The Journal of Korean Institute of Communications and Information Sciences '24-11 Vol.49 No.11
https://doi.org/10.7840/kics.2024.49.11.1510

Comparative Analysis of Oversampling Techniques and Feature Selection for Intrusion Detection

Minkyung Kim◆

ABSTRACT

To accurately detect and defend against ever-evolving cyber-attacks, network security technologies using artificial intelligence are continually advancing. This study analyzed the effective network intrusion detection methods based on the CICIDS2017 dataset, which contains various types of network attacks and has a highly imbalanced class distribution. To enhance detection performance for the minority classes of attacks, five oversampling techniques, including SMOTE, Borderline-SMOTE, ADASYN, GAN, and BiGAN, were applied to the underrepresented Bot and Infiltration classes. Additionally, the impact of feature selection on classification performance was evaluated by selecting features based on the feature importance scores from each machine learning model: Random Forest and XGBoost. The experimental results demonstrated that oversampling with SMOTE and ADASYN improved the recall scores of minority classes. Furthermore, applying feature selection reduced the model's complexity while maintaining or even improving its accuracy.

Key Words : Intrusion Detection, Class Imbalanced Data, Oversampling, Feature Selection

Ⅰ. Introduction

With the rise of malicious network traffic and unauthorized access attempts in large-scale environments, the development of effective Intrusion Detection Systems (IDS) has become increasingly critical. IDS monitor network traffic in real time and distinguish between benign and malicious traffic while providing essential information for defense[1]. In recent years, the advancements in artificial intelligence have led to a growing focus on applying machine learning techniques to enhance intrusion detection performance. However, the data imbalance in both real-world scenarios and training datasets presents a challenge, as certain attack types occur much less frequently, making their detection particularly difficult. Furthermore, standard models are generally designed to optimize overall classification accuracy, leading to poor detection performance for minority attack classes due to their imbalanced distribution[2]. Therefore, it is crucial to develop approaches that enhance detection performance for minority attack classes while also improving overall classification accuracy.

In this study, we used the CICIDS2017 dataset, which includes diverse attack traffic reflecting recent and widely observed patterns. However, this dataset presents significant data imbalance in certain attack classes and requires efficient preprocessing due to its large scale and high dimensionality. To address these challenges, we refined the dataset by reducing it to 68 features and reclassifying the 14 attack types into 6 new labels. The minority classes such as Bot and Infiltration constitute less than 0.07% of the total dataset. To address this imbalance, we applied five oversampling techniques: SMOTE, Borderline-SMOTE, ADASYN, GAN, and BiGAN. Additionally, feature selection was applied using the feature importance scores from both the Random Forest and XGBoost models to improve learning efficiency and performance. The effectiveness of each method was

◆ First Author : Dasan University College, Ajou University, dendlonglove@ajou.ac.kr, 정회원
논문번호 : 202408-189-A-RU, Received August 25, 2024; Revised September 21, 2024; Accepted September 23, 2024

then analyzed to mitigate the curse of dimensionality and enhance model performance by removing low-importance features.

The remainder of this paper is organized as follows. Section 2 provides an overview of the dataset and experimental workflow. This section also describes the oversampling techniques and machine learning models applied in the experiments, along with a discussion of related work. Section 3 presents the analysis of the experimental results. Finally, Section 4 outlines the conclusions and addresses future research directions.

Ⅱ. Data Preparation and Methodology

2.1 Dataset and Preprocessing

The CICIDS2017 dataset[3,4], used in this experiment, is an intrusion detection evaluation dataset released by the Canadian Institute for Cybersecurity in 2017. It contains 78 features and 15 classes, including 14 attack types and a benign class, based on contemporary attack data that closely represent real-world network environments. However, the dataset contains some missing and duplicate values. Since model accuracy depends on the quality of input data, effective preprocessing is essential to balance the dataset and improve performance by removing unnecessary and duplicate data, as well as adjusting labels.

To eliminate null values, redundancy, and any potential noise from the dataset, we first removed the unnecessary 'DestinationPort' column and the duplicate 'FwdHeaderLength.1' column. Additionally, eight columns where all values were identical across rows were removed. Next, we eliminated rows containing Null, NaN, or positive infinity values in the 'FlowBytes/s' and 'FlowPackets/s' columns. This preprocessing resulted in a refined dataset with 68 features and 2,827,876 samples. As the final step, the 14 attack types in the label column were reclassified into 6 attack classes using the new labeling method adopted by Kurniabudi et al.[5], as shown in Table 1. This reclassification consolidates attack types with similar patterns and behaviors to improve model generalization and address data imbalance. After preprocessing, it was crucial to verify that the dataset is balanced. Specifically, the Bot class was found to

Table 1. Number of training and test data with new labels

Original Labels	New Labels	Training data	Test data	Total
BENIGN	Benign	1,589,924	681,396	2,271,320
PortScan	Port Scan	111,163	47,641	158,804
Bot	Bot	1369	587	1,956
Infiltration	Infiltration	25	11	36
Brute Force	Web Attack	1,526	654	2,180
Sql Injection				
XSS				
FTP-Patator	Brute Force	9,682	4,150	13,832
SSH-Patator				
DDoS	DDos/Dos	265,824	113,924	379,748
DoS GoldenEye				
DoS Hulk				
DoS Slowhttptest				
DoS slowloris				
Heartbleed				
Total		1,979,513	848,363	2,827,876

consist of 1,956 samples out of the total 2,827,876, while the Infiltration class had only 36 samples.

The overall experimental procedure for intrusion detection is shown in Fig. 1, based on the CICIDS2017 dataset. The preprocessed dataset was split into 70% training data and 30% testing data. First, the performance of the preprocessed training dataset was assessed using the Random Forest and XGBoost classification models. Although the overall accuracy of the classification models was generally high, it was observed that predictions for the minority classes remained inaccurate. As ensuring class balance is crucial, five different oversampling techniques were applied to the Bot and Infiltration classes. The goal was to balance the dataset and assess how oversampling affects overall improvement. To further enhance the efficiency of learning, feature selection was applied based on the feature importance scores determined by the Random Forest and XGBoost ensemble models. Finally, the performance of the oversampled datasets was compared with the results after applying both oversampling and feature selection to identify the most effective model-building approach.

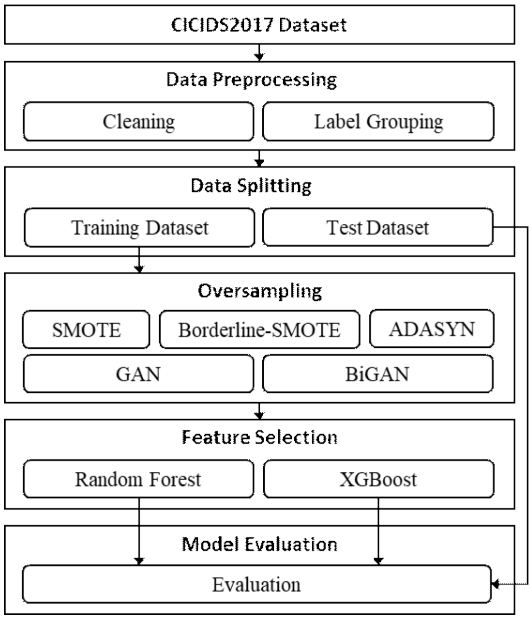

Fig. 1. Experimental workflow of the proposed methods

2.2 Oversampling Techniques

To address the imbalance in an intrusion detection evaluation dataset, oversampling techniques can be used to increase the number of samples in minority classes. These techniques involve generating new synthetic samples from existing minority class data or randomly duplicating samples. In this study, oversampling techniques were applied to address the issue of models underperforming on minority classes due to insufficient training data. This imbalance can lead to a biased model towards majority classes, resulting in low detection rates for minority classes and higher false positive rates. By applying oversampling techniques, the dataset becomes more balanced, enabling the model to learn effectively from all classes and thus improving its overall performance. In this experiment, we applied five different oversampling techniques to the Bot and Infiltration classes to improve class balance.

A widely used and effective oversampling technique is the Synthetic Minority Oversampling Technique (SMOTE)[6]. SMOTE selects a reference sample and identifies its k-nearest neighbors, then generates new samples by interpolating between the reference sample and these neighbors, effectively filling the space between them[7]. This approach helps create a more balanced distribution of samples. However, according to Elreedy et al.[2], while SMOTE improves classification performance, a gap still exists compared to using a fully balanced dataset. One of the limitations of SMOTE is its difficulty in handling samples that lie near the borderline, where minority and majority class samples overlap.

To overcome this limitation, Han et al. proposed Borderline-SMOTE, an enhancement of the SMOTE that specifically oversamples minority class samples near the decision boundary[8]. This technique improves the model's performance in classifying minority samples close to the majority class. Experimental results show that Borderline-SMOTE outperforms SMOTE, achieving better True Positive Rate (TPR) and F1-score. Furthermore, a method was proposed that combines denoising with Borderline-SMOTE to remove noise and prevent the generation of noisy samples[9].

Haibo et al. proposed a novel adaptive synthetic sampling approach called Adaptive Synthetic Sampling (ADASYN) to enhance learning from imbalanced datasets[10]. ADASYN extends SMOTE by generating new samples in a density-weighted manner, producing more samples near the majority class and fewer for those farther away. Dey et al.[11] showed that while ADASYN generates more samples near the decision boundary, it doesn't focus exclusively on these boundary samples like Borderline-SMOTE. This approach more effectively balances the distribution based on each sample's position and, as a result, improves overall performance.

The fourth oversampling technique used in the experiment is the Generative Adversarial Network (GAN), which employs two neural networks, the generator and the discriminator, that compete to create new samples[12]. The generator produces samples from random noise, while the discriminator distinguishes between real and generated data. Through training, the generator learns to create samples that closely resemble the real ones. Lee et al.[13] used GAN to address class imbalance in CICIDS2017 dataset and found that it outperformed other techniques with the Random Forest model. Liu et al.[1] combined GAN with ANOVA-based feature selection on datasets such as NSL-KDD, UNSW-NB15, and CICIDS2017, showing the significant performance enhancements. Additionally, Zareapoor et al.[14] introduced the Minority Oversampling Generative Adversarial Network (MoGAN), which improved classification and fault detection performance.

Finally, the Bidirectional Generative Adversarial Network (BiGAN) is an extension of GAN that incorporates an encoder in addition to the generator and discriminator[15]. The encoder learns a bidirectional mapping between the input data and the generated data by capturing representations in the latent space, enabling the generation of higher-quality data. BiGAN provides a deeper understanding of the underlying structure of the data, allowing for the creation of more sophisticated and diverse samples. Furthermore, Zhang et al.[16] introduced BiGAN-based training structures to improve the effectiveness of minority class oversampling. This study aims to compare the effects of these five oversampling techniques on classification performance.

2.3 Feature Selection

This study employs Random Forest and XGBoost as the primary classification models. These two models were selected for their effectiveness in achieving optimal classification performance through the evaluation of feature importance and the use of ensemble learning methods. By identifying the most relevant features, both models enable effective feature selection, especially after applying oversampling techniques. Random Forest[17] is an ensemble learning method that uses multiple decision trees for classification. It combines predictions from randomly constructed individual trees to produce a final output and determines feature importance by evaluating the reduction in impurity when each feature is used at the node. Features with higher importance are then selected for classification. This approach helps prevent overfitting in individual trees and ensures predictive performance by effectively capturing key features in the data. Li et al.[18] found that employing Random Forest for feature selection, followed by prediction with an Auto-Encoder capturing underlying data structures, significantly reduces detection time and improves accuracy.

Extreme Gradient Boosting (XGBoost)[19] is another ensemble learning method based on decision trees that uses gradient boosting to sequentially train multiple trees, where each tree corrects the errors of the previous one. It calculates feature importance by measuring how often each feature is used at the nodes and evaluating its contribution to improving model performance. This process helps XGBoost identify the most important features, enabling more refined feature selection and improving predictive accuracy. Its built-in regularization also controls model complexity and prevents overfitting to ensure better generalization to new data.

Devan et al.[20] proposed an XGBoost-DNN model that applies XGBoost for feature selection and then uses a Deep Neural Network (DNN) for network intrusion classification with the NSL-KDD dataset. Their study showed that feature selection reduces computational complexity while improving both algorithm performance and data generalization. In cases of severe imbalance, this suggests that feature selection for dimensionality reduction may be more effective in enhancing classification performance. Tsai et al.[21] experimented with 10 imbalanced datasets, comparing the performance of combining ensemble feature selection methods with the SMOTE oversampling algorithm. They found that applying feature selection with XGBoost before SMOTE oversampling achieved the best results, outperforming both the reverse order and feature selection alone.

In this study, we chose to apply feature selection after oversampling, as performing it first could exclude important features containing critical information about the minority class. In preliminary tests conducted with the CICIDS2017 dataset across various scenarios, we found that feature selection is more effective when applied after oversampling. Applying feature selection after oversampling the minority class helps maintain the key features of the minority class while balancing the class distribution. As a result, feature selection using Random Forest and XGBoost was applied to the oversampled data to evaluate the efficiency of this approach.

Ⅲ. Experimental Results and Analysis

In this experiment, five oversampling techniques were applied to a preprocessed dataset containing 68 features to evaluate the performance. Table 2 shows the performance evaluation results on the training dataset using Random Forest and XGBoost before applying oversampling. The hyperparameter 'n_estimators', controlling the number of decision trees generated, was set to 100 for both algorithms. While both algorithms demonstrate high overall accuracy, the recall score is lower in the Bot and Infiltration classes due to class imbalance. In network security, reliable detection of critical attacks is crucial. This study focuses on identifying oversampling techniques that help improve the recall score.

In the first stage of this experiment, five different oversampling techniques were applied to address class imbalance by increasing the Bot class from 1,369 to 11,369 samples and the Infiltration class from 25 to 10,025 samples. The classification performance was then evaluated using Random Forest and XGBoost on the oversampled training datasets generated by each technique. In the second stage, feature selection based on importance measured by Random Forest and XGBoost was conducted to enable faster and more efficient learning while improving classification performance. The oversampled data was sorted by importance, and feature importance was visualized in Fig. 2 and 8. Accuracy was then compared across models with different numbers of selected features, as shown in Fig. 3 – 7 for Random Forest (RF) and Fig. 9 – 13 for XGBoost (XGB). Based on this comparison, the models were then evaluated using the feature set with the highest accuracy. This approach ensured that the final models used only the most relevant and significant features for effective performance evaluation. Since feature importance differs between Random Forest and XGBoost, each dataset was evaluated using its corresponding model.

Table 2. Performance evaluation of Random Forest and XGBoost on preprocessed data (68 features)

Random Forest on preprocessed data (68 features) – Accuracy: 99.885%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.85	0.80	0.82	587
Infiltration	1.00	0.64	0.78	11
Web Attack	0.99	0.93	0.96	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
XGBoost on preprocessed data (68 features) – Accuracy: 99.904%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.88	0.77	0.82	587
Infiltration	1.00	0.64	0.78	11
Web Attack	0.98	0.97	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

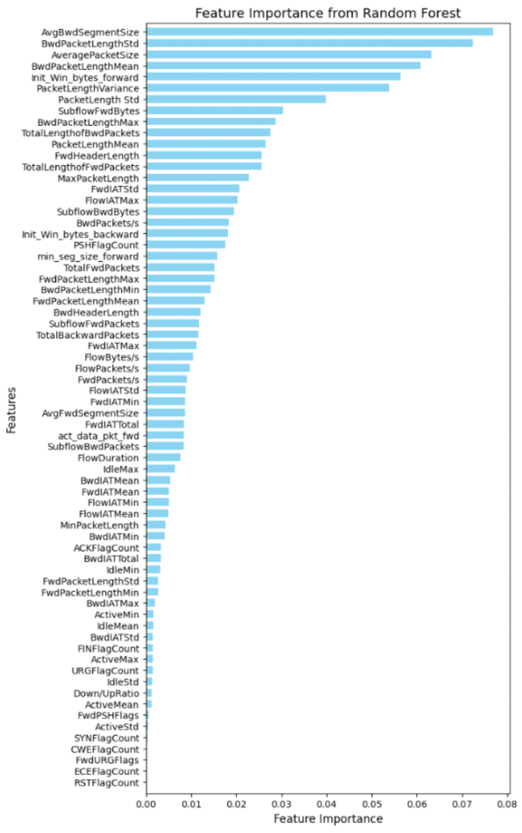

Fig. 2. Feature importance from Random Forest

3.1 Random Forest

3.1.1 SMOTE and Feature Selection with RF

The performance evaluation results of oversampling and feature selection using the Random Forest model are first presented. SMOTE was applied to oversample the Bot and Infiltration classes, and the resulting dataset was used to train the Random Forest model. The model's performance was then evaluated on the test dataset and is presented in Table 3.

As seen from comparing Table 2 and the first result of Table 3, overall accuracy slightly decreased after oversampling as a trade-off, but performance for the minority classes improved. Although the precision score for the Bot class decreased, the model correctly identified more Bot attacks overall. The recall score for Bot class increased from 0.80 to 0.88, and from 0.64 to 0.73 for Infiltration class. The precision score measures the proportion of correctly identified positive cases out of all cases predicted as positive, whereas the recall score measures the proportion of actual positive cases that were correctly identified by the model.

Table 3. Performance evaluation of Random Forest on SMOTE oversampled data: 68 features vs. 20 features

Random Forest on SMOTE oversampled data (68 features) – Accuracy: 99.877%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.70</u>	<u>0.88</u>	<u>0.78</u>	587
Infiltration	<u>1.00</u>	<u>0.73</u>	<u>0.84</u>	11
Web Attack	0.99	0.93	0.96	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
Random Forest on SMOTE oversampled data (20 features) – Accuracy: 99.891%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.71</u>	<u>0.90</u>	<u>0.79</u>	587
Infiltration	<u>1.00</u>	<u>0.82</u>	<u>0.90</u>	11
Web Attack	0.99	0.97	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

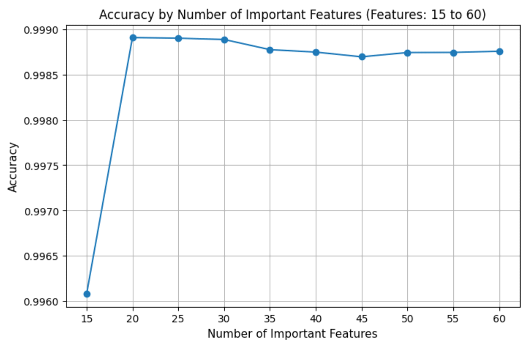

Fig. 3. Accuracy by number of important features from Random Forest (SMOTE)

The feature importance scores calculated using the Random Forest model were analyzed and visualized in descending order, and are shown in Fig. 2. Based on this ranking, feature selection was conducted, and model performance was evaluated as the number of selected features varied. Fig. 3 shows the evaluation of model accuracy as the number of top features increased in increments of 5 from 15 to 60. The highest accuracy was observed when the model was evaluated using the top 20 features. Consequently, the performance evaluation of the Random Forest model on the test data using these top 20 features is presented as the second result of Table 3.

The first result in Table 3 shows the oversampled data with 68 features, while the second result represents the data after selecting the top 20 features. The accuracies are 99.877% and 99.891%, respectively, showing a slight improvement after feature selection. The recall score improved in the Bot and Infiltration classes after SMOTE oversampling, with the Bot class increasing from 0.88 to 0.90 and the Infiltration class increasing from 0.73 to 0.82. The recall score for Web Attack also showed a slight increase, while performance in the remaining classes remained stable. Random Forest calculates feature importance by averaging the importance scores of features used for splits across all trees. The model performed well with the top 20 features because they contain the key information required for accurate predictions. Thus, reducing dimensionality through feature selection lowers model complexity, which makes training more efficient and helps to maintain or slightly improve performance.

3.1.2 Borderline-SMOTE and Feature Selection with RF

In the second experiment, Borderline-SMOTE was applied to oversample the Bot and Infiltration classes. The model's performance was evaluated on the oversampled data, and the impact of feature selection on accuracy was analyzed. Fig. 4 shows that the top 25 features resulted in the best performance. Comparing the results in Table 2 with the first results in Table 4, Borderline-SMOTE produced a recall increase for the Bot and Infiltration classes similar to that achieved by SMOTE. However, after feature selection, the recall score for Bot class increased by only 0.01, while there was no improvement in the Infiltration class. While Borderline-SMOTE contributed to some improvements, its performance was less effective compared to SMOTE.

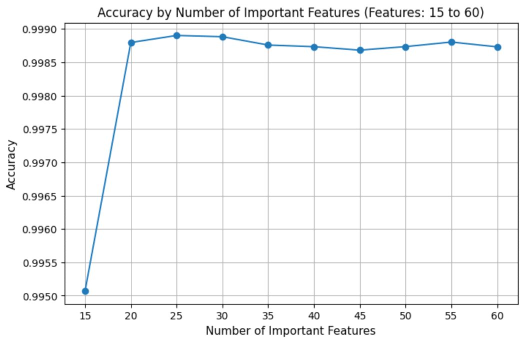

Fig. 4. Accuracy by number of important features from Random Forest (Borderline-SMOTE)

3.1.3 ADASYN and Feature Selection with RF

For the third experiment, Bot and Infiltration classes were oversampled using ADASYN, which was followed by an evaluation of the model's performance. The precision score for the Bot class decreased from 0.85 to 0.67, but the recall score increased from 0.80 to 0.90, as shown in Table 5. In the Infiltration class, the recall score rose from 0.64 to 0.73. These results are similar to the recall improvements observed with SMOTE. These results indicate

Table 4. Performance evaluation of Random Forest on Borderline-SMOTE oversampled data: 68 features vs. 25 features

Random Forest on Borderline-SMOTE oversampled data (68 features) – Accuracy: 99.874%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.70	0.89	0.78	587
Infiltration	1.00	0.73	0.84	11
Web Attack	0.99	0.93	0.96	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
Random Forest on Borderline-SMOTE oversampled data (25 features) – Accuracy: 99.890%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.70	0.90	0.79	587
Infiltration	1.00	0.73	0.84	11
Web Attack	0.99	0.97	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

Table 5. Performance evaluation of Random Forest on ADASYN oversampled data: 68 features vs. 25 features

Random Forest on ADASYN oversampled data (68 features) – Accuracy: 99.873%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.67	0.90	0.77	587
Infiltration	1.00	0.73	0.84	11
Web Attack	0.99	0.93	0.96	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
Random Forest on ADASYN oversampled data (25 features) – Accuracy: 99.885%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.66	0.92	0.77	587
Infiltration	1.00	0.73	0.84	11
Web Attack	0.98	0.96	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

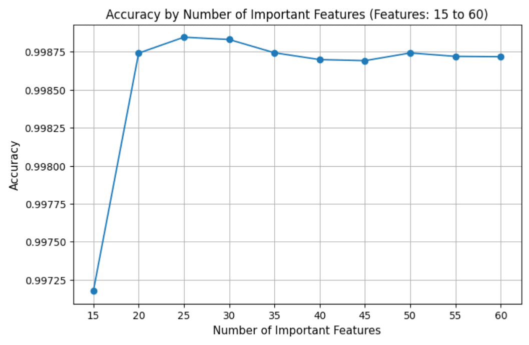

Fig. 5. Accuracy by number of important features from Random Forest (ADASYN)

that while precision may be sacrificed, the ability to accurately identify minority classes has improved. The changes in accuracy depending on the number of features are shown in Fig. 5.

3.1.4 GAN and Feature Selection with RF

In this fourth experiment, GAN, an oversampling technique using neural networks to generate new data, was applied to the Bot and Infiltration classes. After oversampling, the performance of the Random Forest model was evaluated. Compared to the pre-oversampling results in Table 2, the recall score for the Infiltration class increased from 0.64 to 0.73. Unlike the improvements observed in the Bot class with SMOTE, Borderline-SMOTE, and ADASYN, the GAN oversampling technique resulted in minimal to no improvement. This suggests that the data generated by GAN may lack valuable information or contain redundant data, leading to no significant performance enhancement. The results for the top 25 features are

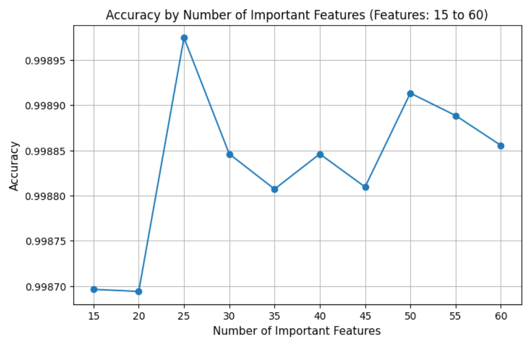

Fig. 6. Accuracy by number of important features from Random Forest (GAN)

Table 6. Performance evaluation of Random Forest on GAN oversampled data: 68 features vs. 25 features

Random Forest on GAN oversampled data (68 features) – Accuracy: 99.884%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.84</u>	<u>0.80</u>	<u>0.82</u>	587
Infiltration	<u>1.00</u>	<u>0.73</u>	<u>0.84</u>	11
Web Attack	0.99	0.93	0.96	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
Random Forest on GAN oversampled data (25 features) – Accuracy: 99.897%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.84</u>	<u>0.80</u>	<u>0.82</u>	587
Infiltration	<u>1.00</u>	<u>0.73</u>	<u>0.84</u>	11
Web Attack	0.99	0.96	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

presented in the second result of Table 6. Fig. 6 shows the accuracy differences based on the number of features.

Performance evaluation with the top 25 selected features after oversampling showed only slight differences in the Bot and Infiltration classes compared to the first results in Table 6. However, there was a slight increase in the recall score for the Web Attack class, leading to an overall improvement in accuracy. Therefore, in the fifth experiment, we utilized BiGAN, which incorporates an encoder to better capture the latent structure of the data during oversampling.

3.1.5 BiGAN and Feature Selection with RF

After oversampling Bot and Infiltration classes using BiGAN, the model's performance was evaluated, and feature selection was applied. The results for the top 30 features ranked according to accuracy are shown in Table 7, with accuracy variations based on the number of features presented in Fig. 7.

When comparing BiGAN with the basic GAN, the recall and precision scores for each class differed by approximately 0.01. The results from BiGAN oversampling and feature selection were comparable to those from GAN. Although BiGAN includes the ability to bidirectionally transform data through an encoder, this addition appears to have had minimal impact on the quality or diversity of the generated data. Further experiments are needed to refine the architectures and hyperparameters of both GAN and BiGAN beyond the current configurations.

In the Random Forest model, oversampling with SMOTE and ADASYN led to slight improvements in both the Bot and Infiltration classes. Evaluating the model with the top features showed that performance was maintained or even improved despite reduced dimensionality. This demonstrates that feature selection is an effective method for reducing model complexity and training time while maintaining or enhancing generalization performance.

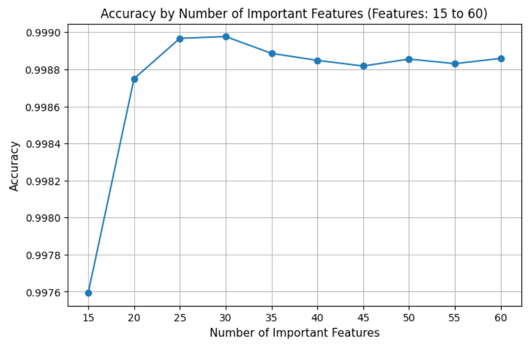

Fig. 7. Accuracy by number of important features from Random Forest (BiGAN)

Table 7. Performance evaluation of Random Forest on BiGAN oversampled data: 68 features vs. 30 features

Random Forest on BiGAN oversampled data (68 features) – Accuracy: 99.890%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.86</u>	<u>0.80</u>	<u>0.83</u>	587
Infiltration	<u>1.00</u>	<u>0.73</u>	<u>0.84</u>	11
Web Attack	0.99	0.94	0.96	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
Random Forest on BiGAN oversampled data (30 features) – Accuracy: 99.898%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.85</u>	<u>0.80</u>	<u>0.82</u>	587
Infiltration	<u>1.00</u>	<u>0.73</u>	<u>0.84</u>	11
Web Attack	0.99	0.96	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

3.2 XGBoost

3.2.1 SMOTE and Feature Selection with XGB

While the previous experiments used the Random Forest model, this section evaluates the XGBoost model with the same five oversampling techniques and feature selection. First, the Bot and Infiltration classes were oversampled using SMOTE, and the

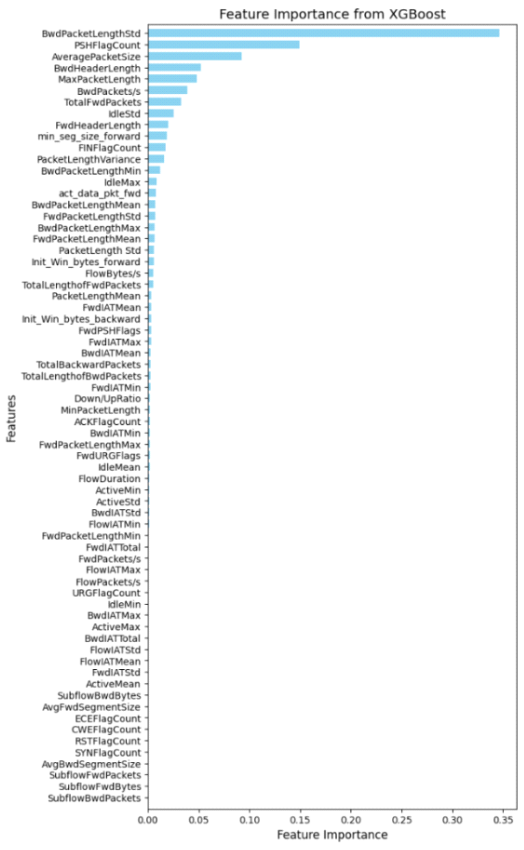

Fig. 8. Feature importance from XGBoost

XGBoost model was trained and evaluated on the test data. Fig. 8 shows that only a few features have high importance in XGBoost, with a significant drop-off beyond the top 25.

As shown in Fig. 9, performance was also assessed using the top 55 features ranked by accuracy. Based on this feature importance result, it seemed at first that XGBoost would perform well with fewer features. However, unlike Random Forest, XGBoost achieved better performance with a larger number of features. When oversampling was applied to the Bot and Infiltration classes, the recall score for the Bot class improved from 0.80 to 0.96, as shown in Table 2 and the first result of Table 8. Similarly, the recall score for the Infiltration class also showed an improvement, increasing from 0.64 to 0.82, which indicates a positive effect. After calculating feature importance and comparing accuracy using the top features, reducing the number of features to 55 still maintained performance similar to that of the full oversampled dataset, as shown in Table 8.

Table 8. Performance evaluation of XGBoost on SMOTE oversampled data: 68 features vs. 55 features

XGBoost on SMOTE oversampled data (68 features) – Accuracy: 99.891%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.67</u>	<u>0.96</u>	<u>0.79</u>	587
Infiltration	<u>1.00</u>	<u>0.82</u>	<u>0.90</u>	11
Web Attack	0.98	0.97	0.97	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
XGBoost on SMOTE oversampled data (55 features) – Accuracy: 99.892%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.68</u>	<u>0.96</u>	<u>0.79</u>	587
Infiltration	<u>1.00</u>	<u>0.82</u>	<u>0.90</u>	11
Web Attack	0.98	0.98	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

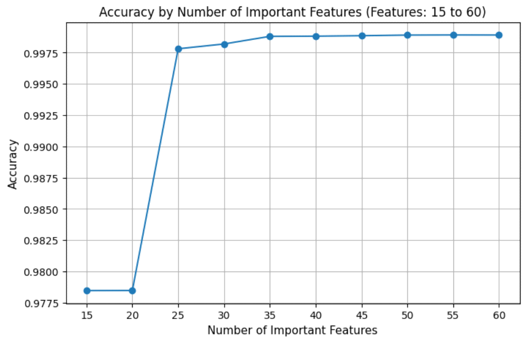

Fig. 9. Accuracy by number of important features from XGBoost (SMOTE)

3.2.2 Borderline-SMOTE and Feature Selection with XGB

In this experiment, Borderline-SMOTE oversampling was applied to Bot and Infiltration classes. As shown in Table 9, the model was then assessed using the top 55 features selected based on the feature importance scores. A comparison of accuracy based on the number of features is presented in Fig. 10.

While Borderline-SMOTE increases the recall

Table 9. Performance evaluation of XGBoost on Borderline-SMOTE oversampled data: 68 features vs. 55 features

XGBoost on Borderline-SMOTE oversampled data (68 features) – Accuracy: 99.886%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.67</u>	<u>0.95</u>	<u>0.79</u>	587
Infiltration	<u>1.00</u>	<u>0.73</u>	<u>0.84</u>	11
Web Attack	0.98	0.98	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
XGBoost on Borderline-SMOTE oversampled data (55 features) – Accuracy: 99.886%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.67</u>	<u>0.95</u>	<u>0.79</u>	587
Infiltration	<u>1.00</u>	<u>0.73</u>	<u>0.84</u>	11
Web Attack	0.98	0.98	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

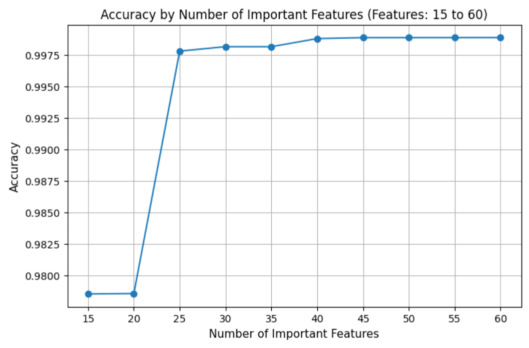

Fig. 10. Accuracy by number of important features from XGBoost (Borderline-SMOTE)

scores for the Bot and Infiltration classes, the improvement is less noticeable compared to SMOTE. This technique specifically focuses on borderline samples of the minority class, which can enhance recall for these classes. However, this approach may reduce the representation of typical samples, potentially leading to lower overall performance than with SMOTE or ADASYN. This suggests that although the recall scores for specific classes may improve, the overall performance gains could be limited.

3.2.3 ADASYN and Feature Selection with XGB

In this third experiment, ADASYN oversampling was applied to the Bot and Infiltration classes. Performance was evaluated using the top 55 features, with the results presented in Table 10 and accuracy comparisons shown in Fig. 11. The recall scores for the Bot and Infiltration classes with ADASYN were comparable to those obtained with SMOTE. Since

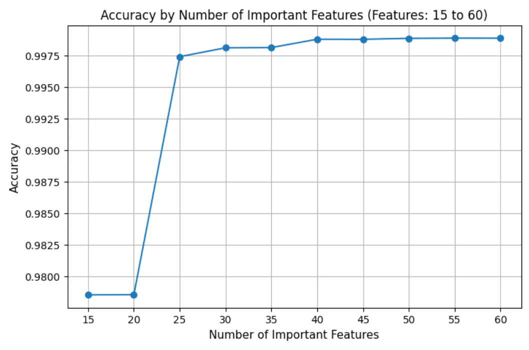

Fig. 11. Accuracy by number of important features from XGBoost (ADASYN)

Table 10. Performance evaluation of XGBoost on ADASYN oversampled data: 68 features vs. 55 features

XGBoost on ADASYN oversampled data (68 features) – Accuracy: 99.886%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.64</u>	<u>0.97</u>	<u>0.77</u>	587
Infiltration	<u>1.00</u>	<u>0.82</u>	<u>0.90</u>	11
Web Attack	0.97	0.97	0.97	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
XGBoost on ADASYN oversampled data (55 features) – Accuracy: 99.887%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	<u>0.64</u>	<u>0.97</u>	<u>0.77</u>	587
Infiltration	<u>1.00</u>	<u>0.82</u>	<u>0.90</u>	11
Web Attack	0.98	0.97	0.97	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

both SMOTE and ADASYN provide generally balanced oversampling, the performance differences between these techniques are minimal. Both SMOTE and ADASYN demonstrated improved performance by improving the recall scores of the minority classes, which effectively improves the identification of these classes.

3.2.4 GAN and Feature Selection with XGB

In the previously used Random Forest model, performance evaluation after GAN oversampling showed an improvement in the recall score only for the Infiltration class. Similarly, feature selection on the oversampled data also led to an improvement in the recall score for the Infiltration class. However, Fig. 12 and Table 11 show that with the XGBoost model, oversampling led to a decrease in the recall scores for both Bot and Infiltration classes, resulting in an overall decline in performance. Evaluating performance with the top 45 features that provided the highest accuracy revealed no meaningful improvements.

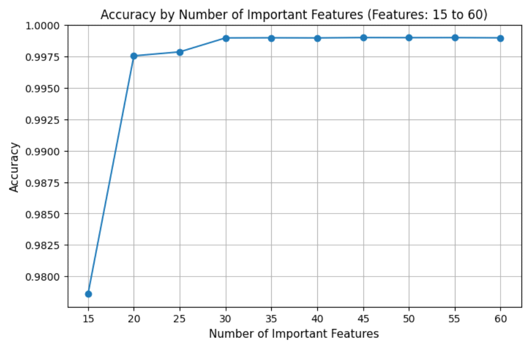

Fig. 12. Accuracy by number of important features from XGBoost (GAN)

Table 11. Performance evaluation of XGBoost on GAN oversampled data: 68 features vs. 45 features

XGBoost on GAN oversampled data (68 features) – Accuracy: 99.901%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.92	0.75	0.82	587
Infiltration	1.00	0.55	0.71	11
Web Attack	0.98	0.97	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
XGBoost on GAN oversampled data (45 features) – Accuracy: 99.902%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.93	0.74	0.83	587
Infiltration	1.00	0.64	0.78	11
Web Attack	0.98	0.97	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

3.2.5 BiGAN and Feature Selection with XGB

Lastly, after applying BiGAN to the Bot and Infiltration classes, the performance remained nearly the same as it was before oversampling. The results in Table 12 show that using the top 45 features, which achieved the highest accuracy in Fig. 13, led to a decrease in the recall score for the Bot class, while the recall score for the Infiltration class slightly increased. Similar to the previous GAN experiment, the performance improvement for both GAN and BiGAN was limited.

To summarize all the experimental results, both SMOTE and ADASYN were effective in enhancing the performance of the Bot and Infiltration classes with Random Forest and XGBoost models. These oversampling techniques improved the recall scores for these classes. Additionally, feature selection using the top-ranked features by feature importance resulted

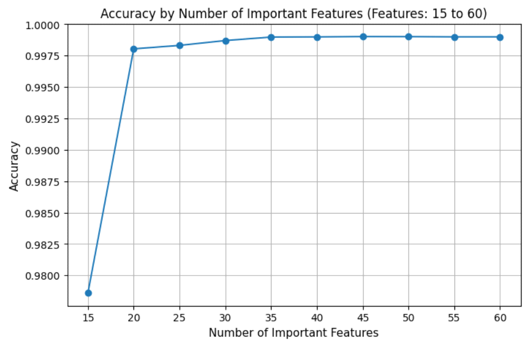

Fig. 13. Accuracy by number of important features from XGBoost (BiGAN)

Table 12. Performance evaluation of XGBoost on BiGAN oversampled data: 68 features vs. 45 features

XGBoost on BiGAN oversampled data (68 features) – Accuracy: 99.900%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.89	0.76	0.82	587
Infiltration	1.00	0.64	0.78	11
Web Attack	0.98	0.97	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924
XGBoost on BiGAN oversampled data (45 features) – Accuracy: 99.902%				
	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	681,396
Port Scan	0.99	1.00	1.00	47,641
Bot	0.93	0.74	0.83	587
Infiltration	1.00	0.73	0.84	11
Web Attack	0.98	0.98	0.98	654
Brute Force	1.00	1.00	1.00	4,150
DDos/Dos	1.00	1.00	1.00	113,924

in either maintaining or improving model performance. Particularly, feature selection with Random Forest reduced the number of features to 20 for SMOTE and 25 for ADASYN, while XGBoost required 55 features for both techniques to achieve comparable performance. Despite this reduction, performance was maintained, demonstrating that feature selection can significantly enhance the efficiency of the learning process.

Ⅳ. Conclusion and Future Work

This study aimed to improve the performance of network intrusion detection systems (IDS) by applying various oversampling techniques and analyzing the effectiveness of feature selection methods. The CICIDS2017 dataset used in this experiment exhibits class imbalance, with some attack classes being under-represented, which may degrade the performance of the classification model. To enhance the classification performance of the minority Bot and Infiltration classes, five oversampling techniques, including SMOTE, Borderline-SMOTE, ADASYN, GAN, and BiGAN, were applied to address the class imbalance. The performance was then evaluated using the Random Forest and XGBoost models. The results showed that SMOTE and ADASYN were particularly effective in improving the recall scores of the minority classes, indicating that the model's ability to detect these classes was enhanced.

Subsequent feature selection experiments revealed that selecting approximately one-third of the most important features from the total 68 features in Random Forest resulted in comparable or slightly improved overall accuracy. This finding suggests that removing less important features can reduce the model's complexity and shorten training time while maintaining or enhancing performance. XGBoost, on the other hand, required around two-thirds of the total features to maintain or improve performance after feature selection. This approach can significantly improve data processing efficiency in environments that require real-time handling of large data volumes. In conclusion, this study demonstrates that employing various oversampling techniques and feature selection methods can effectively enhance the performance of network intrusion detection systems.

In future research, comprehensive fine-tuning experiments will be conducted on oversampling techniques using GAN and BiGAN, as these approaches have not yet shown significant performance improvements. To improve the effectiveness of neural network-based oversampling, it will be essential to employ fine-tuning methods that optimize classification models. The goal is to further enhance performance by applying detailed optimization methods to the data distribution of various network attack types.

References

[1] X. Liu, T. Li, R. Zhang, D. Wu, Y. Liu, and Z. Yang, "A GAN and feature selection-based oversampling technique for intrusion detection," *Secur. and Commun. Netw.*, pp. 1-15, Jul. 2021. (https://doi.org/10.1155/2021/9947059)

[2] D. Elreedy and A. F. Atiya, "A comprehensive analysis of Synthetic Minority Oversampling Technique (SMOTE) for handling class imbalance," *Inf. Sci.*, vol. 505, pp. 32-64, Jul. 2019. (https://doi.org/10.1016/j.ins.2019.07.070)

[3] A. M. Oyelakin, A. O. Ameen, T. S. Ogundele, T. Salau-Ibrahim, U. T. Abdulrauf, H. I. Olufadi, I. K. Ajiboye, S. Muhammad-Thani, and I. A. Adeniji, "Overview and exploratory analyses of CICIDS 2017 intrusion detection dataset," *J. OSEIT*, vol. 2, no. 2, pp. 45-52, Sep. 2023. (https://doi.org/10.29207/joseit.v2i2.5411)

[4] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *IEEE Access*, vol. 9, pp. 22351-22370, Feb. 2021. (https://doi.org/10.1109/ACCESS.2021.3056614)

[5] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with

information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911-132921, Jul. 2020. (https://doi.org/10.1109/ACCESS.2020.3009843)

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artificial Intell. Res.*, vol. 16, pp. 321-357, Jun. 2002. (https://doi.org/10.1613/jair.953)

[7] M. Kim and K. Kim, "Comparison of oversampling methods for effective network intrusion detection in class imbalanced data," *KNOM Rev.*, vol. 26, pp. 21-32, Dec. 2023. (https://doi.org/10.22670/knom.2023.26.2.21)

[8] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *LNCS*, vol. 3644, pp. 878-887, Aug. 2005. (https://doi.org/10.1007/11538059_91)

[9] M. Revathi and D. Ramyachitra, "A modified borderline smote with noise reduction in imbalanced datasets," *Wireless Personal Commun.*, vol. 121, pp. 1659-1680, Jul. 2021. (https://doi.org/10.1007/s11277-021-08690-y)

[10] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *2008 IEEE Int. Joint Conf. Neural Netw.*, pp. 1322-1328, Atlanta, USA, Jun. 2008. (https://doi.org/10.1109/IJCNN.2008.4633969)

[11] I. Dey and V. Pratap, "A comparative study of SMOTE, Borderline-SMOTE, and ADASYN oversampling techniques using different classifiers," *2023 3rd ICSMDI*, pp. 294-302, Trichy, India, Mar. 2023. (https://doi.org/10.1109/ICSMDI57622.2023.00060)

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. of the ACM*, vol. 63, pp. 139-144, Oct. 2020. (https://doi.org/10.1145/3422622)

[13] J. Lee and K. Park, "GAN-based imbalanced data intrusion detection system," *Personal Ubiquitous Comput.*, vol. 25, pp. 121-128, Feb. 2021. (https://doi.org/10.1007/s00779-019-01332-y)

[14] M. Zareapoor, P. Shamsolmoali, and J. Yang, "Oversampling adversarial network for class-imbalanced fault diagnosis," *Mech. Syst. and Signal Process.*, vol. 149, pp. 1-16, Feb. 2021. (https://doi.org/10.1016/j.ymssp.2020.107175)

[15] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *ICLR*, pp. 1-18, Toulouse, France, Apr. 2017. (https://doi.org/10.48550/arXiv.1605.09782)

[16] W. Zhang, P. Peng, and H. Zhang, "Using Bidirectional GAN with improved training architecture for imbalanced tasks," *IEEE 24th Int. Conf. CSCWD*, pp. 714-719, Dalian, China, May 2021. (https://doi.org/10.1109/CSCWD49262.2021.9437750)

[17] L. Breiman, "Random forests," *Machine Learn.*, vol. 45, pp. 5-32, Oct. 2001. (https://doi.org/10.1023/A:1010933404324)

[18] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Comput. & Secur.*, vol. 95, pp. 1-15, Aug. 2020. (https://doi.org/10.1016/j.cose.2020.101851)

[19] T. Chen and C. Guestrins, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 785-794, San Francisco, USA, Aug. 2016. (https://doi.org/10.1145/2939672.2939785)

[20] P. Devan and N. Khare, "An efficient XGBoost–DNN-based classification model for network intrusion detection system," *Neural Computi. and Appl.*, vol. 32, pp. 12499-12514, Jan. 2020. (https://doi.org/10.1007/s00521-020-04708-x)

[21] C.-F. Tsai, K.-C. Chen, and W.-C. Lin, "Feature selection and its combination with data over-sampling for multi-class imbalanced datasets," *Applied Soft Computing*, vol. 153, pp. 1-16, Mar. 2024. (https://doi.org/10.1016/j.asoc.2024.111267)

Minkyung Kim

Aug. 2019 : Ph.D. degree, Ajou University

Mar. 2023~Current : Assistant Professor at Dasan University College, Ajou University

<Research Interests> Big Data Analysis, Machine Learning and Deep Learning, Network Security