Hindawi
Journal of Sensors
Volume 2023, Article ID 5997304, 9 pages
https://doi.org/10.1155/2023/5997304

Research Article

Identification of Network Traffic Intrusion Using Decision Tree

Dasheng Chen, Qi Song, Yinbin Zhang, Ling Li, and Zhiming Yang

Research Institute of China Telecom Corporation Limited, Guangzhou, 510660 Guangdong, China

Correspondence should be addressed to Dasheng Chen; chends4@foxmail.com

Received 8 September 2022; Revised 17 October 2022; Accepted 24 November 2022; Published 15 April 2023

Academic Editor: Dong Chen

Network technology plays an increasingly important role in all aspects of social life. The Internet has brought a new round of industrial revolution and industrial upgrading. The arrival of the "Internet" era is accompanied by a large-scale increase in network applications and the number of netizens. At the same time, the number and severity of cyberattacks continue to increase. Therefore, intrusion detection systems (IDSs) have become an important part of the current network security infrastructure in various industries. Anomaly detection of network traffic data is an effective method for network protection. In order to better realize the detection of network traffic anomalies, several algorithms have been successfully applied. Most of them come from artificial intelligence (AI), but there is a general problem of excessive model execution processing time and low detection rates. And through a lot of research, it is found that most models do not pay enough attention to the data processing in the early stage. Therefore, in this paper, we optimize the data normalization process through a series of experiments and combine the PCA feature selection method to propose an optimized MaxAbs-DT classifier model. To train and measure the performance of the model, we used the NSL-KDD dataset, which is the benchmark dataset for most network anomaly detection models. The experimental results show that MaxAbs-DT outperforms other existing models and validates the effectiveness of the method. In addition, its execution time is greatly reduced compared to many models.

1. Introduction

At the beginning of the development of the Internet and industrial networks, network security issues have attracted much attention. Although network administrators have deployed various security mechanisms over time, such as packet encryption technology and firewalls, most networks are still not immune to attacks because security breaches are proliferating [1]. Among the many defense methods, intrusion detection systems (IDSs) are one of the best solutions for protecting systems from malicious attacks [2]. Commonly used methods are based on anomaly detection and misuse detection [3]. Misuse detection performs comparison and detection based on predefined abnormal features, which is equivalent to a blacklist mechanism. Anomaly detection mainly detects abnormality based on normal behavior, which is equivalent to a whitelist mechanism, and everything outside the whitelist is abnormal. Now, intrusion detection plays a vital and increasingly important role in network defense, most notably to allow network security administrators to warn of intrusions, attacks, and malicious behaviors such as malware. Having an IDS system is a mandatory line of defense to protect critical internal networks from these ever-increasing malicious intrusion activities. Therefore, in order to propose better intrusion detection systems, the research in the field of intrusion detection has been vigorously developed in recent years. The network traffic detection part, which aims to maintain network security through the analysis of network traffic data, is the most important part of IDSs. In order to implement IDS, class methods are required. It is the process of classifying objects with knowledge extracted from a set of data during the learning step.

In the early days, based on the principle of network communication, we focused on some specific characteristics of network traffic data for abnormal detection and classification, and a relatively complete intrusion detection system appeared. Subsequently, network traffic data anomaly detection methods that combine network information such as network status, transmission packets, frequency of occurrence of specific data emerge in an endless stream, and various anomaly detection technologies based on supervised,

unsupervised, and clustering methods have also been successfully used in network traffic. Data anomaly detection. However, with the rapid development of network technology, most of the classical methods are no longer suitable for today's network traffic data with more complex structures and can no longer meet the huge demand for abnormal detection of network traffic data. At present, most of the research work aims to improve the accuracy and reduce the consumption of time resources. Because machine learning and deep learning can well meet the accuracy requirements, and the development of computer hardware has also greatly reduced the model calculation time, in recent years, most relevant scholars have focused on exploring network traffic data anomalies based on machine learning or deep learning. It is well known that data preprocessing and feature selection are an important step in the training of intelligent classifiers, but many research works have not done enough in-depth research here.

Since the NSL-KDD dataset [4] we used is labeled, we employ a supervised classification approach throughout the work. Each piece of data in this dataset contains 41 attributes, some of which are more informative and more important for classification prediction, while some unimportant features interfere with the detection process. Feature selection is a primitive preprocessing stage in which irrelevant features are ignored. To accomplish this task, we use the dimensionality reduction method by PCA (principal component analysis) to select suitable features.

This paper proposes a network intrusion detection (NIDS) method using decision tree classifier. In this study, we developed MaxAbs-DT, a computational predictor for predicting abnormal network traffic using machine learning. In this predictor, we choose the optimal data preprocessing method as well as the PCA feature selection method, tune the decision tree, and finally train the DT-based prediction model. Our main goal is to maximize detection rates and reduce false alarms while considering processing time, a very important factor in real-time applications. The rest of the paper is organized as follows: Section 2 presents related research work on different types of NIDs. Section 3 describes the datasets used in experiments, the feature processing method, and the proposed method. Section 4 summarizes the obtained results and comparisons. Section 5 gives conclusion and outlook that are drawn.

2. Related Works

In order to determine whether the traffic samples belong to the intrusion behavior, the researchers explored different binary classification algorithms to obtain better detection results.

The paper [5] introduced a number of methods for network anomaly detection using decision trees (DT), support vector machines (SVM), and Naive Bayesian networks (NB). Kevric et al. [6] pointed out that combining two tree algorithm models can achieve better performance than separate tree classification models, the best combination they reported was random tree and NB tree, the model was tested on the KDD dataset, and an accuracy of 89.24% was obtained. In the field of machine learning classifier research, feature selection is a key component that selects important features from a dataset. Alazzam et al. [7] proposed a feature selection method based on a wrapper method using a pigeon-inspired optimizer that selects informative attributes from a feature set. This method was tested on UNSW-NB15, KDDCUP 99, and NLS-KDD, respectively, and good experimental results were successfully obtained. Support vector machines have been widely used to detect intrusions in networks, Bachar et al. [8] stated that using support vector machines to detect network anomalies, the model was tested on the UNSW-NB15 dataset, which achieved an accuracy rate of 94%, and compared it with some machine learning classifiers such as MLP, REPTree, and RF.

Some IDS classifiers utilize ANN as a pattern recognition technique. ANNs are implemented using feedforward propagation, and during the learning step, their parameters are optimized so that the output matches the corresponding input model. The authors [9] proposed an artificial neural network (ANN) model, proposed a hybrid model that improves detection performance by combining different state-of-the-art algorithms, and achieved 81.2% accuracy for the NSL-KDD dataset. Gautam and Om [10] proposed two neural network classification models for host ID (HIDS), namely, generalized regression neural network (GRNN) and multilayer perceptron (MLP), and obtained relatively good results. Autoencoder (AE) methods commonly used for feature extraction are now widely used in the first stage of mixture models, existing as preprocessing work for downstream classifiers. It generates an efficient compressed representation of the original input by removing noisy features ([11, 12]. Al-Qatf et al. [13] successfully combined upstream AE and downstream SVM, and the model obtained 84.96% binary classification accuracy when tested on KDD dataset. Niyaz et al. [14] proposed a sparse autoencoder method for feature learning combined with a neural network classifier, which finally achieved 88.39% accuracy on KDD dataset.

3. Materials and Methods

In this section, we first describe the NSL-KDD dataset used to train the model and then introduce the classic decision tree classifier. Then, we introduce other variants of decision tree. In this paper, we only carry out the experimental process of binary classification and introduce several classical classifier performance indicators in the binary classification model.

3.1. Datasets. The NSL-KDD dataset [15, 16] generated in 2009 is widely used in intrusion detection experiments. It is an enhanced form of the KDD Cup 1999 dataset. The dataset covers the $KDDTrain^{+}$ dataset as the training set and $KDDTest^{+}$ dataset as the testing set, which has different normal records and four different types of attack records, as shown in Table 1.

In addition, in order to make intrusion detection more realistic, the test data set includes many attack categories that do not appear in the training set. In the several data sets

Table 1: Breakdown of traffic records in the NSL-KDD.

Dataset	Total	Normal	Abnormal
KDDTrain$^+$	125973	67343	58630
KDDTest$^+$	22544	9711	12833

used, in addition to the 22 attack types in the training set, there are 17 different attack types in the test set.

The NSL-KDD dataset contains 41 features, including 3 nonnumeric features and 38 numeric features (i.e., protocol_type, service, and flag), as shown in Table 2. It has a classification label, which is divided into two categories (abnormal and normal) for binary classification. For multiclass classification experiments, labels can be divided into five categories (i.e., normal, denial of service (DoS), user-to-root (U2R), remote-to-local (R2L), and probe), but in this paper, we only do binary classification experiments.

3.2. Evaluation Metrics. Before introducing the evaluation indicators, we first introduce several concepts: TP, FP, TN, and FN. TP represents the actual positive example, and the prediction is positive example; FP represents that the actual is a negative example, and the prediction is a positive example; TN represents the actual negative example, the prediction is the negative example, FN represents the actual positive example, and the prediction is the negative example. After understanding these concepts, we will then introduce the concepts of ACC, recall, precision, F1-score, MCC, kappa, and AUC. Their definitions are given below:

Accuracy (ACC): this is the ratio of the number of correctly detected intrusions to the total number of traffic records:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}. \tag{1}$$

Recall: it refers to the ratio of the number of intrusion records correctly detected as an intrusion to the overall anomaly:

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{2}$$

Precision: this is the ratio of true anomaly records to total traffic records identified as intrusions

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{3}$$

F1-score: it refers to the harmonic average of accuracy and true positive rate and is a relatively comprehensive evaluation mark

$$\mathrm{F}_1 - \mathrm{Score} = \frac{2 * \mathrm{recall} * \mathrm{precision}}{\mathrm{recall} + \mathrm{precision}}. \tag{4}$$

Matthews correlation coefficient (MCC): it will return a value between -1 and+1. Its meaning is to describe the correlation coefficient between the actual classification and the prediction classification, and the value range is -1 to 1. A value of 1 indicates the perfect prediction of the tested object, and a value of 0 indicates that the prediction result is not as good as the random prediction result; -1 means that the prediction classification is completely inconsistent with

Table 2: Features of NSL-KDD dataset.

No.	Feature	Type
1	duration	Numeric
2	protocol_type	Nonnumeric
3	service	Nonnumeric
4	flag	Nonnumeric
5	src_bytes	Numeric
6	dst_bytes	Numeric
7	land	Numeric
8	wrong_fragment	Numeric
9	urgent	Numeric
10	hot	Numeric
11	num_failed_logins	Numeric
12	logged_in	Numeric
13	num_compromised	Numeric
14	root_shell	Numeric
15	su_attempted	Numeric
16	num_root	Numeric
17	num_file_creations	Numeric
18	num_shells	Numeric
19	num_access_files	Numeric
20	num_outbound_cmds	Numeric
21	is_host_login	Numeric
22	is_guest_login	Numeric
23	count	Numeric
24	srv_count	Numeric
25	serror_rate	Numeric
26	srv_error_rate	Numeric
27	rerror_rate	Numeric
28	srv_rerror_rate	Numeric
29	same_srv_rate	Numeric
30	diff_srv_rate	Numeric
31	srv_diff_host_rate	Numeric
32	dst_host_count	Numeric
33	dst_host_srv_count	Numeric
34	dst_host_same_srv_rate	Numeric
35	dst_host diff_srv_rate	Numeric
36	dst_host_same_src_port_rate	Numeric
37	dst_host_srv_diff_host_rate	Numeric
38	dst_host_serror_rate	Numeric
39	dst_host_srv_serror_rate	Numeric
40	dst_host_rerror_rate	Numeric
41	dst_host_srv_rerror_rate	Numeric

the actual classification

$$\mathrm{MCC} = \frac{\mathrm{TP} * \mathrm{TN} - \mathrm{FN} * \mathrm{FP}}{\sqrt{(\mathrm{TP} + \mathrm{FN}) * (\mathrm{TP} + \mathrm{FP}) * (\mathrm{TN} + \mathrm{FP}) * (\mathrm{TN} + \mathrm{FN})}}. \tag{5}$$

Kappa: kappa coefficient is an index for consistency test, which can be used to measure the effect of classification. For classification problems, the so-called consistency is whether the predicted results of the model are consistent with the actual classification results. The calculation of kappa coefficient is based on the confusion matrix, and the value is between -1 and 1, usually greater than 0, where P_O is the overall accuracy and P_e is the accidental consistency error

$$\mathrm{kappa} = \frac{P_O - P_e}{1 - P_e}. \tag{6}$$

Area under the curve (AUC): ROC (receiver operating characteristic) is a curve connected by the points of FPR and TPR. The horizontal axis is fpr, which means the proportion of negative samples is wrongly classified as positive samples in all positive samples, and the vertical axis is TPR, that is, Sn mentioned above, which means the proportion of positive samples is correctly classified in all positive samples. The often mentioned AUC value is the area under the ROC curve. In other words, AUC is the area formed by the ROC "curve" and the straight line $x = 0$ and $y = 1$. AUC is mainly used to measure the performance or generalization ability of algorithms in binary classification problems. As a numerical value, we can intuitively evaluate the quality of the classifier. The larger the value, the better

$$\begin{aligned} \mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \\ \mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \end{aligned} \tag{7}$$

TP (true positive), TN (true negative), FP (false positive), and FN (false negative) are the number of true positive, true negative, false positive, and false negative samples, respectively. TP means true positive, that is, the number of positive cases predicted; TN indicates true positive, that is, the number of positive cases predicted as negative cases; FP indicates false positive, that is, the number of negative cases predicted to be positive; and FN means false negative, that is, the number of positive cases predicted as negative cases.

3.3. Experimental Method

3.3.1. Crossvalidation. In this study, tenfold crossvalidation was selected to measure the performance of the model. The process of this verification method is described as follows: first, the entire data set will be randomly divided into ten copies, and the contents will not be repeated. Then, a single subset is randomly selected and retained as validation data to test the model, while the remaining 9 are used as training data to train the prediction model. This process goes through 10 times; that is, every piece of data will be used as a test data. Finally, the 10 results are averaged to obtain the final prediction results. Crossvalidation is often used in machine learning model evaluation, and it is rarely used in the field of deep learning. Because it is expensive to train and verify many times in the process of deep learning, it can be used when the amount of data is small.

3.3.2. Independent Testing. Compared with crossvalidation, independent testing is time-consuming and logically simple. First, the algorithm is trained on the training set. Secondly, the parameters of the model are adjusted by observing the performance of the model according to the evaluation indicators each time. At the same time, independent testing is also a method to test the effect of the model. Generally, independent test sets are used to verify the effect of the model in the end of experiments. The specific way is to use independent test sets as common data to compare with other methods. The above two experimental methods have been applied in this study. Generally, crossvalidation and independent testing experiments at the same time will make the experimental results more convincing.

3.4. The Proposed Predictive Framework. The prediction process can be concluded as two stages: (1) model training and (2) prediction. In the training phase, training samples are encoded and integrated by the feature representation algorithm. Then, the features are optimized to obtain the best feature subset and then fed into the decision tree algorithm to train the prediction model MaxAbs-DT. In the prediction stage, given the uncharacteristic traffic samples, we follow a similar process to encode the samples and use a trained model to predict whether the query sequence is an abnormal sample. The decision tree model gives a score for each traffic sample to measure the probability of its normal traffic. If the score is higher than 0.5, it is considered as a normal sample, otherwise, no.

3.5. Classification Algorithm. The decision tree [17] is converted into an if-then rule: a rule is constructed from each path from the root node of the decision tree to the leaf node; the characteristics of the internal nodes on the path correspond to the conditions of the rule, and the class label of the leaf node corresponds to the conclusion of the rule. It is also the final result of the decision.

The basic principle of the construction of the decision tree is the following strategy. A recursive process from root to leaf is looking for a "partition" attribute at each intermediate node.

(1) Start: build the root node, all training data are placed at the root node, select an optimal feature, divide the training data set into subsets according to this feature, and enter the child nodes

(2) All subsets are recursively divided according to the attributes of internal nodes

(3) If these subsets have been basically correctly classified, then construct leaf nodes and assign these subsets to the corresponding leaf nodes

(4) Each subset is assigned to leaf nodes; that is, it has a clear class, thus generating a decision tree

The depth of the tree structure is a very important parameter. Generally speaking, the deeper the tree structure, the better the fitting effect on the data, but it may also lead to overfitting. In this study, we implemented the DT algorithm (version 2.7.15) using the DT library in Python. In this study, we finally set the depth of the tree to 15 through a series of experiments, which can better capture the information of the data. The classification algorithm optimization results can be seen in "Classifier Optimization."

3.6. One-Hot Feature Representation. In many experimental studies, using one hot coding, the value of discrete features of data can be extended to European space, and a value of discrete features will correspond to a point in European space. In this way, if the discrete features are encoded by one-hot, the distance calculation between features will be more reasonable and easier to understand. The one hot feature extraction method [18] is often used in the field of sequence recognition and NLP and other related fields. In most cases, it can obtain excellent experimental results.

As mentioned above, the NSL-KDD dataset has 38 numeric and 3 nonnumeric features. Like many models, the proposed decision tree model only deals with numerical data input. Therefore, we need to convert all nondigital features into digital representation. Features (protocol_type, service, flag) are nondigital features that needs to be converted to digital form in NSL-KDD dataset. In the training set, features, protocol_type, service, and flag, have 3, 70, and 11 categories, respectively. In the test set, features, protocol_type, service, and flag, have 3, 64, and 11 categories, respectively. For the consistency of data, we uniformly set protocol_type, service, and flag as 3, 70, and 11 categories, respectively, and then perform the one-hot encoding process.

3.7. Feature Scaling. After successfully converting these features into digital form, the next appropriate thing is feature scaling. Feature scaling ensures that the dataset is in a normalized form. The values of some features (such as "src_bytes" and "dst_bytes") in NSL-KDD dataset are unevenly distributed; so, it is necessary for us to use feature scaling technology. In this way, we can ensure that our classifier will not produce biased results. There are several feature scaling methods as follows:

Z-score normalization: The Z-score method is standardized based on the average value (mean) and standard deviation of the original data. The average data after processing is 0, and the square difference is 1, which meets the standard normal distribution. The main purpose is to unify different dimensions of data into the same order of magnitude and measure the calculated Z-score value uniformly to ensure comparability between data. The formula is as follows

$$x_{\text{normalization}} = \frac{x - \mu}{\mathrm{O}}. \tag{8}$$

Among them, x represents the original data, μ represents the mean value of the original data, O represents the standard deviation of the original data, and $x_{\text{normalization}}$ represents the data after the normalization process.

min-max standardization: min-max standardization refers to the linear transformation of the original data, and the value is mapped between [0, 1]

$$x_{\text{normalization}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}. \tag{9}$$

MaxAbs normalization: it is usually used for sparse matrices. Using this method for standardization, the data can fall into the specified range of [-1, 1], and the original structure of the data will not be damaged. The formula is as follows

$$x_{\text{normalization}} = \frac{X}{|X_{\max}|}. \tag{10}$$

Robust scaler normalization: for data with outliers, if the Z-score method is used for standardization, the characteristics of outliers are often lost after standardization; so, the standardized data is not ideal. In this case, the robust scaler method can be considered. Robust scaler has a standardized processing method for outliers, which has stronger parameter control for data centralization and data scaling robustness

$$x_{\text{normalization}} = \frac{X - X_{\text{median}}}{\mathrm{IQR}}. \tag{11}$$

X_{median} is the median of the sample, and IQR is the interquartile distance of it.

3.8. Feature Selection. PCA (principal component analysis) is a commonly used data analysis method. PCA transforms the original data into a set of linearly independent representations of each dimension through linear transformation, which can be used to extract the main feature components of the data, and is often used for dimensionality reduction of high-dimensional data.

4. Results and Discussion

4.1. Classifier Optimization. To achieve the best performance, we conducted the following experiments to optimize the DT classifier. We performed parameter optimization on the depth of the decision tree in order to find the optimal max-depth value. Figure 1 shows the classifier evaluation index curves under different decision tree depths. Next, we need to determine which depth of the decision tree model is best for our dataset. Therefore, we compared the performance of the three cores. We can observe in Figure 1 that when the max depth of the tree goes from 15 onwards, all performance indicators basically do not change, and all the curves are in a parallel state. Consequently, the DT with max depth 15 was used to train this model in our predictor.

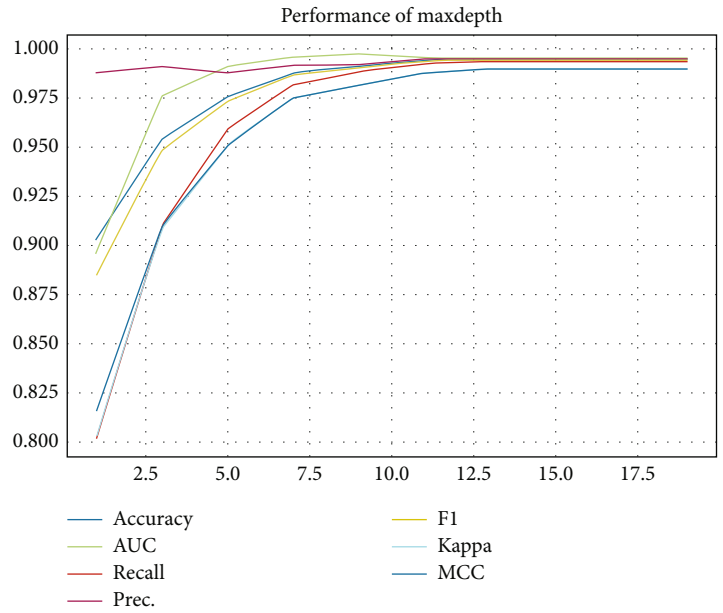

Figure 1: Evaluation indicator curves of different max depths in DT.

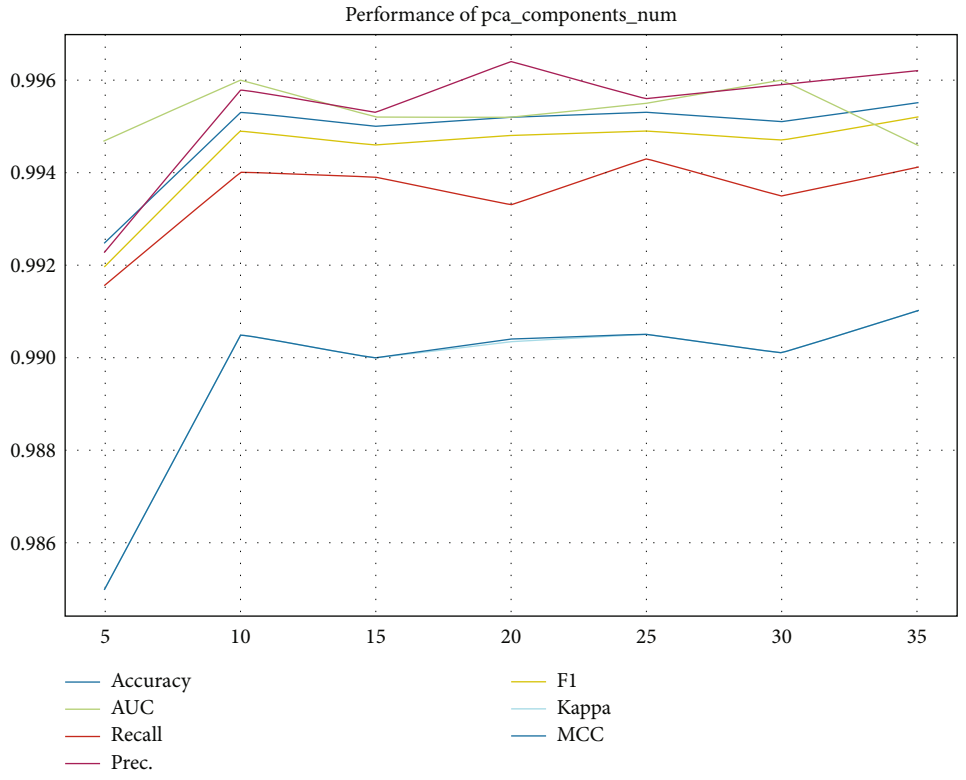

Figure 2: Evaluation indicator curves of different pca_components_nums in PCA.

In order to obtain the optimal feature subset, we conduct the following experiments to optimize the features. We perform parameter optimization on DT classifier using the PCA feature selection method to find the best number of features to keep. Figure 2 shows the classifier evaluation index curves under different numbers of retained features. We can observe in Figure 2 that when the number of features is at the position of 10, all the performance indicators basically reach the highest level. Consequently, the number of features to retain is set to 10 by the PCA method in our predictor.

4.2. Scale and Transform. There are generally four methods for data normalization. In order to get the optimal normalization method, we carried out the following experiments to

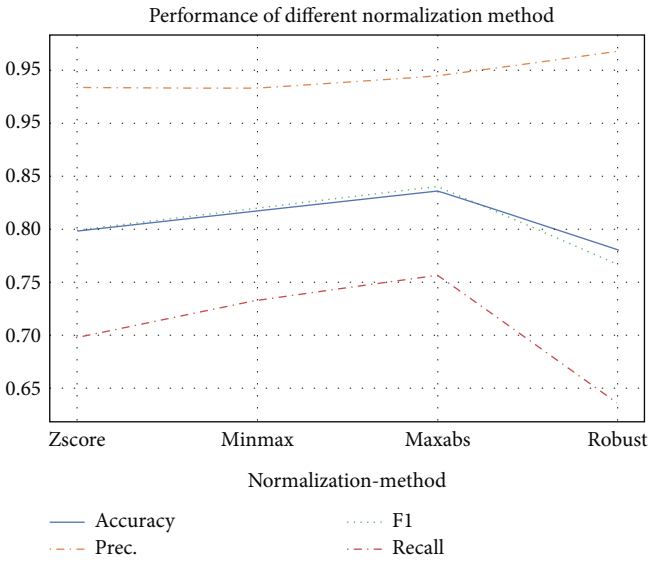

Figure 3: Evaluation indicator curves of different normalization methods in DT.

Table 3: Performance of different normalization methods.

Classifier	Accuracy	Prec.	F1	Recall
Z-score	0.7999	0.9344	0.7988	0.6975
min-max	0.8182	0.9332	0.8211	0.733
MaxAbs	0.8364	0.9452	0.8404	0.7565
Robust	0.7807	0.9682	0.7675	0.6357

Table 4: Performance of different classifiers.

Classifier	Accuracy	Prec.	F1	Recall
et	0.7919	0.9675	0.7822	0.6565
rf	0.7616	0.9681	0.7416	0.601
rbfsvm	0.7847	0.9233	0.7819	0.6781
knn	0.772	0.9231	0.7656	0.654
dt	0.8364	0.9452	0.8404	0.7565
gbc	0.7796	0.9684	0.766	0.6335
ada	0.7819	0.9225	0.7786	0.6735
svm	0.7569	0.9178	0.7466	0.6292

determine. We use 4 data normalization methods, namely, *Z* -score, min-max, MaxAbs, and robust scaler on the test dataset to compare the metrics of the models to determine the best data normalization method. Figure 3 shows the classifier evaluation index curves under different normalization methods. We observed in Figure 3 that when the normalization method is MaxAbs, all performance indicators basically reached the highest value, specifically ACC, F1, and recall obtained the highest value and obtained the second highest value on Prec. The results are listed in Table 3. As can be seen, amongst the four individual normalization methods, the method called MaxAbs performs the best than the other three. This indicates that the MaxAbs method is more useful for NSL-KDD abnormal prediction.

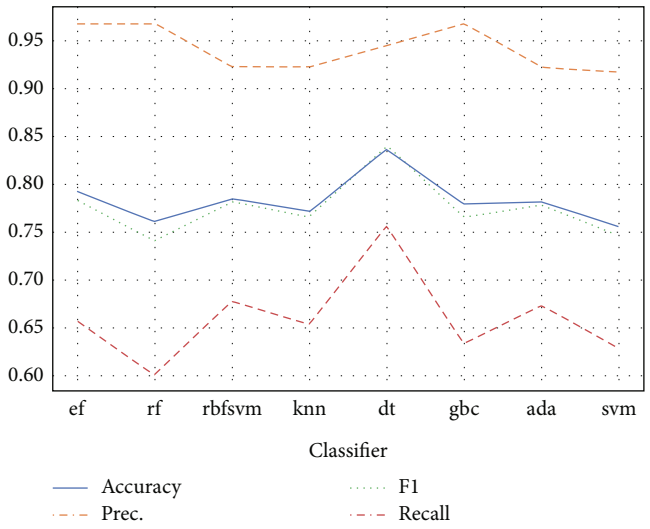

Figure 4: Evaluation indicator curves of different classifiers on KDDTest$^+$.

4.3. Comparison with Different Classification Algorithms. To measure the effectiveness of our proposed work, we compared its performance with several well-known classifiers, such as the extra trees classifier [19], random forest (rf) [20], SVM with RBF core (rbfsvm) [21], SVM with linear core (svm) [22], *K*-nearest Neighbor (knn) [23], gradient boosting classifier (gbc) [24] and AdaBoost classifier (ada) [25]. For a fair comparison, we trained classifiers on the same train dataset KDDTrain$^+$ with our feature set and then fine-tuned the classifiers one by one to achieve the optimal performance. The classifiers are also evaluated by tenfold crossvalidation, and the evaluation results on the same test dataset KDDTest$^+$ are presented in Table 4. We can see that DT achieved the highest value in three indicators, namely, ACC reached 81.86%, F1 reached 81.68%, and recall reached 71.04% outperforming other classifiers in three of the four metrics. Specifically, the performance of DT classifier on ACC, F1, and recall was higher than the second-place classifier ET by 2.67%, 3.46%, and 5.39%, respectively. The results presented in Figure 4 show that compared with the other seven classifiers in this study, the DT classifier has better discriminative ability to distinguish abnormal traffic from normal traffic.

As shown in Figure 5, in terms of the consumption time of the classification model, which is an important consideration in the actual industrial application, our proposed work takes the least time compared to other classifiers and has high practical significance. The specific values are shown in Table 5.

4.4. Comparison with Existing Classifiers. To better demonstrate the performance of our proposed MaxAbs-DT model, we compared its performance with seven existing intrusion detection techniques. As shown in Table 6, we compared the performance with other techniques mentioned in Tavallaee et al. [16] and Yin et al. [26]. The results prove that the MaxAbs-DT classifier is superior in detecting network anomalies. From Table 6, the MaxAbs-DT model improves

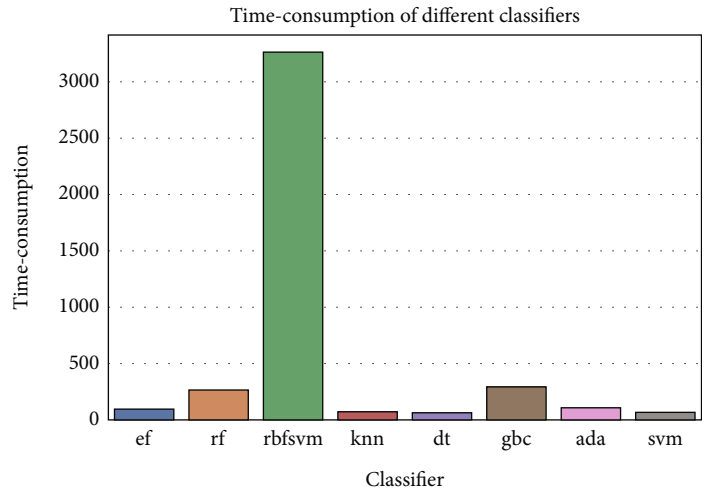

Figure 5: Time consumption histogram of different classifiers.

Table 5: Time consumption of between different classifiers.

Classifier	Time consumption (sec)
et	99.1
rf	270.448
rbfsvm	3265.208
knn	75.077
dt	67.582
gbc	294.056
ada	111.741
svm	69.829

Table 6: Comparison between existing methods using KDDTest$^+$ for binary classification.

Classifier	Accuracy
SVM [16]	69.52%
Naïve Bayes [16]	76.56%
Multilayer perceptron [16]	77.41%
Random tree [16]	80.67%
J48 [16]	81.05%
NB tree [16]	82.02%
RNN [26]	83.28%
DT (this study)	83.64%

the detection accuracy of the RNN model by 0.44% on the KDDTest$^+$ datasets.

The results clearly demonstrate that the detection performances of the proposed model are more effective. Although the processing time of decision tree classifier is short, the overall classification accuracy is not significantly increased. On the contrary, deep learning may take a long time, but it may have a better ability to capture other potential features of data. For this, it is worth further exploring in the future research work. For scenarios where the response time is required to be as short as possible, the classifier proposed in this paper may be more useful.

5. Conclusions

This work proposed the application of an improved decision tree classifier to detect network intrusions. The proposed work showed good performance and achieved nice results. To validate the performance of our model, we train and validate the model using the NSL-KDD dataset, which is currently widely used as a benchmark dataset for intrusion detection by most researchers. After the experiment, the MaxAbs-DT model obtained a higher accuracy, recall, and F-score and less execution time than other models, and it also outperforms other existing models on the accuracy metric in literature. However, there are many deep learning technologies that may have potential optimization capabilities for this research. In the future, we plan to integrate some optimal feature selection methods with novel deep learning models, such as graph convolutional neural networks, to explore better network anomaly classifier performance.

Data Availability

The datasets generated for this study can be found in https://github.com/Kevin-chen-sheng/MaxAbs-DT.

Conflicts of Interest

The authors declare that there are no conflict of interest.

Acknowledgments

The authors would like to give thanks for the financial support provided by the Research Institute of China Telecom Corporation Limited and encouragement of colleagues in the Research Institute.

References

[1] F. Masoodi, S. Alam, and S. T. Siddiqui, "Security & privacy threats, attacks and countermeasures in Internet of Things," *International Journal of Network Security & Its Applications*, vol. 11, no. 2, pp. 67–77, 2019.

[2] S. Alam, M. Shuaib, and A. Samad, "A collaborative study of intrusion detection and prevention techniques in cloud computing," in *International Conference on Innovative Computing and Communications*, Singapore, 2019.

[3] L. Lv, W. Wang, Z. Zhang, and X. Liu, "A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine," *Knowledge-Based Systems*, vol. 195, article 105648, 2020.

[4] F. Z. Belgrana, N. Benamrane, M. A. Hamaida, A. M. Chaabani, and A. Taleb-Ahmed, "Network intrusion detection system using neural network and condensed nearest neighbors with selection of NSL-KDD influencing features," in *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS),.*, Bali, Indonesia, 2021.

[5] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[6] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Computing and Applications*, vol. 28, Supplement1, pp. 1051–1058, 2017.

[7] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert Systems with Applications*, vol. 148, article 113249, 2020.

[8] A. Bachar, N. El Makhfi, and O. E. L. Bannay, "Machine learning for network intrusion detection based on SVM binary classification model," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 4, pp. 638–644, 2020.

[9] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *2015 international conference on signal processing and communication engineering systems*, Guntur, India, 2015.

[10] S. K. Gautam and H. Om, "Computational neural network regression model for host based intrusion detection system," *Perspectives in Science*, vol. 8, pp. 93–95, 2016.

[11] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: a survey," *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.

[12] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, Kyoto, Japan, 2012.

[13] 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.

[14] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," *EAI Endorsed Transactions on Security and Safety*, vol. 4, no. 12, 2017.

[15] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 12, pp. 1848–1853, 2013.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, Ottawa, ON, Canada, 2009.

[17] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[18] P. Rodríguez, M. A. Bautista, J. Gonzalez, and S. Escalera, "Beyond one-hot encoding: lower dimensional target embedding," *Image and Vision Computing*, vol. 75, pp. 21–31, 2018.

[19] A. Sharaff and H. Gupta, "Extra-tree classifier with metaheuristics approach for email classification," in *Advances in Computer Communication and Computational Sciences*, pp. 189–197, Springer, 2019.

[20] Z. Lv, S. Jin, H. Ding, and Q. Zou, "A random forest sub-Golgi protein classifier optimized via dipeptide and amino acid composition features," *Frontiers in Bioengineering and Biotechnology*, vol. 7, p. 215, 2019.

[21] Y. Liu, D. Chen, R. Su, W. Chen, and L. Wei, "iRNA5hmC: the first predictor to identify RNA 5-hydroxymethylcytosine modifications using machine learning," *Frontiers in Bioengineering and Biotechnology*, vol. 8, p. 227, 2020.

[22] B. Liu, H. Wu, and K. C. Chou, "Pse-in-one 2.0: an improved package of web servers for generating various modes of pseudo components of DNA, RNA, and protein sequences," *Natural Science*, vol. 9, no. 4, p. 67, 2017.

[23] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2018.

[24] Z. Liao, S. Wan, Y. He, and Q. Zou, "Classification of small GTPases with hybrid protein features and advanced machine learning techniques," *Current Bioinformatics*, vol. 13, no. 5, pp. 492–500, 2018.

[25] T. K. An and M. H. Kim, "A new diverse AdaBoost classifier," in *2010 International conference on artificial intelligence and computational intelligence*, Sanya, China, 2010.

[26] 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.