in various sequential data analyses [10]. Particularly, CNNs are effective in image-based analysis and can accurately find the spatial features in images or other grid-structured data, such as network traffic. That is why modern IDSs are able to find many patterns and anomalies which could be hidden by the use of traditional approaches. On the other hand, compared with RNNs, the capability of CNNs in catching temporal variations is limited because CNNs rely on the snapshot representation of log or flow states without any time-series information. As a result, their anomaly detection is achieved for isolated time frames only, though many real-world problems rely on finding patterns across more than one consecutive timestamp [11]. However, despite the advantages of deep learning models, they are limited in that they require very large-sized training datasets and computational resources.

2.2.4. Semi-Supervised Learning

Semi-supervised learning generally uses a small amount of labeled data with a large pool of unlabeled data, which comes in handy when labeled data are not enough compared with enormous unlabeled data [12]. Using both types of data, models derived from semi-supervised learning can achieve higher accuracy and robustness [13]. The other major advantage of this approach is the improvement in model accuracy and generalization without essentially requiring large labeled datasets, which makes it promising for practice. This is very helpful in intrusion detection, since labeled data are difficult to obtain. Semi-supervised algorithms within this domain can generally detect both known and unknown threats effectively by leveraging labeled data and augmenting it with a smaller amount of unlabeled data.

2.2.5. Hybrid Learning

The hybrid model utilizes a combination of various learning techniques to hone its detection edge. For instance, a combination of supervised, unsupervised, and various categories of neural networks can increase robustness and accuracy in intrusion detections [7]. Hybrid models employ the power of each technique, creating a holistic solution for discovering both known and unknown threats. The high accuracy of supervised methods and the versatility of unsupervised models are embodied in the hybrid to provide a win-win scenario that combines both types. One standard approach is to use supervised learning (e.g., classify known attacks) in conjunction with unsupervised learning (e.g., detection of novel threats, etc.).

3. Problem Statement

One of the prevalent challenges in NIDS is related to the class imbalance problem that is targeted in this work. Addressing the class imbalance is crucial in network intrusion detection, as real-world datasets tend to follow certain patterns; the vast majority of network traffic is routine, with only a small fraction representing attack-related incidents [14]. This imbalance skews model predictions, making it difficult to detect rare but critical types of attacks, such as user-to-root (U2R) and remote-to-local (R2L) intrusions. With high ratios of normal-to-attack samples, this issue demands attention to restore balance and improve detection capabilities.

Class imbalance often increases false negatives, where attack instances are mistakenly classified as normal traffic patterns [15]. Models trained on imbalanced data tend to overfit to the majority class, optimizing for overall accuracy while failing to detect the rare attack cases. To mitigate this, it is essential to address the bias towards the majority class and accurately represent the minority class, ensuring that the model is equipped to identify these uncommon but significant threats.

Most conventional methods are designed to consider an equal number of class samples in the training data. Thus, the IDSs that do not use class imbalance learning must be handled with extreme care. This is because these models fail to detect minority-class attacks efficiently. To address this, class imbalance learning techniques such as SMOTE and ADASYN are usually used to artificially extend the available data. Ensemble and hybrid models go further and enhance the performance of the skewed datasets. However, due to evolving cyber threats, for effective performance, an IDS model should possess the ability to resample and make cost-sensitive decisions to counteract new and emerging threats [16,17].

Case Study

This study simulates four scenarios by inducing four different ratios of class imbalance in the CIC-DDoS2019 dataset. The considered imbalance ratios are 1:10, 1:100, 1:500, and 1:1000, indicating the existing number of major samples in proportion to one minority sample. In this dataset, the minority class contains benign samples, whereas the major class includes adversarial samples (Dr_DOS_DNS). The datasets consist of features each representing an attribute of the data points. Features in this dataset are listed and described in Table 1.

The CIC-DDoS2019 dataset has been specifically designed to be representative of real-world network traffic distributions, with severe class imbalance, comprising more than 50 million DDoS attack records and fewer than 57,000 benign instances. The dataset includes timestamps for all network flow records; thus, temporal correlations remain intact, which is vital for studying attack patterns and developing models that are sensitive to sequence and timing.

Table 1. Attributes of the network flow that are used in the CIC-DDoS2019 dataset.

Field	Description
Flow ID	Identifier for network flows, helping with tracking and analyzing sessions.
Source IP and Destination IP	Source and destination IP addresses for routing the network traffic.
Source Port and Destination Port	These denote the ports utilized by the source and destination.
Protocol	Indicates the network protocol used in a flow, such as TCP, UDP, or ICMP.
Timestamp	The time when the flow is observed.
Flow Duration	The overall duration of the flow, which can reveal the nature of communication.
Total Fwd Packets and Total Backward Packets	The count of sent packets in each direction offering insights into traffic volume.
Total Length of Fwd Packets and Total Length of Bwd Packets	The size of packets in the backward directions aiding in understanding data volume and potential content.
Fwd Packet Length Max/Min/Mean/Std and Bwd Packet Length Max/Min/Mean/Std	These metrics give insights into sizes in both directions assisting in detecting anomalies.
Flow Bytes/s and Flow Packets/s	The rate at which bytes and packets are transmitted per second, indicating usage and flow intensity.
Flow IAT Mean/Std/Max/Min	The average, deviation, maximum, and minimum inter-arrival times of the flow indicate its burstiness and regularity.
Fwd IAT Total/Mean/Std/Max/Min and Bwd IAT Total/Mean/Std/Max/Min	Similar statistics for inter-arrival times in both forward and backward directions.
Fwd PSH Flags and Bwd PSH Flags	Determines whether data should be pushed immediately.
Fwd URG Flags and Bwd URG Flags	The TCP packets with flags indicate the data of utmost importance.
Fwd Header Length and Bwd Header Length	Understanding the length of headers in both forward and backward packets is essential for grasping the intricacies of overhead and protocol details.
Fwd Packets/s and Bwd Packets/s	The rate at which packets are transmitted in both directions.
Min Packet Length and Max Packet Length	The recorded packet lengths in the flow vary from the smallest to the largest observed lengths.
Packet Length Mean/Std/Variance	Statistical measures of packet length, providing insights into the consistency and variability of packet sizes.
FIN Flag Count, SYN Flag Count, RST Flag Count, PSH Flag Count, ACK Flag Count, URG Flag Count, CWE Flag Count, ECE Flag Count	Counts of various TCP flags, which are crucial for understanding the control mechanisms of the TCP connections.
Down/Up Ratio	The ratio of download to upload, indicating the balance of traffic direction.
Average Packet Size	The average size of packets in the flow.
Avg Fwd Segment Size and Avg Bwd Segment Size	The average size of segments in forward and backward directions.
Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Bwd Avg Bulk Rate	Bulk statistics for forward and backward directions, indicating bulk data transfer patterns.
Subflow Fwd Packets and Subflow Fwd Bytes, Subflow Bwd Packets and Subflow Bwd Bytes	Subflow statistics, which can help in breaking down larger flows into manageable parts for detailed analysis.
Init_Win_bytes_forward and Init_Win_bytes_backward	Initial window sizes for forward and backward directions, which are important for understanding the flow control mechanisms.
act_data_pkt_fwd and min_seg_size_forward	Active data packets in the forward direction and minimum segment size.
Active Mean/Std/Max/Min and Idle Mean/Std/Max/Min	Active and idle times for the flow, providing insights into the flow's activity patterns.
SimillarHTTP	Indicates similarity to HTTP traffic, which is useful for identifying web-based traffic.
Inbound	Indicates whether the traffic is inbound.
Label	The class label (e.g., BENIGN or DrDoS_DNS).

4. Class Imbalance Learning

Class imbalance learning (CIL) bears the responsibility of handling the classification scenarios that engage overwhelming instances of one class compared to the rest. CIL taxonomy includes techniques on three levels: the data, algorithm, and hybrid level [18,19]. Data-level procedures seek to create a balance in the dataset before the model is trained by increasing the instance of the minority class or decreasing the incidence of the majority class [20]. Algorithmic strategies reconfigure the learning strategies in such a way that they are less biased towards the majority category. The hybrid strategy imitates all the strategies by utilizing the merits of the two strategies.

4.1. Algorithm-Level

Algorithm-level techniques modify existing algorithms or the process of handling imbalanced data. These methods incorporate the class imbalance directly into the model training process. One of the ways to achieve this is through cost-sensitive approaches whereby there are distinct misclassification costs for the minority and majority classes. This ensures that the process of learning corrects more on the minority class errors, thus making the model to be more sensitive to instances of the minority class. Weighted decision trees and extended cost-sensitive methods, such as AdaCost [21] among others, also excel in this category [22].

Ensemble methods, such as RUSBoost [23] and SMOTEBoost [24], combine multiple weak classifiers for improved prediction accuracy [25]. Algorithms like EasyEnsemble and BalanceCascade target previously misclassified or hard-to-classify samples as part of model training to boost the overall detection accuracy [26]. Other interesting approaches that combine cost-sensitive learning with ensemble strategies are EasyEnsemble and BalanceCascade, which also have the ability to improve the identification of minority class instances. These techniques find a good trade-off between the sensitivity of the minority class and overall classification accuracy. The introduction of deep learning models such as generative adversarial networks has also contributed to enhancing the perceptive power of such unpopular classes [27].

4.2. Data Level

To remedy the issue of class imbalance, there are methods of data pre-processing that attempt to balance the dataset. Some of these tools such as SMOTE, ADASYN, and ROS are schemes for sampling where the synthetic observations are prepared for the underrepresented class [28]. Another approach is RUS [29], which excludes samples from the overrepresented set at random to create balance in data [30]. These methods may also be integrated into more hybrid ones, such as SMOTEENN (SMOTE with Edited Nearest Neighbors) and SMOTETomek, which not only balance the classes but also clean the dataset of noisy samples [31]. Class imbalance in IDS is mostly addressed using data-level techniques [32]. However, using these methods may result in introducing noise to the data.

The SMOTE family of methods, or Borderline-SMOTE and Safe-Level-SMOTE, have been derived to resolve the challenges of class imbalance. Borderline-SMOTE deals with samples that are located around the decision zone, whereas Safe-Level-SMOTE deals with the target by sampling at appropriate levels, and Safe-Level-SMOTE deals with the target by sampling at appropriate levels. Another approach is offered by ADASYN, which increases the rate of generating extra samples in the area where the ratio of the minority class is low. This rate increase is aimed at improving classifier performance on imbalanced datasets, as it tends to focus on more difficult-to-classify areas.

4.3. Cost-Sensitive

Cost-sensitive learning is a crucial aspect of handling class imbalance at the algorithm level. It concerns the issue of classifying the instances of different classes correctly by putting priority or cost on false positives and false negatives. The method employs the use of a cost matrix, which states how costly information misclassification is so that the system will direct its learning process to concentrating on reducing the errors of the high cost. Common cost-sensitive techniques include MetaCost, AdaCost, AdaC1, and AdaC2. These techniques seek to minimize the decision threshold from the classifiers given a cost matrix and include cost-sensitive SVMs and decision trees [33]. Cost-sensitive boosting approaches such as AdaC2 and CSB2 have also been introduced, where the direct boosting procedure is modified to allow for cost embedding in the boosting process. These methods make the classifier more focused on the minority class and therefore help the classifier in the successful classification of minority samples, even when the dataset is very skewed.

4.4. Hybrid

The exchange of the cost-sensitive and data sampling strategy in a hybrid approach aims to utilize the assets and weaknesses of each minimization technique. These methods generally involve initial treatment of the data to balance the classes and then apply a cost-sensitive or adaptive strategy to the altered dataset. Hybrid approaches can provide a more robust solution to class imbalance by addressing the issue from multiple angles [34]. Techniques such as SMOTEBoost, RUSBoost, and EasyEnsemble leverage both re-sampling and boosting to enhance the detection capabilities of classifiers.

Recent advancements in hybrid methods include the integration of ensemble techniques with re-sampling methods to create more robust models. For instance, methods similar to RUSBoost combine RUS with boosting to enhance the model's performance on imbalanced datasets. These approaches leverage the strengths of multiple techniques, providing a more comprehensive solution to the challenges posed by class imbalance.

5. Experimental Results

This section first explains the experimental setup and then proceeds to evaluating and analyzing the obtained results.

5.1. Experimental Setting

The dataset utilized in this study comprises various features representing different aspects of network traffic. Each column in the dataset holds specific information that contributes to the comprehensive analysis of network behavior.

5.1.1. Scenarios

Various ratios of imbalances are investigated to gain insight into the topic. Different degrees of imbalance can influence how well the model performs in scenarios. The ratios examined in this research are 1:10, 1:100, 1:1500, and 1:1000. Table 2 reports the class population for cross-validation splits over different scenarios.

Table 2. Population of samples in the cross-validation splits.

Ratio	Benign Samples	Dr_DoS_DNS	Total Samples	Training (90%)	Testing (10%)
1:10	1644	16,443	18,087	16,278	1809
1:100	3222	328,691	331,913	298,721	33,192
1:500	3223	1,634,044	1,637,267	1,473,540	163,727
1:1000	3223	3,246,951	3,250,174	2,925,157	325,017

1. 1:10 Ratio: In this scenario, for every 1 instance of the minority class, there are 10 instances of the majority class. The dataset has 1644 BENIGN samples and 16,443 DrDoS_DNS samples.
2. 1:100 Ratio: At a 1 to 100 ratio, for every 1 instance of the minority class, there are 100 instances of the majority class. The dataset has 3222 BENIGN samples and 328,691 DrDoS_DNS samples.
3. 1:500 Ratio: With a 1 to 500 ratio, the imbalance is quite high. The dataset has 3223 BENIGN samples and 1,634,044 DrDoS_DNS samples.
4. 1:1000 Ratio: This ratio is extremely imbalanced, where for every 1 instance of the minority class, there are 1000 instances of the majority class. The dataset has 3223 BENIGN samples and 3,246,951 DrDoS_DNS samples.

5.1.2. Data Processing

The data processing phase involves several crucial steps to prepare the data for machine learning model training. Initially, redundant features such as 'Unnamed: 0', 'Flow ID', 'Source IP', 'Destination IP', and 'Timestamp' are removed to reduce noise in the dataset. Constant features, which provide unuseful information for the model, are also dropped. The dataset is then cleaned by handling missing values and removing duplicate entries.

Next, non-numeric columns are encoded using one-hot encoding, specifically for the 'SimillarHTTP' column. This transformation converts categorical variables into a format suitable for machine learning algorithms. Infinite values are replaced with NaNs, and subsequently, these NaNs are dropped to ensure data integrity.

The dataset is then split into features (X) and the target variable (y). The target labels are encoded using `LabelEncoder` to convert them into numeric form. `StandardScaler` is applied to scale the features, ensuring that each feature contributes equally to the model. Feature selection is performed using `SelectKBest` with the ANOVA F-test (`f_classif`) to select the top 20 most important features, which helps in improving model performance and reducing overfitting.

5.1.3. Model Training

The model training phase involves initializing various classifiers, including Random Forest, KNN, Decision Tree, Naive Bayes, and MLP. Each classifier is evaluated with different resampling techniques to handle class imbalances effectively. The utilized resampling techniques are RUS, ROS, SMOTE, and ADASYN.

A 10-fold stratified cross-validation is employed to ensure robust and unbiased model evaluation. The classifiers are trained and evaluated within each fold of the cross-validation. Hyperparameter tuning is performed for certain models using GridSearchCV and RandomizedSearchCV to find the best parameters. The obtained parameter setting that leads to optimal performance is reported in Table 3.

Table 3. Parameter setting used for the employed algorithms. #. indicates the quantity.

Algorithm	Parameter Setting
AdaBoostCost	#. estimators = 173, learning rate = 0.8387, algorithm = SAMME
Weighted Decision Tree	Class weight = [{0: 100, 1: 1}, {0: 10, 1: 1}, {0: 1, 1: 1}, {0: 1, 1: 10}, {0: 1, 1: 100}]
EasyEnsemble	#. estimators = (50, 500), sampling strategy = ['auto', 'not minority', 'all']
RUSBoost	#. estimators = 50, learning rate = 0.1, base estimator = decision tree

Table 3. *Cont.*

Algorithm	Parameter Setting
UnderOverBagging	Estimator = decision tree, sampling strategy = 'auto', replacement = False
SMOTEBoost	#. estimators = 87, learning rate = 0.4262, estimator = decision tree
Random Forest	#. estimators = 100, resampling methods = rus, ros, smote, adasyn
KNN	$k = 3$, Resampling methods = rus, ros, smote, adasyn
Decision Tree	Resampling methods = rus, ros, smote, adasyn
Naive Bayes	Resampling methods = rus, ros, smote, adasyn
MLP	Max iter = 300, resampling methods = rus, ros, smote, adasyn

Specialized classifiers, namely AdaBoostCost, Weighted Decision Tree, RUSBoost, Under Over Bagging, EasyEnsemble, and SMOTEBoost, are also trained and evaluated. These classifiers are specifically designed to handle class imbalances and improve the performance of the model on imbalanced datasets.

5.1.4. Evaluation

The evaluation phase involves calculating various performance metrics to assess the models' effectiveness. Metrics such as F1-score and G-mean are calculated for each model. Confusion matrices are also generated to provide insights into the models' classification performance.

Results from the cross-validation folds are aggregated and analyzed. Class-specific results are printed to show the performance of each classifier on individual classes. This detailed evaluation helps in understanding how well each model performs and which resampling techniques are most effective.

Visualization techniques, such as box plots, are used to compare the performance of different models and resampling methods. These visualizations provide a clear comparison of the models' performance across various metrics, helping in identifying the best-performing models and techniques.

5.2. Results' Analysis

At a 1:10 imbalance ratio, most techniques achieve high F1-scores with minimal variation. Techniques such as Random Forest+SMOTE and Random Forest (ADASYN) perform exceptionally well, both achieving perfect F1-scores of 1.000 ± 0.0000. Decision Tree (ADASYN) and KNN (ROS) also score close to 1.000, showcasing their effectiveness in handling moderate imbalance. The box plot in Figure 3 illustrates tight clusters, indicating consistent performance across these methods. The results are also detailed in Table 4 in terms of F1-score.

In terms of G-Mean, SMOTEBoost, Under Over Bagging, and EasyEnsemble lead with near-perfect scores, as shown in Table 5. The minimal spread in the box plot (Figure 3) further underscores their stability in managing a 1:10 class imbalance.

As the imbalance increases to 1:100, F1-scores remain high, but variability starts to appear. Random Forest (SMOTE) and Decision Tree (ADASYN) continue to perform well, though methods such as Naive Bayes (ROS) begin to show a slight performance decline. Despite this, MLP (ROS) and EasyEnsemble maintain robustness, with only minor drops in F1-scores, as shown in Figure 4.

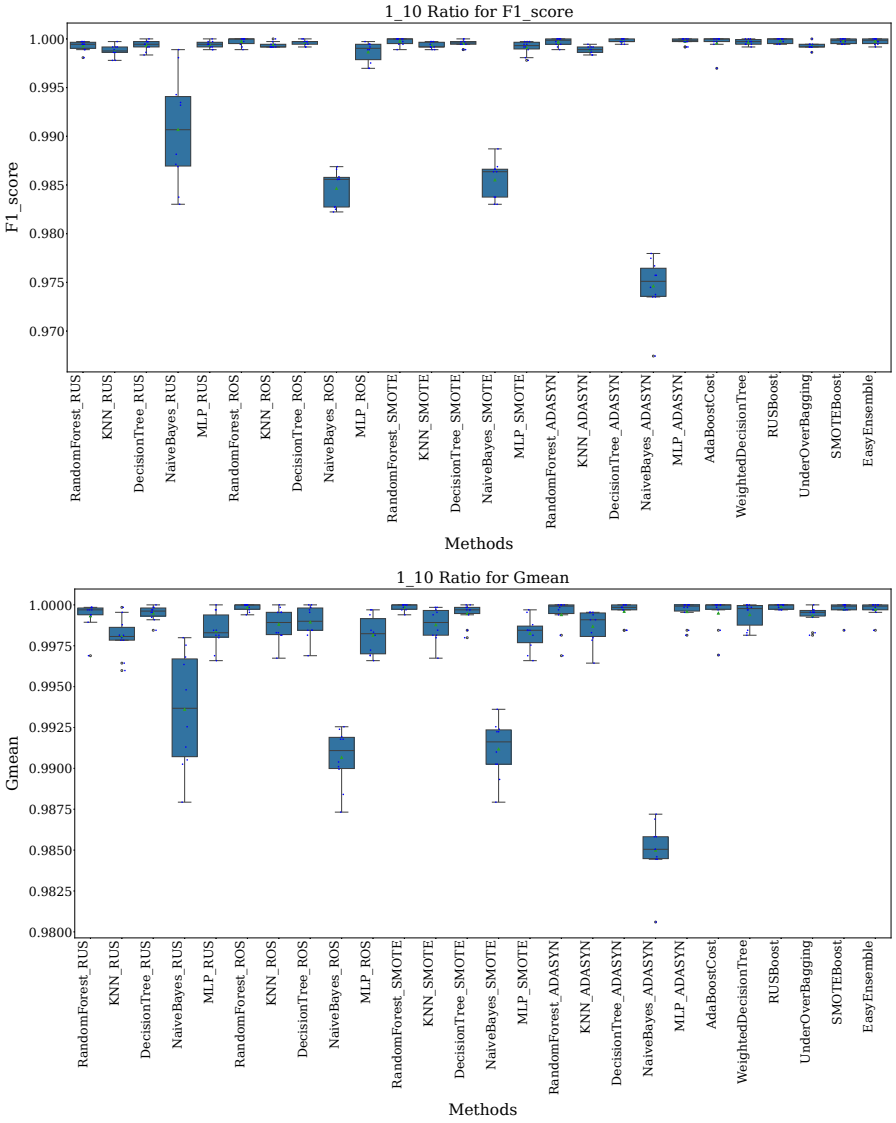

Figure 3. Recorded F1-score and G-mean for all methods with a 1:10 imbalance ratio.

Table 4. Ranked F1-Score for different techniques at 1:10, 1:100, 1:500, and 1:1000 imbalance ratios.

Rank	Technique	1:10 Ratio	1:100 Ratio	1:500 Ratio	1:1000 Ratio
1	Random Forest (SMOTE)	1.0000 ± 0.0000 (1)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
1	Random Forest (ADASYN)	1.0000 ± 0.0000 (1)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
1	Decision Tree (ADASYN)	1.0000 ± 0.0000 (1)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
2	KNN (ROS)	0.9999 ± 0.0001 (2)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
2	KNN (ROS)	0.9999 ± 0.0001 (2)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
2	MLP (RUS)	0.9999 ± 0.0001 (2)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
2	WeightedDecision Tree	1.0000 ± 0.0000 (1)	0.9997 ± 0.0005 (2)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
3	MLP (ROS)	0.9999 ± 0.0000 (3)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
3	MLP (ADASYN)	0.9999 ± 0.0000 (3)	0.9997 ± 0.0004 (1)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
4	Decision Tree (SMOTE)	1.0000 ± 0.0000 (1)	0.9997 ± 0.0004 (1)	0.9999 ± 0.0000 (2)	0.9998 ± 0.0001 (3)
5	MLP (SMOTE)	0.9999 ± 0.0000 (3)	0.9996 ± 0.0004 (3)	1.0000 ± 0.0000 (1)	1.0000 ± 0.0000 (1)
6	Decision Tree (ROS)	0.9999 ± 0.0000 (3)	0.9997 ± 0.0004 (1)	0.9999 ± 0.0000 (2)	0.9998 ± 0.0001 (3)

Table 4. *Cont.*

Rank	Technique	1:10 Ratio	1:100 Ratio	1:500 Ratio	1:1000 Ratio
6	Random Forest (ROS)	0.9999 ± 0.0000 (3)	0.9996 ± 0.0004 (3)	0.9999 ± 0.0000 (2)	1.0000 ± 0.0000 (1)
7	KNN (SMOTE)	0.9999 ± 0.0001 (2)	0.9996 ± 0.0004 (3)	0.9999 ± 0.0000 (2)	0.9997 ± 0.0002 (4)
7	KNN (RUS)	0.9999 ± 0.0001 (2)	0.9997 ± 0.0004 (1)	0.9992 ± 0.0006 (6)	0.9999 ± 0.0002 (2)
7	KNN (ADASYN)	0.9999 ± 0.0001 (2)	0.9996 ± 0.0004 (3)	0.9996 ± 0.0002 (4)	0.9999 ± 0.0002 (2)
8	AdaBoostCost	0.9999 ± 0.0000 (3)	0.9992 ± 0.0014 (6)	0.9999 ± 0.0000 (2)	0.9999 ± 0.0000 (2)
9	UnderOverBagging	0.9999 ± 0.0000 (3)	0.9995 ± 0.0006 (4)	0.9998 ± 0.0001 (3)	0.9996 ± 0.0002 (5)
9	EasyEnsemble	0.9999 ± 0.0000 (3)	0.9994 ± 0.0008 (5)	0.9998 ± 0.0001 (3)	0.9997 ± 0.0002 (4)
9	RUSBoost	0.9997 ± 0.0001 (5)	0.9996 ± 0.0004 (3)	0.9998 ± 0.0001 (3)	0.9997 ± 0.0002 (4)
9	Naive Bayes (SMOTE)	0.9998 ± 0.0001 (4)	0.9915 ± 0.0040 (8)	1.0000 ± 0.0000 (1)	0.9999 ± 0.0000 (2)
10	SMOTEBoost	0.9998 ± 0.0001 (4)	0.9995 ± 0.0006 (4)	0.9996 ± 0.0002 (4)	0.9997 ± 0.0002 (4)
11	Decision Tree (RUS)	0.9996 ± 0.0002 (6)	0.9996 ± 0.0004 (3)	0.9999 ± 0.0000 (2)	0.9994 ± 0.0002 (7)
12	Random Forest (RUS)	0.9999 ± 0.0000 (3)	0.9992 ± 0.0012 (7)	0.9995 ± 0.0002 (5)	0.9995 ± 0.0002 (6)
13	Naive Bayes (RUS)	0.9993 ± 0.0002 (7)	0.9888 ± 0.0044 (9)	0.9980 ± 0.0004 (8)	0.9985 ± 0.0004 (8)
14	Naive Bayes (ROS)	0.9887 ± 0.0008 (8)	0.9854 ± 0.0044(11)	0.9985 ± 0.0004 (7)	0.9985 ± 0.0004 (8)
15	Naive Bayes (ADASYN)	0.9818 ± 0.0012 (9)	0.9876 ± 0.0040(10)	0.9911 ± 0.0009 (9)	0.9981 ± 0.0008 (9)

At a 1:500 ratio, the performance gap widens among techniques. While Random Forest (SMOTE) and Random Forest (ADASYN) maintain strong performance, others, like Decision Tree (ROS) and KNN (ADASYN), show increased variability (Figure 5). Despite the challenges, MLP (ROS) and EasyEnsemble continue to deliver consistent results.

Finally, at the extreme 1:1000 ratio, the difficulty of managing such imbalance becomes clear. Although Random Forest (SMOTE) and Random Forest (ADASYN) still achieve high F1-scores, techniques such as Naive Bayes (ROS) and KNN (ADASYN) exhibit significant drops, as evidenced in Figure 6. However, methods such as Weighted Decision Tree maintain commendable stability, suggesting their resilience under severe imbalance.

For G-Mean, techniques such as SMOTEBoost and RUSBoost remain competitive, though variability increases, especially for methods like Naive Bayes (SMOTE) (Figure 6). Despite the heightened difficulty, Under Over Bagging and EasyEnsemble manage to retain high G-Means, reflecting their robustness in handling severe data imbalance.

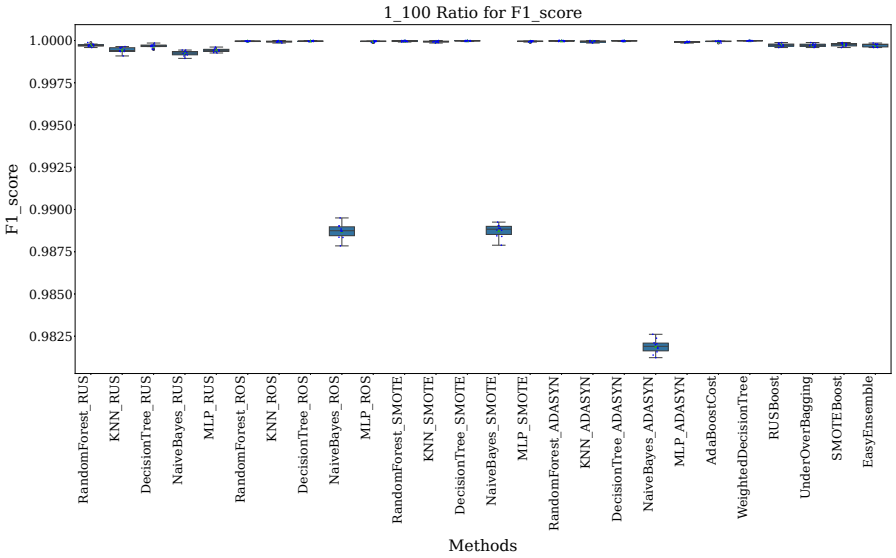

Figure 4. *Cont.*

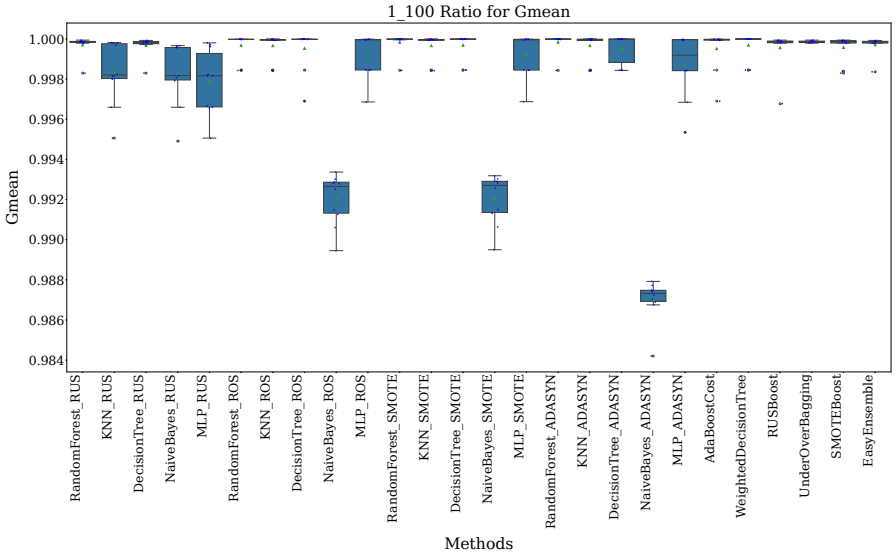

Figure 4. Recorded F1-score and G-mean for all methods with a 1:100 imbalance ratio.

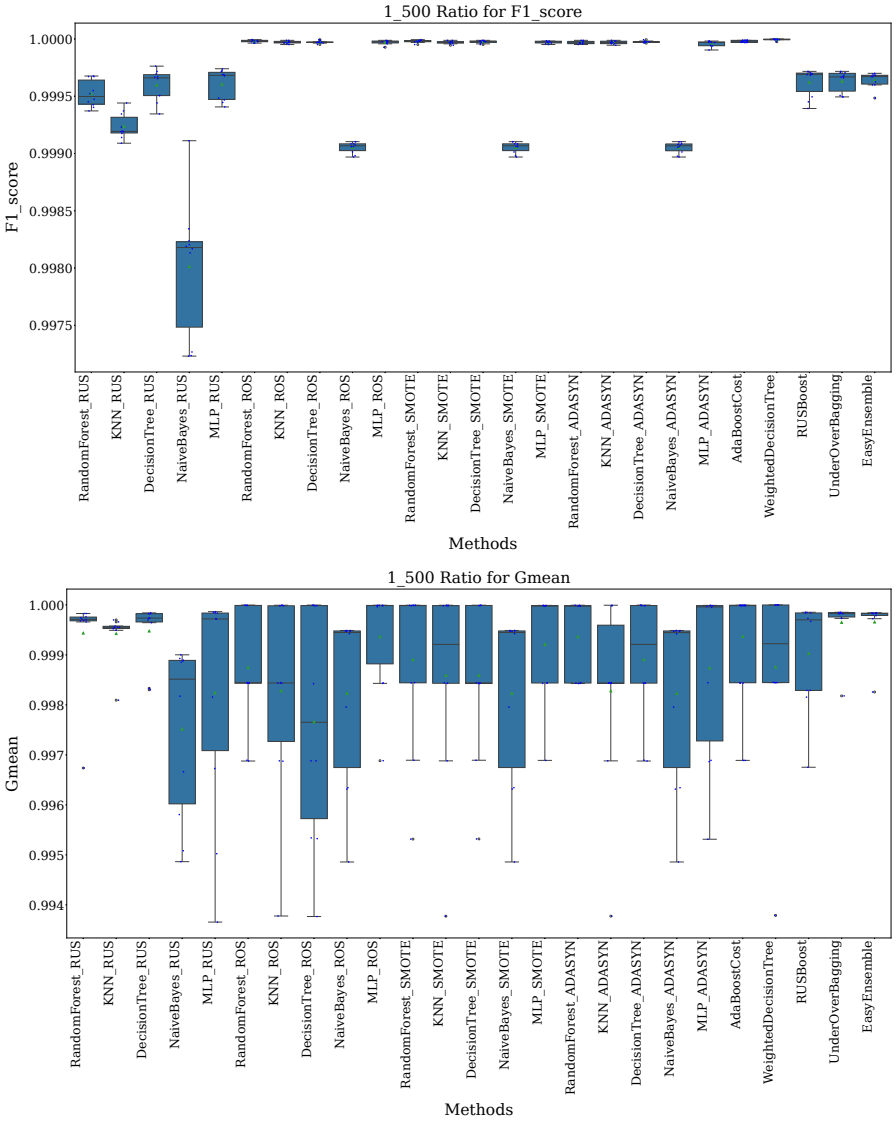

Figure 5. Recorded F1-score and G-mean for all methods under a 1:500 imbalance ratio.

Tables 4 and 5 provide a comparative performance analysis of various machine learning techniques across class imbalance ratios of 1:10, 1:100, 1:500, and 1:1000. Each table is organized by ranking the techniques based on their F1-scores and G-Mean metrics, respectively. This ranking helps to identify which methods are most effective at handling different degrees of class imbalance, which is critical for optimizing model performance in scenarios where data distribution among classes is uneven. Table 4 focuses on F1-scores, reflecting the balance between precision and recall, while Table 5 details G-Mean scores, highlighting the balance between sensitivity and specificity across the classes. These insights are instrumental for selecting appropriate machine learning strategies in practical applications where the class imbalance is a significant factor.

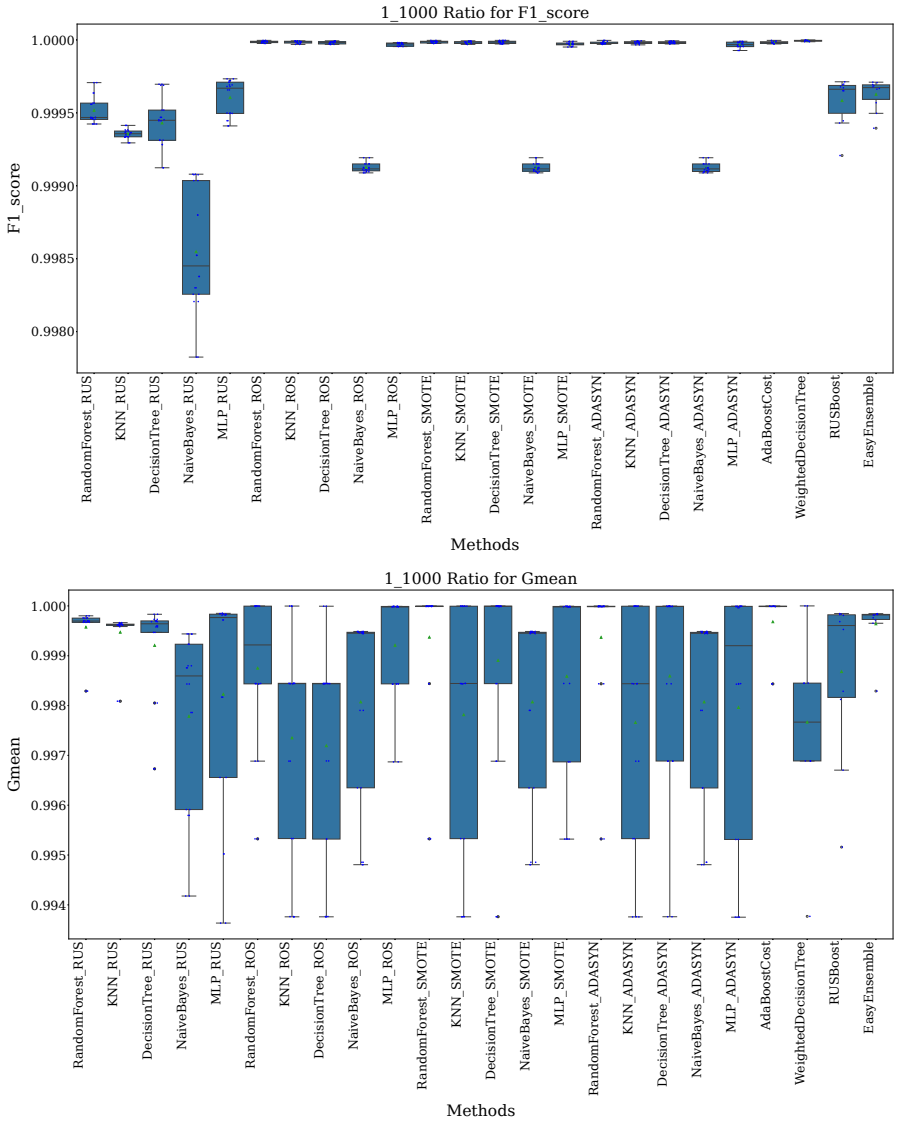

Figure 6. Recorded F1-score and G-mean for all methods at a 1:1000 imbalance ratio.

Table 5. Ranked G-mean for different techniques at 1:10, 1:100, 1:500, and 1:1000 imbalance ratios.

Rank	Technique	1:10 Ratio	1:100 Ratio	1:500 Ratio	1:1000 Ratio
1	SMOTEBoost	0.9999 ± 0.0002 (1)	0.9999 ± 0.0002 (1)	0.9997 ± 0.0004 (3)	0.9997 ± 0.0006 (7)
2	UnderOverBagging	0.9999 ± 0.0006 (2)	0.9999 ± 0.0006 (2)	0.9997 ± 0.0005 (4)	0.9997 ± 0.0005 (6)
3	RUSBoost	0.9998 ± 0.0010 (3)	0.9998 ± 0.0010 (3)	0.9998 ± 0.0010 (2)	0.9997 ± 0.0015 (9)
4	EasyEnsemble	0.9997 ± 0.0005 (5)	0.9996 ± 0.0004 (4)	0.9997 ± 0.0005 (4)	0.9996 ± 0.0005 (13)
5	KNN (ROS)	0.9990 ± 0.0012 (14)	0.9990 ± 0.0012 (9)	0.9996 ± 0.0007 (6)	0.9997 ± 0.0002 (5)

Table 5. *Cont.*

Rank	Technique	1:10 Ratio	1:100 Ratio	1:500 Ratio	1:1000 Ratio
6	KNN (SMOTE)	0.9991 ± 0.0008 (13)	0.9988 ± 0.0015 (11)	0.9998 ± 0.0008 (1)	0.9997 ± 0.0024 (11)
6	Random Forest (ADASYN)	0.9998 ± 0.0010 (3)	0.9991 ± 0.0008 (8)	0.9996 ± 0.0008 (7)	0.9994 ± 0.0014 (18)
8	Decision Tree (RUS)	0.9995 ± 0.0010 (9)	0.9985 ± 0.0012 (13)	0.9995 ± 0.0006 (8)	0.9997 ± 0.0010 (8)
9	KNN (ADASYN)	0.9993 ± 0.0010 (12)	0.9988 ± 0.0012 (10)	0.9993 ± 0.0010 (16)	0.9998 ± 0.0019 (4)
10	MLP (SMOTE)	0.9951 ± 0.0021 (16)	0.9985 ± 0.0027 (14)	0.9995 ± 0.0025 (12)	0.9998 ± 0.0016 (3)
11	Random Forest (RUS)	0.9998 ± 0.0010 (3)	0.9983 ± 0.0016 (17)	0.9994 ± 0.0009 (14)	0.9996 ± 0.0004 (12)
12	Random Forest (SMOTE)	0.9998 ± 0.0005 (4)	0.9995 ± 0.0010 (6)	0.9989 ± 0.0010 (20)	0.9994 ± 0.0014 (18)
13	Decision Tree (SMOTE)	0.9995 ± 0.0009 (8)	0.9982 ± 0.0025 (22)	0.9995 ± 0.0010 (10)	0.9997 ± 0.0020 (10)
14	KNN (RUS)	0.9988 ± 0.0021 (15)	0.9986 ± 0.0015 (12)	0.9995 ± 0.0008 (9)	0.9995 ± 0.0005 (16)
15	Decision Tree (ROS)	0.9994 ± 0.0008 (11)	0.9983 ± 0.0012 (16)	0.9994 ± 0.0008 (13)	0.9996 ± 0.0006 (14)
16	Random Forest (ROS)	0.9997 ± 0.0010 (7)	0.9992 ± 0.0012 (7)	0.9987 ± 0.0012 (22)	0.9988 ± 0.0015 (19)
16	WeightedDecision Tree	0.9997 ± 0.0009 (6)	0.9996 ± 0.0011 (5)	0.9988 ± 0.0018 (21)	0.9977 ± 0.0017 (23)
18	Decision Tree (ADASYN)	0.9995 ± 0.0010 (9)	0.9982 ± 0.0022 (21)	0.9995 ± 0.0010 (10)	0.9995 ± 0.0020 (17)
19	Naive Bayes (SMOTE)	0.9920 ± 0.0040 (19)	0.9982 ± 0.0022 (21)	0.9993 ± 0.0012 (17)	0.9998 ± 0.0012 (1)
19	Naive Bayes (RUS)	0.9920 ± 0.0040 (19)	0.9982 ± 0.0016 (19)	0.9993 ± 0.0013 (18)	0.9998 ± 0.0015 (2)
21	MLP (ADASYN)	0.9995 ± 0.0010 (9)	0.9977 ± 0.0037 (24)	0.9997 ± 0.0010 (5)	0.9980 ± 0.0025 (22)
22	Naive Bayes (ROS)	0.9920 ± 0.0030 (18)	0.9982 ± 0.0017 (20)	0.9995 ± 0.0012 (11)	0.9996 ± 0.0019 (15)
23	AdaBoostCost	0.9995 ± 0.0011 (10)	0.9982 ± 0.0013 (18)	0.9993 ± 0.0013 (18)	0.9986 ± 0.0008 (20)
24	MLP (RUS)	0.9921 ± 0.0037 (17)	0.9984 ± 0.0031 (15)	0.9992 ± 0.0017 (19)	0.9986 ± 0.0019 (21)
25	Naive Bayes (ADASYN)	0.9870 ± 0.0124 (20)	0.9981 ± 0.0019 (23)	0.9994 ± 0.0012 (15)	0.9980 ± 0.0025 (22)

5.3. Discussion

This particular research addresses the problem of class imbalance and does not seek to model temporal relations explicitly. However, such information can be explored in future studies using time-series-based methods. Furthermore, synthetic over-sampling techniques such as SMOTE and ADASYN, which are commonly used for CIL, can introduce overfitting issues, especially in greater ratio imbalances. To counter this, the proposed method embeds sophisticated cross-validation and feature selection to boost generalization and reduce complexity. Moreover, other sophisticated sampling techniques can be studied in future works in order to reduce overfitting, while the variety of the synthetic samples is preserved.

The scalability of the evaluated techniques is indeed different, especially when it comes to handling high-dimensional data and supporting real-time detection in large networks. Distributed architectures are one such scalable solution, where monitoring is decentralized across network segments to enable faster responses; centralized systems, while much easier to manage, tend to limit the processing of high-volume data. Feature selection methods, such as SelectKBest with ANOVA F-tests, work effectively in reducing dimensionality, thus improving computational efficiency without affecting detection accuracy. These are ensemble techniques like EasyEnsemble and SMOTEBoost, which show robustness in performance with a balance in processing demand, thus being more suitable for large-scale operations. Synthetic over-sampling techniques, such as SMOTE and ADASYN, may introduce computational overhead and probably cause overfitting problems, especially for real-time application scenarios. Future development could also consider distributed processing frameworks or edge computing for scalability enhancement in dynamic and high-volume network environments.

Discretization, as presented in works such as [35], can have a remarkable impact on the performance of intrusion detection methods. As it was underlined, the discretization may

enhance the interpretability of algorithms and reduce computational complexity, but if the intervals are defined imprecisely, information loss is possible. Discretization, when applied to the proposed methods, might improve the performance of ensemble methods such as Random Forest and AdaBoostCost, since it simplifies feature spaces, which helps in the formation of decision boundaries. Methods relying on synthetic sample generation, such as SMOTE, ADASYN, and their boosting variants, might suffer from reduced effectiveness, since discretization may disrupt interpolation and limit the diversity of synthetic data. Similarly, for neural network-based approaches, though discretization may simplify input patterns, it may detract from the network's ability to capture fine-grained continuous feature relationships.

6. Conclusions

To summarize this study on machine learning approaches for IDS, we found that techniques such as Random Forest (ADASYN) and KNN (ROS) stood out as effective options for addressing imbalanced datasets across various scenarios, with elevated class imbalance ratios. These methods consistently scored well in terms of F1-score and G-mean metrics. They displayed reliability and consistency when paired with sophisticated sampling methods. In addition, Under Over Bagging and SMOTEBoost are two methods that handle the balance between precision–recall and sensitivity–specificity across various imbalance ratios. On the other hand, EasyEnsemble along with different KNN and Decision Tree versions displayed consistency but might need some fine-tuning for better results. Conversely, Naive Bayes variations such as Naive Bayes (ADASYN) were not deemed ideal for imbalanced datasets, suggesting a combination with other techniques to improve performance.

The examination of how Random Forest (RUS) and KNN (RUS) deal with degrees of class imbalance revealed interesting trends. When faced with an imbalance ratio like 1:10, Random Forest (RUS) displayed strong performance with an F1-score of 0.9996 ± 0.0003. However, with an increase in the imbalance ratio, both Random Forest (RUS) and KNN (RUS) showed signs of performance deterioration, especially when handling a 1:500 ratio imbalance situation. A similar pattern was observed at the 1:1000 ratio, where several methods showed a decline in performance and an increase in variability.

At levels of imbalance in Random Forest (RUS) as well as in KNN (RUS), the decline in performance was mainly due to RUS. This technique involves decreasing the size of the training dataset by sampling from the majority class to align it with the minority class. This results in significant data loss and may cause overfitting to the minority class or underfitting to the majority class. As the class imbalance worsens, so does the influence of this data reduction on the model's capacity to generalize effectively. This higlights the necessity of approaches and improved sampling methods to enhance the model's resilience in datasets, with significant imbalances.

This study opens up several avenues for future work. Firstly, this work can be validated further by expanding the evaluations to diverse datasets. Temporal correlations in the CIC-DDoS2019 dataset should be exploited using time-series-based models. Exploring advanced sampling techniques could address the overfitting risks associated with synthetic sampling. Finally, the adaptation of those approaches for real-time intrusion detection in dynamic networks remains a critical challenge that requires further research.

Author Contributions: Conceptualization, R.R.-F.; methodology, V.S. and R.R.-F.; software, V.S.; validation, V.S.; formal analysis, V.S. and E.H.; investigation, V.S.; resources, R.R.-F.; data curation, E.H.; writing—original draft preparation, V.S.; writing—review and editing, R.R.-F. and E.H.; visualization, V.S.; supervision, R.R.-F.; project administration, R.R.-F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Available upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328.
2. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Int. Res.* **2002**, *16*, 321–357. [CrossRef]
3. He, K.; Kim, D.D.; Asghar, M.R. Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey. *Commun. Surv. Tutor.* **2023**, *25*, 538–566. [CrossRef]
4. Jamalipour, A.; Murali, S. A Taxonomy of Machine-Learning-Based Intrusion Detection Systems for the Internet of Things: A Survey. *IEEE Internet Things J.* **2021**, *9*, 9444–9466. [CrossRef]
5. Ou, C.M. Host-based Intrusion Detection Systems Inspired by Machine Learning of Agent-Based Artificial Immune Systems. In Proceedings of the 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), Sofia, Bulgaria, 3–5 July 2019; pp. 1–5. [CrossRef]
6. Malek, Z.S.; Trivedi, B.; Shah, A. User behavior Pattern -Signature based Intrusion Detection. In Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 27–28 July 2020; pp. 549–552. [CrossRef]
7. Nisioti, A.; Mylonas, A.; Yoo, P.; Katos, V. From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3369–3388. [CrossRef]
8. Sowmya, T.; Anita, E. A comprehensive review of AI based intrusion detection system. *Meas. Sens.* **2023**, *28*, 100827. [CrossRef]
9. Bilot, T.; El Madhoun, N.; Al Agha, K.; Zouaoui, A. Graph Neural Networks for Intrusion Detection: A Survey. *IEEE Access* **2023**, *11*, 49114–49139. [CrossRef]
10. Kiran, A.; Prakash, S.W.; Kumar, B.A.; Likhitha; Sameeratmaja, T.; Charan, U.S.S.R. Intrusion Detection System Using Machine Learning. In Proceedings of the 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 23–25 January 2023; pp. 1–4. [CrossRef]
11. Gaber, T.; Awotunde, J.B.; Torky, M.; Ajagbe, S.A.; Hammoudeh, M.; Li, W. Metaverse-IDS: Deep learning-based intrusion detection system for Metaverse-IoT networks. *Internet Things* **2023**, *24*, 100977. [CrossRef]
12. Fosić, I.; Žagar, D.; Grgić, K. Network traffic verification based on a public dataset for IDS systems and machine learning classification algorithms. In Proceedings of the 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 23–27 May 2022; pp. 1037–1041. [CrossRef]
13. Li, S.; Cao, Y.; Liu, S.; Lai, Y.; Zhu, Y.; Ahmad, N. HDA-IDS: A Hybrid DoS Attacks Intrusion Detection System for IoT by using semi-supervised CL-GAN. *Expert Syst. Appl.* **2024**, *238*, 122198. [CrossRef]
14. Milosevic, M.S.; Ciric, V.M. Extreme minority class detection in imbalanced data for network intrusion. *Comput. Secur.* **2022**, *123*, 102940. [CrossRef]
15. Chen, W.; Yang, K.; Yu, Z.; Shi, Y.; Chen, C.L.P. A survey on imbalanced learning: Latest research, applications and future directions. *Artif. Intell. Rev.* **2024**, *57*, 137. [CrossRef]
16. Ren, H.; Tang, Y.; Dong, W.; Ren, S.; Jiang, L. DUEN: Dynamic ensemble handling class imbalance in network intrusion detection. *Expert Syst. Appl.* **2023**, *229*, 120420. [CrossRef]
17. Bagui, S.; Li, K. Resampling imbalanced data for network intrusion detection datasets. *J. Big Data* **2021**, *8*, 6. [CrossRef]
18. Razavi-Far, R.; Farajzadeh-Zanjani, M.; Saif, M. An Integrated Class-Imbalanced Learning Scheme for Diagnosing Bearing Defects in Induction Motors. *IEEE Trans. Ind. Inform.* **2017**, *13*, 2758–2769. [CrossRef]
19. Farajzadeh-Zanjani, M.; Razavi-Far, R.; Saif, M. Efficient sampling techniques for ensemble learning and diagnosing bearing defects under class imbalanced condition. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–7. [CrossRef]
20. Zhihao, P.; Fenglong, Y.; Xucheng, L. Comparison of the Different Sampling Techniques for Imbalanced Classification Problems in Machine Learning. In Proceedings of the 2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Qiqihar, China, 28–29 April 2019; pp. 431–434. [CrossRef]
21. Fan, W.; Stolfo, S.J.; Zhang, J.; Chan, P.K. AdaCost: Misclassification Cost-Sensitive Boosting. In Proceedings of the Sixteenth International Conference on Machine Learning, Bled, Slovenia, 27–30 June 1999; pp. 97–105.
22. Rezvani, S.; Wang, X. A broad review on class imbalance learning techniques. *Appl. Soft Comput.* **2023**, *143*, 110415. [CrossRef]

23. Seiffert, C.; Khoshgoftaar, T.M.; Van Hulse, J.; Napolitano, A. RUSBoost: A Hybrid Approach to Alleviating Class Imbalance. *IEEE Trans. Syst. Man Cybern.-Part A Syst. Humans* **2010**, *40*, 185–197. [CrossRef]
24. Moniz, N.; Ribeiro, R.; Cerqueira, V.; Chawla, N. SMOTEBoost for Regression: Improving the Prediction of Extreme Values. In Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–4 October 2018; pp. 150–159. [CrossRef]
25. Kusdiyanto, A.Y.; Pristyanto, Y. Machine Learning Models for Classifying Imbalanced Class Datasets Using Ensemble Learning. In Proceedings of the 2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 8 December 2022; pp. 648–653. [CrossRef]
26. Gu, Y.; Yang, Y.; Yan, Y.; Shen, F.; Gao, M. Learning-based intrusion detection for high-dimensional imbalanced traffic. *Comput. Commun.* **2023**, *212*, 366–376. [CrossRef]
27. Tabassum, A.; Erbad, A.; Lebda, W.; Mohamed, A.; Guizani, M. FEDGAN-IDS: Privacy-preserving IDS using GAN and Federated Learning. *Comput. Commun.* **2022**, *192*, 299–310. [CrossRef]
28. Yakshit.; Kaur, G.; Kaur, V.; Sharma, Y.; Bansal, V. Analyzing various Machine Learning Algorithms with SMOTE and ADASYN for Image Classification having Imbalanced Data. In Proceedings of the 2022 IEEE International Conference on Current Development in Engineering and Technology (CCET), Bhopal, India, 23–24 December 2022; pp. 1–7. [CrossRef]
29. Than, S.S.M.; Soe, A.M.; Maw, A.H. Investigation of Oversampling in IoT-IDS. In Proceedings of the 2024 IEEE Conference on Computer Applications (ICCA), Yangon, Myanmar, 16 March 2024; pp. 1–6. [CrossRef]
30. Abdelkhalek, A.; Mashaly, M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *J. Supercomput.* **2023**, *79*, 10611–10644. [CrossRef]
31. Liu, Y.; Li, B.; Yang, S.; Li, Z. Handling missing values and imbalanced classes in machine learning to predict consumer preference: Demonstrations and comparisons to prominent methods. *Expert Syst. Appl.* **2024**, *237*, 121694. [CrossRef]
32. Farajzadeh-Zanjani, M.; Hallaji, E.; Razavi-Far, R.; Saif, M. Generative-Adversarial Class-Imbalance Learning for Classifying Cyber-Attacks and Faults - A Cyber-Physical Power System. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 4068–4081. [CrossRef]
33. Santhiappan, S.; Ravindran, B. Class Imbalance Learning. *Adv. Comput. Commun.* **2017**, *1*. [CrossRef]
34. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*, 27. [CrossRef]
35. Yoo, J.; Min, B.; Kim, S.; Shin, D.; Shin, D. Study on Network Intrusion Detection Method Using Discrete Pre-Processing Method and Convolution Neural Network. *IEEE Access* **2021**, *9*, 142348–142361. [CrossRef]