Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2020.011705
Article

Tech Science Press

Oversampling Methods Combined Clustering and Data Cleaning for Imbalanced Network Data

Yang Yang[1,*], Qian Zhao[1], Linna Ruan[2], Zhipeng Gao[1], Yonghua Huo[3] and Xuesong Qiu[1]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100000, China
[2]Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, 3000, Australia
[3]The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, 050000, China
[*]Corresponding Author: Yang Yang. Email: yyang@bupt.edu.cn

Abstract: In network anomaly detection, network traffic data are often imbalanced, that is, certain classes of network traffic data have a large sample data volume while other classes have few, resulting in reduced overall network traffic anomaly detection on a minority class of samples. For imbalanced data, researchers have proposed the use of oversampling techniques to balance data sets; in particular, an oversampling method called the SMOTE provides a simple and effective solution for balancing data sets. However, current oversampling methods suffer from the generation of noisy samples and poor information quality. Hence, this study proposes an oversampling method for imbalanced network traffic data that combines the SMOTE algorithm and FINCH clustering algorithm to filter out minority sample clusters, proposes a scheme to allocate the number of synthetic samples per cluster according to the clustering sparsity and sample weight, and finally uses multi-layer sensors for noisy sample cleaning during sampling. We compare the proposed method with other oversampling methods, verifying that a data set processed using this method works better in network traffic anomaly detection.

Keywords: Imbalanced learning; oversampling; SMOTE; network anomaly detection

1 Introduction

Network anomaly traffic detection can provide a basis for detecting network attacks, network misconfigurations, network failures, etc., by learning network traffic data. Current researchers have proposed many network anomaly detection techniques to address a variety of different current application scenarios. The main methods of representation are logistic regression classification and Bayesian classification. Regardless of which anomaly detection technique is used, there is the problem of imbalance in the network data itself. For example, most of the network traffic data are standard traffic data, and only a small percentage are abnormal traffic data, which is a natural imbalance in the network data set due to the inherent characteristics of the network. For most classifications that rely on data set

learning, the imbalance of the data set leads to overfitting of the learning model in the direction of majority samples. Even if overfitting occurs, the general detector can obtain a reasonable accuracy rate because by merely judging most samples as usual, the significance of the accuracy rate does not provide reliable information for network maintenance. For example, if 90% average network traffic data and 10% abnormal network traffic data exist, then the classifier needs to judge all of them only as regular network traffic data to be 90% correct. However, in reality, this 90% correct rate is far less meaningful than being able to correctly identify 10% of the anomalous network traffic data because the main problem with network anomaly detection is to detect minority samples because the minority class of network anomalous traffic data information can bring more meaningful information to network maintenance.

Problems such as data imbalance in network traffic anomaly detection have occurred in other areas with similar problems. The problem of imbalanced data is widespread across research areas. The oversampling technique is often used to solve the problem of unbalanced data. This technique addresses the imbalanced nature of the data by generating additional training samples for the minority samples. The most representative oversampling technique is synthetic minority oversampling (SMOTE). This oversampling technique targets the data itself, trying to solve the problem of data imbalance at the root; hence, it is suitable for various application situations.

In the course of research and practice, researchers have also found certain shortcomings in oversampling technology, mainly in the possible generation of noisy samples and mislabelled samples, and the raw data information is not processed. The interpolation scheme is prone to centralized overlap. Therefore, to address the problem of sample imbalance in network traffic anomaly detection, this paper proposes a data synthesis method for imbalanced samples, based on the SMOTE algorithm, to improve the synthesized samples to better adapt to network traffic anomaly detection and to enhance the effectiveness of network traffic anomaly detection as follows:

1. Based on the idea of using the clustering algorithm to avoid the synthesis of noisy samples, the current clustering SMOTE algorithm has many parameter configurations that are difficult to adjust, and for the problem of low adaptability to different data sets, the use of the FINCH clustering algorithm is proposed to reduce the parameters and improve the generalizability.
2. The distribution of the number of synthetic samples after clustering is improved. It avoids the extreme situation where the cluster has a small number of minority samples but distributes an excessive number of synthetic samples, giving an initial weight to minority samples considering their sparsity and safety.
3. The multi-layer perceptron is used to clean the synthetic samples while constantly correcting the weights of the minority samples and the multi-layer perceptron to learn new minority samples.

2 Related Work

In the 1990s, as increasingly more machine learning and data mining techniques became widespread, researchers were faced with a significant challenge: how to process data with the desired classification accuracy in the face of imbalanced distributions. Researchers began to identify research areas for imbalanced learning in 2000; various approaches to imbalanced learning have gradually been proposed.

Faced with unbalanced network data, one can improve the effect by changing the detection model itself. Yang et al. [1] proposed a new hierarchical distributed agent model for network risk assessment from the perspective of modifying the assessment model to avoid the imbalance of network traffic data. Another way to face the problem of unbalanced data is to use oversampling technology, which improves class balance by increasing the number of minority samples. The simplest oversampling method to balance a data set is simple random sampling, which is the random replication of minority samples so that the new

samples generated do not add more information. In 2002, Chawla et al. [2] proposed the SMOTE (synthetic minority oversampling technique) algorithm, and since then, the SMOTE algorithm has become the mainstream algorithm in imbalanced learning data enhancement, has been widely used in various fields, and has yielded subsequent variants. The SMOTE algorithm differs from the simple replication sample mechanism of random oversampling by synthesizing new samples between two minority samples by linear interpolation.

Fig. 1 shows the oversampling process of the SMOTE algorithm. The main process is as follows:

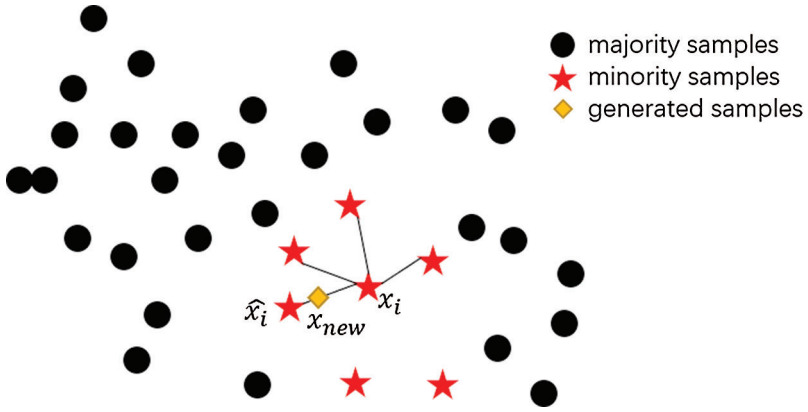

Figure 1: The SMOTE oversampling process

1) Randomly select x_i of the minority sample, calculate the distance from it to the other samples in the minority sample set based on the Euclidean distance and obtain the nearest K near-neighbor minority sample via the KNN algorithm. Then, set the K samples as $x_{i(near)}$, $near \in \{1, \ldots, k\}$.

2) A sample $\widehat{x_i}$ is randomly selected from the K samples, a random number w from 0 to 1 is generated, and a new sample x_{new} is synthesized by Eq. (1).

$$x_{new} = x_i + w(x_i - \widehat{x_i}) \tag{1}$$

3) Set a sample multiplier N according to the sample imbalance ratio, which is generally set to 1 by default, i.e., the number of samples generated is:

$$samples\_num = N * (majority\_num - minority\_num) \tag{2}$$

Repeat Steps 1) and 2) *samples_num* times until the number of minority samples equals the number of majority samples in the data set.

The classic SMOTE algorithm has certain drawbacks; thus, researchers have proposed many SMOTE-based variants for different data sets in their continuous research and practice. To avoid the possibility that the SMOTE algorithm will overlap with the majority samples, Han et al. [3] proposed an oversampling method, Borderline-SMOTE, for the boundary of the minority samples. The study of Devi et al. [4] revealed that the presence of redundant borderline instances and outliers in the data space severely catalyzes the effect of class imbalance. Barua et al. [5,6] proposed an algorithm to generate effective weight values for minority data samples based on each sample's proximity information, i.e., the distance from the boundary, which results in a proper distribution of generated synthetic samples across the minority data set. Perezortiz et al. [7] explored the notion of an empirical feature space to derive a kernel-based synthetic oversampling technique based on borderline instances, which are considered crucial for establishing the decision boundary. Xie et al. [8] proposed a new method called MOT2LD that first maps each training sample into

a low-dimensional space and then clusters their low-dimensional representations. For data noise problems, Sun et al. [9] applied the fractional wavelet transform in fault detection analysis to eliminate noise information and improve the recognition performance of weak fault features. Fernandez et al. [10] summarized the development of the SMOTE algorithm, which has been successfully used in various applications in many different fields due to its simplicity in programming and robustness for different types of problems. Georgios et al. [11] combined the K-means algorithm with the SMOTE algorithm to avoid noise among samples but also brought the problem of too many difficult parameters to adjust. In terms of image data enhancement, Liu et al. [12] introduced a new method to data augmentation and improved the diversity of limited data by changing the texture, contrast, and color of the image. Sarfraz et al. [13–16] proposed new clustering methods, which are suitable for oversampling to avoid the generation of wrong samples. Sandhan et al. [17] tried to balance the dataset with a mixed sampling method. Researchers such as Gao considered the improvement of weight and density in avoid overfitting [18–20]. Young et al. [21] proposed oversampling method based on voronoi diagrams and Douzas improved oversampling method based on self-organizing map [22]. Ramentol et al. [23–25] have applied unbalanced learning methods in machine diagnosis and intrusion detection.

In summary, researchers have explored various SMOTE variants and applications in different scenarios. However, the SMOTE and its variants still have the following problems:

1. The synthesis of new samples is prone to the generation of noisy samples; for example, the synthesis of new samples may occur in the majority sample areas.
2. Samples are synthesized without considering the informational importance of the original sample, e.g., a minority sample of the boundary class would provide more information and therefore require more attention.
3. Oversampling in combination with clustering algorithms usually requires more parameter adjustment, and the performance is unstable.

Therefore, we propose FSMOTE oversampling, which combines a stable FINCH clustering method without parameters and the use of multi-layer perceptron cleaning when synthesizing new samples.

3 FSMOTE: The Proposed Oversampling Method

3.1 Overview

Our proposed FSMOTE method uses a first-neighbor efficient covariant-free class algorithm and a multi-layer sensor combined with the SMOTE algorithm to rebalance the network traffic data set and ultimately achieve the goal of improving network anomaly detection. Our algorithm process is roughly as follows:

As shown in Fig. 2, first, calculate the imbalance ratio for the input imbalanced data set, and then use the FINCH clustering algorithm to cluster the original data set. After clustering, for each cluster, calculate the imbalance ratio. Note that at this time, there will be zero samples; hence, the number of calculations needs to be increased by 1 to facilitate the representation and calculation. The clusters in which the imbalance ratio is smaller than the imbalance ratio of the original data set are selected as filter clusters. The cluster sparsity is calculated according to the distribution of the minority samples in the filtered cluster in the overall minority sample, and the number of minority samples included in the cluster itself is considered. Finally, the number of synthesized samples is assigned to the filtered cluster. In the oversampling phase, considering the safety and sparsity of minority samples as initial weights, the minority samples closer to the majority class samples and the sparser samples around them will receive higher weights and be more homogeneous in the oversampling process. An MLP classifier is trained using the input data, and after synthesizing the new sample, the MLP is used to determine if it is a minority sample, to perform data cleaning and to reduce the weight of the minority sample that produces the noisy sample. The synthesis steps that require the sample size are eventually repeated to achieve a data balance. Fig. 3 shows the case of FSMOTE oversampling safe areas and clean noisy samples.

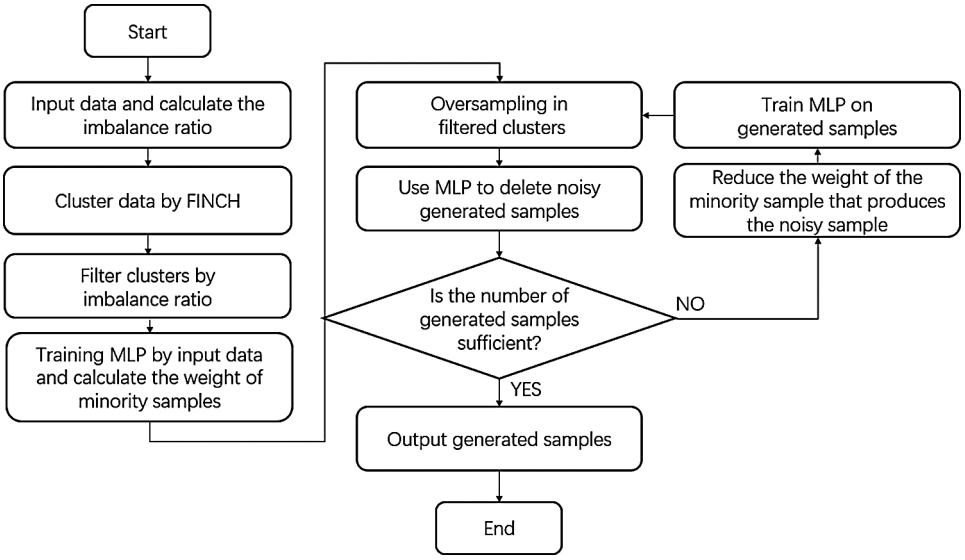

Figure 2: The FSMOTE oversampling process

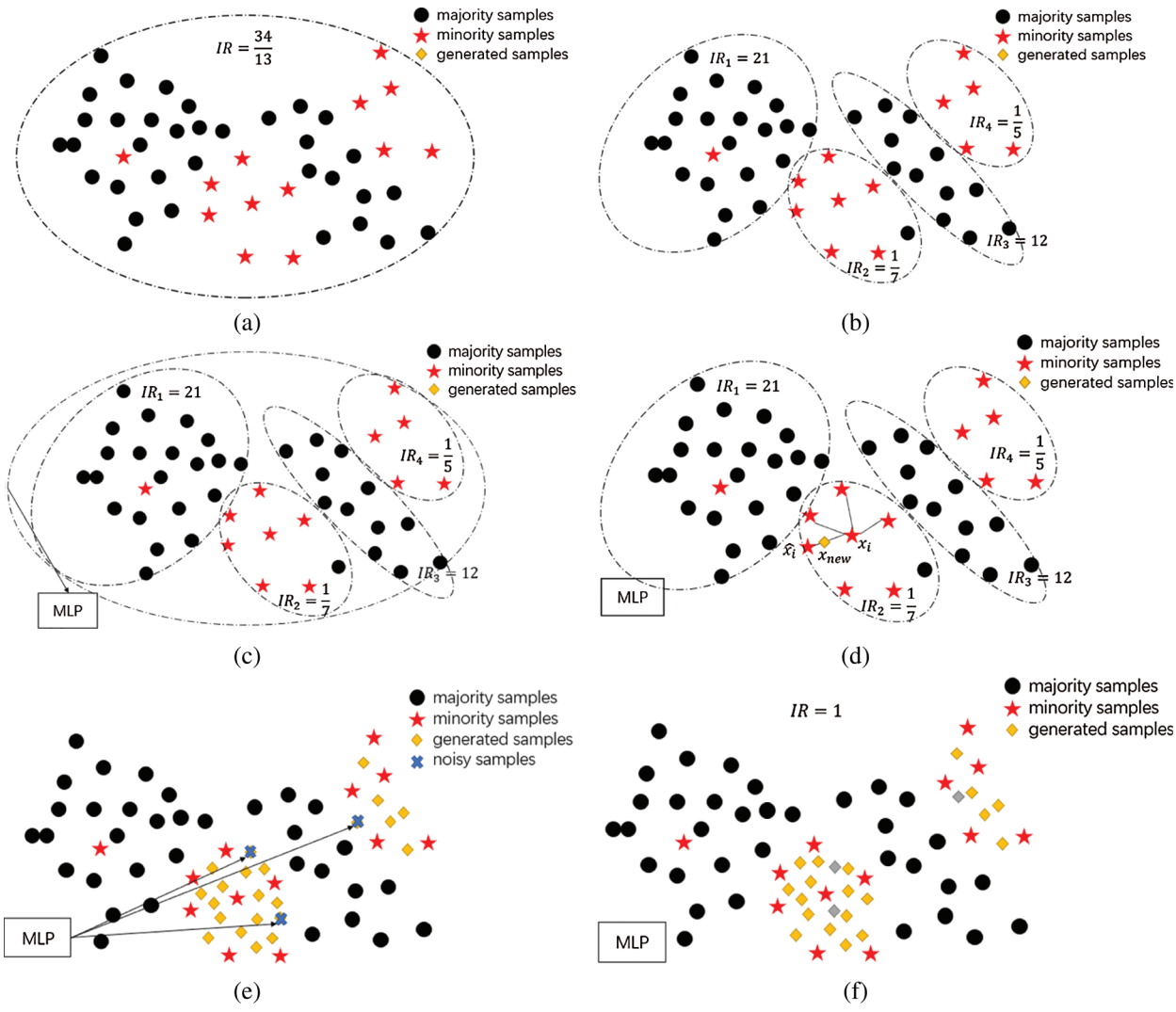

Figure 3: FSMOTE oversampling safe areas and clean noisy samples

It can be seen that our work is broadly focused on the following three modules:

1. Data pre-processing based on FINCH clustering.
2. Synthetic sample size allocation based on the cluster sparsity and sample size.
3. Data cleaning based on the multi-layer perceptron.

Therefore, in the following, we present data pre-processing based on FINCH clustering, methods for assigning a synthetic sample size based on the clustering sparsity and sample proportion, and data cleaning based on the multi-layer perceptron.

3.2 Data Pre-processing Based on FINCH Clustering

We first use the clustering algorithm to pre-process the incoming data, and effective clustering can form a safe region of the sample, improving the quality of the synthetic sample. The most representative oversampling method with clustering is KMeans-SMOTE, which requires three additional parameters to be set, requiring more time to find the best parameters in practice and having an unstable performance. Therefore we choose to introduce a simple and efficient clustering algorithm, i.e., FINCH, which requires no parameters, and use it to pre-process the input data for clustering.

FINCH is a clustering algorithm proposed by Sarfraz et al. [13]. The algorithm proposes a new clustering method in the form of a single clustering equation that directly discovers groups in the data. Compared to most existing clustering algorithms, FINCH does not require any hyperparameters, such as distance thresholds and the number of clusters. FINCH is a hierarchical aggregation method with very low computational overhead, easy to scale and suitable for large practical problems. Here, a brief introduction to the FINCH clustering process is given.

$$A(i,j) = \begin{cases} 1 & \text{if } j = k_i^1 \text{ or } k_j^1 = i \text{ or } k_i^1 = k_j^1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Given an integer index for the first neighbor of each data point, FINCH directly defines the neighbor link matrix according to Eq. (3). k_i^1 symbolizes the first neighbor of point i. The adjacency matrix links each point i to its first neighbor via $j = k_i^1$, enforces symmetry via $k_j^1 = i$, and links points (i, j) that have the same neighbor by using $k_i^1 = k_j^1$. Eq. (3) returns a symmetric sparse matrix directly specifying a graph whose connected components are the clusters. With its amazing simplicity, the clustering equations transfer clusters in the data without relying on any threshold or further analysis.

By using a directed graph or an undirected graph G = (V, E), one can efficiently obtain the component of connections from the adjacency matrix, where V is the set of nodes (points to be clustered) and edge E connects nodes A(i, j) = 1. Intuitively, the condition in Eq. (3) is to combine the 1-nearest neighbor (1-NN) and shared nearest neighbor (SNN) graphs. After the first partition has been calculated, these clusters can be merged recursively. For this, Eq. (3) requires the first neighbor of each cluster. Finding these neighbors requires calculating the distance between clusters. This is also relevant for all linking methods; for example, the average link in a hierarchical aggregation cluster uses the average of paired distances between all points in the two clusters. However, following this is computationally very expensive. Therefore, it cannot be easily scaled to millions of samples, and it needs to maintain a full distance matrix in memory. With the FINCH method, there is no need to compute a complete paired distance matrix for large-scale data since FINCH can obtain the first neighbor by a fast approximation of the nearest neighbor method.

3.3 Synthetic Sample Size Allocation Based on the Cluster Sparsity and Sample Size

After FINCH clustering, the original sample is transformed into multiple clusters. It is necessary to further determine which clusters are suitable for the synthesis of minority samples, that is, to filter the clusters. We generally use the imbalance rate of the original data as a benchmark. The imbalance rate is greater than the original data set imbalance rate, which means that it is a cluster in the majority sample set. Even if there are minority samples, this cluster is likely to belong to the noisy sample; thus, it is filtered. Conversely, clustering with an imbalance rate is less than the imbalance rate of the original data set means that the clusters in the minority sample set should be oversampled within the cluster.

After cluster filtering, there may be multiple minority clusters that need to be oversampled, and we allocate the number of synthetic samples for each filtered cluster based on the sparsity of the minority samples and the number of minority class samples in the cluster. This avoids the extreme case of considering only clustering sparsity or only the number of samples within the clusters. For example, if only the sample size is considered, there will be some cases where the sample size is small but sparse and more samples need to be synthesized, while if only the clustering sparsity is considered, the sample size can effectively avoid some extreme cases where the clustering sparsity is high but will itself be small. Fig. 4(a) shows the extreme case where only the number of clustered samples is considered, in which Cluster B is assigned a small number of synthetic samples due to the small number of minority samples and cannot achieve an effective equilibrium effect, and Fig. 4(b) shows the extreme case where only the sparsity of clusters is considered, in which Cluster B is very sparse and therefore assigned a larger number of synthetic samples, but its number is too small to generate a high sample repeatability.

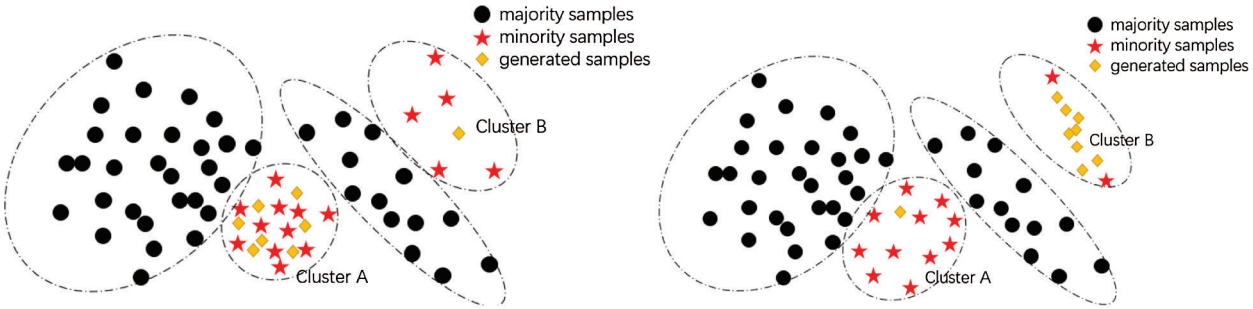

Figure 4: Problems when considering only the cluster sample size or sparsity

We therefore carried out an integrated consideration that exploited both the clustering sparsity and the number of samples in the clusters as a means of jointly determining the number of samples that a cluster needs to generate. The specific steps for cluster filtering and sample size allocation are as follows:

1) For each filtered cluster *fc*, calculate the Euclidean distance matrix for the minority samples in it.

2) Calculate the average distance of minority samples per filtered cluster, i.e., add all non-diagonal elements in the distance matrix and divide by the number of non-diagonal elements.

3) Calculate the sparsity factor for minority samples in a filtered cluster:

$$Sparsity\_factor(fc) = \frac{Average\ minortity\ distance(fc)^2}{minority\ count(fc)} \tag{4}$$

4) The sparsity weight for each filtered cluster is the sum of the sparsity factors of the filtered clusters divided by the sparsity factors of all filtered clusters.

5) The quantitative weight for each filtered cluster is the sum of the number of minority classes in the filtered cluster divided by the number of minority classes in all filtered clusters.

6) The final sampling weights for each filtered cluster are determined by a combination of the sparsity and the quantity weights.

7) Multiply the sampling weights by the total number of samples to be synthesized to obtain the number of samples to be synthesized for each filtered cluster.

3.4 Data Cleaning Based on the Multi-Layer Perceptron

The oversampling process is not effective at avoiding all noisy samples; thus, we use a multi-layer perceptron to make an initial judgement of the new samples synthesized and delete the synthesized samples that do not meet the conditions, and based on the judgement, we reduce the weight for the minority classes of samples that have generated noisy samples and then proceed to the next round of oversampling until the number of samples synthesized is sufficient. We chose the MLP because it is simple enough to be widely used and is more suitable for unbalanced network data sets because the number of samples in unbalanced data sets is usually small. We use the MLP classifier to judge the synthetic samples and provide feedback to adjust the weight of the minority samples, and the neural network continuously learns new information from the synthetic samples.

A multi-layer perceptron generally contains an input layer, an output layer, and a hidden layer, where the hidden layer may have multiple layers, whereas for a simple multi-layer perceptron, there is only one hidden layer, thus having a simple three-layer structure. The layers of the multi-layer perceptron are fully connected, meaning that the neurons in the previous layer are connected to all the neurons in the next layer. The neurons in the input layer provide information from the outside world and do not perform any computation in the input layer but only transmit information to nodes in the hidden layer. Hidden nodes in the hidden layer are not directly connected to the outside world; these nodes perform computations and pass information from the input layer to the output layer. The output layer is also responsible for the calculations that will output the information.

The approximate process of reassignment and sample cleaning is as follows:

1) **Calculation of the sparsity of minority samples:** Calculate the Euclidean distance matrix between all minority samples within each filtered cluster. *mintomin_distance* is the sum of the distances in each row divided by the sum of the triangles on the matrix. The sparsity of a minority sample is the proportion of the *mintomin_distance* of the sample to the total *mintomin_distance*.

$$Sparsity_i = \frac{mintomin\_distance}{\sum mintomin\_distance} \tag{5}$$

2) **Calculation of the safety of the minority samples:** Calculate the Euclidean distance matrix from each minority sample to each majority sample. *mintomaj_distance* is the sum of each row of the matrix divided by the sum of all the numbers in the matrix. Calculate the sample safety for the minority samples; the closer the sample point is to the majority class, the less safe it will be. num_{min} represent the number of minority samples.

$$Safety_i = \frac{\left(1 - \frac{mintomaj\_distance}{\sum mintomin\_distance}\right)}{minority\_num - 1} \tag{6}$$

3) **Minority sample weight calculation:**

$$minweights_i = \partial_1 * Sparsity_i + \partial_2 * Safety_i \tag{7}$$

where $minweights_i$ denotes the weight of sample x_i, ∂_1 *and* ∂_2 are the mean density factors, the sum of which is 1, and the sample density is determined by the sparsity and safety of the two samples together.

4) **Assign weights to the minority samples and train an MLP determinator on the original samples.**

5) **Oversampling and cleaning of the samples:** The oversampling and cleaning process is repeated when the number of synthetic samples is less than the required number to be generated.

a) Oversampling is carried out while retaining the minority sample sequences for each selection.

b) The MLP adjudicator makes judgements on the synthetic samples, deleting synthetic samples judged to be in the majority and keeping synthetic samples judged to be in the minority as new samples.

c) The weight of the minority samples that generate the majority samples is reduced.

d) The continuous learning characteristics of neural networks are utilized, and new samples are input into the MLP classifier for training.

3.5 FSMOTE Algorithm

In summary, we have introduced several important modules of this algorithm. The specific algorithm flow is as follows:

Algorithm 1: FSMOTE Algorithm

Input: Imbalanced data set X.

Imbalanced data set label y, include *minority* and *majority.*

Newly synthesized sample ratio N, the default value of which is 1.

Number of nearest neighbor samples K.

Output: Generated samples *Newsamples*.

1. **Calculate the imbalance rate Irt of the data set.**

$$Irt = \frac{count(majority(X))}{count(minority(X))}$$

2. **Calculate the number of samples to be generated.**

$$samples\_num = N * (count(majority) - count(minority))$$

3. **Generate clusters of the dataset by the FINCH algorithm and filter these clusters.**

$clusters \leftarrow FINCH(X)$

$filtered\_clusters = \varnothing$

for c∈ clusters:

$$c\_Irt = \frac{count(majority) + 1}{count(minority) + 1}$$

if c_ Irt < Irt:

$$filtered\_clusters = filtered\_clusters \cup \{c\}$$

4. **Calculate the number of samples to be generated for each cluster in** *filtered_clusters*.

for fc $\in$ *filtered_clusters*:

$$Average\ minortity\ distance(fc) = mean(Euclidean\ distances(f))$$

(Continued)

Algorithm 1 (continued).

$$Sparsity\_factor(fc) = \frac{Average\ minortity\ distance(fc)^2}{minoritycount(fc)}$$

$$Sparsity\_weight(fc) = \frac{Sparsity\_factor(fc)}{sum(Sparsity\_factor(fc))}$$

$$num\_weight(fc) = \frac{count(minority(fc))}{count(sum(minority(X)))}$$

$$fc\_weight(fc) = \partial_1 * num\_weight(fc) + \partial_2 * Sparsity\_weight(fc)$$

$$fc\_num(fc) = fc\_weight(fc) * samples\_num$$

5. **Calculate the minority sample weight in filtered_clusters:**

for fc $\in$ filtered_clusters :

Calculate the Euclidean distance matrix between all minority samples in each filtered cluster

$$mintomin\_distance = \frac{sum\ (row\ of\ the\ matrix)}{sum(the\ triangle\ of\ the\ matrix)}$$

$$Sparsity = \frac{mintomin\_distance}{\sum mintomin\_distance}$$

Calculate the Euclidean distance matrix from each minority sample to each majority sample

$$mintomaj\_distance = \frac{sum\ (row\ of\ the\ matrix)}{sum(matrix)}$$

$$Safety = \frac{\left(1 - \frac{mintomaj\_distance}{\sum mintomin\_distance}\right)}{minority\_num - 1}$$

$$minweights = \frac{Sparsity + Safety}{2}$$

6. **Oversampling.**

newsamples $= \varnothing$

Train the MLP classifier on X

for fc $\in$ filtered_clusters:

generated_samples $= \varnothing$

while num(generated_samples) $<$ fc_num(fc):

$temp\_samples = SMOTE(fc, fc\_num(fc) - num(generated\_samples), K)$

$mintemp\_samples, majtemp\_samples \leftarrow MLP(temp\_samples)$

$minweights \leftarrow 0.5 * minweights(majtemp\_samples)$

$generated\_samples = generated\_samples \cup \{mintemp\_samples\}$

train the MLP classifier on X$\cup\{$ *generated_samples*$\}$

$newsamples = newsamples \cup \{$ *generated_samples*$\}$

return *newsamples*

4 Performance Evaluation Results

4.1 Comparison Algorithm

To verify the effectiveness of our proposed oversampling method, the unsampled model was selected as the base model in the simulation, and three mainstream oversampling models, i.e., SMOTE [2], Borderline-SMOTE [3] and KMeans-SMOTE [11], were selected for comparison.

SMOTE is the most classic oversampling algorithm. The basic idea is to analyze the minority samples and artificially synthesize new samples based on the minority samples to add to the data set. Despite its flaws, but due to its simplicity, it has a good performance on some data sets, and this oversampling idea is still mainstream, derived from many variants of the algorithm; thus, it still plays an important role in imbalanced learning.

Borderline-SMOTE is an improved SMOTE-based algorithm for boundary samples that uses only minority samples on the boundary to synthesize new samples, thereby improving the class distribution of the sample. The Borderline-SMOTE sampling process divides minority samples into three classes: safe, dangerous, and noisy. Borderline-SMOTE uses only dangerous samples during oversampling.

KMeans-SMOTE is the latest improved SMOTE-based algorithm and has been widely used in recent years. It uses the simple and popular k-means clustering algorithm combined with SMOTE oversampling to rebalance the data set. It manages to avoid noise by oversampling only in safe areas. It differs from related methods not only because of its low complexity but also because it generates samples based on clustering distributions.

4.2 Data Set

Five imbalanced data sets resampled from KDDCUP99 are used in this paper. The data set comprises nine weeks of network connectivity data collected from a simulated U.S. Air Force LAN. Two network anomalies are included in the five imbalanced data sets extracted, and the numbers of both are very imbalanced and suitable for the evaluation of imbalanced network traffic anomaly detection. We also use part of the UNSW_NB15 data, a network data set for an intrusion detection system with a lower imbalance rate but a larger sample size and more complex features. Tab. 1 Shows the information of data sets.

Table 1: Data sets used in the experiment

Datasets	Instances	Features	IR
kddcup-buffer_overflow_vs_back	2233	41	73.43
kddcup-guess_passwd_vs_satan	1642	41	29.98
kddcup-land_vs_portsweep-names	1061	41	49.52
kddcup-land_vs_satan-names	1610	41	75.67
kddcup-rootkit-imap_vs_back-names	2225	41	100.14
UNSW_NB15	175341	44	2.13

4.3 Performance Measures

Because this paper is oriented towards imbalanced network traffic data, some of the traditional indicators used to evaluate network traffic anomaly detection are not applicable. This paper selects indicators that are more suitable for evaluating the classification effect of imbalanced data sets. The positive samples in this article are minority samples, and the negative samples are majority samples.

The evaluation indicators in this paper are first based on the four basic indicators TP, FN, FP and TN of the confusion matrix. The recall rate represents how many of the samples judged to be positive are true positive, the specificity rate represents how many of the samples judged to be negative are true negative, and the accuracy rate indicates how many positive samples are judged correctly. This article focuses more on the accuracy rates and recall rates.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$Specificity = \frac{TN}{TN + FP} \tag{10}$$

On this basis, two three-level indicators, i.e., the F1-score and G-mean, commonly used in imbalanced data sets, are introduced.

$$\text{F1Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

$$\text{Gmean} = \sqrt{Recall * \text{Specificity}} \tag{12}$$

The F1-score integrates the results of the precision and recall. The values range from 0 to 1, with 1 representing the best model output and 0 representing the worst model output. The G-mean indicator integrates the results of the output of the recall and specificity. The larger the G-mean value is, the better the sample evaluation effect of the two categories.

The PR curve is a curve with two variables, i.e., precision and recall, where recall is the horizontal coordinate and precision is the vertical coordinate. The model can be evaluated according to the size of the area under the curve (AUC). The area is represented by the area under the precision-recall curve (AUPRC), and the larger the AUPRC is, the better the model effect.

4.4 Results

To verify the effectiveness of the proposed algorithm in network traffic anomaly detection, two classifiers commonly used in network traffic anomaly detection, i.e., the logistic regression (LR) and simple Bayesian (BNB) classifiers, were selected, and both were implemented based on sklearn. These two classifiers are widely used in classification applications and are easy to implement. The logistic regression classifier assumes that the data obey the Bernoulli distribution. After maximizing the likelihood estimation method, the gradient descent method is used to solve the parameters to achieve the purpose of classifying the data. Bayesian classifiers are the classifiers with the smallest classification error probability or the lowest average risk with a given cost. Its design method is a basic statistical classification method. The classification principle is to calculate the posterior probability by using the Bayesian formula through the prior probability of an object, that is, the probability that the object belongs to a certain class, and to select the class with the largest posterior probability as the class to which the object belongs.

Fig. 5 is a comparison of the AUPRC, F1-score and G-mean under the detection of the LR classifier for different oversampling models. First, it is clear that the FSMOTE proposed in this paper achieves first place on three of the five data sets, which shows that the proposed algorithm is more stable and better than the other oversampling methods. Using the F1-score metric as an example, on the kddcup_BvsB data set, the FSMOTE was better by 5%, 2%, and 3% over the original model, SMOTE, and Borderline-SMOTE,

respectively, and tied with KMeans-SMOTE. On the kddcup_GPvsS data set, the FSMOTE was 4%, 2%, 2%, and 5% better than the original model, SMOTE, Borderline-SMOTE, and KMeans-SMOTE. On the kddcup_LvsS data set, the FSMOTE was 5%, 2%, 2%, and 2% better than the original model, SMOTE, Borderline-SMOTE, and KMeans-SMOTE. On the kddcup_LvsP data set, the SMOTE was 2% better than the original model, SMOTE, KMeans-SMOTE, and Borderline-SMOTE in parallel. The other two indicators show a similar pattern. Although KMeans-SMOTE and Borderline-SMOTE were able to achieve similar performance to the FSMOTE on some data sets, neither of them performed as consistently as the FSMOTE, presenting different results on the five data sets. In particular, this is because KMeans-SMOTE itself requires more parameter tuning, which results in it being less stable than the other models. The FSMOTE also adopts the FINCH clustering algorithm without parameter adjustment and optimizes the oversampled sample size distribution after clustering, which improves the stability from the results. However, it is also noted that on the kddcup_LvsP data set, all models achieved the same results, suggesting that under partially extreme conditions, despite the imbalance of the data set, there is essentially no need for further equilibrium adjustment and optimization because the classification boundaries themselves are more obvious. Since the oversampling algorithm is specific to the data itself, it is normal to have different outcomes for different data sets.

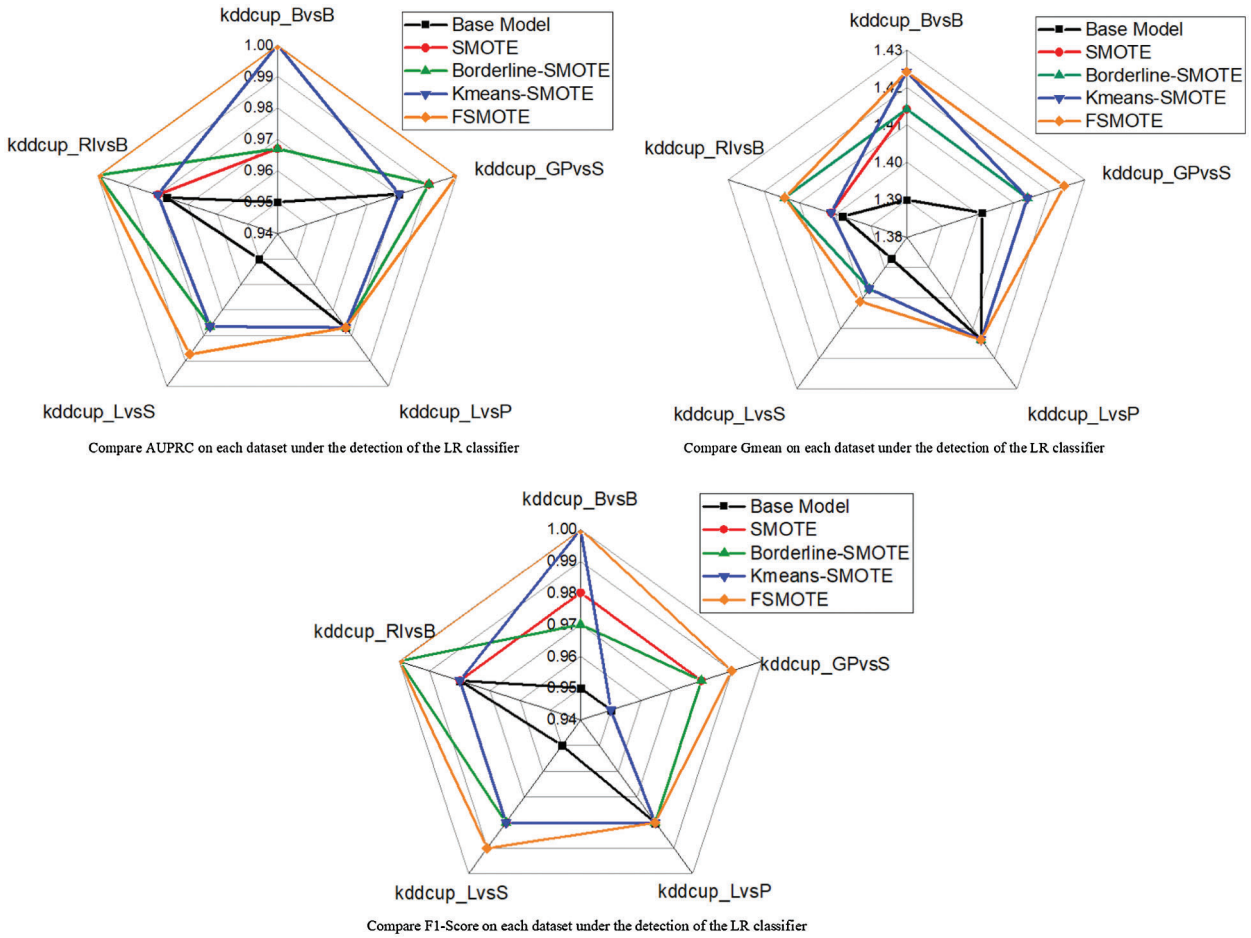

Figure 5: Comparison of the AUPRC, G-mean, and F1-score according to the LR classifier

Fig. 6 shows the result of using the Bayesian classifier to verify the boosting effect of oversampling. As shown, the FSMOTE algorithm proposed in this paper still achieves first place on three of the first four data sets and is only slightly behind KMeans-SMOTE on the last data set. Because the three metrics present similar rankings, we analyze them as F1-scores. On the kddcup_BvsB data set, a large gap emerged when the Bayesian classifier was used for classification, with only Borderline-SMOTE and FSMOTE maintaining similar detection effects to those achieved when the LR classifier was used. On the kddcup_GPvsS data set, the other oversampling failed to achieve an effect boost, and only the FSMOTE synthesized a valid sample, resulting in a 1% effective boost upon final classification. On the kddcup_LvsP data set, the original model, Borderline-SMOTE and KMeans-SMOTE all achieved poor detection and showed no improvements, while the FSMOTE was better than the SMOTE by 7%. On the final kdcup_RlvsB data set, the FSMOTE and Borderline-SMOTE were compared side-by-side, being 4% and 2% better than the original model and SMOTE, respectively, but slightly lower than KMeans-SMOTE. Compared to the LR classifier in Fig. 2, the Bayesian classifier is more inclined towards the majority class on the imbalanced data set and performs more erratically on different data sets; thus, in the next simulation experiment, we chose the LR classifier and the UBSW_NB15 data set for further verification.

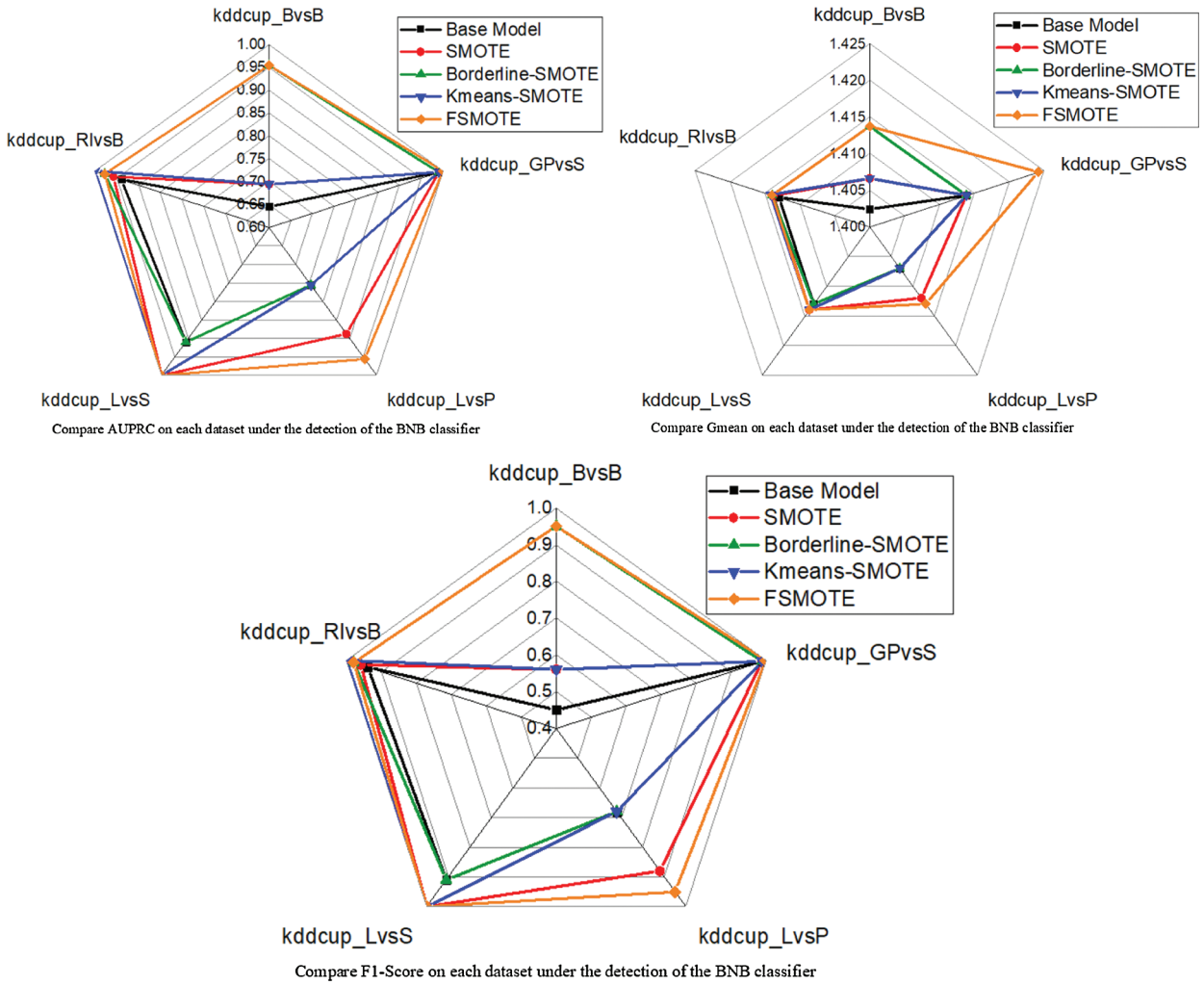

Figure 6: Comparison of the AUPRC, G-mean and F1-score according to the BNB classifier

Fig. 7 shows a comparison of the indicators using the LR classifier and the UBSW_NB15 data set. The FSMOTE still leads on all three metrics, suggesting that its synthesis quality is better than that of the other samplers for the minority samples. Regarding the F1-score, network traffic anomaly detection via the FSMOTE is 30% better than direct anomaly detection on the original data set, 10% better than the SMOTE, and 1% better than Borderline-SMOTE and KMeans-SMOTE. Regarding the AUPRC, the FSMOTE with the LR classifier was better by 2.4%, 2.3%, 0.7% and 0.1% over the original model, SMOTE, Borderline-SMOTE, and KMeans-SMOTE, respectively, and ranked the same on the G-mean as for the previous two metrics. The experiment further illustrates that the FSMOTE has a more stable performance than that of the other models for network traffic anomaly detection data sets. Overall, the oversampling methods have a certain enhancement effect on the imbalanced data set, but it is easier to depend on the quality of the data set and the classifier model. The FSMOTE in this paper is more stable and performs better under different data sets and classifiers. The reason for the improved performance of the FSMOTE is that it avoids synthetic noisy data more effectively through clustering and data cleaning. At the same time, because it refers to the minority sample position to give weight to it, it makes the samples at the boundary easier to synthesize, that is, the minority and majority samples are clearer. Thus, the effect on the logistic regression classifier is improved.

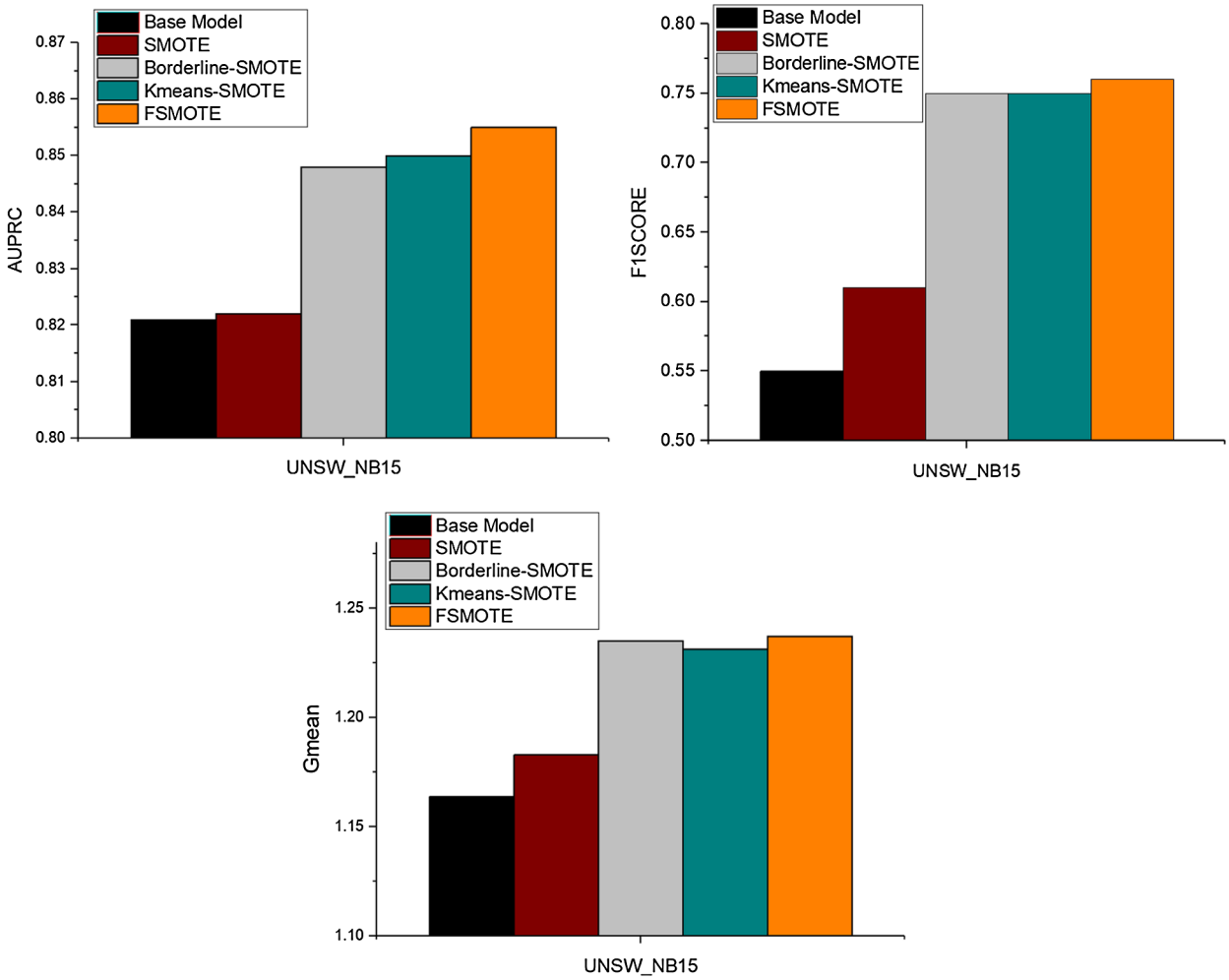

Figure 7: Comparison of the AUPRC, F1-score, and G-mean according to the LR classifier on UNSW_NB15

In summary, the FSMOTE oversampling method proposed in this paper has better stability on an imbalanced network traffic data set than do other comparison methods, avoids a certain degree of noisy sample generation, ultimately improves the data set balance, adapts to common Bayesian and logistic regression classifiers, and improves the logistic regression classifier even more.

5 Conclusions

Network anomaly detection plays a vital role in network operation and maintenance, but there is inevitably an imbalance in the network traffic data. Specifically, the standard network data samples tend to be more than abnormal, but the abnormal network data need to be identified. Therefore, to address the problem of imbalanced network traffic data, we adopt the oversampling method to adjust, explore and summarize the data according to the current oversampling method and propose a new oversampling method, i.e., FSMOTE. The FSMOTE method first uses a parameter-free hierarchical clustering algorithm based on the first nearest neighbor to cluster the samples and filter the clusters that need to be sampled. Then, the FSMOTE assigns initial weights to the minority samples according to their sparsity and security and continuously adjusts the weights and learns the synthetic sample information while cleaning the synthetic samples with multi-layer sensors during oversampling. The method finally completes the oversampling of the data set and improves the data balance. We verify that the data set processed by the oversampling algorithm proposed in this paper can improve the classification effect of classifiers commonly used for network anomaly detection, especially on minority samples.

Funding Statement: This work is supported by National Key Research and Development Program of China (2019YFB2103200), the Fundamental Research Funds for the Central Universities (500419319 2019PTB-019), Open Subject Funds of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory (SKX182010049), the Industrial Internet Innovation and Development Project 2018 & 2019 of China.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Yang, T. Li, G. Liang, W. He and Y. Zhao, "A hierarchy distributed-agents model for network risk evaluation based on deep learning," *Computer Modeling in Engineering & Sciences*, vol. 120, no. 1, pp. 1–23, 2019.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[3] H. Han, W. Wang and B. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *Int. Conf. on Intelligent Computing*, Springer, Berlin, Heidelberg, pp. 878–887, 2005.

[4] D. Devi, S. K. Biswas and B. Purkayastha, "Redundancy-driven modified tomek-link based undersampling: A solution to class imbalance," *Pattern Recognition Letters*, vol. 93, pp. 3–12, 2017.

[5] S. Barua, M. M. Islam and K. Murase, "A novel synthetic minority oversampling technique for imbalanced data set learning," in *Int. Conf. on Neural Information Processing*, Springer, Berlin, Heidelberg, Part II, pp. 735–744, 2011.

[6] S. Barua, M. M. Islam and K. Murase, "Prowsyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning," in *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Springer, Berlin, Heidelberg, pp. 317–328, 2013.

[7] M. Perezortiz, P. A. Gutierrez and C. Hervas-Martinez, "Borderline kernel based over-sampling," *Hybrid Artificial Intelligence Systems*, pp. 472–481, 2013.

[8] Z. Xie, L. Jiang and T. Ye, "A synthetic minority oversampling method based on local densities in low-dimensional space for imbalanced learning," *Database Systems for Advanced Applications*, Springer, Hanoi, Vietnam, pp. 3–18, 2015.

[9] Y. Sun, H. Chen, L. Tang and S. Zhang, "Gear fault detection analysis method based on fractional wavelet transform and back propagation neural network," *Computer Modeling in Engineering & Sciences*, vol. 121, no. 3, pp. 1011–1028, 2019.

[10] A. Fernandez, S. Garcia, F. Herrera and N. V. Chawla, "Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *Journal of Artificial Intelligence Research*, vol. 61, pp. 863–905, 2018.

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

[12] D. Liu, J. Zhao, A. Xi, C. Wang and X. Huang *et al.,* "Data augmentation technology driven by image style transfer in self-driving car based on end-to-end learning," *Computer Modeling in Engineering & Sciences*, vol. 122, no. 1, pp. 593–617, 2020.

[13] M. S. Sarfraz, V. Sharma and R. Stiefelhagen, "Efficient parameter-free clustering using first neighbor relations," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*Long Beach California, pp. 8934–8943, 2019.

[14] L. L. He, J. Wang, H. T. Bai, Y. Jiang and T. L. Li, "A novel parallel and distributed magnetotelluric inversion algorithm on multi-threads workloads cluster," *Cluster Computing*, vol. 22, no. 4, pp. 1073–1083, 2019.

[15] L. L. Zhou, F. Tan, F. Yu and W. Liu, "Cluster synchronization of two-layer nonlinearly coupled multiplex networks with multi-links and time-delays," *Neurocomputing*, vol. 359, pp. 264–275, 2019.

[16] S. S. Li, T. J. Cui and J. Liu, "Research on the clustering analysis and similarity in factor space," *Computer Systems Science and Engineering*, vol. 33, no. 5, pp. 397–404, 2018.

[17] T. Sandhan and J. Y. Choi, "Handling imbalanced datasets by partially guided hybrid sampling for pattern recognition," in *Int. Conf. on Pattern Recognition*, pp. 1449–1453, 2014.

[18] M. Gao, X. X. Hong, S. Chen, C. J. Harris and E. Khalaf, "PDFOS: PDF estimation based over-sampling for imbalanced two-class problems," *Neurocomputing*, vol. 138, pp. 248–259, 2014.

[19] F. Koto, "Smote-out, smote-cosine, and selected-smote: An enhancement strategy to handle imbalance in data level," in *Int Conf. on Advanced Computer Science and Information Systems*, Jakarta, Indonesia, pp. 280–284, 2014.

[20] S. Barua, M. M. Islam, X. Yao and K. Murase, "Mwmote-majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2014.

[21] W. A. Young, S. Nykl, G. R. Weckman and D. M. Chelberg, "Using Voronoi diagrams to improve classification performances when modeling imbalanced datasets," *Neural Computing and Applications*, vol. 26, no. 5, pp. 1041–1054, 2015.

[22] G. Douzas and F. Bacao, "Self-organizing map oversampling (somo) for imbalanced data set learning," *Expert Systems with Applications*, vol. 82, pp. 40–52, 2017.

[23] E. Ramentol, I. Gondres, S. Lajes, R. Bello and Y. Caballero *et al.,* "Fuzzy-rough imbalanced learning for the diagnosis of high voltage circuit breaker maintenance," *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 134–139, 2016.

[24] L. Tan, C. Li, J. M. Xia and C. Jun, "Application of self-organizing feature map neural network based on k-means clustering in network intrusion detection," *Computers Materials & Continua*, vol. 61, no. 1, pp. 275–288, 2019.

[25] Y. T. Chen, J. Xiong, W. H. Xu and J. W. Zuo, "A novel online incremental and decremental learning algorithm based on variable support vector machine," *Cluster Computing*, vol. 22, no. 3, pp. 7435–7445, 2019.