Hindawi
Security and Communication Networks
Volume 2024, Article ID 9848463, 1 page
https://doi.org/10.1155/2024/9848463

Retraction

Retracted: IoT-Oriented Distributed Intrusion Detection Methods Using Intelligent Classification Algorithms in Spark

Security and Communication Networks

Received 8 January 2024; Accepted 8 January 2024; Published 9 January 2024

Copyright © 2024 Security and Communication Networks. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Q. Xu, X. Jia, B. Jia, and Y. Liang, "IoT-Oriented Distributed Intrusion Detection Methods Using Intelligent Classification Algorithms in Spark," *Security and Communication Networks*, vol. 2022, Article ID 2842624, 11 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 2842624, 11 pages
https://doi.org/10.1155/2022/2842624

RETRACTED

Research Article

IoT-Oriented Distributed Intrusion Detection Methods Using Intelligent Classification Algorithms in Spark

Qiang Xu, Xiteng Jia, Bin Jia, and Yongquan Liang

College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, Shandong 266590, China

Correspondence should be addressed to Bin Jia; jiab_sdust@126.com

Received 16 December 2021; Revised 6 April 2022; Accepted 14 April 2022; Published 22 September 2022

Academic Editor: Xingsi Xue

Copyright © 2022 Qiang Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the rapid development and wide application of the 5G mobile communication and the explosive security threats of the Internet of things (IoT), distributed intrusion detection is one of the hot topics in the intrusion detection field of network security. The classification algorithm is a kind of the most representative and classical algorithms of artificial intelligence (AI), and it is an important technique for intrusion detection in order to distinguish the attack traffic from massive network data. In order to solve the problem to detect massive and complex network attack traffic in IoT, in this study, we propose the distributed intrusion detection framework and method using intelligent classification algorithms in Spark. We first introduce several mainstream classification algorithms provided by Spark. Second, the distributed intrusion detection procedure using intelligent classification algorithms is given. Next, the overall framework of the proposed model is built. Finally, a series of comparison experiments by the binary classification and quintuple classification in six evaluation indicators (i.e., recall, precision, *F*1-score, FNR, FPR, and ROC curve) indicate that the naive Bayes has a worse classification performance than that of other classification algorithms, and the classification effect in a cluster environment is almost the same as that in a stand-alone environment.

1. Introduction

Nowadays, the cyberattack is an ongoing, destructive network intrusion behavior. The threat is aimed at information service systems, computer network, industrial infrastructure, and smart terminals. Wherever computers and the Internet go, the cyberattack behavior haunts us. The intrusion detection system (IDS) differs from other network security devices, and it leverages big data and proactive strategies to perform real-time detection. Based on the trusted detection results, some forward-looking security protection measures are adopted into the intrusion prevention system (IPS). It includes hardware and software, which can actively or passively control hosts or network to detect some violations [1]. Its function is to detect and take countermeasures against the intrusion behavior of the host and network system, and it is the important equipment to identify an attempted intrusion or ongoing detriment [2]. The traditional IDS mainly includes the host-based intrusion detection system (HIDS), the network-based intrusion detection system (NIDS), and the hybrid intrusion detection system (hybrid IDS).

With the rapid development and wide application of the 5G mobile communication and the explosive security threats of the Internet of things (IoT) [3], it posed some new challenges to network security and IDS. The dispersibility of big data and the multisource peculiarity of compound attacks are the key features of the future Internet. Therefore, the single host-based intrusion detection and the network-based intrusion detection technologies have been increasingly unable to meet the security requirements of the current complex and diverse attack behavior recognition. In addition, the IoT devices based on 5G have a primary function for which computation of massive data is required [4]. The high capacity and complexity of safety audit data on large-scale and high-speed networks are overwhelming to the traditional IDS. The distributed intrusion detection is one of the hot topics in the intrusion detection field of network security [5].

RETRACTED

Spark and Hadoop supported by the Apache Software Foundation are the most famous and widely used open-source parallel distributed computing platforms for massive data processing [6]. A work in Hadoop is called the "Job," and a Job is divided into the Map Task and the Reduce Task. The work submitted by Spark users is called the "Application" that corresponds to a SparkContext. Multiple jobs exist in an application. While one time of operation "Action" is triggered, a job is created. These jobs can be executed in parallel or in a serial way, and each job has multiple stages. The stages are acquired by dividing the jobs by the DAGScheduler based on the dependency between every two resilient distributed datasets (RDDs) in the shuffle. Each stage contains multiple tasks, which constitute task sets. The task sets are distributed to each executor for execution by TaskScheduler. The life cycle of an executor is the same as that of an application. Even if there is no running of jobs, the tasks can be quickly started to read the memory for calculation.

Nowadays, the machine learning and deep learning methods in artificial intelligence (AI) have incarnated their unique advantages in intrusion detection except for the scattered cyberattacks like distributed denial of service (DDoS). The classification algorithm is one of the most representative and classical algorithms. From the perspective of classification, the intrusion detection based on intelligent classification algorithms can extract the features of network flow and host session from a bulk of Internet data, and they learn the classification model to discover the classification rules of hidden intrusion behavior [7]. The classification algorithms include binary classification and multi-classification. Some binary classification methods can be directly extended to multiclassification methods; however, the binary classification learner is usually used to solve multiclassification problems based on some basic strategies. A distributed computing environment (i.e., Apache Spark) is incorporated to accelerate the implementation process of these classification algorithms [8].

In order to overcome the existing shortcomings of IDS in current IoT, this study proposes the IoT-oriented distributed intrusion detection methods using intelligent classification algorithms in Spark. Compared with the previous work, the proposed method and model have the following advantages.

(1) The four typical classification algorithms provided by Spark are used, which combine the advantages of traditional machine learning. The distributed detection framework deployed in Spark based on different intelligent classification algorithms is innovatively proposed.

(2) A set of novel data processing methods by the LabelEncoder, one-hot, and principal component analysis (PCA) are built. LabelEncoder coding is used to process the classification features of character data. One-hot coding represents the eigenvalue by multidimensional vectors with LabelEncoder. The PCA technique is used to select typical features and reduce the feature dimension. Our method eliminates the uncorrelated and redundant data from the dataset to achieve better classification performance.

(3) We deploy the Spark cluster to compare with the stand-alone environment. The experiments prove the feasibility of using intelligent classification algorithms for network traffic intrusion detection in the distributed environment.

A series of comparison experiments by the binary classification and quintuple classification in recall, precision, $F1$-score, FNR, FPR, and ROC curve indicate that the naive Bayes has a worse classification performance than that of other classification algorithms, and the classification effect in a cluster environment is almost the same as that in the stand-alone environment. Thus, the three other algorithms besides the naive Bayes are given priority as our distributed detection algorithms.

The rest of this study is arranged as follows. Section 2 mainly presents the related work to IDS research. Section 3 introduces the NSL-KDD datasets and the classification algorithms provided by Spark in this study and analyzes the related data preprocessing procedures. Section 4 gives our method and model in distributed intrusion detection. Section 5 carries out the experiments to verify our method and model, and Section 6 concludes the work.

2. Related Work

Although the method and technology of IDS have been developed over the years, there are still some urgent things to detect and resist complex distributed cyberattacks in IoT. For example, the traditional IDS mostly employs individual classification methods, which do not provide a satisfactory attack detection rate. The technique of a single model is more difficult to accurately predict the different types of invasion. Meanwhile, the generalization ability of a single model is insufficient, and its detection ability is not enough as facing distributed multipoint attacks.

In addition, with the booming development of AI, many machine learning and deep learning methods have been increasingly used in the intrusion detection field. Some typical methods applied to intrusion detection are as follows: dimensionality reduction method, supervised machine learning, semisupervised machine learning, unsupervised machine learning, deep learning, and ensemble learning [9].

The smart IDS should have the ability to analyze the representative data characteristics to reduce their dimensions. The correlational studies on this aspect mainly include the following. Jia et al. [10] focused on how to distinguish the malicious traffic from normal flows in big data. They proposed a novel real-time DDoS attack detection mechanism based on multivariate dimensionality reduction analysis (MDRA). In the mechanism, the authors first reduced the dimensionality of multicharacteristic variables in a network traffic record by PCA. Then, the correlation of the lower dimensional variables is analyzed. Finally, the malicious traffic can be differentiated from the normal flows by MDRA and Mahalanobis distance. Hussain et al. [11] realized a set of linear discriminant analysis (LDA) and PCA feature extraction algorithms. The whole PCA-LDA method generates better results and shows a higher precision ratio than the existing single feature extraction method. The eigenvalue decomposition of PCA has some limitations. The foremost components obtained by the PCA may not be optimal in the case of non-Gaussian distribution.

Some typical research in recent years with regard to the supervised learning, the semisupervised learning, and the unsupervised learning in machine learning are as follows. Mebawondu et al. [12] presented the lightweight IDS based on information gain and neural network with multilayer perceptron. The gain ratio was used to select some relevant features of attack and normal traffic prior to classification by using a neural network. Some pre-existing solutions by adopting supervised learning-based intrusion detection need a big labeled set for better accuracy. However, it is not easy to source the labeled dataset due to the huge size of IoT. In order to overcome the impediments in the pre-existing solutions, Ravi and Shalinie [13] proposed a unique SDRK (semisupervised machine learning and deep feedforward neural network and repeated random sampling and K-means) machine learning method to detect intrusion behavior. The SDRK leverages the supervised deep neural networks (DNNs) and the unsupervised clustering techniques. The intrusion detection and mitigation schemes are placed in the fog nodes that lie between the IoT and the cloud. Nisioti et al. [14] provided a comprehensive outlook of the hybrid unsupervised methods to detect intrusion behavior, discussing their potential in the field. The authors highlighted the importance of feature engineering that was proposed for intrusion detection, and they also discussed that the pre-existing IDS should evolve from simple detection to correlation and attribution.

There are the problems that the supervised classifiers are prone to adversarial evasion, and the existing countermeasures suffer from some limitations. Most solutions degrade the performance in the absence of adversarial perturbations, and they are unable to face new attack variants. Apruzzese et al. [15] built a novel framework to protect botnet detectors from adversarial attacks through deep reinforcement learning mechanisms. It automatically generates realistic attack samples evading detection, and the samples are used to produce an augmented training dataset to yield the hardened detectors. In such a way, more resilient detectors are obtained, and they can work even against unforeseen evasion attacks with the great merit of not penalizing the performance in the absence of specific attacks. Gamage and Samarabandu [16] first introduced the taxonomy of deep learning models in intrusion detection, and they summarized the research on this topic. Then, the four key deep learning models are trained and evaluated, i.e., feedforward neural network, autoencoder, deep belief network, and long short-term memory network, for the intrusion classification tasks on four legacy datasets and two modern datasets.

In addition, ensemble learning has also been an important branch of AI and has paid growing attention. Adaptive boosting (AdaBoost) and random forest algorithms are two typical methods of ensemble learning [17]. Hu et al. [18] proposed two online AdaBoost-based intrusion detection methods. In the former, a traditional online AdaBoost is used where the decision stumps are used as weak classifiers. In the latter, an improved online AdaBoost is achieved, and the online Gaussian mixture models (GMMs) are used as weak classifiers. Resende and Drummond [19] told us a survey of methods based on the random forest applied in IDS, considering the particularities involved in some models.

Although the abovementioned work has made the updated developments and research fruits, however, the ability to detect massive and complex network attack traffic in IoT needs further improvement. To the best of our knowledge, there are some innovations to solve the distributed security vulnerabilities by providing the proposed distributed detection framework based on the intelligent classification in Spark, which is analyzed in a subsequent discussion. Our research has great application value in real-time big data intrusion detection in IoT.

Compared with the existing research and application, the classification algorithms provided by Spark can better adapt to the distributed computing platform. In addition, compared with other stand-alone environment, the classification algorithms have the same outstanding detection performance in the binary classification and multiclassification.

3. Preliminaries

In this study, we select the four typical classification algorithms provided by Spark as the core techniques of distributed intrusion detection, and they are logistic regression, naive Bayes, decision tree, and multilayer perceptron, respectively.

In addition, how to select a credible experimental dataset is crucial. The KDD CUP 99 dataset [20] is a classic and authoritative dataset of network intrusion detection, and it has become an effective benchmark in this field. The NSL-KDD dataset [21] is the version to improve the KDD CUP 99 dataset [22]. Some redundant and duplicate records have been removed from the NSL-KDD dataset. Therefore, we use the NSL-KDD dataset. However, it needs to be preprocessed in order to make the experimental results reliable.

3.1. Classification Algorithms

3.1.1. Logistic Regression. Logistic regression is a generalized linear regression analysis model. Binary logistic regression and multivariate logistic regression are provided by Spark MLlib for binary classification and multiclassification, respectively.

First, in the binary logistic regression, the formula of the prediction function is as follows:

$$g(z) = \frac{1}{1 + e^{-z}}, \tag{1}$$

where if $z > 0$, then $0.5 < g < 1$; else if $z < 0$, then $0 < g < 0.5$. By this time, the output in regression is the input of the function $g\ (z)$, and the final output is the probability of a certain category. The complete prediction function is shown as follows:

RETRACTED

$$h_\theta(x) = g\left(\theta^T x\right) = \frac{1}{1+e^{-\theta^T x}}. \tag{2}$$

The purpose of machine learning is to get a training model for calculating the parameter θ, and the coefficient model about θ can be solved by the maximization likelihood function in probability theory. The classification probability formula of binary logistic regression is denoted as follows:

$$\begin{aligned} P(y=1|x,\theta) &= h_\theta(x), \\ P(y=0|x,\theta) &= 1-h_\theta(x), \\ P(y|x,\theta) &= h_\theta(x)^y\left(1-h_\theta(x)\right)^{1-y}. \end{aligned} \tag{3}$$

The likelihood function represents the similarity between the actual situation and the whole estimated situation. The logarithmic formula of the likelihood function is expressed as follows:

$$\log L(\theta) = \sum_{i=1}^{n} y^{(i)}\log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right). \tag{4}$$

However, the loss function is the difference between the overall actual situation and the estimated situation, and it is as opposed to the likelihood function. Therefore, the loss function of binary logistic regression is "$-\log L(\theta)$," and the problem of maximizing the likelihood function is transformed into the problem of minimizing the loss function. In this study, we adopted the loss function after $L2$ regularization shown as follows:

$$\text{Loss}(\theta) = -\log L(\theta) + \lambda \sum_{j=1}^{m}\left(\theta_j\right)^2. \tag{5}$$

In addition, the L-BFGS algorithm is used to optimize loss function.

Next, the multivariate logistic regression provided by Spark MLlib uses the softmax function to make multiple classifications in nature. For the k categories, one of the classes is considered as the main class. First, the $k-1$ binary logistic regressions are performed. Then, the main class and the other $k-1$ classes are to perform the categorical regression.

3.1.2. Naive Bayes. A naive Bayes classifier [23] is a probabilistic model, which is also used for binary classification and multiclassification. The naive Bayes method assumes that the conditions of feature attributes in a dataset are mutually independent, that is, there is no correlation between every two features. The mathematical models of the algorithm are expressed as follows:

$$\begin{aligned} P(Y=y_i|X) &= \frac{P(Y=y_i)P(X|Y=y_i)}{P(X)}, \\ P(X|Y=y_i) &= \prod_{j=1}^{d} P\left(X_j|Y=y_i\right). \end{aligned} \tag{6}$$

According to the input eigenvectors, to which the class of the eigenvector belongs, the probability of y_i is able to judge. Next, all categories get traversal. Finally, the category with the highest output probability is chosen as the category of this eigenvector.

In this study, the Laplacian smooth class-conditional probability is used, and it is to avoid the problem of $P(Y = y_i|X) = 0$ due to no eigenvalues in the sample.

3.1.3. Decision Tree. The decision tree [24] is a tree model that uses the probability to classify, and it includes leaf nodes, internal nodes, and branches. In the decision tree, the internal nodes represent to divide a decision tree by a certain feature, the branches represent the types of eigenvalues of the feature, and the leaves represent the final classification results.

There are three main steps in building a decision tree. (a) The optimal feature is selected as the internal node to delimit the molecular node. (b) The subtrees are split according to the selected optimal features, and internal nodes or leaf nodes are recursively generated until the dataset is completely divided or reaches the given depth of the tree. (c) Because the decision tree is prone to overfitting, it is necessary to prune the generated decision tree model to reduce the size of the decision tree and prevent overfitting.

In the ideal state, the nodes should be divided by the optimal features, and the purity of partitioned nodes should be as high as possible. There are three important indexes to select the optimal features. They are information gain, information gain ratio, and Gini index, and the corresponding decision tree algorithms are ID3, C4.5, and CART, respectively. In the next experiment section, we use the CART algorithm provided by Spark MLlib, and the Gini index is used as the criterion to select the optimal feature of the divided nodes. Compared with ID3 and C4.5, the CART algorithm has better classification performance and only generates the binary trees during classification, while the former two algorithms both generate multiway trees. The formula of the Gini index is defined as follows:

$$\text{Gini}(p) = \sum_{k=1}^{K} p_k\left(1-p_k\right) = 1 - \sum_{k=1}^{K} p_k^2, \tag{7}$$

where K represents the number of classes, and p_k denotes the probability of which the sample point belongs to a certain class.

3.1.4. Multilayer Perceptron. Multilayer perceptron known as artificial neural network (ANN) is the most classical feedforward neural network algorithm. It is composed of multiple node layers, and they are the input layer node, hidden layer node, and output layer node, respectively. The input layer and output layer are both one layer of nodes, and the hidden layer contains multilayer nodes that each layer of nodes is connected through full connection. Each node in the hidden layer and the output layer contains a nonlinear activation function. The classical activation functions include the ReLU function, sigmoid function, softmax function, and tanh function. In this study, the sigmoid function is used as the activation function in the neure of the hidden layer, and the sigmoid function and softmax function are

RETRACTED

RETRACTED

used as the activation function in the neure of the output layer to make dichotomies and multiclassification.

The purpose of training multilayer perceptron is to calculate the optimal weight and the bias of each layer so that the output results have a smaller difference from the actual results. The loss function can be minimized by the gradient descent method.

Here, we use a four-layer neuronal architecture, and it includes one layer of input neurons, one layer of output neurons, and two layers of hidden neurons. The number of input neurons *InputLayers* is equal to the dimension of the eigenvectors. The number of hidden neurons in the first layer $Hide_1Layers$ is $\log_2 InputLayers$. The number of hidden neurons in the second layer $Hide_2Layers$ is $\sqrt{InputLayers + OutputLayers} + 1$. The number of output neurons *OutputLayers* is equal to the number of label types.

3.2. Data Preprocessing. First, the classification algorithms in machine learning and deep learning are based on the features of the dataset. After a series of processing, the features are transformed into the eigenvector as the input of an algorithm. In the NSL-KDD dataset, the feature data need to be converted into numeric data and then combined with other features to form the eigenvectors. In this study, the LabelEncoder coding is used to process the classification characters of feature data. The encoded features can be used as the input of some classification algorithms, such as naive Bayes and decision tree algorithms, which are not sensitive to the numerical value. However, each eigenvalue has a logical ordering relationship after feature coding. Some algorithms that are sensitive to the numerical value will cause a larger error, such as logistic regression and multilayer perceptron. This is because the values between every two variables will affect the output result of the model in the loss function of the algorithm. Therefore, the one-hot coding is adopted. The one-hot coding represents the eigenvalue by multidimensional vectors with LabelEncoder. For example, "0," "1," and "2" can be changed into three trivectors, i.e., "000," "001," and "010," respectively. The result of the above processing has less impact on the model of selected parameters after converting character features into numeric features. The one-hot coding is suitable for disordered classification features, and these features will not generate the ranking relations after one-hot coding.

Second, the features in the dataset have some different values in the light of the measuring unit. For example, the src_bytes and dst_bytes in the NSL-KDD represent the size of transmitted data between every two hosts, and the range is [0, 1379963888]. The serror_rate and srv_serror_rate indicate the proportion of SYN errors in TCP connections, and it ranges from 0 to 1. When the above features are input into the used algorithms, the features with a larger value range will take the dominant position, which causes the features with a smaller value range to weaken the effect of the trained model. In order to solve the problem that there is no comparability between every two features due to the different measuring units, the dataset should be standardized so that the continuous feature attributes in the dataset are at the same level and the practical significance of continuous attributes is eliminated.

The standardized formulae are as follows:

$$\begin{aligned} x' &= \frac{x-\mu}{\sigma}, \\ \mu &= \frac{1}{N}\sum_{1}^{N} x_i, \\ \sigma &= \sqrt{\frac{1}{N}\sum_{1}^{N}\left(x_i-\mu\right)^2} \end{aligned} \quad . \tag{8}$$

Standardization is usually applied to the algorithms that are greatly influenced by the size of eigenvalues, such as logistic regression and multilayer perceptron. However, the algorithms that are not affected by the size of feature variables and are greatly affected by the distribution or probability of feature variables, such as decision tree and naive Bayes, do not need standardization processing before data input.

Third, the continuous feature discretization is to process continuous eigenvalue in segments, and each segment is divided by a number. In the application of naive Bayes, decision tree, and other algorithms concerned with data probability, the discrete features have better interpretability than the continuous features, and they are easier to understand the model.

The discretization methods in this study are quantile discretizer and binarizer. The quantile discretizer automatically divides the continuous features according to the number of intervals given by the developer and tries to acquire the same number of samples in each interval. The developers do not need to specify the critical value of the interval and only need to give the number of intervals after dividing the interval. The binarizer divides the continuous attributes into two types of discrete features based on a threshold given by the developer.

Last but not least, when the classification features are processed by one-hot coding, high-dimensional vectors are used to denote the classification eigenvalue, which will make the dimension of the eigenvectors input to the algorithms higher. In addition, it results in a decline in the efficiency and the performance of the classification algorithm. In order to eliminate data redundancy and noise, a dimensionality reduction technique is introduced into our detection method. The PCA algorithm is used to extract less dimensional and more representative features [9]. One advantage of the PCA is the data-driven design by keeping the foremost components of feature messages and eliminating the correlated and measured feature messages.

4. Method and Model Using Classification Algorithms for Distributed Intrusion Detection in IoT

In the IoT application, by the method and model for distributed intrusion detection, the operating state of the system and network is real-time monitoring and intelligent analysis in order to find all kinds of the behaviors or results of attack and anomaly and makes responses. The ultimate

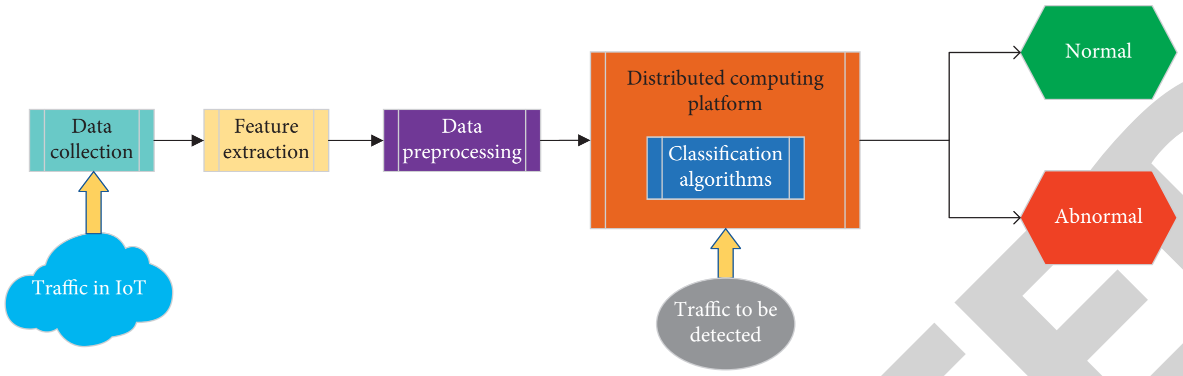


Figure 1: Distributed intrusion detection procedure using intelligent classification algorithms.

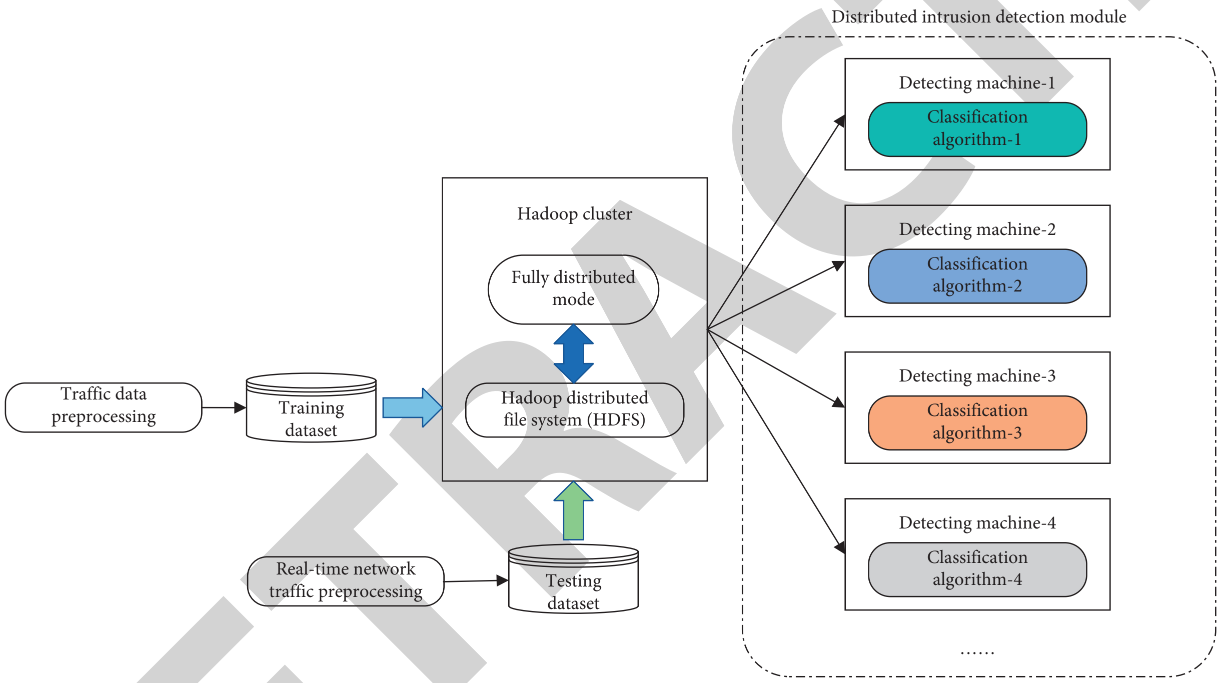

Distributed intrusion detection module
Detecting machine-1
Classification algorithm-1
Detecting machine-2
Classification algorithm-2
Detecting machine-3
Classification algorithm-3
Detecting machine-4
Classification algorithm-4
......
Hadoop cluster
Fully distributed mode
Hadoop distributed file system (HDFS)
Traffic data preprocessing
Training dataset
Real-time network traffic preprocessing
Testing dataset

Figure 2: The overall framework of the proposed model.

RETRACTED

goal is to ensure the confidentiality, integrity, and availability of system and network resources.

In this section, first we will introduce the distributed intrusion detection procedure using intelligent classification algorithms. Second, the whole framework of the proposed model in our study will be given, as shown in Figures 1 and 2.

(1) The data collection module obtains the network traffic from different types of IoT devices, and then, the feature extraction is finished in the collected data. Next, the LabelEncoder coding and the one-hot coding are used as our data preprocessing methods. Again, the processed data are put into a distributed computing platform like Spark. Finally, the traffic to be detected would be estimated as normal or abnormal by the classification algorithms. The distributed intrusion detection procedure using intelligent classification algorithms is shown.

(2) The overall framework of our model is shown. Here, we first preprocess the collected network traffic data, and the preprocessed data are used as the training dataset. The training dataset is deployed in multiple nodes of a Hadoop cluster. Hadoop has three running modes, i.e., local mode, pseudo distributed mode, and fully distributed mode. The first two modes use stand-alone simulation; in this study, we adopt the fully distributed mode. The fully distributed mode interacts with the Hadoop distributed file system (HDFS) to perform the distributed computing and data processing. Next, the real-time network traffic needs to be fleetly preprocessed by the predefined methods, and they are used as the testing dataset. The data in the testing dataset are input to every distributed node in the Hadoop cluster. Finally, the data in the testing dataset are sent to the distributed intrusion

detection module. The four types of classification algorithms are deployed on every machine of four detecting machines to perform distributed detection, respectively.

5. Experiment and Analysis

5.1. Experimental Environment and Dataset. The two experimental environments are chosen in this study, namely, the stand-alone environment and the cluster environment. In the former, Python v3.7 and Spark v3.1.1 are used. The Jupyter Notebook, which is an interactive computing environment, is adopted to facilitate data interaction and result visualization. For the cluster environment, VMware is used to build three virtual machines with a memory of 2G and a processor of 2 cores. The distributed environments consist of four components, which are Hadoop3.1.3, Spark3.1.1, Scala2.13.5, and Java1.8.

Our experiment is set up in a fully distributed mode. The deployment schemes of the three virtual machines and their HDFS in this study are shown in Tables 1 and 2. In Table 1, Hadoop 102, 103, and 104 represent three virtual machines, respectively. The master node, worker node, and worker node in Spark are deployed in Hadoop 102, 103, and 104, respectively. The HDFS has three types of nodes, i.e., NameNode, Secondary NameNode, and DataNode. Their deployment schemes are shown in Table 2. Based on the above experimental environment, we simulate a distributed intrusion detection scenario in IoT.

We use the NSL-KDD [25] datasets to demonstrate the superiority of distributed intrusion detection methods using the intelligent classification algorithms. The NSL-KDD datasets are to divide all kinds of attacks into four categories, and they are described as DoS, probe, remote to local (R2L), and user to root (U2R). As for each record, it includes the information that has been separated into 41 features plus 1 class label [26] as the same as the KDD CUP 99 dataset. The training dataset and the testing dataset in the NSL-KDD have a reasonable distribution ratio, in which the former contains 125973 records, and the latter contains 22544 records.

5.2. Evaluation Indicators. In order to evaluate the performance of the IoT-oriented distributed intrusion detection methods using intelligent classification algorithms in Spark, in this study, recall (i.e., true-positive rate (TPR)), precision, false-negative rate (FNR), false-positive rate (FPR), $F1$-score, and receiver operating characteristic (ROC) curve are selected. Recall is the percentage that is ultimately predicted to be positive in the total positive samples. Precision is the percentage that is ultimately predicted to be positive in the parts identified as positive samples. FNR is the rate of false alarm. FPR is the rate of missing alarm. The ideal situation is that the recall and precision are as high as possible, and the FPR and TPR are as low as possible. $F1$-score is related to recall and precision. The corresponding calculation formulae are shown as follows:

Table 1: Deployment scheme of the virtual machines.

	Hadoop 102	Hadoop 103	Hadoop 104
Spark	Master	Worker	Worker

Table 2: Deployment scheme of HDFS.

	Hadoop 102	Hadoop 103	Hadoop 104
HDFS	NameNode DataNode	DataNode	Secondary NameNode DataNode

$$\begin{aligned}
\text{Recall} &= \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},\\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}},\\
\text{FNR} &= \frac{\text{FN}}{\text{TP} + \text{FN}},\\
\text{FPR} &= \frac{\text{FP}}{\text{FP} + \text{TN}},\\
\text{F1} - \text{Score} &= \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}},
\end{aligned} \tag{9}$$

where true positive (TP) is the number of positive samples correctly identified. True negative (TN) is the number of negative samples correctly identified. False positive (FP) is the number of positive samples identified by mistake. False negative (FN) is misidentified as the number of negative samples.

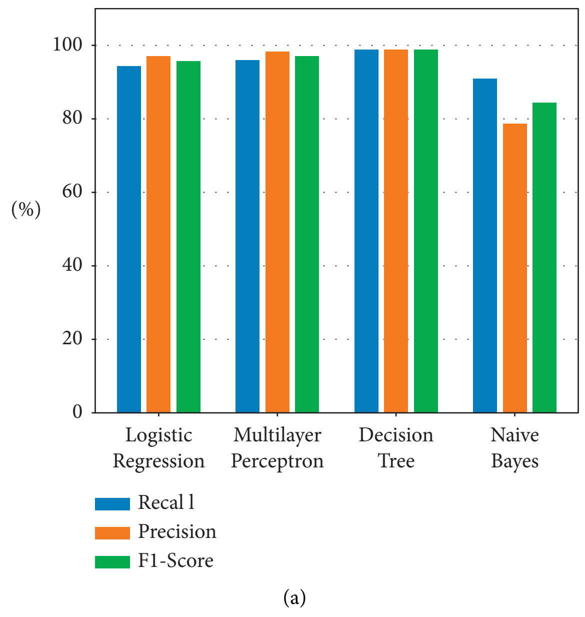
In addition, the abscissa is FPR, and the ordinate is TPR in the ROC space [27]. Every point on the ROC curve reflects the sensitivity to the same signal stimulus. The curve is obtained by setting different thresholds, and there is a trade-off between TPR and FPR.

5.3. Analysis of Experimental Results. In our experiment, the features of three character types, i.e., protocol_type, service, and flag, are used by LabelEncoder and one-hot to obtain the vectors of 117 dimensions. The vectors of 117 dimensions are reduced the dimensionality by the PCA algorithm, and the vectors of 40 dimensions are finally chosen. In addition, TCP traffic features within 2 seconds are divided into discrete features by binarizer, and the threshold value is 0.5. The remaining continuous features are assigned the numbers of divided intervals according to the value range of the eigenvalue.

Here, we conduct a binary classification experiment and a quintuple classification experiment based on the classification algorithms, respectively. The former is performed to distinguish normal against abnormal, and all other attack types are abnormal. The latter is based on normal traffic and four different attack types.

In binary classification, if a certain category accounts for the majority of proportion, then the recall and precision are close to 1; however, the classifiers have no practical significance. Because the classifiers do not screen out the few categories, the use of recall and precision cannot measure the

RETRACTED



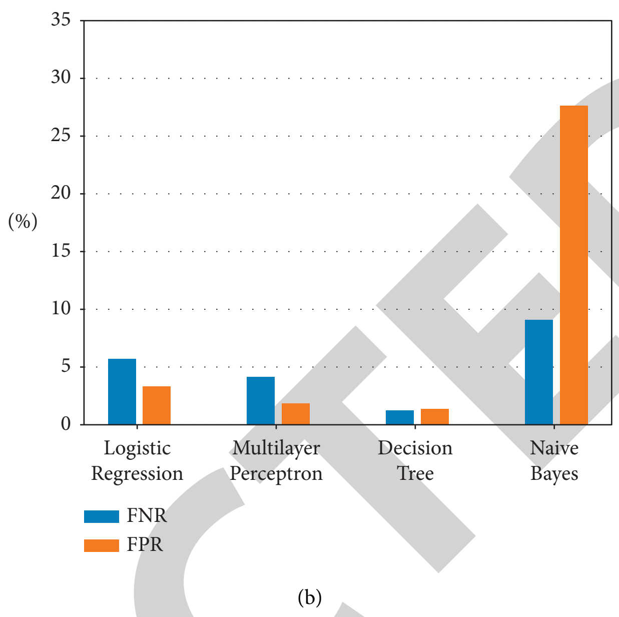


Figure 3: Comparisons of binary classification performance. (a) Comparisons in recall, precision, and F1-score. (b) Comparisons in FNR and FPR.

RETRACTED

Table 3: Comparative data of binary classification performance.

	Recall (%)	Precision (%)	F1-score (%)	FNR (%)	FPR (%)
Logistic regression	94.28	96.96	95.61	5.71	3.32
Multilayer perceptron	95.87	98.32	97.08	4.13	1.84
Decision tree	98.75	98.79	98.77	1.25	1.35
Naive Bayes	90.91	78.69	84.36	9.09	27.63

advantages and disadvantages of the classification algorithms. Therefore, we bring in FNR, FPR, and ROC curve.

In Figure 3 and Table 3, the comparisons of binary classification performance are shown. The performance of the logistic regression, multilayer perceptron, and decision tree is better than the naive Bayes in the binary classification. Because the naive Bayes takes for condition independence between every two features, and some features in the NSL-KDD dataset have stronger correlations, the naive Bayes has poor performance compared with other classification algorithms. In Figure 4, the ROC curve of four classification models also shows that the naive Bayes has the worst classification effect and the decision tree has the best classification performance.

In Figures 5 and 6, and Tables 4 and 5, the comparisons of quintuple classification performance are shown. We find that the classification performance of naive Bayes is still low. Due to the proportion of R2L and U2R in abnormal traffic types being smaller than that of other abnormal traffic types in the training dataset and testing dataset, the detection effect of R2L and U2R in abnormal traffic types is lower than that of other abnormal traffic types. Therefore, we usually deploy the naive Bayes on the detection machine that owns fewer data samples.

According to the optimal parameters of the classification algorithms running in the local environment, the

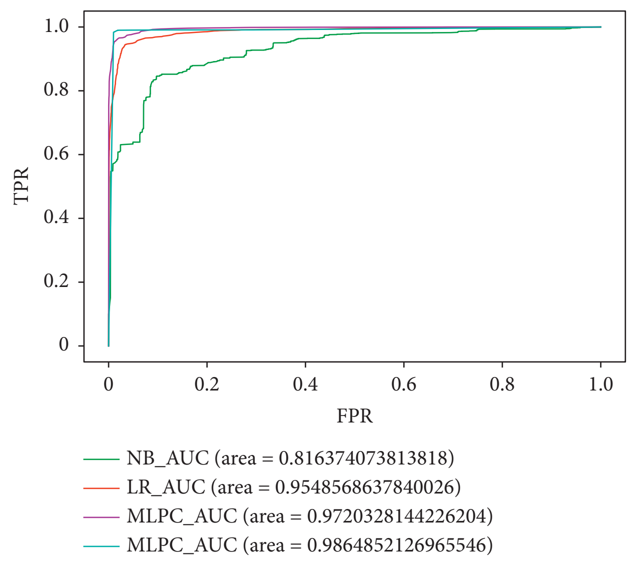


Figure 4: ROC curve of four classification models.

corresponding classification algorithm program is written in Scala and ran in the Spark cluster to compare the performance of classification algorithms in the stand-alone and the cluster environment. In Figures 7 and 8, the performance comparisons of binary classification and quintuple

RETRACTED

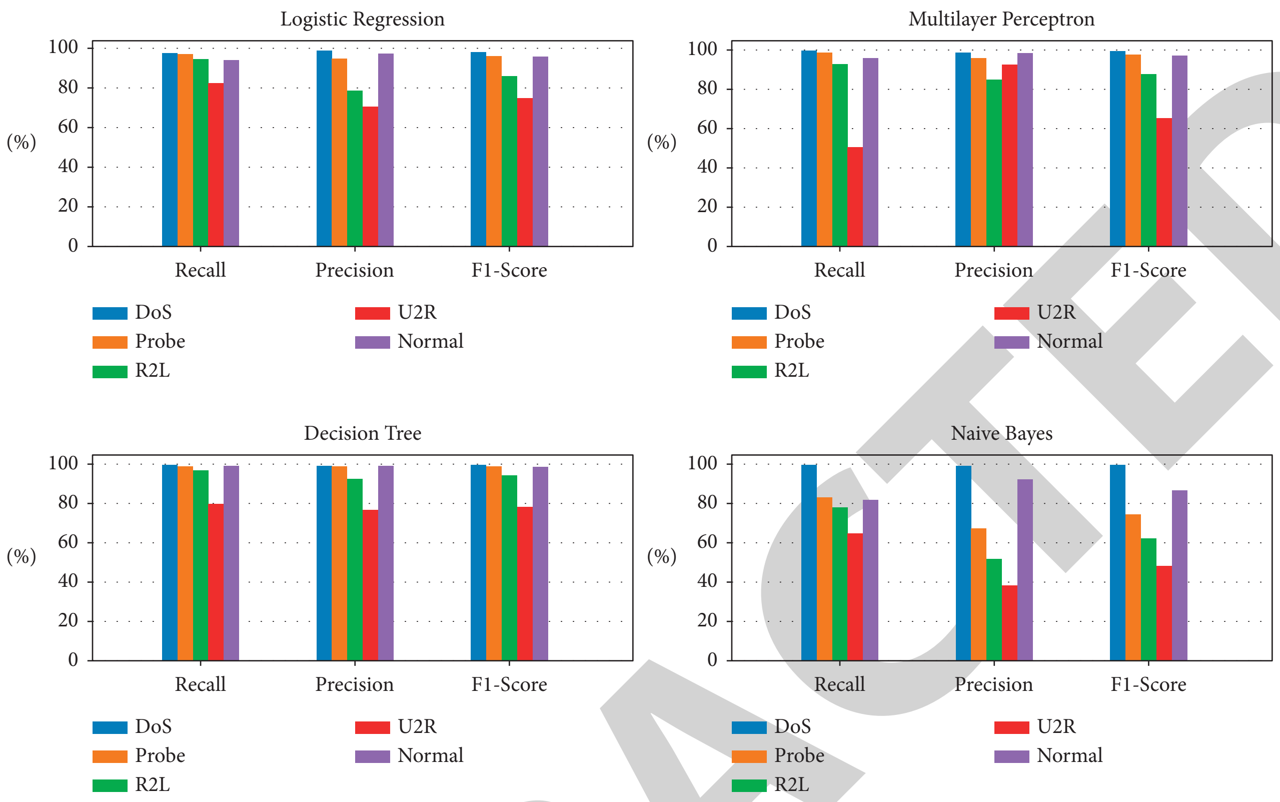


Figure 5: Comparisons of quintuple classification performance in recall, precision, and *F*1-score.

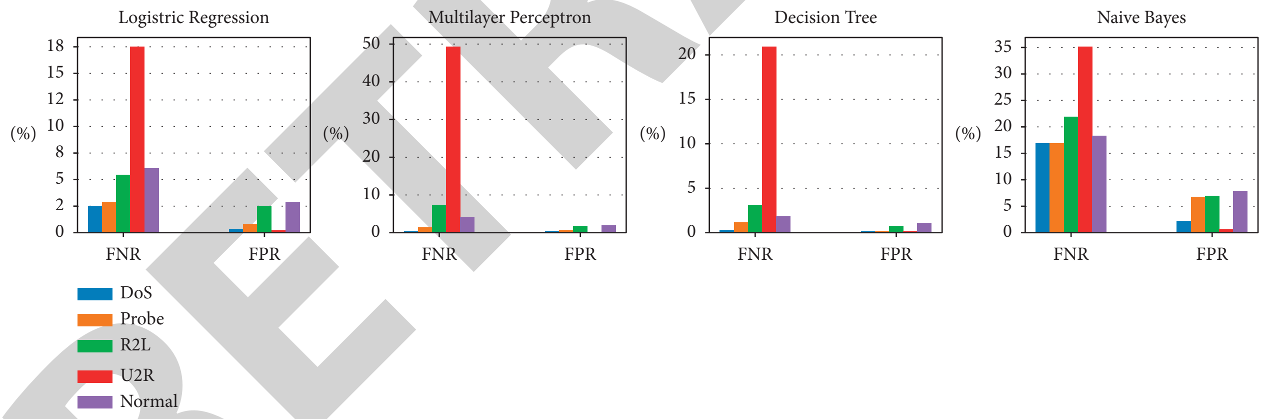


Figure 6: Comparisons of quintuple classification performance in FNR and FPR.

Table 4: Comparative data of quintuple classification performance in recall, precision, and *F*1-score.

Evaluation indicator	Classification method	DoS	Probe	R2L	U2R	Normal
Recall (%)	Logistic regression	97.49	97.12	94.56	82.47	93.92
	Multilayer perceptron	99.73	98.61	92.85	50.51	95.86
	Decision tree	99.66	98.85	96.94	79.73	99.03
	Naive Bayes	83.07	83.11	78.06	64.86	81.73
Precision (%)	Logistic regression	98.86	94.85	78.69	70.54	97.43
	Multilayer perceptron	98.79	96	84.93	92.45	98.3
	Decision tree	99.15	98.8	92.45	76.62	99.03
	Naive Bayes	91.8	67.34	51.7	38.4	92.2
*F*1-score (%)	Logistic regression	98.17	96.05	85.9	74.77	95.65
	Multilayer perceptron	99.4	97.59	87.78	65.33	97.09
	Decision tree	99.63	98.83	94.4	78.15	98.58
	Naive Bayes	87.21	74.4	62.2	48.24	86.64

Table 5: Comparative data of quintuple classification performance in FNR and FPR.

Evaluation indicator	Classification method	DoS	Probe	R2L	U2R	Normal
FNR (%)	Logistic regression	2.51	2.88	5.44	17.5	6.07
	Multilayer perceptron	0.27	1.4	7.43	49.28	4.14
	Decision tree	0.33	1.14	3.06	20.91	1.85
	Naive Bayes	16.92	16.89	21.94	35.13	18.28
FPR (%)	Logistic regression	0.35	0.84	2.48	0.219	2.87
	Multilayer perceptron	0.41	0.68	1.75	0.02	1.86
	Decision tree	0.12	0.2	0.76	0.14	1.08
	Naive Bayes	2.26	6.8	6.97	0.61	7.76

RETRACTED

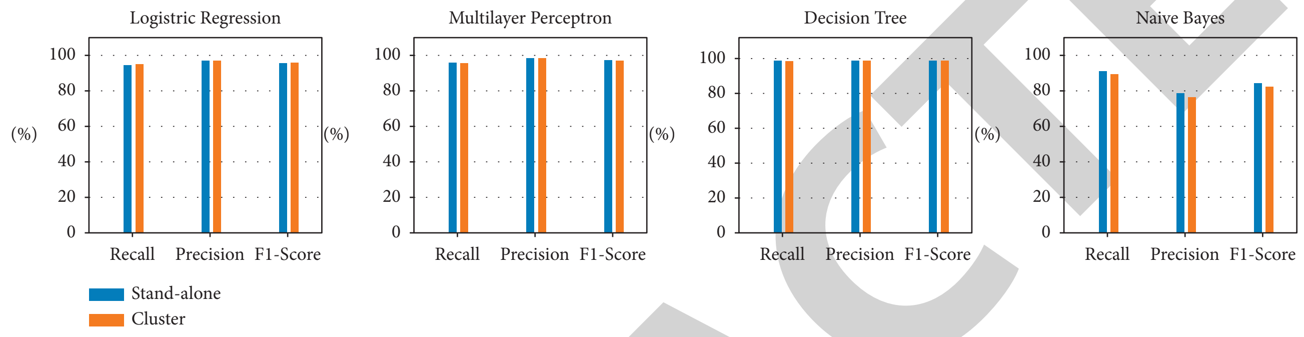


Figure 7: Comparisons of binary classification performance in stand-alone and cluster environments.

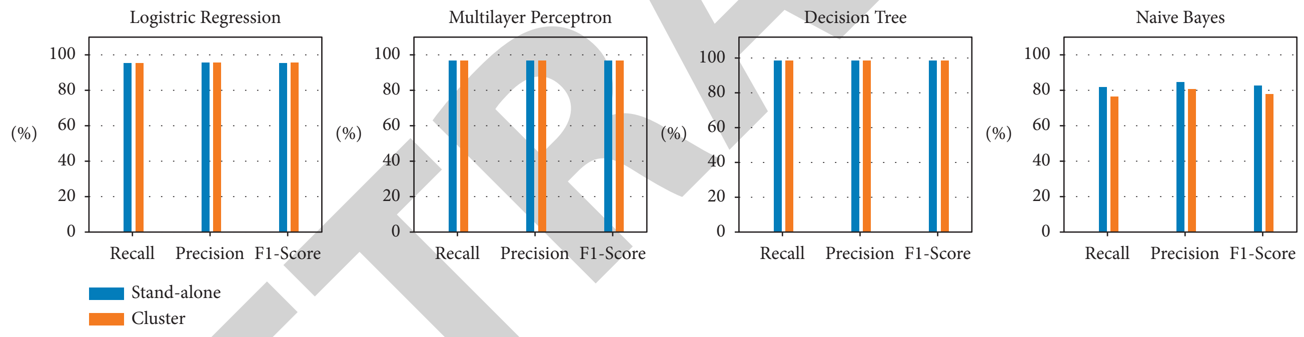


Figure 8: Comparisons of quintuple classification performance in stand-alone and cluster environments.

classification in stand-alone and cluster environments are given, respectively. The classification effect in a cluster environment is almost the same as that in the stand-alone environment, which proves the feasibility of using intelligent classification algorithms for network traffic intrusion detection in a distributed environment.

6. Conclusions

In this study, in order to solve the problem to detect massive and complex network attack traffic in IoT, an IoT-oriented distributed intrusion detection framework and methods using the classification algorithms provided by Apache Spark are proposed and built. Some comparison experiments by the binary classification and quintuple classification in six evaluation indicators (i.e., recall, precision, $F1$-score, FNR, FPR, and ROC curve) indicate that the naive Bayes has a worse classification performance than that of other classification algorithms, and the classification effect in a cluster is almost the same as that in a stand-alone environment. We usually deploy the naive Bayes on the detection machine that owns fewer data samples. It proves the feasibility of using intelligent classification algorithms for network traffic intrusion detection in the distributed environment.

Data Availability

The data used to support the results of this study have been given as links in this article. The datasets can be accessed on the online websites [21].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Key Research and Development Program of Shandong Province (soft science project) (2020RKB01364).

References

[1] N. Lionel, "Tidjon, marc frappier, and amel mammar, "intrusion detection systems: a cross-domain overview," *IEEE Communications surveys & tutorials*, vol. 21, no. 4, pp. 3639–3681, 2019.

[2] W. Lian, G. Nie, B. Jia, D. Shi, Q. Fan, and Y. Liang, "An intrusion detection method based on decision tree-recursive feature elimination in ensemble learning," *Mathematical Problems in Engineering*, vol. 2020, Article ID 2835023, 15 pages, 2020.

[3] B. Jia, X. Zhang, J. Liu, Y. Zhang, K. Huang, and Y. Liang, "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4049–4058, 2022.

[4] A. Mudgerikar, P. Sharma, and E. Bertino, "Edge-based intrusion detection for IoT devices," *ACM Transactions on Management Information Systems*, vol. 11, no. 4, pp. 1–21, 2020.

[5] Y. Lian, Y. Dai, and Y. Hu, "A study of a distributed intrusion detection model," *Journal of Computer Research and Development*, vol. 40, no. 8, pp. 1195–1202, 2003.

[6] J. Lee, B. Kim, and J. M. Chung, "Time estimation and resource minimization scheme for Apache spark and hadoop big data systems with failures," *IEEE Access*, vol. 7, pp. 9658–9666, 2019.

[7] F. Zhao, X. Yang, K. Zhang, and X. Niu, "Representativeness-based Instance Selection for Intrusion Detection," *Security and Communication Networks*, vol. 2021, Article ID 6638134, 13 pages, 2021.

[8] A. Alsirhani, S. Sampalli, and B. Peter, "DDoS detection system: using a set of classification algorithms controlled by fuzzy logic system in Apache spark," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 936–949, 2019.

[9] K. Kim, M. E. Aminanto, and H. Chandra Tanuwidjaja, *Network Intrusion Detection Using Deep Learning: A Feature Learning Approach*, Springer, Salmon, NY USA, 2018.

[10] B. Jia, Y. Ma, X. Huang, Z. Lin, and Y. Sun, "A novel real-time ddos attack detection mechanism based on MDRA algorithm in big data," *Mathematical Problems in Engineering*, vol. 2016, Article ID 1467051, 10 pages, 2016.

[11] J. Hussain, S. Lalmuanawma, and L. Chhakchhuak, "A two-stage hybrid classification technique for network intrusion detection system," *International Journal of Computational Intelligence Systems*, vol. 9, no. 5, pp. 863–875, 2016.

[12] J. Olamantanmi Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, "Network intrusion detection system using supervised learning paradigm," *Scientific African*, vol. 9, pp. 1–11, Article ID e00497, 2020.

[13] R. Nagarathna and S. Mercy Shalinie, "Semisupervised-learning-based security to detect and mitigate intrusions in IoT network," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 11041–11052, 2020.

[14] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: a comprehensive survey of unsupervised methods," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3369–3388, 2018.

[15] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, and M. Colajanni, "Deep reinforcement adversarial learning against botnet evasion attacks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 1975–1987, 2020.

[16] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: a survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, pp. 1–21, 2020.

[17] B. Jia and Y. Liang, "Anti-D chain: a lightweight DDoS attack detection scheme based on heterogeneous ensemble learning in blockchain," *China Communications*, vol. 17, no. 9, pp. 11–24, 2020.

[18] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, "Online adaboost-based parameterized methods for dynamic distributed network intrusion detection," *IEEE Transactions on Cybernetics*, vol. 44, no. 1, pp. 66–82, 2013.

[19] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Computing Surveys*, vol. 51, no. 3, pp. 1–36, 2018.

[20] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.

[21] NSL-KDD Dataset, "NSL-KDD Dataset," 2009, https://www.unb.ca/cic/datasets/nsl.html.

[22] KDD CUP 99 Dataset, "KDD CUP 99 Dataset," 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[23] N. Ben Amor, S.f Benferhat, and Z. Elouedi, "Naive Bayes vs decision trees in intrusion detection systems," in *Proceedings of the conference 2004 ACM Symposium on Applied Computing*, pp. 420–424, Nicosia, Cyprus, March 2004.

[24] J. H. Lee, J. H. Lee, S. G. Sohn, J. H. Ryu, and T. M. Chung, "Effective value of decision tree with KDD 99 intrusion detection datasets for intrusion detection system," in *Proceedings of the 10th International Conference on Advanced Communication Technology*, vol. 2, pp. 1170–1175, Phoenix Park, Korea, February 2008.

[25] T. D. Diwan, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 11, pp. 2954–2968, 2021.

[26] C. Bae, W.-C. Yeh, M. A. M. Shukran, Y. Y. chung, and T.-J. Hsieh, "A novel anomaly-network intrusion detection system using ABC algorithms," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 12, pp. 8231–8248, 2012.

[27] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

RETRACTED