Research Article

Evolutionary Data Preprocessing to Alleviate Class Imbalance

Jae-Hyun Seo

Department of Computer Engineering, Gwangju University, 277 Hyodeok-ro, Nam-gu, Gwangju 54649, Republic of Korea

Correspondence should be addressed to Jae-Hyun Seo; delphia7979@gmail.com

Received 27 June 2022; Revised 26 August 2022; Accepted 29 September 2022; Published 11 October 2022

Copyright © 2022 Jae-Hyun Seo. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Intrusion detection technology for network attacks is developing rapidly with the development of artificial intelligence technology. Recently, machine learning-based methods that can detect new types of attacks have been developed. To improve the classification performance of the rare classes in the intrusion detection dataset, we study the efficient data preprocessing method based on machine learning. The UNSW-NB15, a well-known network intrusion detection dataset, is used in the experiments. The dataset includes 9 attack types and has severe class imbalance and overlap, so it is difficult to improve the classification performance above a certain level. To improve the classification performance by adjusting the number of instances of rare classes is needed. SMOTE techniques and genetic algorithms are used to optimize the ratio between classes in the training dataset. The computation time is reduced by creating a training dataset that samples only a few percent of the UNSW-NB15 dataset. Many new training datasets are generated based on the small training dataset according to the randomly generated SMOTE ratios. The classification experiments are conducted with these new training datasets. A new dataset is generated by combining the results of the experiments, and a regression model is generated by training the dataset. The best tuple of SMOTE ratios is searched by applying the model as a fitness function of the genetic algorithm. The D-S-1G combination exhibited the best performance among the test results. It consists of a decision tree classifier and the support vector regressor (SVR). As a result, the computation time was significantly reduced, and the optimal SMOTE ratios showed better results than the experimental results of the original UNSW-NB15 dataset. It was found that the classification result of each rare class relies heavily on the type of classifier.

1. Introduction

An intrusion detection system based on artificial intelligence [1–3] is needed to detect new and malicious network attacks that traditional firewalls cannot detect. This system protects against network attacks on vulnerable services, data-driven attacks in applications, and privilege escalation and intruder login/access to major files by intruders/malicious software (computer viruses, Trojan horses, and worms) and host-based attacks.

Intrusion detection can be divided into signature-based and anomaly-based detection. Signature-based detection has a low false positive rate, but it has limited ability to detect new attack types because it works by matching patterns after inputting information for anomalous behavior. Anomaly detection is a method based on statistical analysis and machine learning. Anomaly detection has a higher false positive rate than signature-based methods, but it has the advantage of being able to detect new attack types.

Intrusion detection systems can be divided into network-based IDS (N-IDS), host-based IDS (H-IDS), and multihost-based IDS based on the detection location. In the N-IDS system, the intrusion detection system equipment is installed in the front stage, and the traffic going to and from the network is analyzed. Although the N-IDS system has the advantage of being independent of the operating system, considerable packet loss occurs in high-speed networks. In addition, it is difficult to detect abnormal behavior that occurs inside the host. In the H-IDS system, audit data collected from a specific host system are analyzed to detect abnormal behavior. The H-IDS system accurately detects attacks, decrypts and analyzes encrypted packets, and has the advantage of no packet loss. In contrast, the H-IDS has the disadvantage of being dependent on the operating system and using system resources. A multihost-based IDS is a system that comprehensively analyzes all host systems to detect abnormal behavior.
Recently, with the rapid development of artificial intelligence technology, research on anomaly-based IDSs has been actively conducted. To promote intrusion detection research, datasets such as KDD’99 and UNSW-NB15 have been released. Many scholars have conducted research on intrusion detection using these public datasets. However, since there is a limit to how much the performance of intrusion detection can be improved using only artificial intelligence technology, the trend is to combine intrusion detection systems with the existing signature-based method.

Class imbalance and class overlap make it difficult to improve detection performance using only the intrusion detection dataset and artificial intelligence technology. In general, when learning models are created using a dataset with a high proportion of a specific class, most observations tend to be classified as belonging to the majority class based on the learning model, and it is uncommon for observations to be assigned to the minority class (rare class).

In this study, SMOTE [4], classification, regression, and genetic algorithms (GAs) [5] are used to improve the detection performance for rare classes in the UNSW-NB15 dataset. The classification results are derived by varying the class ratio of the training dataset, and the regression model is created using the experimental results. This model is used as the fitness function of GAs. After finding the optimized SMOTE ratio and creating a training dataset using it, we apply the classification algorithm to assess the classification performance. Finally, the test dataset is input into the training model to measure the attack type detection performance.

The contributions of this study are as follows: (1) there are many classification experiments, so only some (approximately 3%) of the UNSW-NB15 dataset is used as a training dataset to reduce the computation time; (2) an optimal solution is derived only with a relatively small number of classification experiments; (3) a regression model is used as a fitness function of GAs to drastically reduce the computation time. In other words, it is not necessary to perform a classification experiment every time the fitness function is called. (4) We used RMC (root mean cube) to reflect the characteristics of classes well and reduce the effect of outliers instead of the RMS (root mean square) of the previous research [6].

The remainder of this study is organized as follows: Section 2 describes previous studies related to this study. In Section 3, we present a description of the UNSW-NB15 dataset. In Section 4, the problem to be solved is defined, and the proposed methods are explained. In Section 5, we explain our experimental environments, plan, and results. The study ends with some concluding remarks in Section 6.

2. Related Work

Soltanzadeh and Hashemzadeh [7] conducted a study of the class imbalance problem. To improve the SMOTE algorithm, they tried to overcome the following three issues: (1) overgeneralization due to oversampling of noisy samples, (2) oversampling of uninformative samples, and (3) the increasing overlap of different classes around class boundaries.

To address issues 1 and 2, they applied a sample categorization method to identify minor samples that are suitable for oversampling. To address the third issue, they proposed an improved sample creation process that generates synthetic samples within an accurately calculated safe range. This range is calculated based on the characteristics of the input data to provide a safe oversampling region for each dimension of the feature space. The extracted range is used to control the position of the new synthetic sample in the data space and to prevent it from drifting into the majority class domain.

Bagui and Li [8] used resampling to better balance the classes in a dataset by adjusting the ratios of different classes. In experiments using the benchmark cybersecurity datasets KDD99, UNSW-NB15, UNSW-NB17, and UNSW-NB18, evaluation using macro precision, macro recall, and macro F1-score values led to the following conclusions: first, oversampling increases the training time, and undersampling decreases the training time; second, both oversampling and undersampling significantly increase recall when the data are extremely imbalanced; third, resampling does not have a significant effect if the data imbalance is not severe; and fourth, resampling detects more minority data.

A study by Zoghi and Serpen [9] presented a visual analysis of the UNSW-NB15 intrusion detection dataset. PCA, t-SNE, and k-means clustering algorithms were used to develop graphs and plots for visualization. After visualizing the results, they identified and described two main problems for this dataset: class imbalance and class overlap. This shows that it is necessary to solve the problems of class imbalance and class overlap before using this dataset for classifier model development.

Choudhary and Kesswani [10] used a deep neural network (DNN) to identify IoT attacks. An intelligent intrusion detection system requires an effective dataset. The performance of DNNs to accurately identify attacks was evaluated on the most popular datasets (e.g., KDD-Cup’99, NSL-KDD, and UNSW-NB15). Experimental results showed that the accuracy of the proposed method using DNN was more than 90%.

Kumar et al. [11] proposed a new misuse-based intrusion detection system that detects five categories in the network, namely, exploit, DOS, probe, generic, and normal. They designed their own unified classification-based model using the UNSW-NB15 dataset. This model showed significantly higher performance than other existing decision tree-based models that detect five categories. Additionally, the NIT Patana CSE lab, which published this study, generated its own real-time dataset, RTNITP18. The RTNITP18 dataset was used as an experimental dataset to evaluate the performance of the proposed intrusion detection model. When the performance of the proposed model was analyzed using UNSW-NB15 and the real-time dataset RTNITP18, it exhibited better performance in terms of accuracy, attack detection rate, average F1-score, average accuracy, attack accuracy, and false alarm rate compared to other models.

Sun et al. [12] conducted a study on class imbalance. The class imbalance problem has been reported to seriously impair the classification performance of many standard
learning algorithms and thus has received much attention from researchers in various fields. Therefore, several methods have been proposed to solve these problems, such as sampling methods, cost-sensitive learning methods, and ensemble methods based on bagging and boosting. However, the conventional methods for handling class imbalances potentially suffer from loss of useful information, unexpected mistakes, or increased likelihood of overfitting because they can alter the original data distribution. Therefore, the imbalanced dataset was first transformed into several balanced datasets. The authors proposed a novel ensemble method to build multiple classifiers based on multiple datasets using a specific classification algorithm. Finally, the classification results of these classifiers on the new dataset were combined by specific ensemble rules. In an empirical study, the authors compared their method with the existing various class imbalance data processing methods. Experiments were conducted on 46 imbalanced datasets, and the experimental results showed that their method was generally superior to the existing imbalanced datasets, and the experimental results showed that their methods. Experiments performed on eight imbalanced datasets in the KEEL repository helped highlight the importance of the method. Comparative analysis showed performance improvement in terms of the area under the ROC curve (AUC).

Haixiang et al. [16] provided an in-depth review of rare event detection from the perspective of imbalance learning. For this analysis, 517 related papers published over the past 10 years were collected. The authors reviewed all the papers collected from both a technical and a practical point of view. The modeling methods discussed included techniques such as data preprocessing, classification algorithms, and model evaluation. The authors provided a comprehensive classification of existing application domains of imbalance learning and then detailed the applications for each category. Integrating some of the suggestions from the reviewed papers with their experience and judgment, they provided directions for further research in the fields of imbalance learning and rare event detection.

Zheng et al. [17] tried to alleviate the class imbalance problem using SMOTE. SMOTE is the most widely used data-level method, and many derivatives of the original model have been developed to alleviate the class imbalance problem. The authors found that SMOTE has serious flaws and proposed a new oversampling method SNOCC that can compensate for the shortcomings of SMOTE. In SNOCC, increasing the number of seed samples prevents new samples from connecting on the line segment between the two seed samples in SMOTE. The authors used a new algorithm that differs from the previous one to find the nearest neighbor of the sample. With these two improvements, new samples generated by SNOCC can naturally reproduce the distribution of original seed samples. Experimental results showed that SNOCC outperformed SMOTE and CBSO (SMOTE-based methods).

Suleiman and Issac [18] attempted to improve the detection performance of intrusion detection systems (IDS) using machine learning. IDS suffers from setbacks such as false positives (FP), low detection accuracy, and false negatives (FN). To improve the performance of IDS, machine learning classifiers are used to support detection accuracy and significantly reduce false positive and false negative rates. In their study, they used six classifiers based on machine learning. For three types of datasets, such as NSL-KDD, UNSW-NB15, and phishing datasets, their results show that k-NN and decision tree were the best classifiers in terms of detection accuracy, test time, and false positive rate.

Nawir et al. [19] attempted to build a network anomaly detection system using an efficient, effective, and fast machine learning algorithm. They performed binary classification experiments using the UNSW-NB15 dataset. The experimental results showed that the AODE algorithm was superior in terms of accuracy and computation time for binary classification on the UNSW-NB15 dataset.

Douzas and Baco [20] approximated the actual data distribution using the conditional version of cGAN.
(generative adversarial networks) and generated data for minority classes in various imbalanced datasets. They compared the performance of cGAN with several standard oversampling algorithms. They presented empirical results showing that the quality of the generated data was significantly improved when cGAN was used as the oversampling algorithm.

Douzas and Bacao [21] proposed a new oversampling method called self-organizing map-based oversampling (SOMO). This method enables the effective generation of artificial data by generating a two-dimensional representation in the input space through the application of a self-organizing map. SOMO consists of three main phases. Initially, the self-organizing map creates an original two-dimensional space. Next, it generates intracluster synthetic samples and finally intercluster synthetic samples. Additionally, the authors presented empirical results showing that the performance of the algorithm was improved when using artificial data generated from SOMO and showed that their method outperformed various oversampling methods.

Gong and Kim [22] proposed an effective ensemble classification method called RHSBoost to solve the imbalance classification problem. Their classification rule uses random undersampling and ROSE sampling in the boosting scheme. The experimental results suggested that RHSBoost is an attractive classification model for imbalanced data.

3. Background

3.1. UNSW-NB15 Dataset and Data Preprocessing. The UNSW-NB15 intrusion detection dataset [23] was generated through the IXIA traffic generation testbed in Figure 1. The IXIA traffic generator consisted of three virtual servers. Server 1 and Server 3 generate normal traffic, and Server 2 generates abnormal or malicious activity in network traffic. After establishing internal communication between the servers and collecting public and private network traffic, there are two virtual interfaces with IP addresses 10.40.85.30 and 10.40.184.30. Servers connect to hosts through two routers. Router 1 is configured with IP addresses of 10.40.85.1 and 10.40.182.1, and Router 2 is configured with IP addresses of 10.40.184.1 and 10.40.183.1. These routers connect to firewall devices configured to allow both normal and abnormal traffic to pass through. The tcpdump tool is installed on Router 1 to capture pcap files during simulation uptime.

The goal of this testbed is to collect normal or abnormal traffic that originates from the IXIA tool and spreads out to network nodes (e.g., server and clients). The IXIA tool generates attack traffic in addition to normal traffic. To generate attack traffic similar to the actual attack environment, the attack behavior is generated from the common vulnerabilities and exposures (CVE) site [24]. Using the IXIA tool, the first simulation was configured to include 1 attack per second, and the second simulation was configured to include 10 attacks per second. The data captured during the simulation process were 50 GB each.

There were 47 features provided by the UNSW-NB15 dataset. In the proposed method, the features were extracted from the srcIP and dstIP features, and 53 features were configured, as shown in Table 1. The attack_cat feature was 1–10, and it was used for class classification. When network traffic was normal, the class value was 1. The srcIP feature was divided into srcIP1, srcIP2, srcIP3, and srcIP4 features. The srcIP is the IP address, and the extracted features were srcIP1-4 mean class A–D of the IP address. The dstIP feature was also divided into dstIP1, dstIP2, dstIP3, and dstIP4 in the same way.

The rare class has a relatively small number of instances or shows low classification performance compared to other classes. In this study, five classes were considered rare, namely, reconnaissance (3), DoS (4), worms (8), backdoor (9), and analysis (10).

Given a dataset $D$, and a set of labels $L$, where the labels of an example are denoted with $Y_i$, we can define label cardinality as below. Label cardinality of $D$ is the average number of labels of the examples in $D$:

$$\text{imbalance ratio} = \frac{|Y_i|}{LC(D)} \sum_{i=1}^{D} |Y_i|$$

The classification experiment using the original training dataset requires too much computation time. The proposed method reduces the computation time by adjusting the class ratio of the training and validation datasets, as shown in the experimental results in Table 2–5 [25]. Tables 2 and 3 show the changes in the distribution of normal and generic classes. Table 2 shows the difference in the number of instances before and after undersampling, and Table 3 shows the change in the class imbalance ratio. The class imbalance ratio is calculated by (1) [26].

While reducing the number of instances occupied by normal and generic classes, undersampling was attempted to the extent that the classification performance of the two classes was not significantly degraded. Of the 229,110 instances, 1/3 was used as the training dataset, and another 1/3 was used as the validation dataset. The test dataset was used by adding the reduced number of instances of normal and generic classes to the remaining 1/3. That is, the test dataset was restored as if there were no undersampling processes.

![Figure 1: IXIA traffic generation testbed.](image-url)
Table 4 shows that the recall values of all classes changed while reducing the number of instances of the normal class by 1/2, that is, 1/2\(n\). As the number of instances of the normal class decreased, the weighted average decreased. The change in classification performance for each class was relatively small. When \(n = 5\), the class imbalance ratio of the normal class was reduced to 2.81%, and its classification performance was not significantly affected.

After adjusting the number of instances of the normal class, Table 5 shows the change in recall values according to the change in the number of instances of the generic class. In the case of the generic class, the class imbalance was mitigated to an appropriate level when \(n = 2\). If the number of instances of normal and generic classes was reduced, the recall value of those classes was somewhat lowered, but the computation time was significantly reduced.

Since the proposed method requires more than many SVM classification experiments, a very large amount of computation time is needed. By reducing the number of instances of normal and generic classes, the computation time can be significantly reduced. StratifiedRemoveFolds [27] was used to reduce the number of instances.

Table 6 shows the class imbalance ratios of the training, validation, and test datasets. In the test dataset, the class imbalance ratio of normal and generic classes was large. This is because we did not reduce the number of instances of those classes.

3.2. SMOTE: Synthetic Minority Oversampling Technique. The synthetic minority oversampling technique (SMOTE) is an oversampling method that generates a new sample by adding a random value after taking a class sample and adding it to the data. SMOTE uses a \(k\)-NN (\(k\) nearest neighbor) algorithm to add points at slightly shifted positions from existing instances. SMOTE is similar to random oversampling in which it increases the number of instances of rare classes. However, it does not recreate the same instance. Instead, it creates new instances by appropriately combining existing instances, avoiding the disadvantage of overfitting.

4. Modeling

4.1. Problem definition. We attempt to solve the UNSW-NB15 classification problem, a network intrusion detection dataset. The dataset has severe class importance and overlap, so it is not easy to improve classification performance.
Figure 2 is a two-dimensional scatter plot, which is derived by inputting the original training dataset to the t-SNE (t-distributed stochastic neighbor embedding) algorithm. This plot shows that the classes had multiple clusters of different sizes, and the boundaries between the classes were not clear; that is, it shows that there were class overlaps. Many attack classes mimic the behavior of the normal class. Figure 3 shows the degree of class overlap using PCA. Figures 2 and 3 suggest that it is very difficult to increase the classification performance of all classes at the same time. However, by overcoming these difficulties using data preprocessing techniques, the classification performance for the five rare classes can be improved.

If there is a class imbalance problem, most of the examples are classified into majority classes, not minority classes. The ratios between classes of the training dataset need to be adjusted to solve this problem. The SMOTE algorithm is used as a method of adjusting the ratio between classes. It is needed to test all tuples of SMOTE ratios to optimize the ratio of each class; however, there are time and cost constraints to test all cases. Therefore, it is needed to find the best tuple of SMOTE ratios just by testing with a small number of experiments. Formula (1) is a method of calculating the class imbalance ratio of each class, and Table 6 shows class imbalance ratios and the number of training, validation, and test datasets. In summary, data preprocessing methods that optimize the ratio between classes should be found, and the performance should be proven through experiments.

4.2. Proposed Methods. To find the optimal class ratio, one approach is to experiment with all combinations of SMOTE ratios. Considering time and cost, it is not efficient to

| Classes   | Original imbalance ratio (%) | Adjusted imbalance ratio (%) |
|-----------|-----------------------------|-----------------------------|
| Normal    | 690.59                      | 2.81                        |
| Exploits  | 1.78                        | 1.78                        |
| Reconnaissance | 0.55                     | 0.55                        |
| DoS       | 0.65                        | 0.65                        |
| Generic   | 9.27                        | 2.17                        |
| Shellcode | 0.06                        | 0.06                        |
| Fuzzers   | 0.96                        | 0.96                        |
| Worms     | 0.01                        | 0.01                        |
| Backdoor  | 0.09                        | 0.09                        |
| Analysis  | 0.11                        | 0.11                        |

Standard deviation 217.93 1.00

**Table 4:** Recalls according to the number of instances of the normal class.

| Class | 0  | 1  | 2  | 3  | 4  | 5  |
|-------|----|----|----|----|----|----|
| 1     | 0.999 | 0.997 | 0.995 | 0.993 | 0.989 | 0.987 |
| 2     | 0.850 | 0.842 | 0.840 | 0.843 | 0.839 | 0.846 |
| 3     | 0.776 | 0.777 | 0.773 | 0.777 | 0.777 | 0.777 |
| 4     | 0.394 | 0.404 | 0.417 | 0.404 | 0.422 | 0.404 |
| 5     | 0.990 | 0.990 | 0.989 | 0.990 | 0.990 | 0.990 |
| 6     | 0.901 | 0.904 | 0.918 | 0.909 | 0.901 | 0.911 |
| 7     | 0.764 | 0.81  | 0.845 | 0.869 | 0.886 | 0.895 |
| 8     | 0.569 | 0.701 | 0.667 | 0.655 | 0.603 | 0.621 |
| 9     | 0.112 | 0.112 | 0.111 | 0.116 | 0.111 | 0.110 |
| 10    | 0.160 | 0.162 | 0.173 | 0.169 | 0.171 | 0.173 |
| W. Avg| 0.986 | 0.976 | 0.962 | 0.947 | 0.933 | 0.923 |

**Table 5:** Recalls according to the number of instances of the generic class.

| Class | 0  | 1  | 2  |
|-------|----|----|----|
| 1     | 0.999 | 0.997 | 0.995 |
| 2     | 0.850 | 0.842 | 0.84 |
| 3     | 0.776 | 0.777 | 0.773 |
| 4     | 0.394 | 0.404 | 0.417 |
| 5     | 0.990 | 0.990 | 0.989 |
| 6     | 0.901 | 0.904 | 0.918 |
| 7     | 0.764 | 0.810 | 0.845 |
| 8     | 0.569 | 0.701 | 0.667 |
| 9     | 0.112 | 0.112 | 0.111 |
| 10    | 0.160 | 0.162 | 0.173 |
| W. Avg| 0.986 | 0.976 | 0.962 |

Figure 2 [9] is a two-dimensional scatter plot, which is derived by inputting the original training dataset to the t-SNE (t-distributed stochastic neighbor embedding) algorithm. This plot shows that the classes had multiple clusters of different sizes, and the boundaries between the classes were not clear; that is, it shows that there were class overlaps. Many attack classes mimic the behavior of the normal class. Figure 3 shows the degree of class overlap using PCA. Figures 2 and 3 suggest that it is very difficult to increase the classification performance of all classes at the same time. However, by
evaluate all possible combinations. Therefore, we suggest a method to maximize the classification performance while dramatically reducing the computation time.

Figure 4 shows the flowchart of the data preprocessing and experimental process of the proposed method. After the data preprocessing process described in Section 3, tuples of SMOTE ratios are randomly generated according to the range in Table 7. New training datasets are created according to the tuples. A classification experiment is performed using these training datasets and the validation dataset. A regression model is created using the recall values that are the results of these experiments. There are two optimization methods using the regression model. First, a large number of tuples of SMOTE ratios are randomly generated. The tuples are entered into the regression model to find the tuple with the best performance. Next, the regression model is used as the fitness function of the GA to find the optimal SMOTE ratios. In the following steps, the pseudocode and formula of the proposed method are shown:

(1) **Step 1:** Randomly generate many tuples of SMOTE ratios for five rare classes according to the range in Table 7. The new training datasets are generated by inputting the ratios of the tuples as one of the parameters of the SMOTE algorithms [4], and the datasets are added to the set, TS.

\[
\text{SMOTE}(T, N, k); /\ast \text{ SMOTE pseudocode } /\ast
\]

**Input:**
- \(T\): Train dataset;
- \(N\%\) amount of synthetic samples;
- \(k\): Number of nearest neighbors.

**Output:**
- \(S\): Set of synthetic samples

**begin**

\[
\begin{align*}
X & \leftarrow \text{MinorityInstances}(T); \\
n & \leftarrow \text{NumberOfInstances}(X); \\
p & \leftarrow \text{NumberOfVariables}(X); \\
S & \leftarrow \emptyset; \\
\text{if } N < 100 \text{ then} \\
\beta & \leftarrow \text{RandomNumber}(1, k); /\ast \text{ Chose one neighbor of } i /\ast \\
\text{for } i & \leftarrow 1 \text{ to } n \text{ do} \\
\alpha & \leftarrow \text{RandomNumber}(0, 1); \\
S_{ij} & \leftarrow X_{ij} + (\bar{X}_i - X_{ij}) \times \alpha; \\
N & \leftarrow N - 1;
\end{align*}
\]

**end**

| Classes       | Train and validation | Imbalance ratio (%) | Test         | Imbalance ratio (%) |
|---------------|----------------------|---------------------|--------------|---------------------|
| Normal        | 23,112               | 2.81                | 739,588      | 43.40               |
| Exploits      | 14,842               | 1.78                | 14,841       | 24.14               |
| Reconnaissance| 4,662                | 0.55                | 4,663        | 6.50                |
| DoS           | 5,451                | 0.65                | 5,451        | 7.69                |
| Generic       | 17,957               | 2.17                | 71,828       | 30.76               |
| Shellcode     | 504                  | 0.06                | 504          | 0.66                |
| Fuzzers       | 8,082                | 0.96                | 8,082        | 11.84               |
| Worms         | 58                   | 0.01                | 58           | 0.08                |
| Backdoor      | 776                  | 0.09                | 776          | 1.03                |
| Analysis      | 893                  | 0.11                | 892          | 1.18                |
| Sum           | 76,337               | 1.00                | 846,683      | 142.35              |

Table 6: Class imbalance ratios of the training, validation, and test datasets.
end
end
return $S$;

$T; / *$ The original training dataset * /

$TS→∅; / *$ An empty set of training datasets * /

$SR→$ A set of randomly generated tuples of SMOTE ratios for five rare classes;

$size→$ size of $SR$;

$k→$ the number of neighbors;

for $i→1$ to $size$ do

$TS_i←SMOTE(T, SR_i, k)$
(2) **Step 2:** Classification experiments are performed with TS and the validation dataset. The RMC is calculated with the classification results (recall values).

\[ V; / \ast validation\ dataset / \]

\[ Tuples\rightarrow A\ set\ of\ very\ large\ number\ of\ SMOTE\ ratios; \]

\[ BestTuple\rightarrow \arg \max (RModel(Tuples)); \]

(4.2) A genetic algorithm is used to search for an optimal solution. Randomly generated tuples of SMOTE ratios are used as Population, and the regression model is used as a fitness function. Figure 5 shows the distribution of the initial population of the genetic algorithm.

(5) **Step 5:** Generate a training dataset with the best tuple of SMOTE ratios. A training model is created by performing a classification experiment using the training dataset and the validation dataset. After inputting the test dataset to the model, the classification experiment results are derived.

\[ Test; / \ast test\ dataset / \]

\[ BestTrain\rightarrow SMOTE(T, BestTuple, k); \]

\[ TModel\rightarrow classifiers(BestTrain, V); \]

\[ BestResult\rightarrow TMode l(Test); \]

5. **Experiments**

Let \( m \) be the maximum number of experiments you can do with your computation device. We randomly generated \( m \) tuples of SMOTE ratios and generated \( m \) training datasets using the tuples. We obtained \( m \) recall values by applying classifiers such as SVM, decision tree, and random subspace to the generated training datasets and validation dataset. The RMC was calculated using the recall values of 10 classes. The dataset used to create the regression model consisted of RMC and the tuples for 5 rare classes, and the number of instances was \( m \). The regression algorithms used to generate the model were MLP regressor, SVR, and random forest. In previous studies [6], RMS was used, but in this study, RMC was used to reflect the characteristics of classes well and reduce the effect of outliers. (5) is an equation for calculating RMC.

We conducted two experiments with the regression model. In the first method, after randomly generating a very large number of tuples of SMOTE ratios, the tuples were input into the regression model to find the best tuple. The second method used generational GAs. The representation consisted of 5 real numbers, the tuples for 5 rare classes. The RMC was calculated using the regression model as the fitness function of the GA.

The following were the parameters of the GA: the population was set to \( m \), and the roulette wheel method was used for selection. One-point crossover was used with 100% probability, and bitwise mutation was set to 5%. Replacement left 1% of the superior solution and replaced the remaining 99% with a new child solution. The stop condition
occurred when the iteration of 10,000 generations ended or there was no change in the optimal value for more than 50 generations.

The parameters of the classifiers were as follows: in SVM, a polykernel was used, and the value of $c$ was 1. In the decision tree, the confidence factor was set to 0.25. REPTree was used as a classifier for random subspace, and subspaceSize was set to 0.5.

The parameters of the regressors were as follows: in the MLPRegressor, approximate sigmoid was used as the activation function, and the squared error was used as the loss function. In SVR, a polykernel was used, $c$ was set to 1, and the RegSMOImproved optimizer was used as the optimizer. The epsilon parameter was set to 0.001, and the tolerance was set to 0.001. In random forest, bagSizePercent was set to 100%, and numIterations were set to 100 times.

For a group of $n$ values involving $\{x_1, x_2, x_3, \ldots, x_n\}$, the RMC is given by

$$\text{RMC} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}$$  \hspace{1cm} (5)

Table 8 shows the performance of regression models generated from classification results using SVM, decision tree, and random subspace algorithms. The regression algorithms used to generate the model were MLP regressor, SVM, and random forest. In the SVM and random subspace experiments, the MLP regressor exhibited the best performance, and in the decision tree experiments, the random forest showed the best performance. Among all the experimental results, the experiment using the SVM classifier and random forest was the best. The performance of the experiment using the SVM classifier and random forest regressor was excellent.

Table 9 and Figure 6 show the results of the classification experiment using the original training dataset. The classification performance was compared between algorithms. The accuracy of the experiments is as follows: the random subspace is 89.12%, the decision tree is 97.4%, and the SVM is 97.06.

In the decision tree experiment, recall’s W. Avg. was the best at 0.974. The experimental results in Table 10 and Figure 7 show that the performance of D-S-1G was excellent. When the classification experiment was performed with a decision tree and the regression experiment was performed with SVR, SMOTE ratios were derived that exhibited good performance. Table 11 shows the best tuple of SMOTE ratios according to the type of experiment.

Each letter in the name of the test type in Table 10 has a specific meaning. The first letter is the algorithm used in the initial classification experiment: $D$ (decision tree), $R$ (random subspace), and $S$ (SVM). The second letter is the algorithm used to generate the regression model: $M$ (MLP regressor), $R$ (random forest), and $S$ (SVR). The number in the third position represents the classification experiment using the training dataset generated according to the ratios of the best tuple: 1 (random forest), 2 (SVM), 3 (decision tree), and 4 (k-NN). $G$ in the 4th position denotes that the genetic algorithm is used to find the best tuple of SMOTE ratios. For example, D-S-1G is the following meaning: $D=$ perform the decision tree classification experiment, $S=$ generate a regression model using the experimental results and SVR, 1 = the final classification experiment is performed using the generated training dataset and random forest, and $G =$ a genetic algorithm was used to obtain the best tuple of SMOTE ratios. The model is used to find the best tuple of SMOTE ratios, and a training dataset is generated according to the ratios of the best tuple. In Table 10, the accuracy of $S-R-1G$ is 96.70%, $R-R-2$ is 69.28%, $R-R-2G$ is 95.88%, and $D-S-1G$ is 96.55%.

| Classifiers | Measures | MLP regressor | SVR | Random forest |
|-------------|----------|---------------|-----|---------------|
| SVM         | Corr.    | 0.974         | 0.334 | 0.967        |
|             | MAE      | 0.003         | 0.013 | 0.003        |
|             | RMSE     | 0.004         | 0.018 | 0.005        |
|             | RAE      | 19.67         | 90.04 | 22.59       |
|             | RRQE     | 22.64         | 95.04 | 26.24       |
| Decision tree | Corr.    | 0.366         | 0.323 | 0.500        |
|             | MAE      | 0.019         | 0.018 | 0.020        |
|             | RMSE     | 0.028         | 0.029 | 0.026        |
|             | RAE      | 93.73         | 91.03 | 86.77        |
|             | RRQE     | 93.08         | 96.70 | 86.47        |
| Random subspace | Corr.   | 0.224         | 0.205 | 0.202        |
|              | MAE      | 0.042         | 0.041 | 0.042        |
|              | RMSE     | 0.047         | 0.048 | 0.047        |
|              | RAE      | 96.06         | 93.75 | 95.87        |
|              | RRQE     | 97.46         | 100.46 | 98.78      |

*Corr.: correlation coefficient; MAE: mean absolute error; RMSE: root mean squared error; RAE: relative absolute error; RRQE: root relative squared error.

| Class | Random subspace | Decision tree | SVM |
|-------|-----------------|---------------|-----|
| Normal | 0.985           | 0.986         | 0.986 |
| Exploits | 0.884           | 0.810         | 0.909 |
| Reconnaissance | 0.693       | 0.691         | 0.647 |
| DoS    | 0.283           | 0.402         | 0.000 |
| Class  |                 |               |     |
| Generic | 0.003           | 0.981         | 0.972 |
| Shellcode | 0.853           | 0.756         | 0.681 |
| Fuzzers | 0.893           | 0.893         | 0.708 |
| Worms  | 0.000           | 0.052         | 0.000 |
| Backdoor | 0.006         | 0.031         | 0.000 |
| Analysis | 0.103           | 0.030         | 0.027 |

| Result | Recall | 0.891 | 0.974 | 0.971 |
|--------|--------|-------|-------|-------|
| W. Precision | —   | 0.981 | —     | —     |
| Avg. F-measure | —   | 0.976 | —     | —     |
| RMC sub     | 1.112 | 1.324 | 1.190 |
| RMC sub     | 0.119 | 0.132 | 0.090 |
| Stdev sub   | 0.426 | 0.399 | 0.434 |
| Stdev sub   | 0.290 | 0.297 | 0.287 |
Stdev is the standard deviation between the recall values of all classes (10 classes), and $Stdev_5$ is the standard deviation between the recall values of rare classes. $RMC_{10}$ is the value calculated by applying the recall values of all classes to equation (2), and $RMC_5$ is the value calculated by applying the recall values of rare classes to equation (2). In this experiment, if the standard deviation is small and the value of $RMC_{10}$ or $RMC_5$ is large, then the classification performance is considered excellent.

Figure 8 compares the results of the decision tree experiment, which showed the best performance among experiments using the original training dataset, and the D-S-1G experiment, which showed the best performance using SMOTE. Comparing the two experimental results, we found that the experimental result of D-S-1G was superior to that of the decision tree experiment. There was a large difference in the classification performance for rare classes. Figure 8 shows the distribution of the initial population of the genetic algorithm in the D-S-1G experiment. Table 12 shows the results of the D-S-1G experiment for various measures.

The UNSW-NB15 dataset includes 2,540,047 instances. The training dataset includes 76,337 instances, and the validation dataset is the same. The test dataset has 846,683 instances. The SMOTE algorithm makes it easier to classify rare classes, as in [4]. The algorithm is used to randomly generate many tuples of SMOTE ratios. The tuples are used...
to create new train datasets. The classification experiments are performed using the training datasets and the validation datasets, and RMC values are derived from the experimental results. Recalls, the results of experiments, are used for RMC calculation. Recall is the ratio of correctly classified positive samples among the total positive samples, and it is mainly used to evaluate the classification performance. The reason is that when many normal samples are misclassified as attacks, the system is easily overloaded. Therefore, the classification results are mainly evaluated based on the recall metric, and the precision and F1-score are used as auxiliary metrics in evaluating the robustness of the machine learning model.

Next, a regression model was generated using the RMC values and the tuples of SMOTE ratios. In evaluating the performance of the regression model, the correlation coefficient is used as the default metric, and the MAE and RMSE are used as auxiliary metrics. There are two methods to optimize the ratios between classes using the regression model. First, create a very large number of tuples of SMOTE ratios randomly and enter the tuples in the regression model. Then, the best SMOTE ratios are chosen. The second is to use the regression model as the fitness function of GAs to derive the optimal SMOTE ratios. The reason for choosing GAs among evolutionary algorithms is that GAs are used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection between multiple individuals [5].

In Table 10, the D-S-1G experiment shows better performance than other experiments in the exploit, reconnaissance, ShellCode, fuzzers, and backdoor classes. In the experiment, the decision tree classifier is used for classification experiments, the results of the experiments are used to generate a dataset, and a regression model is generated using SVR and the dataset. Then, it is used in the GAs’ fitness function to find the best SMOTE ratios. A new training dataset is generated according to the best SMOTE ratios, and classification experiments are conducted by using the training dataset, the test dataset, and the random forest classifier.

The D-S-1G test results show smaller standard deviations compared to other experiments. That is, the classification

| Class          | Recall | D-S-1G | R-R-2G | S-R-1G |
|----------------|--------|--------|--------|--------|
| Normal         | 1.0    |        |        |        |
| Exploits       | 0.8    |        |        |        |
| Reconnaissance | 0.6    |        |        |        |
| DoS            | 0.4    |        |        |        |
| Generic        | 0.2    |        |        |        |
| Shellcode      | 0.0    |        |        |        |
| Fuzzers        |        |        |        |        |
| Worms          |        |        |        |        |
| Backdoor       |        |        |        |        |
| Analysis       |        |        |        |        |

Figure 7: Experimental results of train dataset preprocessed to the best tuple of SMOTE ratios, validation, and test dataset.

Table 11: Ratios of the best tuples (unit: percentage).

| Class          | Reconnaissance | DoS    | Worms  | Backdoor | Analysis |
|----------------|----------------|--------|--------|----------|----------|
| D-S-1G         | 80             | 14     | 14,106 | 8,900    | 127      |
| R-R-2G         | 1,352          | 177    | 102,609| 4,803    | 5,843    |
| S-R-1G         | 373            | 433    | 29,241 | 3,853    | 3,091    |
results for each class do not have bias and show good performance. The weighted average of the recall and F1-score is 0.965 and 0.971, respectively, which means that the classification performance is likely to be high. The S-R-1G combination includes an SVM classifier and a random forest regressor. Different from D-S-1G’s decision tree-based method, it shows excellent performance in DOS, generic, and worms classes. The R-R-2G is an experiment with a random subspace classifier and random forest regressor, and it shows excellent results in the analysis class.

A comprehensive analysis of the results shows that the type of classifier has a significant impact on the classification performance of each rare class. Therefore, it is expected that the classification performance can be improved by analyzing the classifiers that respond well to each rare class and then applying ensemble techniques such as boosting.

6. Concluding remarks

We studied machine learning-based data preprocessing methods for rare class classification using the UNSW-NB15 dataset, which has severe class imbalances. Only a small percent of the total datasets were used as the training dataset, without having a significant impact on the classification performance. We suggested how to optimize the ratio between classes in the training dataset with SMOTE and genetic algorithms. In the experiment, the optimal SMOTE ratios were found to maximize the classification performance. In the case of the D-S-1G experiment, which showed the best overall classification performance, the SMOTE ratios were 80 times, 14 times, 14,106 times, 8,900 times, and 127 times for reconnaissance, DoS, worms, backdoor, and analysis, respectively. In the case of the S-R-1G experiment, the SMOTE ratios were as follows: 373 times for reconnaissance, 433 times for DoS, 29,241 times for worms, 3,853 times for backdoor, and 3,091 times for analysis.

As a result, the best SMOTE ratio was obtained for rare classes, and the computation time was significantly reduced by generating a regression model. The superiority of the proposed method was verified through experiments, and the classification performance was enhanced by alleviating class importance. Each rare class showed very different classification results depending on the type of classifier.

In the future, it is expected that better results can be derived by applying ensemble methods such as boosting. We would like to present a new data preprocessing method to

| Classes    | Precision | Recall  | F-measure | ROC area |
|------------|-----------|---------|-----------|----------|
| Normal     | 1.000     | 0.986   | 0.993     | 0.998    |
| Exploits   | 0.486     | 0.718   | 0.580     | 0.982    |
| Reconnaissance | 0.954 | 0.704   | 0.810     | 0.957    |
| DoS        | 0.342     | 0.321   | 0.331     | 0.968    |
| Generic    | 1.000     | 0.897   | 0.946     | 0.996    |
| Shellcode  | 0.718     | 0.742   | 0.730     | 1.000    |
| Fuzzers    | 0.446     | 0.866   | 0.588     | 0.967    |
| Worms      | 0.254     | 0.534   | 0.344     | 0.981    |
| Backdoor   | 0.067     | 0.357   | 0.113     | 0.990    |
| Analysis   | 0.102     | 0.206   | 0.137     | 0.983    |
mitigate class imbalance with various data augmentation algorithms. We will also experiment with a few other network anomaly datasets.

**Data Availability**

The UNSW-NB15 data used to support the findings of this study have been deposited in the The UNSW-NB15 network data set repository (DOI: 10.1109/MilCIS.2015.7348942).

**Conflicts of Interest**

The author declares that he has no conflicts of interest.

**Acknowledgments**

This work was supported by National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2020R1F1A1070363) and Gwangju University 2022 funds.

**References**

[1] “Intrusion detection system,” https://en.wikipedia.org/wiki/Intrusion_detection_system.

[2] V. E. Adeyemo, A. Abdullah, N. Jhanjhi, M. Supramaniam, and A. O. Balogun, “Ensemble and deep-learning methods for two-class and multi-attack anomaly intrusion detection: an empirical study,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 9, 2019.

[3] V. Ponnusamy, M. Humayun, N. Jhanjhi, A. Yichiet, and M. Fahhad Almufareh, “Intrusion detection systems in internet of things and mobile Ad-Hoc networks,” *Computer Systems Science and Engineering*, vol. 40, no. 3, pp. 1199–1215, 2022.

[4] T. Gao, *Hybrid Classification Approach of SMOTE and Instance Selection for Imbalanced Datasets*, Iowa State University, Ames, Iowa, 2015.

[5] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts, United States, 1998.

[6] J.-H. Seo and Y.-H. Kim, “Machine-learning approach to optimize smote ratio in class imbalance dataset for intrusion detection,” *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 9704672, 11 pages, 2018.

[7] P. Soltanzadeh and M. Hashemzadeh, “RCSMOTE: range-Controlled synthetic minority over-sampling technique for handling the class imbalance problem,” vol. 542, pp. 92–111, Information Sciences, 2021.

[8] S. Bagui and K. Li, “Resampling imbalanced data for network intrusion detection datasets,” *Journal of Big Data*, vol. 8, no. 1, pp. 6–41, 2021.

[9] Z. Zoghi and G. Serpen, “UNSW-NB15 computer security dataset: analysis through visualization,” arXiv preprint arXiv: 2101.05067, 2021.

[10] S. Choudhary and N. Kesswani, “Analysis of KDD-Cup’99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT,” *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2020.

[11] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, “An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset,” *Cluster Computing*, vol. 23, no. 2, pp. 1397–1418, 2020.