EAI Endorsed Transactions
on Context-aware Systems and Applications
Research Article

The data preprocessing in improving the classification quality of network intrusion detection systems

Hoang Ngoc Thanh[1,*]

[1]The Saigon International University

Abstract

Stream-based intrusion detection is a growing problem in computer network security environments. Many previous researches have applied machine learning as a method to detect attacks in network intrusion detection systems. However, these methods still have limitations of low accuracy and high false alarm rate. To improve the quality of classification, this paper proposes two solutions in the data preprocessing stage, that is, the solution of feature selection and resampling of the training dataset before they are used for training the classifiers. This is based on the fact that there is a lot of class imbalanced data in the training dataset used for network intrusion detection systems, as well as that there are many features in the dataset that are irrelevant to the classification goal, this reduces the quality of classification and increases the computation time. The data after preprocessing by the proposed algorithms is used to train the classifiers using different machine learning algorithms including: Decision Trees, Naive Bayes, Logistic Regression, Support Vector Machines, k Nearest Neighbor and Artificial Neural Network. The training and testing results on the UNSW-NB15 dataset show that: as with the Reconnaissance attack type, the proposed feature selection solution for F-Measure achieves 96.31%, an increase of 19.64%; the proposed oversampling solution for F-Measure achieves 96.99%, an increase of 3.17% and the proposed undersampling solution for F-Measure achieves 94.65%, an increase of 11.42%.

Received on 22 August 2023; accepted on 06 September 2023; published on 06 September 2023

Keywords: Feature Selection, Machine Learning, NIDS, Resampling, UNSW-NB15

doi:10.4108/eetcasa.v9i1.3778

1. Introduction

Internet is the trend of the times, the internet plays an increasingly important role in all areas of social life. On that internet platform, e-commerce is growing strongly, it is an indispensable part of business activities. Besides the benefits of the internet, businesses also face negative aspects of the internet, one of which is the problem of cyber attacks. Cyber attack is all forms of unauthorized intrusion into a computer system, website, database, network infrastructure, equipment of individuals and businesses through the internet for illegal purposes. The target of a cyber attack is very diverse, it can be a data breach (stealing, altering, encrypting, destroying), it can also target the integrity of the system (disruption, service obstruction), or take advantage of the victim's resources (displaying ads, malicious code, mining virtual currency, ...). To protect the network, one of the systems used by network administrators today is the network intrusion detection system (NIDS).

NIDS has the function of monitoring network traffic to detect abnormalities and illegal activities intruding the network of agencies and enterprises. NIDS can detect anomalies based on specific signatures of known threats or by comparing current network traffic with system benchmarks. There are three methods to detect attacks: (1) Signature-based detection; (2) Anomaly-based detection and (3) Hybrid-based detection.

Signature-based detection is designed to detect known attacks using the signatures of those attacks. This is an effective m ethod t o d etect k nown attacks stored in the NIDS database. Therefore, it is much more accurate in identifying a penetration attempt of a known attack. However, with new or variant attacks, NIDS cannot detect because the signature of

Corresponding author. Email: hoangngocthanh@siu.edu.vn

the attack is not stored. To solve the problem, anomaly-based detection compares current user activities with predefined profiles for intrusion detection. Anomaly-based detection is effective against unknown attacks or zero-day attacks without any updates to the system. However, these methods, mainly machine learning (ML) still face challenges in improving accuracy, reducing false alarm rate (FAR) and detecting new attacks [1].

In ML, data preprocessing is an important stage. The main objective comes from data preprocessing which has a major impact on the accuracy and capability of NIDS. With the increasing traffic of network data, ML techniques need a lot of time to train and classify the data. Using big data techniques for NIDS can solve many challenges such as speed and computational time as well as develop accurate NIDS [2]. This paper deals with improving the quality of classification (QoC) of NIDS through two data preprocessing techniques: feature selection and dataset resampling.

2. Related works

2.1. Feature selection

Feature selection is a method to remove irrelevant or noisy features and select the most suitable features to better classify instances belonging to various attack types. According to the researchers, this needs to be done because:

(1) A single selection strategy is not sufficient to obtain consistency across multiple datasets, since network traffic behavior is constantly changing [3].

(2) A suitable subset for each attack type must be determined, since a common subset is not sufficient to represent all of the various attacks [3].

(3) Feature selection can greatly improve not only the detection accuracy but also the computational efficiency, where:

- Irrelevant or noisy features can lead to poor detection rates, so reducing them can increase detection accuracy [4–6].

- Having more features results in higher computational cost and complexity. Reducing extraneous features increases computational efficiency [3, 7].

(4) Finally, some known types of attacks have become challenging to identify because they are too isolated and can be mislabeled as normal data. Researches and experiments have shown that: feature selection can solve this problem by defining a subset of features that adapt to the behavior of each attack type [5–7].

2.2. Resampling dataset

For many years, the problem of imbalanced data has been one of the important issues and received the attention of many researchers [8]. A dataset is said to be imbalanced when the number of instances belonging to one class label is much smaller than that of other class labels. To solve the problem, resampling techniques have been proposed, there are two main approaches used: removing some instances from the majority class, called undersampling (US), and cloning some of the instances from the minority class, is called oversampling (OS). Both oversampling and undersampling aim to change the ratio between majority and minority classes [9]. It is also possible to combine both techniques at the same time to create a more balanced dataset. In this way, resampling allows various classes to have relatively similar influence on the results of the classification model. Researches show that resampling the training dataset improves the accuracy of NIDS [10, 11].

One of the commonly used oversampling techniques is SMOTE (Synthetic Minority Over-Sampling Technique) [12]. The implementation of SMOTE is described as follows: Take a instance $\vec{a}$ from the minority class of the dataset and randomly select one instance $\vec{b}$ from among the k nearest neighbors of the same class $\vec{a}$ (in the feature space). A new synthetic data instance $\vec{x} = \vec{a} + w(\vec{b} - \vec{a})$ is created and added to the dataset, where w is the random weight in the interval $[0, 1]$.

Based on SMOTE, several various techniques have been built and developed. The first is the Cluster SMOTE technique [12]. In this technique, the training data is first classified into k clusters using the k-Means algorithm, for each cluster the imbalance ratio is calculated:

$$\text{IR} = \frac{\text{Number of minority class instances in the cluster}}{\text{Number of majority class instances in the cluster}}$$

Then, use SMOTE to clone the number of minority class instances in clusters with imbalance ratio IR > 1.

Next is the Adaptive Synthetic Sampling technique (ADASYN), which is built by shifting the importance of classification boundaries to difficult minority classes. ADASYN uses a weighted distribution for minority class instances that vary according to training difficulty, where more synthetic data is generated for more difficult minority class instances to learn [13].

Another SMOTE-based innovation is Borderline-SMOTE, Borderline-SMOTE there are two variants Borderline-SMOTE1 and Borderline-SMOTE2. This method oversample of minority instances only near the boundary and nearest neighbors of the same type. The difference between the two versions is that Borderline-SMOTE2 uses both positive and negative nearest neighbor. Compared to conventional SMOTE, Borderline-SMOTE does not clone synthetic instances for noise, but concentrates its efforts near the boundary, thereby helping the decision function to create better boundaries between classes. In terms of performance,

Borderline-SMOTE has also been reported to perform better than SMOTE [14].

The first known undersampling technique is Tomek Link which is defined as follows: provide a pair of objects (x_i, x_j), here where $x_i \in S_{min}, x_j \in S_{max}$ and $d(x_i, x_j)$ is the distance between x_i and x_j, then the pair (x_i, x_j) is called a Tomek Link if there is no x_k satisfy $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$. In this way, if two instances form a Tomek Link, either of these instances is noisy or both are near the contour. So one can use Tomek Link to clean up the overlap between. By removing overlapping instances, one can establish well-defined clusters in the training dataset and lead to improved classification quality.

Another approach is the Neighborhood Cleaning Rule (NCR) can be described as follows: for each instance Ei in the training dataset of the binary classification problem, its three nearest neighbors are found. If Ei belongs to the majority class and the class given by its three nearest neighbors contradicts the original class Ei, then Ei is deleted. If Ei belongs to the minority class and the three nearest neighbors misclassify Ei, then the nearest neighbors belonging to the majority class are discarded [9].

Similarly Tomek Link is an Edited Nearest Neighbors algorithm (ENN). ENN tend to remove more instances than Tomek Links, so it should provide more in-depth data cleaning. Different from NCR which is a method of undersampling, ENN is used to remove instances from both classes. Therefore, any instance that is misclassified by its three nearest neighbors will be removed from the training dataset [15].

Several researches have been carried out on the basis of comparing the oversampling and undersampling methods to deal with the class imbalance problem. Douzas and Bacao [16] developed a method to estimate the distribution of real data and clone data for minority classes of various imbalanced datasets. Douzas et al [17] have presented an oversampling method based on k-Means and SMOTE clustering to avoid generating noise and overcome imbalances between classes.

Amin et al [18] have investigated several well-known oversampling techniques: Mega-trend Diffusion Function (MTDF), SMOTE, ADASYN, Top-N and k-nearest neighbor (TRkNN) inversion, Weighted Minority Oversampling Technique (MWMOTE) and Immune Centroids Oversampling Technique (ICOTE). The research showed that the overall prediction performance of MTDF and genetic algorithm-based rule generation performed best compared to the rest of the oversampling augmentation methods.

2.3. Dataset and evaluation metrics

The dataset is a main component in training classifiers to detect attacks. Choosing the right dataset is important to ensure proper model building. Statistically the most used datasets in the researches are: KDDCup99, NSL-KDD, ISCX2012 and UNSW-NB15. In which, the UNSW-NB15 dataset was chosen to be used in the experiments of this paper, because it has some advantages when compared with other datasets: (1) It contains composite attack activities nowadays; (2) The probability distributions of the training and testing datasets are similar; (3) It consists of a set of features from the packet's payload and header to reflect the effective network packet, and (4) The dataset contains many complex data samples [19].

The selection of evaluation metric plays an important role when building and evaluating NIDS models. In this paper, the evaluation metric F1 Score (F-Measure with $\beta = 1$) was chosen to evaluate the classification quality of NIDS, for the following reasons: (1) Dataset used for training the NIDS is inherently imbalanced; (2) In the NIDS, positive class is the class of instances labeled attack plays an important role; (3) The false positive or negative alerts are equally important and (4) A normal access is interpreted as an attack or conversely an attack is interpreted as a normal access, both are important.

2.4. Shortcomings and Challenges

It is becoming increasingly important to protect computer systems using NIDS for intrusion detection. The above section has detailed related works on the methodology and technology of data preprocessing. Here are the shortcomings and challenges that need to be research:

(1) The use of old datasets such as: KDDCup99 and NSL-KDD can lead to static progress in NIDS, while intrusion attacks are constantly evolving with new technologies and user behaviors. It is therefore important to use a new dataset that is representative of the current environment, both software and hardware.

(2) Researches also show the effectiveness of reducing the features of the training datasets, which not only increases the accuracy of the ML algorithm, but also reduces the training time and cost. A subset of suitable features for each attack type should also be determined.

(3) And finally, like most imbalanced data sources in other fields, the improvement of algorithms for data resampling to improve the classification quality of NIDS should also be researched.

3. PROPOSED SOLUTIONS

3.1. Solution of feature selection (FS)

The proposed feature selection solution here uses ML as an fitness function to determine the best-suited subset of features for each attack type on all feature subsets of the training dataset. Because the training datasets used in NIDS are often very large and have many

irrelevant or noisy features, which causes difficulties such as: taking a long time to train and test, reduce the classification accuracy and increase the FAR. So the goal here is to find the features that are important to the classification results, eliminate irrelevant or noisy features, thereby reducing the time to train and test the classifier; At the same time, it helps to improve accuracy and reduce FAR.

Backward Feature Elimination Algorithm. The Backward Feature Elimination (BFE) algorithm [20] is proposed on the basis of the assumption that the features of the dataset are independent of each other, presented in Algorithm 1. In this algorithm, at each iteration, a classification model is selected to train a dataset of n input features. Then we eliminate one input feature at a time and train the same model on n-1 remaining input features n times. The input feature whose removal produces the smallest increase in error rate is discarded, leaving us with n-1 remaining input features. The classification is then repeated on n-2 features, ... In the k^{th} iteration, the model is trained on n-k features and has an error rate $e(k)$. Choosing the maximum acceptable error rate, we determine the minimum number of features needed to achieve classification accuracy with the chosen ML algorithm.

Proposition 1: Algorithm 1 has a time complexity of $O(N!)$, N is the number of features of the dataset.

Proof:

Let $T(N)$ be the time complexity of the algorithm. According to the lines from (12) to (19), we have due to the recursion of the algorithm:

$$T(N) = N \times T(N-1)$$

From here infer

$$T(N) = N \times (N-1) \times T(N-2)$$

Or

$$T(N) = N \times (N-1) \times \ldots \times T(1) = N! \times T(1)$$

So

$$T(N) = O(N!)$$

Forward Feature Construction Algorithm. The Forward Feature Construction (FFC) algorithm [20] is also proposed on the basis of the assumption that the features of the dataset are independent of each other, presented in Algorithm 2. This is the reverse process of the BFE algorithm. We start with a feature, then increment one feature at a time, the feature that produces the highest quality will be selected to be added to the resulting feature set. Both algorithms, BFE and FFC, are time consuming and computationally expensive. They actually only apply to datasets with a low number of features [20].

Algorithm 1 Feature selection using BFE

Input
D - Dataset
C - Classifier using ML
δ - Minimum error rate
Output
A subset of selected features

1: **begin**
2: Initialize:
3: $S \leftarrow$ Set of all features of the dataset D
4: $R \leftarrow \emptyset$ ▷ Set of features to remove
5: FindNoise(S, D, C)
6: **return** $S \setminus R$
7: **end**
8: **procedure** FindNoise(S, D, C) ▷ Find noisy features
9: $N \leftarrow$ Number of features $\in S$
10: Train C with features $\in S$ on the dataset D
11: $e \leftarrow$ Error rate of classifier C when testing
12: **for** $i \leftarrow 1$ to N **do**
13: $S_1 = S \setminus \{s_i\}$
14: Train C with features $\in S_1$ on the dataset D
15: **if** Error rate of classifier $C < e + \delta$ **then**
16: $R \leftarrow R \cup \{s_i\}$
17: FindNoise(S_1, D, C)
18: **end if**
19: **end for**
20: **end procedure** ▷ End of FindNoise

Proposition 2: Algorithm 2 has a time complexity of $O(N!)$, N is the number of features of the dataset.

Proof:

Let $T(N)$ be the time complexity of the algorithm. According to lines from (11) to (22), we have due to the recursion of the algorithm:

$$T(N) = N \times T(N-1)$$

From here infer

$$T(N) = N \times (N-1) \times T(N-2)$$

Or

$$T(N) = N \times (N-1) \times \ldots \times T(1) = N! \times T(1)$$

So

$$T(N) = O(N!)$$

Proposed feature selection algorithm (pFSA). As shown above, for a dataset with N features, if BFE or FFC is used to select the optimal set of features, the time complexity of the algorithm will be $O(N!)$ (according to Proposition 1 and Proposition 2). This is not suitable

Algorithm 2 Feature selection using FFC

Input
D - Dataset
C - Classifier using ML
Output
R - Selected subset of features

begin
  Initialize:
  $S \leftarrow$ Set of all features of the dataset D
  $R \leftarrow \emptyset$ ▷ Selected best set of features
  $bq \leftarrow 0$ ▷ the best QoC
  FindBest(S, D, C)
  return R
end
procedure FindBest(S, D, C) ▷ Find best features
  $best \leftarrow \emptyset$
  for each $s_i \in \{S \setminus R\}$ **do**
    $R_1 \leftarrow R \cup \{s_i\}$
    Train C with features $\in R_1$ on the dataset D
    if the QoC of $C > bq$ **then**
      $bq \leftarrow$ the QoC of C
      $best \leftarrow s_i$
    end if
  end for
  if $best \neq \emptyset$ **then**
    $R \leftarrow R \cup best$
    FindBest(S, D, C)
  end if
end procedure ▷ End of FindBest

for datasets with a large number of features. Moreover, BFE and FFC are effective only when the features of the dataset are independent of other features. However, with some features that are not independent and they only really work when combined, in this respect BFE and FFC are limited. To address the stated limitations of the BFE and FFC, we propose 3 contents:

(1) Combine BFE and FFC with feature ranking to reduce calculation time and cost, which is especially suitable for large datasets;

(2) Consider the correlation between features when adding or removing a feature. This is intended to address the limitations of BFE and FFC with datasets whose features are not independent of other features;

(3) The order of adding or removing a feature is based on the feature's rank, which is based on the relevance of the feature to the class label. Various reasearches [21] have suggested various metrics of the importance and relevance of features. In this paper, we propose to use the metrics: Information Gain (IG), Gain Ratio (GR) and Correlation Attribute (CA) to rank features.

The this paper proposes two algorithms: The first algorithm is denoted by pFFC, which is an algorithm that uses the improved wrapping model from the FFC algorithm combined with feature ranking, and at the same time considers the correlation between features. The algorithm starts from an empty set of features, then the features will, in turn, be selected for addition if the addition of that feature improves the classification quality of the NIDS. In addition, features that are correlated with the selected feature for inclusion in the selected set of features are also considered to be removed if such removal also improves the classification quality. Features with higher importance will be selected to be added first. The importance of features used here includes: IG, GR and CA.

The second algorithm, denoted pBFE, is an algorithm that uses the improved wrapping model from the BFE algorithm combined with feature ranking, while considering the correlation between features. The algorithm starts from the full set of 42 features, then the features will be selected for removal in turn if the removal of that feature improves the classification quality. In addition, before removing the selected feature, the features that correlate with the selected feature in the previously selected set of features are also calculated and evaluated to choose the best feature to remove. Features with lower importance will be selected for removal first. The feature importance used here also includes: IG, GR and CA.

The pseudocode of the first algorithm pFFC is shown in Algorithm 3. Accordingly, first the importance of 42 features in the UNSW-NB15 dataset is calculated and sorted in descending order. Important of the Features (IoF) used include: IG, GR and CA. Initially, S_{opt} has a feature, the F-Measure is achieved when training and testing on the UNSW-NB15 dataset with a feature, which is the starting value for the journey to find better F-Measure in the next 41 iterations. At each iteration in the next step, the features $s_i \in S$ is in turn added to S_{opt} to form S_1, the more important features (with a larger metric of information) are added first. Next, the data with features in S_1 is used to train and test the classifiers using various ML techniques. The results of evaluation of the classifiers are performed on the independent testing dataset in the UNSW-NB15. The classification quality of the classifiers is shown by the F-Measure. If the F-Measure of the classifier is trained with features in S_1 better than S_{opt}, which is the set of features for the previously stored best F-Measure, then the s_i feature will be recorded. Then, the features correlating with $s_i \in S_{opt}$ are considered for removal if such removal improves the F-Measure. The final obtained feature set will be assigned to S_{opt}. Otherwise, the s_i feature will be dropped, because adding this feature does not improve the classification quality. This process will be repeated until all the features have been added in turn to find the set of features S_{opt} for the best F-Measure.

Proposition 3: Algorithm 3 has a time complexity of $O(N \times (N-1)/2)$, N is the number of features of the dataset.

Proof:

Let $T(N)$ be the time complexity of the algorithm. According to the lines from (6) to (21), we have:

$$T(N) = T(1) + 2 \times T(1) + \ldots + (N-1) \times T(1)$$

From here infer

$$T(N) = N \times (N-1)/2 \times T(1)$$

So

$$T(N) = O(N \times (N-1)/2)$$

The pseudocode of the second algorithm is shown in Algorithm 4. Accordingly, first the importance of 42 features in the UNSW-NB15 dataset is calculated and sorted in descending order. The importance of the features used includes: IG, GR and CA. The initial S_{opt} includes all 42 features of the dataset, the F-Measure is achieved when training and testing on the UNSW-NB15 dataset with 42 features, which is the starting value for the journey to find better F-Measure in the next 41 iterations. At each iteration in the next step, the features s_i, in turn, are considered to be removed from S_{opt} to form S_1, and the less important features (with a smaller information metric) will be considered to be eliminated first. Next, the data with features in S_1 is used to train and test the classifiers using various ML techniques. The results of evaluation of the classifiers are also performed on the independent testing dataset in the UNSW-NB15. The classification quality of the classifiers is shown by the F-Measure. If the F-Measure of the classifier is trained with features in S_1 is better than S_{opt}, which is the set of features for the previously stored best F-Measure, then the s_i feature will be considered for elimination. Then, features that are correlated with s_i and have less importance than s in S_{opt} will be considered for removal instead of s_i if the removal improves the classification quality (shown by F-Measure). The final obtained set of features will be assigned to S_{opt}. In contrast, the removed feature is recovered, because removing this feature does not improve the classification quality. This process will be repeated until all the features have been eliminated in turn to find the set of features S_{opt} for the best F-Measure.

Proposition 4: Algorithm 4 has a time complexity of $O(N \times (N-1)/2)$, N is the number of features of the dataset.

Proof:

Let $T(N)$ be the time complexity of the algorithm. According to the lines from (6) to (21), we have:

$$T(N) = ((N-1) + (N-2) + \ldots + 1) \times T(1)$$

From here infer

$$T(N) = N \times (N-1)/2 \times T(1)$$

So

$$T(N) = O(N \times (N-1)/2)$$

Algorithm 3 pFFC feature selection algorithm

Input
D - Dataset
N - Number of features of the dataset
Output
S_{opt} - Subset of optimal features

```
1: begin
2:     Initialize:
3:     Calculate importance of features on dataset D
4:     S ← All features sorted in descending of IoF
5:     S_opt ← First feature of S
6:     for i ← 2 to N do
7:         S_1 ← S_opt ∪ {s_i}
8:         if the QoC of S_1 better than S_opt then
9:             Best ← the QoC of S_1
10:            for each s_ca in S_opt do
11:                if s_ca correlates with s_i then
12:                    S_2 ← S_opt ∪ {s_i} \ {s_ca}
13:                    if the QoC of S_2 > Best then
14:                        S_1 ← S_2
15:                        Best ← the QoC of S_2
16:                    end if
17:                end if
18:            end for
19:            S_opt ← S_1
20:        end if
21:    end for
22:    return S_opt
23: end
```

3.2. Solution of dataset resampling

The second proposed solution to improve the classification quality of the NIDS: the training dataset resampling solution. As shown, the training dataset UNSW-NB15 is quite imbalanced, attack types are accounting for a very small proportion in the dataset such as: Worms accounting for 0.05%, Shellcode accounting for 0.46%, Backdoor accounting for 0.71%, Analysis accounting for 0.82%, ... Such imbalance of data between classes leads to a situation where minority classes have a low influence on the results of the classification model and thereby reduce the quality of classification.

However, because the training datasets used in NIDS are often very large and have many irrelevant or noisy features, this adversely affects data resampling

Algorithm 4 pBFE feature selection algorithm

Input
D - Dataset
N - Number of features of the dataset
Output
S_{opt} - Subset of optimal features

begin
  Initialize:
  Calculate IoF of all features on the dataset D
  $S \leftarrow$ All features sorted in descending of IoF
  $S_{opt} \leftarrow S$
  for $i \leftarrow 1$ to N **do**
    $S_1 \leftarrow S_{opt} \setminus \{s_i\}$
    if the QoC of S_1 better than S_{opt} **then**
      Best $\leftarrow$ the QoC of S_1
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates and has IoF < s_i **then**
          $S_2 \leftarrow S_{opt} \setminus \{s_{ca}\}$
          if the QoC of S_2 > *Best* **then**
            $S_1 \leftarrow S_2$
            Best $\leftarrow$ the QoC of S_2
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

Algorithm 5 Oversampling combined with pFFC

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using OS

begin
  Initialize:
  Calculate IoF of all features on the dataset D
  $S_{max} \leftarrow$ All features sorted in descending of IoF
  $S_{min} \leftarrow$ The features obtained from pFSA
  $S_{opt} \leftarrow S_{min}$
  $S_{add} = S_{max} \setminus S_{min}$
  OS on training dataset D using features $\in S_{opt}$
  for $i \leftarrow 1$ to len(S_{add}) **do**
    $s \leftarrow S_{add}\,[i\text{ -}1]$
    $S_1 = S_{opt} \cup \{s\}$
    OS on the dataset D using the features $\in S_1$
    if QoC of S_1 is better than S_{opt} after OS **then**
      Best $\leftarrow$ the QoC of S_1 after OS
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates with s **then**
          $S_2 = S_{opt} \cup \{s\} \setminus \{s_{ca}\}$
          if QoC of S_2 after OS > *Best* **then**
            $S_1 \leftarrow S_2$
            Best $\leftarrow$ the QoC of S_2 after OS
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

techniques (which is based on the Euclidean distance between the features) by: cloning the bad instances of the minority class and eliminating the good instances of the majority class. To improve, this paper proposes not to use irrelevant or noisy features when calculating to resample the dataset.

Solution of proposed oversampling. The above oversampling techniques all rely on k nearest neighbors to create the synthetic data instances with the participation of all features. The problem is, there are irrelevant or noisy features when calculating the distance to determine the k nearest neighbors, that can affect the quality of the oversampling. To eliminate these irrelevant or noisy features, this paper proposes to use 2 solutions presented in Algorithm 5 and Algorithm 6.

The first algorithm (Algorithm 5) uses the solution proposed by Algorithm 3 (the pFFC algorithm) to determine the best-fit features participating in the distance calculation when determining the k nearest neighbors used in oversampling. Algorithm 5 is implemented specifically as follows: First, the S_{max} consisting of 42 features of the dataset UNSW-NB15 is calculated and sorted in descending order of importance, the importance of features has can be IG, GR or CA. S_{min} is the initial minimum set of features, these are the features obtained for each type of attack through the feature selection algorithms presented in Section 3.1 (see the feature selection results in Table 2. S_1 is the set of features to be evaluated, S_1 has an initial value of S_{min}. At each loop, the remaining features ($S_{max} \setminus S_{min}$) are added to S_1 in turn, the more important features (with a larger information metric) are added first. Then, oversampling techniques such as: SMOTE, ADASYN, Cluster SMOTE and Borderline SMOTE are respectively used to add the synthetic data instances to the original training dataset to generate the new training dataset, the difference is that only the features in S_1 are used when calculating the distance to determine the k nearest neighbors in the oversampling algorithms. New training datasets with additional synthetic instances used to train classifiers using ML techniques. Evaluation results of classifiers are performed on the testing dataset, which is an independent dataset in the UNSW-NB15. The

classification quality of the classifiers is shown by the F-Measure. If the F-Measure of the classifier is trained with features in S_1 better than S_{opt}, which is the set of features for the previously stored best F-Measure, then the added instance (denoted s) will be recorded. Next, we will find the features s_{ca} that correlates with s in S_{opt}, and perform the removal of s_{ca} and add s to S_{opt} if such removal improves the F-Measure. Otherwise, the added feature will be dropped, because adding this feature does not improve the classification quality. This process will be repeated until all the remaining features other than S_{min} are added in turn to find the set of features S_{opt} for the best F-Measure.

Proposition 5: Algorithm 5 has a time complexity of $O(N \times (N-1)/2)$, N is the number of features of the dataset.

Proof:

Let $T(N)$ be the time complexity of the algorithm. According to the lines from (9) to (26), we have:

$$T(N) = T(1) + 2 \times T(1) + \ldots + (N-1) \times T(1)$$

From here infer

$$T(N) = N \times (N-1)/2 \times T(1)$$

So

$$T(N) = O(N \times (N-1)/2)$$

The second algorithm (Algorithm 6) uses the solution proposed by Algorithm 4 (the pBFE algorithm) to determine the best-fit features participating in the distance calculation when determining the k nearest neighbors used in oversampling algorithms. Algorithm 6 is implemented specifically as follows: First, the set S_{max} consisting of 42 features of the UNSW-NB15 dataset is also calculated and sorted in descending order of importance. The importance of features has can be IG, GR or CA. S_{min} is the initial set of minimal features, which are also features obtained for each attack type through feature selection algorithms presented in Section 3.1 (see the results of feature selection in Table 2). S_1 is the set of features to be evaluated, S_1 has an initial value of S_{max} including all 42 features. At each iteration, the features to be considered ($S_{max} \setminus S_{min}$) are in turn considered to be removed from S_1, and the less important features (with a smaller information metric) will be considered before. Next, oversampling techniques such as: SMOTE, ADASYN, Cluster SMOTE and Borderline SMOTE are also respectively used to add synthetic data instances to the original training dataset to generate the new training dataset, the difference is that only the features in S_1 are used when calculating the distance to determine the k nearest neighbors in the oversampling algorithms. New training datasets with additional synthetic instances used to train classifiers using ML techniques. Evaluation results of classifiers are performed on the testing dataset, which is an independent dataset in the UNSW-NB15. The classification quality of the classifiers is shown by the F-Measure. If the F-Measure of the best classifier of the above trained classifiers is better than the best F-Measure generated from S_{opt}, which is the set of features for the previously stored best F-Measure, then the feature (denoted by s) will be considered for removal. Next, we will find the features s_{ca} that is correlated with s and has less importance than s in S_{opt}. The removal of textits will be replaced by the removal of s_{ca} in S_{opt} if that replacement improves the F-Measure. Otherwise, the removed feature will be recovered, because removing this feature will degrade the classification quality. This process will be repeated until all remaining features other than S_{min} are removed in turn to find the set of features S_{opt} for the best F-Measure.

Algorithm 6 Oversampling combined with pBFE

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using OS

```
1:  begin
2:      Initialize:
3:      Calculate IoF of all features on the dataset D
4:      S_max ← All features sorted in ascending of IoF
5:      S_min ← The features obtained from pFSA
6:      S_opt ← S_max
7:      S_del = S_max \ S_min
8:      OS on training dataset D using features ∈ S_opt
9:      for i ← 1 to len(S_del) do
10:         s ← S_del [i -1]
11:         S_1 = S_opt \ {s}
12:         OS on the dataset D using the features ∈ S_1
13:         if QoC of S_1 is better than S_opt after OS then
14:             Best ← the QoC of S_1 after OS
15:             for each s_ca in S_opt do
16:                 if s_ca correlates and has IoF < s then
17:                     S_2 = S_opt \ {s_ca}
18:                     if QoC of S_2 after OS > Best then
19:                         S_1 ← S_2
20:                         Best ← the QoC of S_2 after OS
21:                     end if
22:                 end if
23:             end for
24:             S_opt ← S_1
25:         end if
26:     end for
27:     return S_opt
28: end
```

Proposition 6: Algorithm 6 has a time complexity of $O(N \times (N-1)/2)$, N is the number of features of the dataset.

Proof:

Call $T(N)$ is the time complexity of the algorithm. According to the lines from (9) to (26), we have:

$$T(N) = ((N-1) + (N-2) + \ldots + 1) \times T(1)$$

From here infer

$$T(N) = N \times (N-1)/2 \times T(1)$$

So

$$T(N) = O(N \times (N-1)/2)$$

Solution of proposed undersampling. The undersampling techniques also rely on k nearest neighbors to remove unwanted overlap between classes with all features involved. The problem is, there are irrelevant or noisy features when calculating the distance to determine k nearest neighbors, that can affect the quality of removing data instances at majority class of undersampling. To eliminate these irrelevant or noisy features, this paper proposes to use 2 solutions presented in Algorithm 7 and Algorithm 8.

The first algorithm (Algorithm 7) uses the solution proposed by Algorithm 3 (the pFFC algorithm) to determine the best-fit features participating in the distance calculation when determining the k nearest neighbors used in undersampling algorithms. Algorithm 7 is implemented specifically as follows: First, the S_{max} consisting of 42 features of the dataset UNSW-NB15 is calculated and sorted in descending order of importance, the importance of features has can be IG, GR or CA. S_{min} is the initial minimum set of features, these are the features obtained for each type of attack through the feature selection algorithms presented in Section 3.1 (see the feature selection results in Table 2). S_1 is the set of features to be evaluated, S_1 has an initial value of S_{min}. At each loop, the remaining features ($S_{max} \setminus S_{min}$) are added to S_1 in turn, the more important features (with a larger information metric) are added first. Then, undersampling techniques such as: TML, NCR, ENN are respectively used to remove noisy and overlapping data instances from the original training dataset to generate a new training dataset. The difference is that only the features in S_1 are used when calculating the distance to determine the k nearest neighbors in the undersampling algorithm. New training datasets with eliminated data instances are used to train classifiers using ML techniques. Evaluation results of classifiers are performed on the testing dataset, which is an independent dataset in the UNSW-NB15. The classification quality of the classifiers is shown by the F-Measure. If the F-Measure of the above classifier is better than the best F-Measure generated from S_{opt}, which is the set of features for the previously stored best F-Measure, then the added instance (denoted s) will be recorded. Next, we will find the features s_{ca} that correlates with s in S_{opt}, and perform the removal of s_{ca} and add s to S_{opt} if such removal improves the F-Measure. Otherwise, the added feature will be dropped, because adding this feature does not improve the classification quality. This process will be repeated until all remaining features other than S_{min} are added in turn to find the set of features S_{opt} for the best F-Measure.

Proposition 7: Algorithm 7 has a time complexity of $O(N \times (N-1)/2)$, N is the number of features of the dataset.

Proof:

Let $T(N)$ be the time complexity of the algorithm. According to the lines from (9) to (26), we have:

$$T(N) = T(1) + 2 \times T(1) + \ldots + (N-1) \times T(1)$$

From here infer

$$T(N) = N \times (N-1)/2 \times T(1)$$

So

$$T(N) = O(N \times (N-1)/2)$$

The second algorithm (Algorithm 8) uses the solution proposed by Algorithm 4 (pBFE algorithm) to determine the best-fit features participating in distance calculation when determining k nearest neighbors is used in oversampling algorithms. Algorithm 8 is implemented specifically as follows: First, the set S_{max} consisting of 42 features of the UNSW-NB15 dataset is also calculated and sorted in descending order of importance, the feature importances can also be IG, GR or CA as in Algorithm 7. S_{min} is the initial minimum set of features, which are also features obtained for each attack type through feature selection algorithms presented in Section 3.1 (see the results of feature selection in Table 2). S_1 is the set of features to be evaluated, S_1 has an initial value of S_{max} including all 42 features. At each iteration, the features to be considered ($S_{max} \setminus S_{min}$) are in turn considered to be removed from S_1, and the less important features (with a smaller information metric) are considered first. Next, undersampling techniques such as: TML, NCR, ENN are respectively used to remove noisy and overlapping data instances from the original training dataset to create a new training dataset. Another point is that only features in S_1 are used when calculating distances to determine k nearest neighbors in undersampling algorithms. New training datasets with eliminated data instances used to train classifiers using ML techniques.

Algorithm 7 Undersampling combined with pFFC

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using US

begin
  Initialize:
  Calculate IoF of all features on the dataset D
  $S_{max} \leftarrow$ All features sorted in descending of IoF
  $S_{min} \leftarrow$ The features obtained from pFSA
  $S_{opt} \leftarrow S_{min}$
  $S_{add} = S_{max} \setminus S_{min}$
  US on training dataset D using features $\in S_{opt}$
  for $i \leftarrow 1$ to len(S_{add}) **do**
    $s \leftarrow S_{add}\,[i\text{ -}1]$
    $S_1 = S_{opt} \cup \{s\}$
    US on the dataset D using the features $\in S_1$
    if QoC of S_1 is better than S_{opt} after US **then**
      $Best \leftarrow$ the QoC of S_1 after US
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates with s **then**
          $S_2 = S_{opt} \cup \{s\} \setminus \{s_{ca}\}$
          if QoC of S_2 after US > $Best$ **then**
            $S_1 \leftarrow S_2$
            $Best \leftarrow$ the QoC of S_2 after US
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

Evaluation results of classifiers are performed on the testing dataset, which is an independent dataset in the UNSW-NB15. The classification quality of the classifiers is shown by the F-Measure. If the F-Measure of the above trained classifier is better than the best F-Measure generated from S_{opt}, which is the set of features for the previously stored best F-Measure, then the feature (denoted by s) will be considered for removal. Next, we will find the features s_{ca} that is correlated with s and has less importance than s in S_{opt}. The removal of s will be replaced by the removal of s_{ca} in S_{opt} if that replacement improves the F-Measure. Otherwise, the removed feature will be recovered, because removing this feature will degrade the classification quality. This process will be repeated until all remaining features other than S_{min} are removed in turn to find the S_{opt} for the best F-Measure.

Proposition 8: Algorithm 8 has a time complexity of $O(N \times (N-1)/2)$, where N is the number of features of the dataset.

Proof:
Let $T(N)$ be the time complexity of the algorithm. According to the lines from (9) to (26), we have:

$$T(N) = ((N-1) + (N-2) + \ldots + 1) \times T(1)$$

From here infer

$$T(N) = N \times (N-1)/2 \times T(1)$$

So

$$T(N) = O(N \times (N-1)/2)$$

Algorithm 8 Undersampling combined with pBFE

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using US

begin
  Initialize:
  Calculate IoF of all features on the dataset D
  $S_{max} \leftarrow$ All features sorted in ascending of IoF
  $S_{min} \leftarrow$ The features obtained from pFSA
  $S_{opt} \leftarrow S_{max}$
  $S_{del} = S_{max} \setminus S_{min}$
  US on training dataset D using features $\in S_{opt}$
  for $i \leftarrow 1$ to len(S_{del}) **do**
    $s \leftarrow S_{del}\,[i\text{ -}1]$
    $S_1 = S_{opt} \setminus \{s\}$
    US on the dataset D using the features $\in S_1$
    if QoC of S_1 is better than S_{opt} after US **then**
      $Best \leftarrow$ the QoC of S_1 after US
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates and has IoF < s **then**
          $S_2 = S_{opt} \setminus \{s_{ca}\}$
          if QoC of S_2 after US > $Best$ **then**
            $S_1 \leftarrow S_2$
            $Best \leftarrow$ the QoC of S_2 after US
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

4. EXPERIMENTAL RESULTS

4.1. Solution of feature selection

The training and testing datasets are the full training and testing datasets of the UNSW-NB15. The features on the UNSW-NB15 dataset are numbered sequentially

Table 1. Features of UNSW-NB15 dataset

ID	Features	ID	Features	ID	Features
1	label	2	dur	3	proto
4	service	5	state	6	spkts
7	dpkts	8	sbytes	9	dbytes
10	rate	11	sttl	12	dttl
13	load	14	dload	15	loss
16	dloss	17	sinpkt	18	dinpkt
19	sjit	20	djit	21	swing
22	stcpb	23	dtcpb	24	dwin
25	tcprtt	26	synack	27	ackdat
28	smeansz	29	dmeansz	30	trans_depth
31	response_body_len	32	ct_srv_src	33	ct_state_ttl
34	ct_dst_ltm	35	ct_src_dport_ltm	36	ct_dst_sport_ltm
37	ct_dst_src_ltm	38	is_ftp_login	39	ct_ftp_cmd
40	ct_flw_http_mthd	41	ct_src_ltm	42	ct_srv_dst
43	is_sm_ips_ports				

Table 2. Results of using pFSA for each attack type

Attack	Selected Features	pFSA
Worms	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	pBFE-IG
Shellcode	2, 3, 4, 6, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 24, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43	pBFE-GR
Backdoor	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43	pBFE-CA
Analysis	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	pBFE-CA
Reconnais	3, 4, 5, 7, 9, 10, 11, 12, 13, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29, 31, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	Ridge
DoS	2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 21, 24, 26, 29, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43	pBFE-CA
Fuzzers	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36, 37, 39, 40, 43	pBFE-IG
Exploits	2, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 37, 38, 39, 40, 41, 42, 43	pBFE-CA
Generic	2, 3, 4, 6, 7, 8, 9, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 24, 27, 28, 29, 31, 33, 37, 38, 39, 42, 43	pBFE-IG

as shown in Table 1. The results of feature selection using pFFC and pBFE with consideration of the correlation of features are shown in Table 2. For each type of attack, the selected features are different in features and quantity. The classifier is mainly used as a decision tree. Figure 1 shows the improvement of classification quality in each attack type when using the proposed feature selection algorithms.

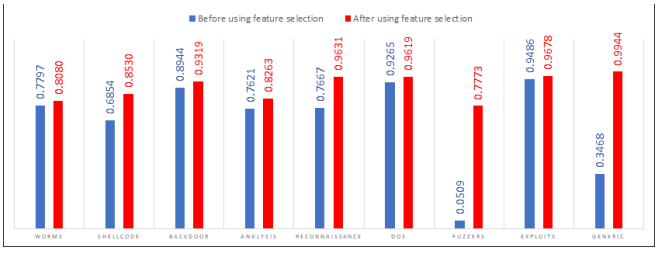

Figure 1. The F-Measure before and after using pFSA

From the experimental results, some conclusions are drawn as follows:

(1) Evaluation results on the UNSW-NB15 dataset show that this dataset has many complex instances, especially Generic and Fuzzers attack types.

(2) The use of feature selection techniques not only reduces the computational cost and time (according to Proposition 3 and Proposition 4) but also improves the classification quality in IDS.

(3) The use of pBFE-IG, pBFE-GR and pBFE-CA algorithms for better feature selection than other known algorithms.

(4) For each different type of attack, different features and ML algorithms will be selected to best improve the classification quality of the intrusion detection system.

4.2. Solution of dataset resampling

Regarding the oversampling techniques: SMOTE, ADASYN, Cluster SMOTE, Borderline SMOTE1 and Borderline SMOTE2 are used. Regarding the undersampling techniques: TML, ENN and NCR techniques are used. The selection of the features participating in the Euclidean distance to find the nearest neighbors in the oversampling and undersampling techniques has also been done with the pFFC and pBFE algorithms.

Oversampling the Dataset. As mentioned in the proposal, the oversampling techniques are based on the kNN algorithm to create a synthetic data instances with the participation of all features. The removal of irrelevant or noisy features when calculating distances to determine k nearest neighbors can improve the quality of oversampling techniques. In the experiments, both proposed solutions, pFFC and pBFE, were performed. The oversampling techniques used include: ADASYN, SMOTE, Cluster SMOTE, Borderline SMOTE1 and Borderline SMOTE2.

The results of using oversampling in combination with feature selection are presented in Table 3. The line with symbol G is the evaluation result obtained when not using oversampling; the line with the symbol O is the evaluation result obtained when using the oversampling on all 42 features; line with symbol F is the evaluation result obtained when using the oversampling in combination with pFFC; line with symbol B is the evaluation result obtained when using the oversampling in combination with pBFE. The line in bold is the line that gives the best F-Measure for each attack type.

Table 4 is a summary of the results obtained by using the oversampling in combination with feature selection for each attack type. The Selected Features column has the features numbered in order as shown in Table 1. The Techniques Used column shows the oversampling technique, the feature selection technique and the number of features left for the best results for each type of attack. Thereby, it is shown that the technique of oversampling in combination with feature selection

Table 3. Detailed results using OS combined with pFSA

Attack Type	OS	Sensitivity	Specificity	Precision	F-Measure
Worms	G	0.7077	0.9998	0.8679	0.7797
	O	0.7692	0.9996	0.8130	0.7905
	F	0.8462	0.9995	0.7914	0.8178
	B	0.7923	0.9997	0.8655	0.8273
Shellcode	G	0.5605	0.9985	0.8819	0.6854
	O	0.8067	0.9954	0.7792	0.7927
	F	0.7520	0.9978	0.8721	0.8076
	B	0.6805	0.9989	0.9256	0.7843
Backdoor	G	0.8803	0.9973	0.9089	0.8944
	O	0.9330	0.9976	0.9229	0.9279
	F	0.9273	0.9977	0.9273	0.9273
	B	0.9198	0.9987	0.9560	0.9375
Analysis	G	0.7055	0.9948	0.8285	0.7621
	O	0.6985	0.9999	0.9957	0.8210
	F	0.7135	0.9997	0.9875	0.8284
	B	0.7140	0.9996	0.9828	0.8271
Reconnaissance	G	0.6287	0.9979	0.9823	0.7667
	O	0.9142	0.9935	0.9634	0.9382
	F	0.9669	0.9945	0.9705	0.9687
	B	0.9635	0.9956	0.9764	0.9699
DoS	G	0.9430	0.9797	0.9106	0.9265
	O	0.9516	0.9901	0.9544	0.9530
	F	0.9486	0.9938	0.9708	0.9596
	B	0.9611	0.9909	0.9584	0.9597
Fuzzers	G	0.0270	0.9755	0.4407	0.0509
	O	0.6193	0.8845	0.6351	0.6271
	F	0.7276	0.8842	0.6711	0.6982
	B	0.6489	0.9010	0.6803	0.6642
Exploits	G	0.9328	0.9798	0.9650	0.9486
	O	0.9584	0.9784	0.9636	0.9610
	F	0.9433	0.9661	0.9431	0.9432
	B	0.9584	0.9784	0.9636	0.9610
Generic	G	0.2284	0.9711	0.7197	0.3468
	O	0.9966	0.9886	0.9842	0.9904
	F	0.9958	0.9967	0.9954	0.9956
	B	0.9959	0.9964	0.9950	0.9954

Table 4. Summary of results using OS combined with pFSA

Attack Type	Selected Features	Techniques Used
Worms	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	Borderline_SMOTE1 pBFE 40 features left
Shellcode	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 24, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43	Borderline_SMOTE1 pFFC 35 features left
Backdoor	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	ADASYN pBFE 41 features left
Analysis	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	ADASYN pFFC 41 features left
Reconnais	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	Cluster SMOTE pBFE 39 features left
DoS	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	Cluster SMOTE pBFE 42 features left
Fuzzers	2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, 25, 27, 28, 29, 31, 33, 34, 37, 38, 39, 40, 42, 43	Cluster SMOTE pFFC 32 features left
Exploits	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	ADASYN No 42 features left
Generic	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36, 37, 39, 40, 43	Borderline_SMOTE2 pFFC 35 features left

helps to improve the classification quality in all types of attacks. But there will be cost for feature selection with time complexity of $O(N \times (N-1)/2)$ (according to Proposition 5 and Proposition 6).

Undersampling the Dataset. Similar to oversampling, the undersampling techniques also rely on the kNN algorithm to remove the overlapping data between classes and the noisy data with the participation of all features. The removal of irrelevant or noisy features when calculating distances to determine k nearest neighbors can improve the quality of undersampling techniques.

In the experiments, both proposed solutions mFF and pBFE were used. The undersampling techniques used include: TML, NCR, ENN.

The results of using undersampling in combination with feature selection are presented in Table 5. The line with symbol G is the evaluation result obtained without using undersampling; the line with the symbol U is the evaluation result obtained when using the undersampling on all 42 features; line with symbol F is the evaluation result obtained when using the undersampling in combination with pFFC; The line with symbol B is the evaluation result obtained when using the the undersampling in combination with pBFE. The line in bold is the line that gives the best F-Measure for each attack type.

Table 6 is a summary of the results obtained by using undersampling techniques combined with feature selection for each attack type. The columns

Table 5. Detailed results using US combined with pFSA

Attack Type	US	Sensitivity	Specificity	Precision	F-Measure
Worms	G	0.7077	0.9998	0.8679	0.7797
	U	0.7308	0.9998	0.8962	0.8051
	F	0.7769	0.9997	0.8559	0.8145
	B	0.7769	0.9997	0.8559	0.8145
Shellcode	G	0.5605	0.9985	0.8819	0.6854
	U	0.6196	0.9986	0.8966	0.7328
	F	0.6664	0.9985	0.9010	0.7661
	B	0.6796	0.9990	0.9322	0.7861
Backdoor	G	0.8803	0.9973	0.9089	0.8944
	U	0.9192	0.9971	0.9073	0.9132
	F	0.8963	0.9980	0.9332	0.9144
	B	0.8935	0.9980	0.9336	0.9131
Analysis	G	0.7055	0.9948	0.8285	0.7621
	U	0.7020	0.9984	0.9404	0.8039
	F	0.7110	0.9998	0.9903	0.8277
	B	0.7105	0.9999	0.9965	0.8295
Reconnaissance	G	0.6287	0.9979	0.9823	0.7667
	U	0.7231	0.9973	0.9802	0.8323
	F	0.9138	0.9962	0.9781	0.9449
	B	0.9169	0.9961	0.9780	0.9465
DoS	G	0.9430	0.9797	0.9106	0.9265
	U	0.9317	0.9941	0.9719	0.9514
	F	0.9459	0.9935	0.9697	0.9576
	B	0.9410	0.9942	0.9725	0.9565
Fuzzers	G	0.0270	0.9755	0.4407	0.0509
	U	0.3051	0.9670	0.7499	0.4337
	F	0.3609	0.9646	0.7678	0.4910
	B	0.3516	0.9550	0.7173	0.4719
Exploits	G	0.9328	0.9798	0.9650	0.9486
	U	0.9530	0.9765	0.9603	0.9566
	F	0.9438	0.9910	0.9843	0.9636
	B	0.9421	0.9914	0.9850	0.9630
Generic	G	0.2284	0.9711	0.7197	0.3468
	U	0.2284	0.9711	0.7197	0.3468
	F	0.9958	0.9971	0.9959	0.9959
	B	0.9947	0.9975	0.9965	0.9956

Table 6. Summary of results using US combined with pFSA

Attack	Selected Features	Techniques
Worms	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	ENN pFFC 40 features left
Shellcode	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43	NCR pBFE 40 features left
Backdoor	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 20, 31	NCR pFFC 40 features left
Analysis	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	ENT pBFE 41 features left
Reconnais	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29, 31, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43	ENN pBFE 37 features left
DoS	2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 21, 24, 26, 29, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 5, 17	ENT pFFC 32 features left
Fuzzers	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36, 37, 39, 40, 43, 22, 31	NCR pFFC 37 features left
Exploits	2, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 8, 31	ENT pFFC 37 features left
Generic	2, 3, 4, 6, 7, 8, 9, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 24, 27, 28, 29, 31, 33, 37, 38, 39, 42, 43, 5, 23, 32, 40	ENT pFFC 32 features left

of Selected Features are numbered features according to Table 1. Thereby, it shows that the technique of undersampling combined with feature selection greatly improves the classification quality. It also helps to remove many noisy instances and overlap between classes in the majority class. But there will be cost for feature selection with time complexity of $O(N \times (N-1)/2)$ (according to Proposition 7 and Proposition 8).

5. CONCLUSION

Experimental results have demonstrated that the proposed solutions to improve the classification quality of NIDS include:

(1) Propose techniques to improve feature selection of training datasets used in NIDS.

(2) Propose techniques to improve the handling of imbalanced data sources inherent in NIDS, through the improvement of oversampling and undersampling techniques.

In the experiments, the UNSW-NB15 dataset was used for training and testing, which is a dataset with many contemporary synthetic attack instances that have not been used by many researchers. The paper proposes to use the F-Measure to evaluate the classification quality of NIDS. This is to contribute to improving the effectiveness of the evaluation.

Besides the obtained results, the research results of the paper also leave the following shortcomings, limitations and future development orientations:

(1) The correct combination of data preprocessing algorithms and classifiers to build a hybrid, multi-label and real-time response classifier is an issue that needs to be further researched.

(2) The system's ability to process data as well as compute plays an important role in exploiting ML algorithms. The improvement of processing efficiency in the direction of parallel processing as well as the selection and optimization of parameters for ML techniques is still an open issue.

References

[1] S.M. Othman, F.M. Ba-Alwi, T. Nabeel, and A.Y. Al-Hashida, "Intrusion detection model using machine learning algorithm on Big Data environment," *J Big Data*, vol. 5, no. 34, https://doi.org/10.1186/s40537-018-0145-4, 2018.

[2] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, vol. 55, pp. 453–563, 2022.

[3] Z. Liu, R. Wang, M. Tao, and X. Cai, "A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion," *Neurocomputing*, vol. 168, pp. 365–381, 2015.

[4] H. Alsaadi, R. Almuttairi, O. Bayat, and A. Osman, "Computational intelligence algorithms to handle dimensionality reduction for enhancing intrusion detection system," *J. Inf. Sci. Eng.*, vol. 36, no. 2, pp. 293–308, 2020.

[5] O. Almomani, "A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms," *Symmetry (Basel)*, vol. 12, no. 6, pp. 1–20, 2020.

[6] M.S. Bonab, A. Ghaffari, F.S. Gharehchopogh, and P. Alemi, "A wrapper-based feature selection for improving performance of intrusion detection systems," *Int. J. Commun. Syst.*, vol. 33, no. 12, pp. 1–25, 2020.

[7] Y. Zhu, J. Liang, J. Chen, and Z. Ming, "An improved NSGA-III algorithm for feature selection used in intrusion detection," *Knowledge-Based Systems*, vol. 116, pp. 74–85, 2017.

[8] J. Leevy, T. Khoshgoftaar, R. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, 2018.

[9] N. Junsomboon, "Combining Over-Sampling and Under-Sampling Techniques for Imbalance Dataset," in *Proceedings of the 9th International Conference on Machine Learning and Computing*, 2017.

[10] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *Journal of Big Data*, vol. 8, no. 6, 2021.

[11] H. Ahmed, A. Hameed, and N. Bawany, "Network intrusion detection using oversampling technique and machine learning algorithms," *PeerJ Computer Science*, 8:e820 DOI 10.7717/peerj-cs.820, 2022.

[12] F. Last, G. Douzas, and F. Bação, "Oversampling for Imbalanced Learning Based on K-Means and SMOTE," CoRR abs/1711.00837, 2017.

[13] Y. Pristyanto, A.F. Nugraha, A. Dahlan, L.A. Wirasakti, A.A. Zein, and I. Pratama, "Multiclass Imbalanced Handling using ADASYN Oversampling and Stacking

Algorithm," in *16th International Conference on Ubiquitous Information Management and Communication*, doi: 10.1109/IMCOM53663.2022.9721632, 2022.

[14] A. Pathak, "Analysis of Different SMOTE based Algorithms on Imbalanced Datasets," *International Research Journal of Engineering and Technology (IRJET)*, vol. 8, no. 8, pp. 4111–4114, 2021.

[15] D. Guan, W. Yuan, Y.K. Lee, and S. Lee, "Nearest neighbor editing aided by unlabeled data," *Information Sciences*, vol. 179, pp. 2273–2282, 2009.

[16] G. Douzas and F. Bacao, "Effective data generation for imbalanced learning using conditional generative adversarial networks," *Expert Systems with Applications*, vol. 91, pp. 464–471, 2018.

[17] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," *Information Sciences*, vol. 465, pp. 1–20, 2018.

[18] A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Quadir, A. Havalah, and A. Hussain, "Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study," *IEEE Access*, vol. 4, pp. 7940–7957, 2016.

[19] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive dataset for network intrusion detection systems," in *Conference on Military Communications and Information Systems*, 2015.

[20] C. Sergio, D.S. Javier, L. Ibai, O. Ignacio, S. Javier, J.S. Javier, and I.T. Ana, "Chapter 5 - Big Data in Road Transport and Mobility Research," in *Intelligent Vehicles*, Butterworth-Heinemann, pp. 175–205, 2018.

[21] M. Torabi, N.I. Udzir, M.T. Abdullah, and R. Yaakob, "A Review on Feature Selection and Ensemble Techniques for Intrusion Detection System," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 5, pp. 538–553, 2021.