Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3794579, 17 pages
https://doi.org/10.1155/2022/3794579

Research Article

Application of a New Feature Generation Algorithm in Intrusion Detection System

Yingchun Niu[1], **Chengdong Chen**[2], **Xuehua Zhang**[3], **Xiaoguang Zhou**[2] and **Hongjie Liu**[1]

[1]*Beijing University of Posts and Telecommunications, Beijing, China*
[2]*Minjiang University, Fuzhou, China*
[3]*National Disaster Reduction Center of China, Beijing, China*

Correspondence should be addressed to Xiaoguang Zhou; zxg@bupt.edu.cn

Received 6 September 2021; Revised 20 November 2021; Accepted 6 January 2022; Published 31 January 2022

Academic Editor: Wenjuan Li

The intrusion detection system is designed to discover the abnormal behavior of the network system, but it has the problems of low detection accuracy, inability to perform fine detection, and huge time cost. Therefore, it is necessary to design a fast and accurate intrusion detection system. Therefore, this paper proposes a multigranularity feature generation + XGBOOST method to improve the intrusion detection system. First, we propose a multigranularity feature generation algorithm, which converts all features into discrete features with different numbers of categories. Different numbers of categories represent different granularities. We believe that the combination of multiple different granular features can achieve better accurate attack detection. Then, we use the proposed method to perform experimental verification on the four datasets of KDD99, NSL-KDD, UNSW_NB15, and CSE-CIC-IDS2018. For the KDD99 dataset, detection rates of 100%, 100%, and 99.43% can be achieved in the two-category, five-category, and multicategory tasks, respectively; for the NSL-KDD dataset, detection rates of 100%, 100%, and 90.84% can be achieved in the two-category, five-category, and multicategory tasks, respectively; for the UNSW_NB15 dataset, 100% detection rate can be achieved in the second and tenth categories; for the CSE-CIC-IDS2018 dataset, 100% detection rate can be achieved in the third classification. Experiments show that the proposed algorithm can achieve accurate and precise detection. Finally, we experiment with the multigranularity feature generation algorithm on multiple classifiers and multiple datasets to prove the generalization ability of the proposed feature generation algorithm and compare the proposed algorithm with the CFS algorithm to prove the efficiency of the algorithm.

1. Introduction

With the development of the Internet, the number of netizens is gradually increasing, which will lead to a larger and more complex network, which puts forward higher requirements for the transmission rate and security of the Internet. At the same time, a variety of new attack techniques emerge in an endless stream, which poses a higher challenge to the network security system. Therefore, how to design a fast, efficient, and accurate prevention system is the primary problem that researchers have to solve at present.

Various systems have been designed in the past to identify and prevent Internet-based attacks. The most important system is the intrusion detection system (IDS), which can effectively resist external attacks. IDS can detect different types of attacks in network communications, but traditional firewalls cannot resist these attacks well. The intrusion detection system is based on the assumption that the behavior of the intruder is different from that of the legitimate user to identify and prevent the attack. Generally speaking, IDS is roughly divided into two types, anomaly detection system and misuse detection system, according to its detection method. Therefore, the development of an intrusion detection system can effectively prevent attacks and intrusions.

Intrusion detection can be considered as a classification problem in data mining. For classification problems, the general processing flow consists of four steps: data preprocessing, feature engineering, model building, and model evaluation. Data preprocessing and feature engineering play a key role in the effectiveness of the detection system. Good feature engineering greatly improves the accuracy of intrusion detection and reduces time cost.

To reduce model complexity, improve detection accuracy, and save time, many feature engineering methods are used in the field of intrusion detection to reduce the features of the dataset; that is, only some of the features of the original dataset are selected. Ganapathy et al. [1] made a good summary of the feature selection method and divided the feature selection method into stepwise feature removal method, improved feature selection algorithm based on mutual information, feature selection based on conditional random field, and wrapper based methods of genetic feature selection. Some scholars use other feature selection methods, such as information gain [2], CFS feature selection [3], RFS feature selection method [4], symmetrical uncertainty (SU) feature measurement method [5], and F-test [6]. Although these methods can reduce features, they also lose some information. It is also a good choice to improve the accuracy of intrusion detection without losing or increasing the characteristic information.

Different feature engineering methods will get different features, which need to be combined with classification algorithms to achieve intrusion detection. Based on the more classic datasets NSL-KDD and KDD99 in the field of intrusion detection, scholars have studied two-class detection, such as whether the detection is normal or abnormal [3, 7–10], to detect whether there is a DOS attack [11, 12]. Some scholars have carried out five classification tests, namely, Normal, Dos, Probe, U2R, and R2L [5, 9, 13], but this part of the research is relatively small. More detailed classification detection based on five classifications is even rarer.

For an intrusion detection system, if you know about anomalous attacks, it will undoubtedly be more valuable for security protection to know more accurate methods of intrusion attacks. However, this part of the research is scarcer than the two-class detection and the five-class detection. The current research has the method of Iwendi C. et al. [3]. They performed more accurate classification detection on the basis of two classifications. They used CFS for feature selection and then used ensemble learning algorithms for classification. Although CFS reduces features, the time cost of this algorithm is too high when selecting features.

So we hope to propose a faster feature generation method, and the intrusion detection system based on this method can perform more refined and accurate intrusion detection. In order to verify the performance of the improved intrusion detection system, we have used multiple datasets on multiple classifiers to conduct different degrees of experimental verification.

The main contributions of this article include the following:

(1) For discrete features in the intrusion detection dataset, the CatboostEncoder method is used to perform feature conversion

(2) A multigranularity feature generation (mgfg) algorithm for feature generation of network intrusion datasets is proposed

(3) An mgfg + Xgboost method for network intrusion detection is proposed

(4) The multigranularity feature generation (mgfg) algorithm is combined with the classifier algorithm to prove the effectiveness of the mgfg algorithm to generate features

(5) It can not only realize the detection of normal flow and abnormal flow but also realize more refined detection

(6) On the basis of not using other sampling techniques or data enhancement methods, the problem of data imbalance is basically solved

The remainder of this paper is organized as follows: Section 2 mainly reviews related research on intrusion detection; Section 3 mainly discusses the proposed method; Section 4 introduces the dataset used and conducts experiments to verify the proposed algorithm; Section 5 is dedicated to discussion, mainly the algorithm comparison; Section 6 gives the conclusion and future work prospects.

2. Related Research

In this section, first, we introduce the research of supervised learning algorithms in machine learning in the related fields of intrusion detection and then the application research of unsupervised learning algorithms. The last part introduces the research progress of deep learning technology.

In the research of applying machine learning algorithms in intrusion detection systems, scholars apply certain feature selection methods to select features, then select a certain classification method, and finally conduct a two-class or five-class experiment. For example, Panda proposed a new hybrid data mining method [7], specifically combining mutual Bayes and decision trees to propose the NBDT algorithm, combining the decision tree with the farthest traversal to propose the DTFF algorithm, and combining NNGE (nonnested generalized exemplars) with JRip (extended repeated incremental pruning) to propose the NNJR algorithm; the FFT and NNGE algorithm were combined to propose the FFNN algorithm; and the Bayesian belief network (BBN) and Tabu search were combined to propose the NNGE algorithm. Finally, the NSL-KDD algorithm was tested and verified, and a high abnormal detection rate was achieved. However, only two classification experiments were carried out on the KDD99 and NSL-KDD datasets, and no multiclass verification was carried out. In addition, the detection rate of this method for normal network access can reach 98%, but the detection rate for abnormal attacks is about 70%. Similarly, S. Sivanantham combined the traditional machine learning algorithm naive Bayes and random tree algorithm with enhancement technology and proposed

the Ada + NB algorithm and ADA + RT algorithm and also proposed the CBFWNB algorithm for weighting features [8]. The CBFWNB algorithm assigns weights to each feature value, not to each feature, so it is not a feature weighting method in the strict sense but a feature value weighting method. Then, the three algorithms are used to verify Intrusion Detection Kaggle Dataset and NSL-KDD dataset. Higher accuracy rate and detection rate and lower false alarm rate were achieved. The same problem is that no experiments were performed on more refined categories.

D. Sudaroli [14] focused on the applicability of traditional machine learning algorithms in wireless intrusion detection systems, introduced many methods to reduce false alarm rates, and verified them on the AWID dataset. However, the detection accuracy is not high. After that, he used the embedded ridge-based decrease method on the AWID dataset [15] to select features and used the ensemble classifier for classification, with an accuracy of 99.94%. Sumaiya Thaseen [16] proposed an intrusion detection model based on Chi-square feature selection and multiclass support vector machine in 2015 and then established an intrusion detection model using Chi-square feature selection and support vector machine and improved naive Bayes and LPBoost in 2017 [17]. However, only a good detection result was achieved on NSL-KDD dataset. They just verified their algorithm on a single dataset.

For multiple types of intrusion detection, many scholars have also provided some solutions. Basically, the multiclassification problem is converted into multiple two-classification problems and solved step by step. For example, Jivitesh Sharma et al. proposed a three-stage anomaly detection method [18]. First, the corresponding characteristics for each attack were selected separately. Then, the classifier was used to detect different types of attacks separately. Finally, softmax was used to combine the results of all the classifiers to get the probability output of each attack type. The idea behind it is that the multicategory task is difficult compared to the two-category task. Therefore, the multiclassification problem is divided into multiple two-classification problems. Finally, the authors tested it on the UNSW and KDD99 datasets. The experimental results showed that the multiclass classification accuracy rates of UNSW and KDD99 datasets reached 98.24% and 99.76%, respectively. In addition, the weighted extremum learning machine was used to alleviate the problem of unbalanced attack classification and further improve the attack performance. Finally, the integration of ELMs was implemented in parallel by GPU, and real-time intrusion detection was realized. Because the algorithm was a three-stage algorithm, as the number of data increased, the time cost of the algorithm would gradually increase, and the corresponding hardware cost would increase accordingly.

In addition, some scholars have introduced unsupervised learning algorithms into the field of intrusion detection. For example, J. V. Anand Sukumar proposed an improved genetic k-means algorithm [19] for network intrusion detection. The traditional k-means algorithm needs to manually set the "k" value, which is quite cumbersome for larger datasets. The IGKM algorithm uses a fitness function to quickly find the optimal "k" value. The fitness function is mainly measured by the intraclass distance fitness function and the intercluster distance function. Finally, it is tested on the KDD99 dataset. However, the limitation of this algorithm is that there are only hundreds of pieces of experimental data, and the final detection accuracy is only 72%, which indicates that there is a big gap compared with the supervised learning algorithm. But the method proposed by Pu G. is relatively more effective. His method is to detect known, unknown, and zero-time attacks. He proposed an unsupervised anomaly detection method, SSC-OCSVM algorithm [6], which combines subspace clustering and single-class support vector machine; the attack can be detected without any prior knowledge. The specific steps are to first initialize a feature vector D and divide the feature space X into N-word spaces and then use the OCSVM algorithm to calculate a partition p_i for each subspace. Then vector D is updated based on the basis, and, finally, vector D is sorted, and the value of vector D is compared with a predetermined threshold. If it is larger, the corresponding sample is considered to be an abnormal attack sample. Finally, the author uses the NSL-KDD dataset to evaluate the proposed method. It is verified on the three indicators of time, correct detection rate, and error detection rate, and, compared with the k-means algorithm and the DBSCAN algorithm, the result of the better one is obtained. However, the problem with this method is that it is only tested on thousands of sample sets, and only 4 classification experiments are performed. As the number of samples increases, this method must find a good feature selection method to solve the time cost problem.

With the development of artificial intelligence, deep learning technology has been introduced into various fields. Recently, some scholars have introduced deep learning technology into the field of intrusion detection. For example, Ömer KASIM integrated deep learning and machine learning algorithms and proposed the AE-SVM algorithm [20]. Autoencoding (AE) was used for feature extraction, and then the generated features were used in SVM; then it was compared with the traditional SVM algorithm and the PCA-SVM algorithm, and experiments were performed on the CIC-IDS dataset and the NSL-KDD dataset. In the case of unbalanced datasets, the DDoS flow detection test's success rate was 99.1%. However, this article only detects DDoS attacks and does not detect more types of attacks. In 2020, B. Riyaz [21] combined the proposed feature selection algorithm with a convolutional neural network and applied it to an intrusion detection system. The overall detection accuracy rate reached 98.88%. However, it was only verified on the KDD99 dataset.

Vinayakumar R. and others [22] completely applied deep learning technology network intrusion detection for five classifications, which can detect four types of attacks. Although the classification effect is better, they have not conducted experiments on more refined attack categories. They modeled network traffic events as a time series, analyzed the effectiveness of convolutional neural networks for intrusion detection, and compared the CNN algorithm with other algorithms. The results proved that, on the KDD99 dataset, the CNN-based algorithm was superior to other

algorithms. They also analyzed the effectiveness of mixing recurrent neural networks and convolutional neural networks, specifically using CNN as the first layer input, then using one-dimensional convolution and one-dimensional pooling to generate feature vectors, and then applying recurrent neural networks or their variants operation, and finally output the classification result. However, the results show that this hybrid model does not improve performance and is equivalent to CNN in most cases. Vinayakumar R. et al. also evaluated the effectiveness of different depths of the network for intrusion detection [23]. They first discussed the performance of deep belief networks and multilayer perceptrons of different scales in the two classifications of KDD99 and NSL-KDD datasets and compared them with LR, NB, KNN, DT, AB, RF, and SVM. The results show that the AUC value of DBN on KDD99 reaches 0.9997 and the AUC value on the NSL-KDD dataset reaches 0.9991, which are better than those based on traditional machine learning algorithms. Then they conducted a five-classification experiment, and the results showed that their method had a detection accuracy of less than 92% for the NSL-KDD and KDD99 datasets, and it could hardly detect PROBE, U2R, and R2L.

Recently, Sumaiya Thaseen Ikram [24] integrated Multilayer Perceptron, Back-Propagation Network, and Long Short-Term Memory through Xgboost. An anomaly recognition model was established, and almost perfect detection results were achieved on the UNSW-NB15 and VIT_SPARC20 datasets, but there was no further detailed intrusion detection classification experiment.

3. Proposed Method

The purpose of this paper is to propose a multigranularity feature generation (mgfg) method and apply the proposed feature generation method to an intrusion detection system so that IDS can achieve accurate prediction of multiple datasets of two classifications, five classifications, and multiple classifications. First, we use LabelEncoder encoding to preprocess the training dataset and then use the multigranularity feature generation method for feature generation to obtain features of different granularities. Then we enter the training classifier stage in the intrusion detection module and finally perform the model evaluation. Our IDS carried out not only two-class attack detection and five-class attack detection but also multiclass attack detection. When performing multiclass attack detection, some categories have very few samples, and our method can also detect well.

This section is divided into four stages, as shown in Figure 1. The first stage is data preprocessing, which mainly converts category labels. The second stage is to use the multigranularity feature generation method for feature generation. The third stage is algorithm training. Finally, in the fourth stage, the model is evaluated.

3.1. Data Preprocessing. As the first step of our method, data preprocessing is mainly to convert discrete type features into continuous features through coding methods. For example, for the KDD99 and NSL-KDD99 datasets, the three attributes and category labels of "protocol_type," "service," and "flag" need to be converted. For the tag column attributes, we use the Category Encoder class in the Category_encoders library to encode. We convert the label to a continuous positive integer starting with 0.

3.2. Multigranularity Feature Generation Algorithm. Although most of the intrusion detection research is to classify by reducing the features, we believe that this will lose the value of the data, and we add features based on the original data. First, new columns are generated by dividing continuous features with different equal-width bins, and then Catboost encoding [28] is performed.

Catboost is the open source machine learning library of Russian Yandex in 2017. Like the popular Xgboost [29] and LightGBM [30], it is a member of the Boosting family. Catboost solves the problems of gradient bias and prediction shift in the Boosting family algorithm, which improves the accuracy of prediction and solves the problem of overfitting to a certain extent. In addition, Catboost can also process discrete data efficiently and reasonably. Therefore, this article uses Catboost to convert discrete features into numerical features again and to generate more advanced features with different granularity.

In data analysis, the commonly used data types include continuous features and discrete features. The discrete features have relatively few values and have better distinguishability. Therefore, there is often an operation to convert continuous features into discrete features, which is called discretization. Discretization operation has many advantages. First, discretization can map finite individuals in infinite space to finite space, which can improve the space-time efficiency of the algorithm. Second, the discretized features are highly distinguishable. Finally, the discretized features are very robust to abnormal data.

There are usually five methods for the discretization of a continuous feature: binning, histogram analysis, clustering, decision tree analysis, and correlation analysis [27]. Relatively speaking, the binning method is relatively simple. There is usually equal width binning and equal frequency binning. The binning method often requires the user to specify the number of bins. Different numbers of bins will convert a continuous feature into discrete features with different values. Then, when the number of bins is large, the numerical feature column will be converted into a feature with multiple discrete values. We believe that the feature at this time represents a relatively fine-grained feature; similarly, when the number of bins is small, we think this represents a relatively coarse-grained feature.

Therefore, for a certain column of continuous features, to draw the features of different degrees and different granularities, first, we start the binning with 10 as the number of bins and then sequentially bin by 100, 1000, and so forth until reaching one half of the maximum value of the continuous feature; then, the discretization-generated feature column is labeled to complete the conversion of the continuous feature into multiple columns of features with

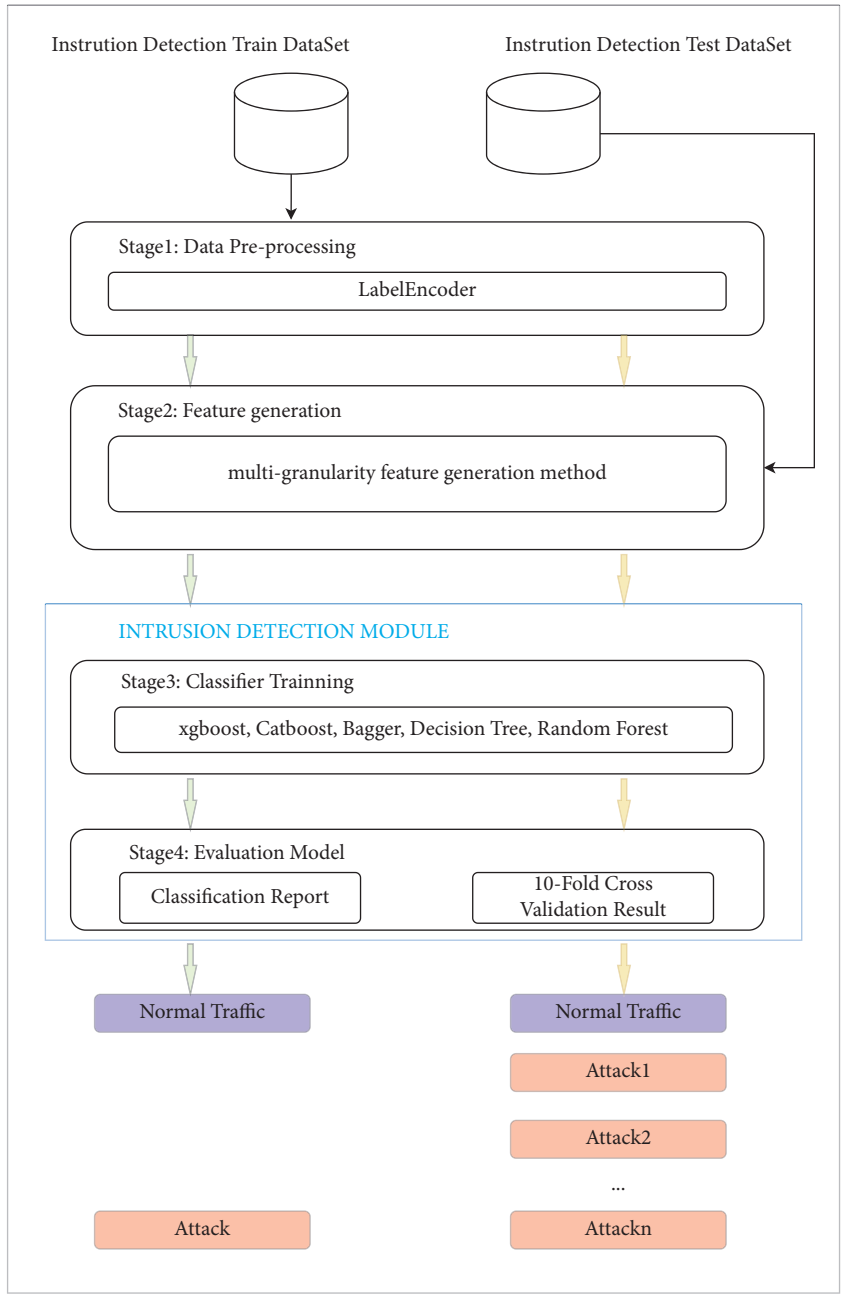

Figure 1: Improved intrusion detection system.

different granularities. Finally, the discrete features are vectorized; that is, the vectorized data combined with the attack category is converted into a new continuous feature. The conversion here is performed by the CatboostEncoder class. At this time, our data not only has a strong discriminability of discrete features but also has a strong correlation with the attack category. We call the above method a multigranularity feature generation algorithm.

The multigranularity feature generation method first obtains the maximum value of the numerical feature in the i-th column, as shown in formula (1), where $x_{\max}^{(i)}$ represents the maximum value of the i-th column of numerical features and $\max(x^{(i)})$ represents the maximum value of the i-th column of numerical features.

$$x_{\max}^{(i)} = \max\left(x^{(i)}\right). \tag{1}$$

Then, multiple new attribute columns are generated based on the characteristics of the i-th column. The following formula of N represents a set of different granularities, where α is a hyperparameter, which represents the granularity level, and larger α will result in a coarser granularity. In the same way, smaller α will result in a finer granularity, and here we default $\alpha = 10$.

$$N = \bigcup_{r=1}^{+\infty} \alpha^r, \tag{2}$$

where $\alpha^r < \text{sqrt}(x_{\max}^{(i)})$. Next, each element n in N is traversed and the value in $x^{(i)}$ is divided into n parts of equal width (i.e., the difference between the maximum value and the minimum value in each group is equal). Therefore, for every n, a new column is generated. The interval of the new column is as shown in the following equation:

$$w = \frac{\left(x_{\max}^{(i)} - x_{\min}^{(i)}\right)}{n}. \tag{3}$$

Formula (3) indicates that the $x^{(i)}$ column is divided into n intervals of equal size. w represents the width of the interval, so the boundary of the interval is $[\min, \min + w)$, $[\min + w, \min + 2w), \ldots, [\min + (n-1)w, \min + nw]$; then $x^{(i)}$ falls within an interval. Then the n interval boundaries are marked, and finally CatboostEncoder encoding is performed.

$$\widehat{x}_i^k = \frac{\sum_{j=0}^{j\le i}\left(y_j \times \left(x_j == k\right)\right) - y_i + \text{prior}}{\sum_{j=0}^{j\le i} x_j == k}, \tag{4}$$

where $\widehat{x}_i^k$ represents the i-th categorical feature of the k-th training sample, x_j represents the value of the j-th categorical feature, y_i and y_j represent the labels of the i-th and j-th samples, respectively, and "prior" represents the smoothing parameter.

3.2.1. Numerical Feature Processing. As shown in Figure 1, all feature data are generally divided into continuous features and discrete features. For each column of continuous features, we find the maximum value in turn and use 10 as the initial number of bins, then 100 as the number of bins, 1000 as the number of bins, and up to the maximum one half of that. At the same time as each binning, the Catboost_encoders operation is executed to convert the binning results into continuous features again. In this way, each bin for each column of features will add a new column of features to the original data. The pseudocode of the above process is shown in Algorithm 1, and the algorithm's steps are shown in Figure 2.

3.2.2. Discrete Feature Processing. For the original discrete features in the dataset, we can directly convert them into continuous features using Catboost_encoders encoding. The detailed operation is shown in Algorithm 2 and Figure 3.

3.3. Model Training. At this stage, the model is mainly trained using the features generated in the previous stage. Generally speaking, the performance of the ensemble learning model is better than that of a single weak classifier. Therefore, the most representative Xgboost ensemble learning classifier is selected. The objective function of Xgboost is shown in the following formula, where $l(y_i, \widehat{y}_i)$ represents the loss function and $\Omega(f_k)$ represents the regular term. Then the objective function is derived, and only one tree $f(t)$ is generated in each round. Then the objective function for the t-th round is shown in the following formula:

$$\begin{aligned} \tau(\varphi) &= \sum_i l(y_i, \widehat{y}_i) + \sum_k \Omega(f_k), \\ \tau^{(t)} &= \sum l\left(y_i, \widehat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t). \end{aligned} \tag{5}$$

This kind of thinking transforms the problem of a globally optimal solution into the process of seeking a locally optimal solution at every step, which is typical greedy thought. After that, we use Taylor Series to treat $f(t)$ as a variable and use the second derivative to approximate $\mathscr{L}^{(t)}$. Finally, the greedy algorithm is adopted to split from a node and repeatedly add branches to the tree to find the best split point. At this stage, the python toolkit Xgboost is mainly used for training without any regulation parameter.

3.4. Model Evaluation. The index for evaluating the performance of a classifier is generally classification accuracy. The commonly used evaluation indicators for two-class classification problems are precision and recall. Usually, the class of interest is positive, and the other classes are negative class. The prediction of the classifier on the test dataset is either correct or incorrect. The total number of the four situations is recorded as follows:

(i) TP: predict the positive class as a positive class number

(ii) FN: predict the positive class as a negative class number

(iii) FP: predict the negative class as a positive class number

(iv) TN: predict the negative class as a negative class number

Precision is defined as

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{6}$$

The recall rate is defined as

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{7}$$

In addition, there is the F_1 value, which is the harmonic mean value of the precision rate and the recall rate. Namely,

$$\begin{aligned} \frac{2}{F_1} &= \frac{1}{P} + \frac{1}{R}, \\ F_1 &= \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \end{aligned} \tag{8}$$

Finally, we use support to represent the number of occurrences of each class label sample.

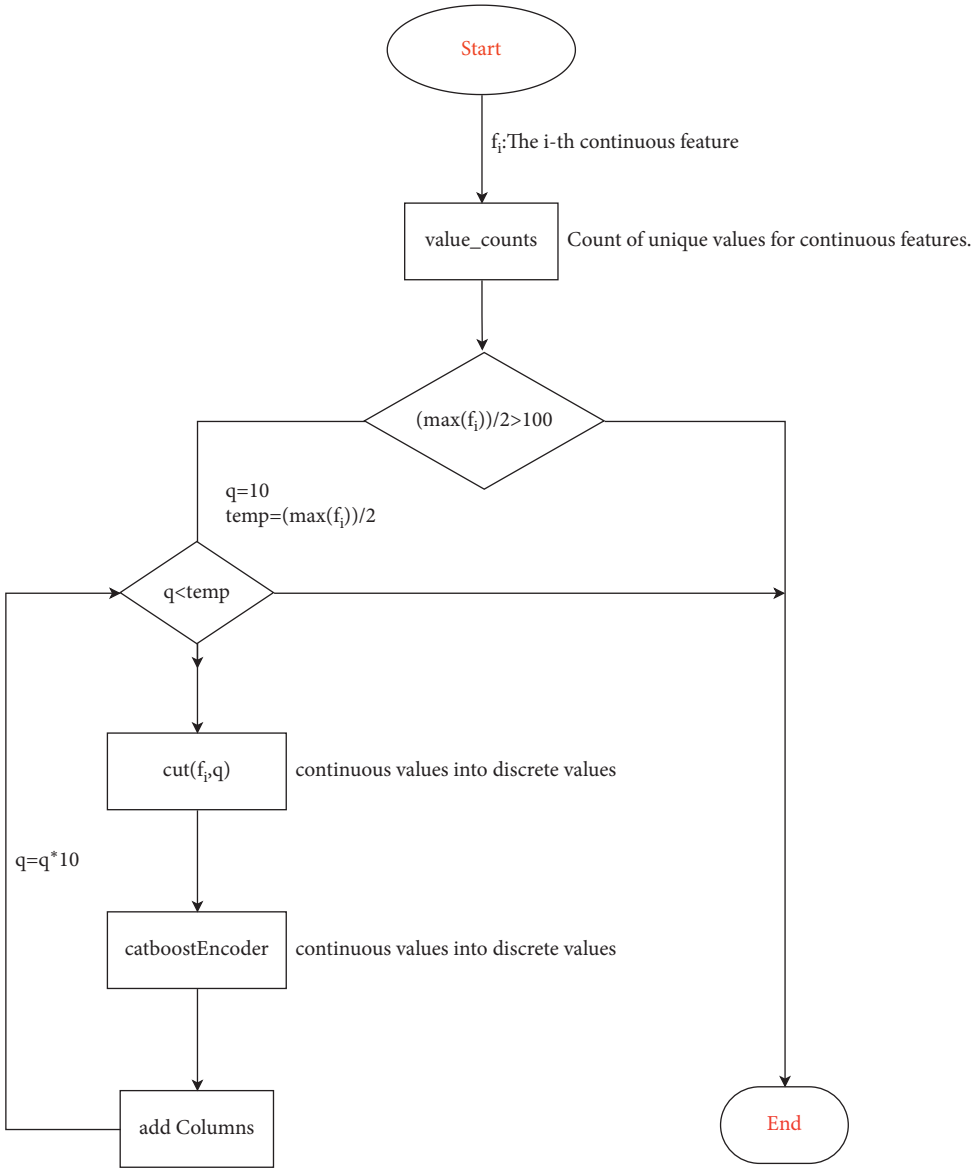

Figure 2: The processing steps of the mgfg algorithm for continuous features.

(i) Input: $F(f1, f2, f3, \ldots, fm)$//Continuous Feature Col
(ii) m: the number of feature cols
(iii) ε: 100
(iv) cut: Converts the f column attribute to the Q class
(v) Catboost_encoders: Converts the discrete attributes of the Q class to continuous values
(vi) **for** $i = 1$ **to** m **do**
(vii) temp = value_counts((f_i))
(viii) temp = (max(f_i))//2
(ix) If (temp > ε)
(x) **for** $q = 10$ **do**
(xi) temp_col = cut(f_i, q)
(xii) f_{i_q} = Catboost_encoders(temp_col)
(xiii) F.append(f_{i_q})
(xiv) $q = q * 10$
(xv) end
(xvi) end
Output: $F(f1, f2, \ldots, fm, \ldots, fn)$//Feature columns with different granularity.

Algorithm 1: mgfg method for continuous feature columns.

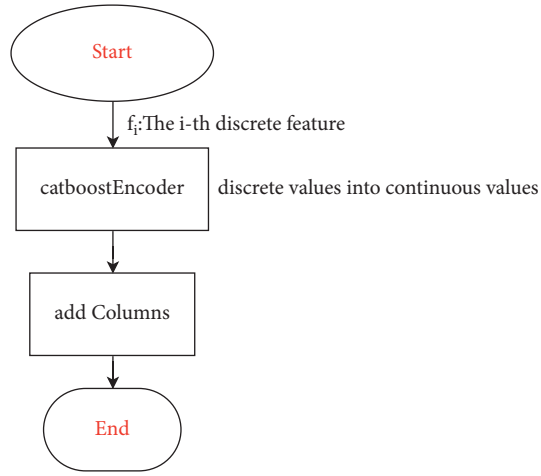

Figure 3: The processing steps of the mgfg algorithm for discrete features.

(i) **Input:** $F(f1, f2, f3, \ldots, fm)$//Discrete feature columns
(ii) m: the number of discrete feature columns
(iii) Catboost_encoders: Converts the discrete attributes of the Q class to continuous feature
(iv) **for** $i = 1$ **to** m **do**
(v) f_{i_col} = Catboost_encoders(f_i)
(vi) F.append (f_{i_col})
(vii) end
(viii) **Output**: $F(f1, f2, \ldots, fm, \ldots, fn)$//Feature columns with different granularity.

Algorithm 2: mgfg method for discrete feature columns.

4. Experiment

In this section, we combine the multigranularity feature generation method proposed in this paper with Xgboost for experimental verification. Experiments were carried out on the KDD99, NSL-KDD, UNSW_NB15, and CSE-CIC-IDS2018 datasets.

Therefore, we first introduced the KDD99, NSL-KDD, UNSW_NB15, and CSE-CIC-IDS2018 datasets. Then three sets of experiments were carried out on the KDD99 and NSL-KDD datasets. Then, two sets of experiments were conducted on the UNSW_NB15 dataset, and, finally, a set of experiments were conducted on the CSE-CIC-IDS2018 dataset.

The development language we use is Python3.6, and the development tools are Anconda3, PyCharm2018, and Xgboost0.72. Our hardware environment is 64-bit Windows 10 Professional, InteI(R) Core(TM) i7-8550U, 16 G memory, 256 SSD + 1 T HDD.

4.1. Data Collection. Intrusion detection data is extracted from network access traffic before it can be used for feature engineering and model training. However, it is not easy to collect real network access traffic data, so we use the classic intrusion detection datasets KDD99, NSL-KDD, UNSW_NB15, and CSE-CIC-IDS2018 as the experimental data in this article.

4.1.1. KDD99 Dataset. The KDD99 dataset was developed in a research lab at MIT, and IDS designers use it as a benchmark to evaluate various methods and technologies. The total number of KDD99 samples was 4,898,431 with 40 columns of features and 1 column of class labels. In this paper, 10% of all samples were selected as the training set (hereafter referred to as KDD99 training set), which contains 494,021 samples. Category labels can be generalized into two types, Normal and Anormal. Normal refers to Normal network access, while Anormal refers to attack access.

Anormal can be divided into four categories: DoS, Probing, U2R, and R2L, and Normal includes 5 types. As shown in Table 1, each attack type has specific classification signs, such as 6 classification signs under DoS attack, 4 classification signs under Probing, 8 classification signs under R2L, and 4 classification signs under U2R. There are 22 types of attack classification identifiers in total, plus the normal identifier, and the total number of categories is 23.

For the test set, we select the corrected dataset in KDD99 (hereinafter referred to as the KDD99 test set), which has the correct classification label, the number of samples is 311,029, and the same is 41 columns of attributes.

Next, we further analyze the KDD99 training set. If all data type labels are divided into normal traffic and abnormal traffic, there are 97,278 samples of normal traffic and 396,743 samples of abnormal traffic. The number of abnormal flow samples is about 4 times the number of normal flow samples,

Table 1: Classification of attack types.

Identify the type	Meaning	Specific classification identification
Normal	Normal Record (Normal)	Normal
Anormal	Denial of Service Attack (DoS)	back, land, neptune, pod, smurf, teardrop
	Probing Attack (Probing)	ipsweep, nmap, portsweep, satan, etc.
	Remote to Local Attack (R2L)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
	User to Root Attack (U2R)	buffer_overflow, loadmodule, perl, rootkit

and there is a problem of extreme imbalance of sample categories.

If we further divide the exception types into 4 categories, DoS, Probing, R2L, and U2R, the largest number of DoS samples is 391,458, while U2R samples only have 52, which is extremely unbalanced.

If we treat the specific classification identifiers as one category, then there are 20 abnormal traffic labels and 1 normal traffic label in the KDD99 dataset at this time and a total of 21 category labels at this time. The reason why there are 21 categories here is that we have selected the intersection of the training set and the test set category labels. At this time, the number of Smurf samples is at most 280,790, and the number of Perl samples is at least 3. The problem of extreme imbalance of sample data categories is even more serious.

4.1.2. NSL-KDD Dataset. NSL-KDD is also a classic dataset, which can be regarded as an improved version of the KDD99 dataset. The dataset is preprocessed to remove redundant and duplicate records. The dataset included 125,973 samples, 42 columns of attributes for training, and 22,544 samples for testing. So the total sample size is 148,517. Next, we also carried out 2-classification, 5-classification, and multi-classification analyses on the NSL-KDD training set. Next, we also perform 2-class, 5-class, and multiclass analyses on the NSL-KDD training set.

First of all, if the sample categories in the NSL-KD training data are treated as two-classification problems (Normal and Normal), then the number of samples of each type is 67,343 and 58,630, respectively, and the data categories are balanced.

However, if the abnormal traffic types are subdivided into 4 major categories (refer to Table 1 for classification details), then the Normal category with the largest sample size has 67,343, and the U2R category with the smallest sample size has 52. The sample category has an extremely unbalanced problem.

Finally, if we treat all the specific classification identifiers as a single category, then there are 21 categories in the training set at this time. Among them, the maximum number of samples in the Normal category is 67,343, while the number of samples in the Perl category is only 3. It can be seen that the sample categories are extremely unbalanced.

4.1.3. UNSW_NB15 Dataset. The UNSW_NB15 [25] dataset was created by the Australian Cyber Security Centre, which contains nine attacks and 49 features. The nine attacks are Fuzzers, Analysis, Backdoors, DoS, Adventurous, Generic, Reconnaissance, Shellcode, and Worms. The total number of samples in this dataset is 2,540,044, which are stored in four files. We choose UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv in this dataset as training set and test set. Among them, the sample size of the training set was 175,341, and that of the test set was 82,332. For the training set, we make statistics of the number of dichotomous and dechotomous samples, respectively. In terms of dichotomies, the sample classification is relatively average. In terms of ten categories, the sample classification is unbalanced, especially for Normal and Generic classes, and the samples numbers are 37,000 and 44, respectively, indicating extreme imbalance.

4.1.4. CSE-CIC-IDS2018 Dataset. CSE-CIC-IDS2018 dataset [26] is a collaborative project of the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). It has become a diversified and comprehensive benchmark dataset for intrusion detection. The dataset includes seven different attack scenarios: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside.

Since downloading the dataset required installing AWS CLI, we downloaded it from Kaggle. After downloading, there were 10 CSV files, each containing 1-day data. Considering the hardware performance, the file 02-14-2018.Csv was selected for the experiment. In total, the file contains 1,048,575 samples, 80 columns of features, and 3 types of attacks. After pretreatment, the actual number of samples used was 1,044,751 samples with 54 column features. The three attack types are Benign, FTP-Brute-force, and SSH-Brute-force, with 663,808, 193,354, and 187,589 samples, which are also unbalanced.

Based on the content of this section, we can see that, for the KDD99, NSL-KDD, and UNSW_NB15 datasets, we need to perform more accurate classification detection. The imbalance of sample categories will greatly affect our final detection effect. The problem of category imbalance is a common phenomenon. The main performance is that the model is more inclined to the classes with a large number of samples, and it is easy to ignore the classes with a small number. The common solution is to solve it by sampling. However, the sampling technique is to increase or decrease the number of samples in some way, which may improve the generalization ability of the model, but there may be a risk of overfitting.

For the problem of multiclass intrusion detection, the imbalance of sample types is common. If this problem

Table 2: Classification results of the proposed method in the KDD99 dataset.

	Precision	Recall	F1-score	Support
Normal	1.00	1.00	1.00	24287
Anormal	1.00	1.00	1.00	99219
Accuracy			1.00	123506

cannot be alleviated, the final prediction result of the model will inevitably be more inclined to the category with the majority of samples, and the detection accuracy of the category with a small number of samples will be very low.

4.2. KDD99 Experiment Results. First, perform three sets of experiments on the KDD99 dataset. Experiment 1 mainly verifies the model's detection of normal and abnormal visits. Experiment 2 verifies the model's ability to recognize normal and 4 types of abnormal visits. Experiment 3 is the most accurate multiclassification experiment.

4.2.1. Experiment 1 on the KDD99 Dataset. Through training, the classification result of the model on the KDD99 test set is obtained. It can be seen from Table 2 that the classification results of our proposed algorithm on Xgboost can reach an accuracy of 100%. Even if the sample ratio of the training set is close to 1 to 4, there is a problem of extreme imbalance, and the algorithm in this paper can still classify very accurately.

4.2.2. Experiment 2 on the KDD99 Dataset. If the abnormal access in the second classification is further divided into DoS, Probing, R2L, and U2R, then 5 classifications can be performed at this time. At this time, the DoS attack type has the largest number of samples, which is 391,458 samples. The U2R category has the smallest number of samples, which is 52 samples. The ratio of the two is far more than 4 : 1, and there is an extremely unbalanced number of categories. But, judging from the experimental results in Table 3, our classification effect is also very good, with 100% accuracy for all 5 categories.

4.2.3. Experiment 3 on the KDD99 Dataset. To further verify the effectiveness of the algorithm proposed in this article, we have further refined the four attack types of DoS, Probing, R2L, and U2R into 20 attack types, plus the Normal category, for a total of 21 types. Again, 21 categories are used as the classification target to verify the effectiveness of the algorithm proposed in this paper. The reason for the 21 categories is that this is the number of categories shared by the training set and the test set. In other words, we only select categories that are shared by the training set and the test set. At this time, the class "smurf" with the largest number of samples in the test set has 280,790 samples, and the classes with the smallest numbers of samples are "perl," "phf," "multihop," "ftp_write," and "loadmodule." Their sample sizes are 3, 4, 7, 8, and 9, respectively. We can perform very good tests on the four categories of "perl," "multihop," "perl," and "loadmodule." For a 21-classification problem, as shown in Table 4, we can achieve a classification accuracy of 99%, and we can also perform relatively accurate classification for categories with an extremely small number of samples. It can be seen that the proposed method can achieve a certain level of category imbalance.

4.3. NSL-KDD Experiment Results. The NSL-KDD dataset is an improved version of the KDD99 dataset, which solves the problem of duplication and inconsistency in the KDD99 dataset. Below we will verify the effectiveness of the algorithm shown on the NSL-KDD dataset.

4.3.1. Experiment 1 on the NSL-KDD Dataset. First, perform a two-classification experiment on the NSL-KDD dataset. From Table 5, it can be seen that the classification results of the data generated by our proposed algorithm on Xgboost can reach an accuracy of 100%. Our algorithm can classify very accurately.

4.3.2. Experiment 2 on the NSL-KDD Dataset. Then, a five-classification experiment was performed on NSL-KDD. At this time, the class with the largest number of samples in the NSL-KDD training set is Normal, which has 67,343 samples. The U2R category has the smallest number of samples, which is 52 samples. There is also the problem of extreme imbalance in the number of categories. But, judging from the experimental results in Table 6, our classification effect is also very good, with 100% accuracy for all 5 categories.

4.3.3. Experiment 3 on the NSL-KDD Dataset. To further verify the effectiveness of the algorithm proposed in this article, we have further refined the four attack types DoS, Probing, R2L, and U2R into 20 attack types, plus the Normal category, for a total of 21 types. Again, 21 categories are used as the classification target to verify the effectiveness of the algorithm proposed in this paper. As shown in Table 7, at this time, the class "Normal" with the largest number of samples has 67,343 samples, and the classes with the smallest numbers of samples are "perl," "phf," "multihop," "ftp_write," and "loadmodule," and there are only 3 samples for the "perl" class. The ratio of the samples to the "Normal" sample is 22447 : 1, and the problem of sample category imbalance is quite serious. For the categories of "perl," "phf," "multihop," and "loadmodule," there are 3, 4, 7, and 9 in the training set. However, as can be seen from the table below, we can all perform relatively good testing. In addition, for such a 21-classification problem in NSL-KDD, we can achieve a classification accuracy of 91%.

4.4. UNSW_NB15 Experiment Results. The KDD99 and NSL-KDD datasets are a bit old and cannot represent network attacks in recent years, so UNSW_NB15 is chosen to further verify the algorithm proposed in this article. For this dataset, we use its divided training set and test set to perform two-class and ten-class experiments, respectively.

Table 3: 5 classification results of our method in the KDD99 dataset.

	Precision	Recall	F1-score	Support
Normal	1.00	1.00	1.00	24394
DoS	1.00	1.00	1.00	97768
Probing	1.00	1.00	1.00	1028
R2L	1.00	1.00	1.00	308
U2R	1.00	1.00	1.00	8

Table 4: Multiclassification results of our method in the KDD99 dataset.

	Precision	Recall	F1-score	Support
smurf	1.0000	1.0000	1.0000	164091
imap	0.0000	0.0000	0.0000	1
land	1.0000	1.0000	1.0000	9
Normal.	1.0000	0.9990	0.9995	60593
ftp_write.	0.0000	0.0000	0.0000	3
rootkit.	0.7500	0.2308	0.3529	13
ipsweep.	1.0000	0.9804	0.9901	306
back	0.4015	1.0000	0.5729	1098
satan	0.9802	1.0000	0.9900	1633
neptune	1.0000	1.0000	1.0000	58001
teardrop	0.4444	1.0000	0.6154	12
multihop	1.0000	0.4444	0.6154	18
loadmodule	1.0000	1.0000	1.0000	2
portsweep	1.0000	1.0000	1.0000	354
warezmaster	1.0000	0.0031	0.0062	1602
nmap	1.0000	1.0000	1.0000	84
buffer_overflow	1.0000	1.0000	1.0000	22
perl	1.0000	1.0000	1.0000	2
pod	1.0000	1.0000	1.0000	87
phf	0.0000	0.0000	0.0000	2
guess_passwd	1.0000	1.0000	1.0000	4367
Accuracy			0.9942	292300
Macro avg.	0.7893	0.7456	0.7211	292300
Weighted avg.	0.9976	0.9942	0.9927	292300

Table 5: Two-classification results on NSL-KDD.

	Precision	Recall	F1-score	Support
Normal	1.00	1.00	1.00	9711
Anormal	1.00	1.00	1.00	12833

Table 6: 5 classification results on the NSL-KDD dataset.

	Precision	Recall	F1-score	Support
Normal	1.00	1.00	1.00	9711
DoS	1.00	1.00	1.00	7460
Probing	1.00	1.00	1.00	2421
R2L	1.00	1.00	1.00	2885
U2R	1.00	1.00	1.00	67

4.4.1. Experiment 1 on UNSW_NB15 Dataset. As shown in Table 8, the indicators of the new dataset generated by the algorithm in this article on the Xgboost classifier have reached 100% classification accuracy.

Table 7: Multiclassification results on the NSL-KDD dataset.

	Precision	Recall	F1-score	Support
nmap	1.0000	1.0000	1.0000	73
smurf	1.0000	1.0000	1.0000	665
perl	0.2000	1.0000	0.3333	2
Normal	1.0000	1.0000	1.0000	9711
satan	0.9986	1.0000	0.9993	735
guess_passwd	0.4861	0.3826	0.4282	1231
pod	0.9762	1.0000	0.9880	41
warezmaster	0.0000	0.0000	0.0000	944
teardrop	1.0000	1.0000	1.0000	12
ftp_write	0.0000	0.0000	0.0000	3
rootkit	0.6429	0.6923	0.6667	13
back	0.2297	1.0000	0.3736	359
ipsweep	0.9929	0.9858	0.9893	141
buffer_overflow	1.0000	0.7000	0.8235	20
portsweep	0.9937	1.0000	0.9968	157
phf	1.0000	0.5000	0.6667	2
land	1.0000	1.0000	1.0000	7
imap	0.0000	0.0000	0.0000	1
neptune	1.0000	1.0000	1.0000	4657
multihop	1.0000	1.0000	1.0000	18
loadmodule	1.0000	1.0000	1.0000	2
Accuracy			0.9084	18794
Macro avg.	0.7390	0.7743	0.7269	18794
Weighted avg.	0.9006	0.9084	0.8995	18794

Table 8: Binary classification experiment on the UNSW_NB15 dataset.

	Precision	Recall	F1-score	Support
Normal	1.0000	1.0000	1.0000	56000
Anormal	1.0000	1.0000	1.0000	119341
Accuracy			1.0000	175341
Macro avg.	1.0000	1.0000	1.0000	175341
Weighted avg.	1.0000	1.0000	1.0000	175341

Table 9: Subclass experiments on the UNSW_NB15 dataset.

	Precision	Recall	F1-score	Support
Normal	1.0000	1.0000	1.0000	56000
Reconnaissance	1.0000	1.0000	1.0000	10491
Backdoor	1.0000	1.0000	1.0000	1746
DoS	1.0000	1.0000	1.0000	12264
Exploits	1.0000	1.0000	1.0000	33393
Analysis	1.0000	1.0000	1.0000	2000
Fuzzers	1.0000	1.0000	1.0000	18184
Worms	1.0000	1.0000	1.0000	130
Shellcode	1.0000	1.0000	1.0000	1133
Generic	1.0000	1.0000	1.0000	40000
Accuracy			1.0000	175341
Macro avg.	1.0000	1.0000	1.0000	175341
Weighted avg.	1.0000	1.0000	1.0000	175341

4.4.2. Experiment 2 on UNSW_NB15 Dataset. In order to further verify the performance of the algorithm, we further carry out a more refined classification. 9 types of anomalous attacks are detected, and there are 10 categories at this time.

Table 10: Results of three-classification experiments on the CSE-CIC-IDS2018 dataset.

	Precision	Recall	F1-score	Support
Benign	1.0000	1.0000	1.0000	219138
FTP-Brute-force	1.0000	1.0000	1.0000	63565
SSH-Brute-force	1.0000	1.0000	1.0000	62065
Accuracy			1.0000	344768
Macro avg.	1.0000	1.0000	1.0000	344768
Weighted avg.	1.0000	1.0000	1.0000	344768

Table 11: Comparison of two-classification test results.

Research	Method	Accuracy	Dataset
Al-Yaseen et al. [13] (2017)	SVM + ELM	95.75%	KDD99
Wang et al. [10] (2017)	Augmented Features + SVM	99.31%	NSL-KDD
Al-Qatf et al. [31] (2018)	SAE-SVM (binary classification)	99.41%	NSL-KDD training dataset with 10-fold cross-validation
Ömer Kasim [20] (2020)	Label encoding + normalization + AE-SVM	99.5%	NSL-KDD training dataset with 10-fold cross-validation
Thaseen, S [24] (2021)	Xgboost + MLP + BPN + LSTM	99%	UNSW_NB15
Proposed method	mgfg + Xgboost (binary classification)	100%	KDD99 test
Proposed method	mgfg + Xgboost (binary classification)	100%	KDD99 training dataset with 10-fold cross-validation
Proposed method	mgfg + Xgboost (five-classification)	100%	KDD99 test
Proposed method	mgfg + Xgboost (five-classification)	100%	KDD99 training dataset with 10-fold cross-validation
Proposed method	mgfg + Xgboost (binary classification)	100%	NSL-KDD test
Proposed method	mgfg + Xgboost (binary classification)	100%	NSL-KDD train dataset with 10-fold cross-validation
Proposed method	mgfg + Xgboost (five-classification)	100%	NSL-KDD test
Proposed method	mgfg + Xgboost (five-classification)	100%	NSL-KDD training dataset with 10-fold cross-validation
Proposed method	mgfg + Xgboost (binary classification)	100%	UNSW_NB15 test
Proposed method	mgfg + Xgboost (binary classification)	100%	UNSW_NB15 training dataset with 10-fold cross-validation
Proposed method	mgfg + Xgboost (ten-classification)	100%	UNSW_NB15 test
Proposed method	mgfg + Xgboost (ten-classification)	100%	UNSW_NB15 training dataset with 10-fold cross-validation
Proposed method	mgfg + Xgboost (three-classification)	100%	CSE-CIC-IDS2018 test
Proposed method	mgfg + Xgboost (three-classification)	100%	CSE-CIC-IDS2018 training dataset with 10-fold cross-validation

As shown in Table 9, the classification effect of our algorithm can still reach 100%.

4.5. CSE-CIC-IDS2018 Experimental Results. The CSE-CIC-IDS2018 dataset is the latest intrusion detection data. To verify the performance of the algorithm, we selected the data on February 14, 2018, for experiments. Table 10 shows the classification results on the CSE-CIC-IDS2018 test set. It can be seen that the detection effect is still accurate.

5. Discussion

For the problem of intrusion detection, most scholars perform two-class detection, such as the DoS attack detection work carried out by Al-Yaseen et al. in 2016 in Table 11. The accuracy of that method on the KDD99 dataset is 95.75%. This is a good accuracy rate, but the KDD99 dataset is old after all. Therefore, experimenting on the NSL-KDD dataset is also a way to verify the effectiveness of the algorithm. For example, in 2017, Wang et al. used Augmented Features + SVM method to achieve a high accuracy rate of 99.31%. In addition, the methods of Al-Qatf et al. in 2018 and Ömer Kasim in 2020 reached a relatively high level. The accuracy rates were 99.41% and 99.5%. But they are all DoS attack detections. In 2021, Thaseen, S. proposed a method of integrating multiple deep learning methods through Xgboost, which achieved 99% classification accuracy on the UNSW_NB15 dataset. But it may take extra time to adjust the parameters. Our method does not have too many parameters to be adjusted, and the classification accuracy is not lower than his method.

Therefore, in contrast to the above-mentioned author's research, our method can perform five-category detection in addition to two-category detection, and the five-category detection includes DoS attack detection. It can be seen that our method can reach a detection rate of 100%. Furthermore, we also conduct two-class and ten-class experiments on the UNSW_NB15 dataset, and the detection accuracy rate is 100%. Because the selected CSE-CIC-IDS2018 dataset has a total of two attack categories and one normal network

Table 12: Comparison with other multiclassification algorithms.

Dataset		KDD99 (%)			NSL-KDD (%)		
Classification		Binary	Five	Multiple	Binary	Five	Multiple
Research	Celestine Iwendi (2020)	99.80	-	99.90	98.98	-	98.60
	Thaseen S. (2016)	-	-	-	-	92.81	-
	Thaseen S. (2019)	-	-	-	-	99.23	-
	B. Riyaz (2020)	-	-	-	-	98.88	-
	mgfg + Xgboost	100	100	99.42	100	100	90.84

access, only one three-classification experiment was performed, and the detection accuracy rate was also 100%.

To further verify the detection performance of the algorithm in this paper, we compare it with Iwendi C.'s method [2], as shown in Table 12, because his method simultaneously performs two-class classification and finer attack detection. For two-classification problems, our method can achieve 100% accurate classification, whether it was on the KDD99 dataset or the NSL-KDD dataset, while Celestine Iwendi's method achieves 99.8% and 98.98%, respectively. Although it seems that there is not much improvement, for a classifier algorithm with ultrahigh accuracy, it is very difficult to improve every time. For the five-classification experiment, Iwendi C. has not verified it, and our algorithm can also achieve 100% accurate detection on the KDD99 dataset and the NSL-KDD dataset. For more detailed multiclassification experiments, on the KDD99 data, the recognition rate of our algorithm is 99.42%, and on the NSL-KDD dataset it is 90.84%, which is lower than the result of Iwendi C., but, for the two classifications with the detection results of five categories, our algorithm is better than his method. In addition, we compared the algorithm to the method of Thaseen, S. Her average accuracy of NSL-KDD detection in 2016 and 2019 was 92.81% and 98.88%, respectively, while in B. Riyaz's test on NSL-KDD in 2020 the detection accuracy rate was relatively high, reaching 98.88%. However, they did not perform more refined classification tests.

Further, in order to verify the effectiveness of the multigranularity feature generation algorithm, we combine the algorithm with multiple classifiers and verify it on multiple datasets, as shown in Tables 13 and 14. Therefore, we combined the multigranularity feature generation algorithm with decision trees, Bagger, and random forests and performed two-class, five-class, and multiclass experiments on KDD99, NSL-KDD, UNSW_NB15, and CSE-CIC-IDS2018. For the two-classification experiment, all algorithms can reach a 100% detection rate on the KDD99, NSL-KDD, and UNSW_NB15 datasets. For the three-classification experiment, all algorithms can reach a 100% detection rate on CSE-CIC-IDS2018. For the five-classification experiments on the KDD99 and NSL-KDD datasets, the detection effect of random forest on NSL-KDD is slightly worse, and the detection rates of other algorithms are above 96%. For the multiclassification experiments on the KDD99 and NSL-KDD datasets, the detection rate of random forest on NSL-KDD is 88.01%, and the detection rates of other algorithms are above 90%. For the ten-classification experiments on the UNSW_NB15 dataset, the detection rate is also above 97.8%. It can be seen that the features generated by the proposed mgfg algorithm have good distinguishability and can achieve relatively high classification accuracy on multiple classifiers and multiple datasets.

In order to further verify the effectiveness of the multigranularity feature generation algorithm, we compare and verify it with the Catboost method. Because Catboost itself is an integrated learning method, it can automatically encode and generate features for numerical data and text data. We compare the features generated by the proposed multigranularity feature generation method with them to further verify our proposed feature generation method Therefore, we conducted further comparative experiments. First, we directly use the original Catboost algorithm to experiment with the original data of KDD99 and NSL-KDD. Then the features generated by the multigranularity feature generation method are sent to the Catboost algorithm for experimentation. As shown in Tables 15 and 16, it can be seen that the classification effect of our proposed algorithm on different classifications on the 4 datasets has been significantly improved, which shows the effectiveness of the algorithm.

In addition, the time cost of the algorithm is also an indicator for evaluating the algorithm. In order to further verify the efficiency of the proposed algorithm, we next compare our feature generation method with the CFS feature selection method in terms of time cost. The reason for choosing the CFS algorithm is that Iwendi C. uses CFS to perform feature selection. The CFS feature selection method is a classic algorithm. Although it will select some features, it will be faster during model training, but the process of selecting features is more time-consuming. We call the CFS algorithm in the skfeature library to perform two-classification, five-classification, and multiclassification experiments on the KDD99 dataset and the NSL-KDD dataset to verify the efficiency of the algorithm in this paper. It can be seen from Figure 4 that, on the KDD99 dataset, the time cost of the CFS feature selection method is several times that of our algorithm, and as the number of classifications increases, this is particularly obvious. The time cost of CFS on multiclassification tasks is 1,2633 seconds, our algorithm is only 1,340 seconds, and the time cost is about 1/10 of CFS. Similarly, as shown in Figure 5, on the NSL-KDD dataset, the efficiency of our proposed algorithm is also better than that of the CFS algorithm. For two-class detection and five-class detection tasks, our algorithm also has obvious advantages. For multiclass detection tasks, the time cost of our algorithm is only 1/22 of the CFS algorithm.

Finally, the total time spent on our feature generation method and model training is compared with the time spent

Table 13: Results of mgfg algorithm with different algorithms on different datasets 1.

Dataset		KDD99 (%)			NSL-KDD (%)		
Classification		Binary	Five	Multiple	Binary	Five	Multiple
Method	mgfg + DT	100	100	96.07	100	100	99.96
	mgfg + Bagger	100	96.18	96.11	100	100	98.77
	mgfg + RF	100	96.12	97.93	100	89.08	88.01
	mgfg + Xgboost	100	100	99.42	100	100	90.84

Table 14: Results of mgfg algorithm with different algorithms on different datasets 2.

Dataset		UNSW_NB15 (%)		CSE-CIC-IDS2018 (%)
Classification		Binary	Ten	Three
Method	mgfg + DT	100	100	100
	mgfg + Bagger	100	100	100
	mgfg + RF	100	97.80	100
	mgfg + Xgboost	100	100	100

Table 15: Comparison of the results of the original Catboost algorithm and mgfg + Catboost algorithm 1.

Dataset		KDD99 (%)			NSL-KDD (%)		
Classification		Binary	Five	Multiple	Binary	Five	Multiple
Method	Catboost	92.67	92.38	97.59	78.95	92.38	85.98
	mgfg + Catboost	100	100	98.22	100	99.99	89.65

Table 16: Comparison of the results of the original Catboost algorithm and mgfg + Catboost algorithm 2.

Dataset		UNSW_NB15 (%)		CSE-CIC-IDS2018 (%)
Classification		Binary	Ten	Three
Method	Catboost	100	75.60	100
	mgfg + Catboost	100	99.93	100

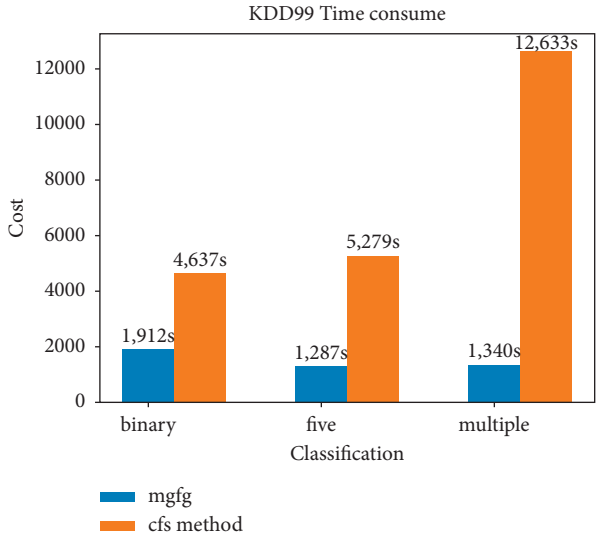

Figure 4: Comparison of feature generation time and cost on the KDD99 dataset.

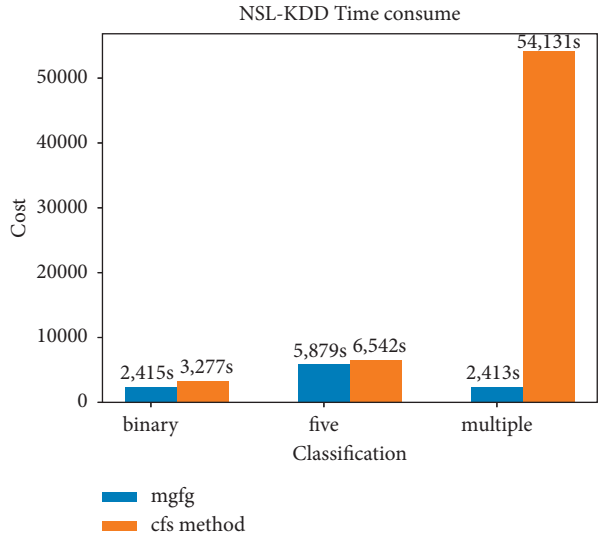

Figure 5: Comparison of feature generation time and cost on the NSL-KDD dataset.

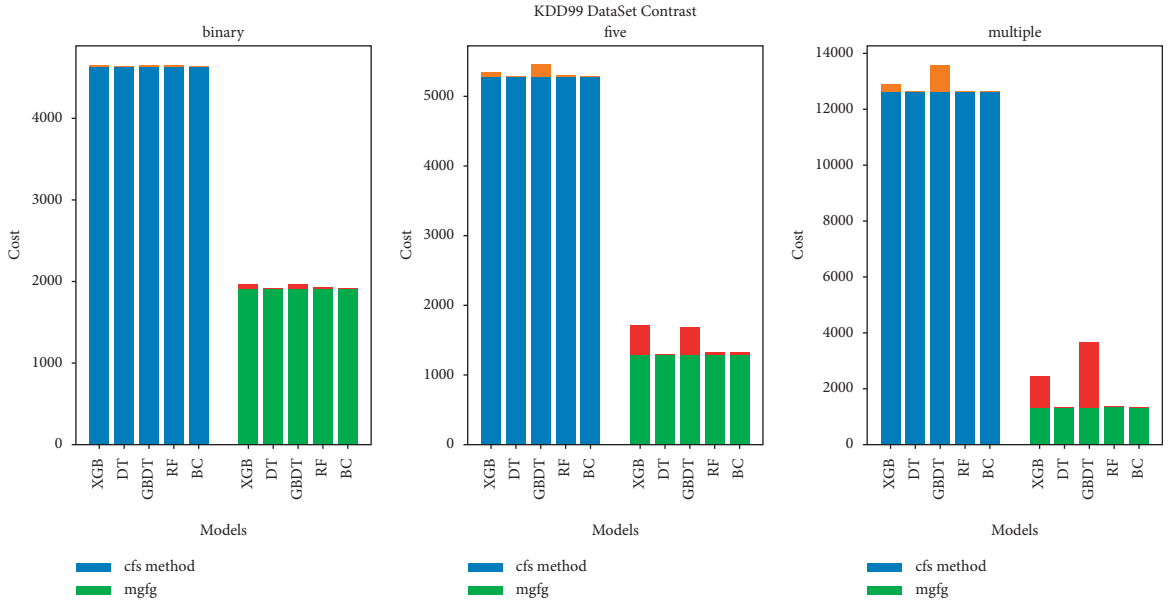

Figure 6: Comparison of feature generation + model training prediction time cost on the KDD99 dataset.

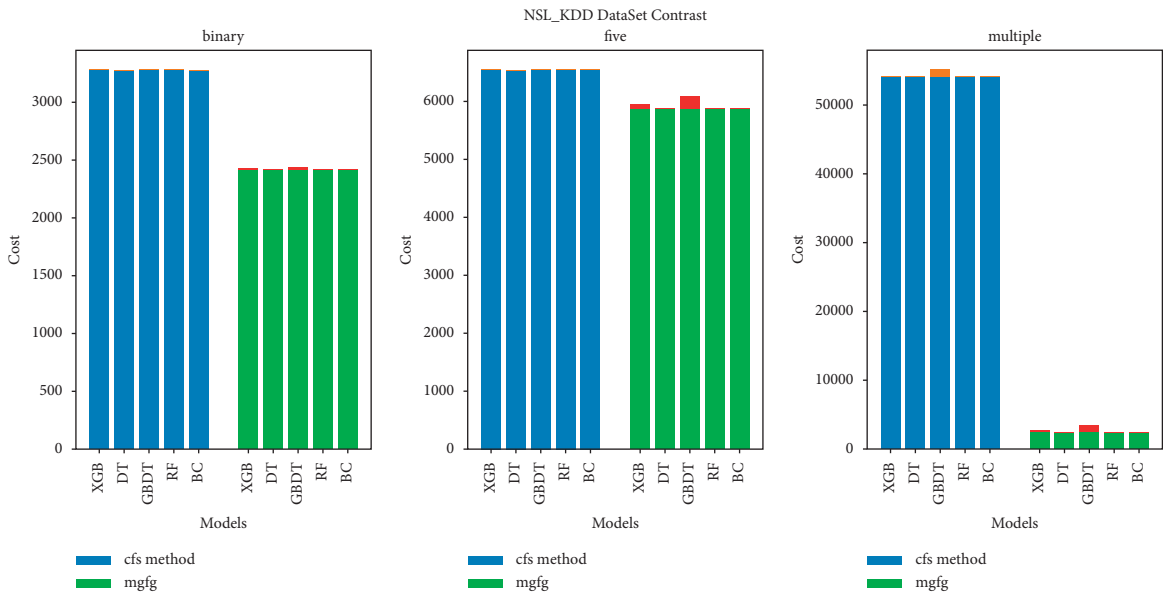

Figure 7: Comparison of feature generation + model training prediction time cost on NSL-KDD dataset.

on CFS + model training. Although we have added some feature columns on the basis of the original dataset, it may make the model training time longer. However, compared with the CFS feature selection first and then the model training, our algorithm still has a greater time advantage on the KDD99 and NSL-KDD datasets. This can be seen in Figure 6. First, we use the CFS method for feature processing on the KDD99 dataset and then use Xgboost, decision trees, gradient boosting trees, random forests, and Bagger classifiers (XGB, DT, GBDT, RF, and BC) for classification experiments. We sequentially perform two-category, five-category, and multicategory tasks on the KDD99 dataset. It can be seen that, for the two classifications, the time cost of our method is less than half of that of the CFS method. For five-category and multicategory tasks, our method takes less time and cost. Then, as shown in Figure 7, we also conducted algorithm comparison experiments on NSL-KDD. For two-class and five-class experiments, our method has obvious advantages compared with the CFS + model method, and, for multiclass experiments, our method has more obvious advantages. The time cost of the CFS method is several times that of our proposed method, which shows the efficiency of our method.

6. Conclusion and Further Work

This paper proposes a method based on multigranularity feature generation to be applied to an intrusion detection system to verify the effect of attack detection on four datasets. We carried out two-classification, five-classification, and multiclassification experiments on KDD99 and NSL-KDD, respectively. The detection results of the two-classification and five-classification experiments reached 100%, and the detection effect on the multiclassification experiment was also at the leading level. For categories with small sample size, there is a certain degree of differentiation. In addition, we also conducted experimental verification on UNSW_NB15 and CSE-CIC-IDS2018 and achieved 100% detection results on some datasets. The limitation of this work is that as the features of the dataset increase, more features will be generated when the multigranularity feature generation method is performed, and it may take more time to train the algorithm. In the future, dimensionality reduction may be considered to save training time.

Data Availability

The experiment data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: a survey," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, Article ID 271, 2013.

[2] H. Saxena and V. Richariya, "Intrusion detection in kdd99 dataset using SVM-PSO and feature reduction with information gain," *International Journal of Computer Application*, vol. 98, no. 6, pp. 25–29, 2014.

[3] C. Iwendi, S. Khan, J. H. Anajemba, M. Mittal, M. Alenezi, and M. Alazab, "The use of ensemble models for multiple class and binary class classification for improving intrusion detection systems," *Sensors*, vol. 20, no. 9, Article ID 2559, 2020.

[4] M. Shafiq, X. Yu, A. K. Bashir, H. N. Chaudhry, and D. Wang, "A machine learning approach for feature selection traffic classification using security analysis," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 4867–4892, 2018.

[5] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[6] G. Pu, L. Wang, and J. Shen, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 146–153, 2020.

[7] M. Panda, A. Abraham, and M. R. Patra, "Hybrid intelligent systems for detecting network intrusions," *Security and Communication Networks*, vol. 8, no. 16, pp. 2741–2749, 2015.

[8] S. Sivanantham, R. Abirami, and R. Gowsalya, "Comparing the performance of adaptive boosted classifiers in anomaly based intrusion detection system for networks," in *Proceedings of the 2019 International Conference on Vision towards Emerging Trends in Communication and Networking (ViTE-CoN)*, pp. 1–5, Vellore, India, March 2019.

[9] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21954–21961, 2017.

[10] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowledge-Based Systems*, vol. 136, pp. 130–139, 2017.

[11] A. R. Yusof, N. I. Udzir, and A. Selamat, "Adaptive feature selection for denial of services (DoS) attack," in *Proceedings of the 2017 IEEE Conference on Application, Information and Network Security (AINS)*, pp. 81–84, Sarawak, Malaysia, November 2017.

[12] S. Hosseini and M. Azizi, "The hybrid technique for DDoS detection with supervised learning algorithms," *Computer Networks*, vol. 158, pp. 35–45, 2019.

[13] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system," *Expert Systems with Applications*, vol. 67, pp. 296–303, 2017.

[14] D. S. Vijayakumar and S. Ganapathy, "Machine learning approach to combat false alarms in wireless intrusion detection system," *Computer and Information Science*, vol. 11, no. 3, pp. 67–81, 2018.

[15] D. S. Vijayakumar and S. Ganapathy, "Multistage ensembled classifier for wireless intrusion detection system," *Wireless Personal Communications*, vol. 122, no. 3, pp. 1–24, 2021.

[16] I. Sumaiya Thaseen and C. Aswani Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 4, pp. 462–472, 2017.

[17] I. S. Thaseen, C. A. Kumar, and A. Ahmad, "Integrated intrusion detection model using chi-square feature selection and ensemble of classifiers," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3357–3368, 2019.

[18] J. Sharma, C. Giri, and O. C. Granmo, "Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation," *EURASIP Journal on Information Security*, vol. 2019, no. 1, pp. 1–16, 2019.

[19] J. V. A. Sukumar, I. Pranav, and M. M. Neetish, "Network intrusion detection using improved genetic K-means algorithm," in *Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2441–2446, Bangalore, India, September 2018.

[20] Ö. Kasim, "An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks," *Computer Networks*, vol. 180, Article ID 107390, 2020.

[21] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Computing*, vol. 24, no. 22, pp. 17265–17278, 2020.

[22] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1222–1228, Udupi, India, September 2017.

[23] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proceedings of the 2017 International Conference on Advances in Computing,*

Communications and Informatics (ICACCI), pp. 1282–1289, Udupi, India, September 2017.

[24] S. T. Ikram, A. K. Cherukuri, B. Poorva et al., "Anomaly detection using Xgboost ensemble of deep neural network models," *Cybernetics and Information Technologies*, vol. 21, no. 3, pp. 175–188, 2021.

[25] N. Moustafa and S. Jill, "UNSW-NB15: a comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset)," in *Proceedings of the Military Communications and Information Systems Conference (MilCIS) 2015*, Canberra, Australia, November 2015.

[26] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Portugal, January 2018.

[27] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques third edition," *The Morgan Kaufmann Series in Data Management Systems*, vol. 5, no. 4, pp. 83–124, 2011.

[28] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," in *Proceedings of the Workshop on ML Systems at NIPS 2018*, Montreal, Canada, December 2018.

[29] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, San Francisco, CA, USA, August 2016.

[30] G. Ke, Q. Meng, and T. Finley, "Lightgbm: a highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3146–3154, 2017.

[31] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.