Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8827185, 12 pages
https://doi.org/10.1155/2020/8827185

Research Article

Valid Probabilistic Anomaly Detection Models for System Logs

Chunbo Liu,[1] **Lanlan Pan**,[2] **Zhaojun Gu**,[1] **Jialiang Wang**,[2] **Yitong Ren**,[2] **and Zhi Wang**[3]

[1]*Information Security Evaluation Center, Civil Aviation University of China, Tianjin 300300, China*
[2]*College of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China*
[3]*College of Cyber Science, Nankai University, Tianjin 300350, China*

Correspondence should be addressed to Zhi Wang; zwang@nankai.edu.cn

Received 28 June 2020; Revised 26 September 2020; Accepted 31 October 2020; Published 16 November 2020

Academic Editor: Weizhi Meng

System logs can record the system status and important events during system operation in detail. Detecting anomalies in the system logs is a common method for modern large-scale distributed systems. Yet threshold-based classification models used for anomaly detection output only two values: normal or abnormal, which lacks probability of estimating whether the prediction results are correct. In this paper, a statistical learning algorithm Venn-Abers predictor is adopted to evaluate the confidence of prediction results in the field of system log anomaly detection. It is able to calculate the probability distribution of labels for a set of samples and provide a quality assessment of predictive labels to some extent. Two Venn-Abers predictors LR-VA and SVM-VA have been implemented based on Logistic Regression and Support Vector Machine, respectively. Then, the differences among different algorithms are considered so as to build a multimodel fusion algorithm by Stacking. And then a Venn-Abers predictor based on the Stacking algorithm called Stacking-VA is implemented. The performances of four types of algorithms (unimodel, Venn-Abers predictor based on unimodel, multimodel, and Venn-Abers predictor based on multimodel) are compared in terms of validity and accuracy. Experiments are carried out on a log dataset of the Hadoop Distributed File System (HDFS). For the comparative experiments on unimodels, the results show that the validities of LR-VA and SVM-VA are better than those of the two corresponding underlying models. Compared with the underlying model, the accuracy of the SVM-VA predictor is better than that of LR-VA predictor, and more significantly, the recall rate increases from 81% to 94%. In the case of experiments on multiple models, the algorithm based on Stacking multimodel fusion is significantly superior to the underlying classifier. The average accuracy of Stacking-VA is larger than 0.95, which is more stable than the prediction results of LR-VA and SVM-VA. Experimental results show that the Venn-Abers predictor is a flexible tool that can make accurate and valid probability predictions in the field of system log anomaly detection.

1. Introduction

With the rise of distributed and cloud computing technology, the scale of the system continues to expand. The explosive growth of large-scale Internet services supported by large-scale server deployment has brought great challenges to operation and maintenance personnel to maintain the normal operation of the system. The generated operation log can locate anomalies, but due to the exponential growth of the log volume generated by the distributed system and different systems will adopt different fault tolerance mechanisms, manual retrieval is time-consuming and labor-intensive. With the development of machine learning, log anomaly detection methods based on machine learning have become a research focus.

Machine learning techniques have been used to detect anomalies. For example, statistical anomaly detection models based on data distribution [1, 2] propose a hypothesis that the dataset obeys a certain distribution or probability model and realizes anomaly detection by judging whether a certain data point conforms to the distribution model, but this method is only suitable for point anomaly detection. With the increase of data dimensions and data volume, the efficiency of this method's anomaly detection would decrease

accordingly. The nearest nearby method based on distance [3, 4], its basic idea is that normal data is similar to its nearby data, while abnormal data is different from nearby data. This method does not need to master the data distribution, nor does it require a labeled training dataset. It is theoretically suitable for high-dimensional data anomaly detection, but due to its high computational complexity, it is difficult to determine the parameters, which limits its application. Clustering based anomaly detection methods [5, 6] assumes that normal data is located in a dense area, and anomalies are far away from this area. Because the results of anomaly detection depend on the effect of clustering, the complexity and time complexity of the calculation method increase with the increase in dimensionality. An anomaly detection method based on neural network [7, 8] is generally realized by comparing the predicted value of the model with the actual measured value. The anomaly detection method based on neural network has strong ability to detect abnormal data, but its shortcoming is that the neural network model parameter setting has a great influence on the model result and is difficult to determine. There is no unified standard for the selection and optimization of the network structure, and it will also increase the time complexity and the computational complexity of the model when processing large amounts of data or high dimensions.

This paper is aimed at detecting anomalies in system logs. System anomaly detection is a necessary condition for the stable operation of a computer system. The system logs record information about hardware, software, and system problems in the system. At the same time, it can also monitor events that occur in the system. It records system states and significant events in detail, which can help administrators troubleshoot or understand what is happening in the system at a detailed level. Zhu et al. [9] summarizes the structure of common system logs and the dependencies between events (note: https://github.com/logpai/loghub provides a complete system log dataset). Therefore, only when the log can be correctly parsed can the rich information in the log be effectively used for system health diagnosis and avoid serious problems such as system downtime. Xu et al. [10] proposes to automatically detect system runtime problems by parsing the console system log, and use a PCA-based feature extraction algorithm to accurately describe the complex state information of a large-scale system. Lou et al. [11] categorizes the console system logs after being structured, and judges abnormal events by counting the distribution of various logs over a period of time. He et al. [12] converts the console system log into an event template, slices the original log into a set of log sequences and forms a feature vector through different grouping techniques, and uses three supervised and three unsupervised methods for log anomaly detection. The LSTM-based deep neural network model Deeplog proposed by Du et al. [13] is an integrated framework for anomaly detection that combines LSTM and online learning. It is used to solve the impact of unknown abnormal events in the future that are difficult to predict on system operation and maintenance diagnosis. Li et al. [14] uses the longest common subsequence method to compare the similarity between new time series data and historical data. Later, Xia et al. [15] propose LogGAN (Log Generative Adversarial Networks), an anomaly detection model based on generative adversarial networks, to generate more abnormal event samples to solve the problem of the imbalance between the numbers of abnormal events and normal events. Recently, Xia et al. [16] further improve the generative confrontation network based on the attention mechanism and train the generator based on the recurrent neural network to converge through the machine of confrontation learning, which is further improved than the anomaly detection accuracy of LogGAN.

These algorithms can only give one prediction (for classification, it is the prediction label; for regression, it is the predicted value), and no reliability evaluation of the prediction result is provided, that is, the evaluation of the credibility of the prediction result and the evaluation guarantee of validity [17, 18]. At present, the most popular probability prediction algorithms are conformal predictor and Venn-Abers predictor. The conformal predictor gives p value as an estimate of prediction reliability under confidence [19], but that is not a direct probability. The paper is aimed at introducing an algorithm that converts the results of the conformal predictor into probabilities and giving estimates of the probabilities of the predicted results, which makes the results more intuitive.

The validity of probabilistic prediction is very important for probabilistic prediction methods. Validity refers to the estimated probability for predicted label is unbiased, which means the estimated probability is equal or close to the observed frequency that predictions are correct [20]. The statistical learning algorithm Venn-Abers predictor used in this paper has a validity guarantee [21]; this method can evaluate the reliability of log anomaly detection results, which means it can make effective probability predictions about the correctness of prediction results. It is a flexible machine learning framework that uses probability to classify data. Any machine learning algorithm can be used as its underlying algorithm. The only assumption required for the Venn-Abers predictor is that the example distribution is the exchangeability assumption, which can be easily satisfied by the log data, and it does not need specific distribution of the log data once the exchangeability assumption is satisfied; thus, the validity of predicted probabilities is guaranteed. At present, Venn-Abers predictors have obtained reliable and effective probabilistic prediction results in many fields [22–24]. The Venn-Abers predictor is introduced in system log anomaly detection so that the system log abnormality can be detected more reliably and effectively.

This paper evaluates proposed method using a real system HDFS log dataset. The Venn-Abers predictor has been proved to be perfectly calibrated [25]. However, the cost is that Venn-Abers predictors are multiprobabilistic predictors, in the sense of issuing a set of probabilistic predictions instead of a single probabilistic prediction; intuitively, the diameter of this set reflects the uncertainty of our prediction. Two Venn predictors are based on two underlying machine learning methods (Logistic Regression and Support Vector Machine) for the system logs, respectively [26]. The multiprobabilistic prediction outputs are replaced by Venn-Abers predictors with a probability prediction value. This approach makes it facilitate a comparison of various

algorithms using loss functions. The validity of the prediction results is compared by the loss function of the underlying machine learning methods and Venn-Abers predictors. Two Venn-Abers predictors base on two underlying machine learning methods separately so as to detect system log anomalies. The probability prediction value of the Venn-Abers predictor is converted into the prediction results. By this way it is convenient to use the loss function to compare the effectiveness of various algorithms. The experimental results turn out that the method of using Venn-Abers predictors to evaluate the correctness of the system anomaly detection is valid and accurate [27].

Moreover, considering the differences in the data processing principles of different algorithms, full play is given to the advantages of each model, and integrated learning is used in order to achieve a stronger generalization ability. Common integrated learning methods such as AdaBoost and Bagging use autonomous sampling (booststrap) [28] to construct different training sets, and Random Forest (RF) uses different random feature spaces. The commonality of these methods is based on the integration of the same algorithm, so the multiple base classifiers produced are different, and the combination of classifiers is generally voting. Stacking is different, and it is based on multiple different classifiers generated by different algorithms and learns again on the prediction of multiple classifiers to achieve a combined integration method. This paper proposes a Stacking multimodel fusion strategy based on 5 different classifiers, named SVM, KNN, DT, RF, and GBDT for combination, and develops a Venn-Abers predictor based on Stacking to achieve high-precision anomaly detection.

The remainder of this paper is organized as follows. Section 2 outlines the overall structure of the paper, including a brief description of the five steps of anomaly detection, and it also introduces the Venn-Abers framework and single model, multimodel fusion methods. Section 3 reviews the evaluation indicators of the experiment. The comparative experiments are reported, and the experimental results are analyzed in Section 4; besides, the experimental conclusions and future plans are given in Section 5.

2. Framework

Figure 1 illustrates the overall framework for log-based anomaly detection. Log anomaly detection framework mainly includes four steps: log collection, log parsing, feature extraction, and anomaly detection. In the process of log anomaly detection, this paper introduces a probability prediction statistical learning method Venn-Abers predictor, which compares with the underlying machine learning algorithm in terms of threshold; thus, it draws the probability of predicting the label, which will make the prediction result more effective.

2.1. HDFS Logs. Modern large-scale systems record system runtime information by generating logs. Each log contains unstructured data such as time stamps, log priorities, system components, and log entries themselves. Typically, a log message records a specific system event with a set of fields. Eight log lines are extracted from the HDFS logs on Amazon EC2 platform as shown in Figure 1 [10], while some fields are omitted here for ease of presentation.

2.2. Log Parsing. The goal of log parsing is to extract a set of log event templates. That is, the constant part (fixed plain text) and the variable part (such as *blk_id* in Figure 1) are distinguished from the log data content [10–12]. The log template event mainly includes a constant part and a wildcard: the constant part constitutes the fixed plain text, which remains the same for every event occurrence and can reveal the event type of the log message, while the wildcard carries the runtime information of interest, such as the values of states and parameters (e.g., the IP address: 10.251.31.5); thus, it may vary among different event occurrences, and it is replaced by a string of the form $<*>$. Each different log event template is numbered as *event_id*, and each log event template corresponds to an identifier *blk_id*.

2.3. Feature Representation. Results obtained from the previous step are used to generate an event count matrix X, which will be fed into a log anomaly detection model. In the event count matrix, each row represents a block, while each column indicates one event type. The value in cell $X_{i,j}$ records how many times event j occurs on block i. X is generated with one pass through the parsed results. Instead of directly detecting anomaly on X, TF-IDF [29] is a well-established heuristic in information retrieval, and it is often used as a feature representation of documents in information retrieval and text mining.

2.4. Anomaly Detection. Based on the previous log preprocessing results, log anomaly detection is used to find out suspicious blocks that may indicate problems. The underlying supervised learning method is the input of a given historical feature vector, and our prediction model outputs the probability of an upcoming failure. If the computed probability exceeds a predefined threshold c, it will be considered abnormality to indicate a failure. In this paper, two supervised learning algorithms are used for experiments, i.e., Logistic Regression and Support Vector Machine.

Logistic Regression (LR) is a widely used machine learning classification model. Firstly, training instances are used to establish the logistic regression model. After obtaining the model, a testing instance X is fed into the logistic function so as to compute its possibility p of anomaly; the label of X is anomalous while $p \geq 0.5$ and normal otherwise.

Support Vector Machine (SVM) is a supervised learning method for classification. The basic idea is to solve the separation hyperplane that can correctly divide the training dataset and have the largest geometric interval. Similar with LR, the training instances are event count vectors together with their labels. In anomaly detection via SVM, if a new instance is located above the hyperplane, then, it would be reported as an anomaly; otherwise, it will mark as normality.

2.5. Integrated Learning Stacking. Ensemble learning is based on statistical learning theory [30]. Stacking is an integrated learning algorithm proposed by Wolpert. Unlike bagging and boosting, which use the same classification algorithm

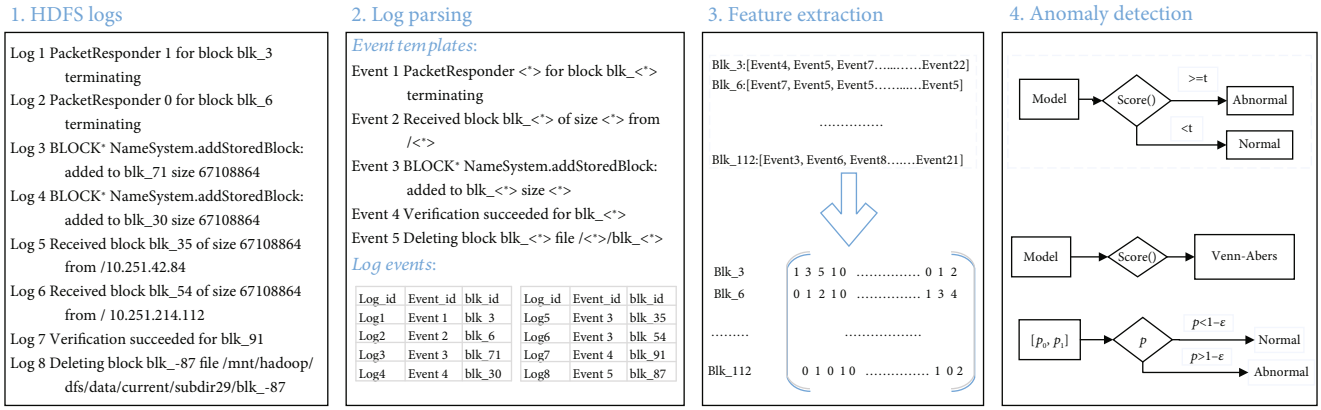

Figure 1: Framework of anomaly detection.

to continuously iteratively train a single learner, Stacking can combine the prediction results of multiple underlying classifiers so as to generate a new model and customize the combination strategy [31]. In the combined model, different types of underlying classifiers are selected in order to reduce the generalization error of the model. In the Stacking integrated learning model, it is necessary not only to analyze the individual prediction ability of each base learner but also to comprehensively compare the combined effect of each base learner, so that the Stacking integrated learning model can obtain the best prediction results.

As shown in Figure 2, a Stacking integrated learning framework firstly divides the original dataset into several subdatasets and inputs to each base learner of the layer 1 prediction model, so each base learner outputs its own prediction result. Then, the output of the first layer is used as the input of the second layer, besides the metalearner of the prediction model of the second layer is trained, and the final prediction result is output by the model located at the second layer. The Stacking learning framework generalizes the output of multiple models to improve the overall prediction accuracy.

The models selected in the first layer of the Stacking model in this paper are as follows: SVM as a classic stability classifier which has a good generalization ability in solving binary classification problems and has certain advantages in solving high-dimensional datasets; KNN has good practical application effects due to its mature theory and high training efficiency; the single-layer tree-structure DT; bagging integrated learning method representing random forest (RF) [32]; and boosting integrated learning method representing gradient boosted decision tree (GDBT). The selected model in the second layer is logistic regression with strong generalization ability, which constantly minimizes the prediction function error through the stochastic gradient descent method to improve the generalization ability of the model; besides, the loss function adopts the corresponding regularization method to ease the model degree of overfitting.

2.6. Venn-Abers Predictor. In this part, the Venn-Abers predictor is used to evaluate the likelihood of underlying machine learning algorithms predicting log anomaly detection results correctly. The Venn-Abers predictor [33] transforms the predicted results into probabilities. It applies isotonic regression to transform the output of other classifiers into probabilities.

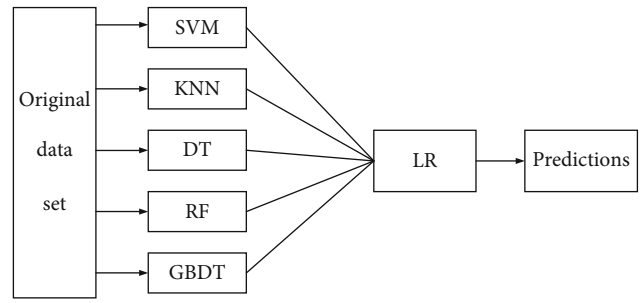

Figure 2: Framework of integrated learning Stacking.

Suppose given a standard binary classification problem: a training set of examples $(z_1, z_2, z_3, \cdots, z_{n-1})$. Each z_i consists of a pair of object x_i and label y_i. The possible labels are binary, that is, $y \in Y = \{0, 1\}$. Besides, given a new object x_n , the goal is to predict the label y_n for the new object x_n and give the estimation of the likelihood that the prediction is correct.

A scoring algorithm trains a classifier on the training set and uses the classifier to output a prediction score $s(x_n)$ for the new object x_n; besides, it predicts the label of x_n to be "1" only while $s(x_n) \geq c$ (c is a fixed threshold). So $s(\cdot)$ is hereby called the scoring function. Many machine learning algorithms for classification are scoring algorithms. In the paper, the decision function in Sections 2.4 and 2.5 is a scoring function. It is "increasing," which means a function $f(\cdot)$ is increasing if its domain is an ordered set and $t_1 \leq t_2 \rightarrow f(t_1) \leq f(t_2)$. For the "isotonic regression," it is a monotonically increasing function on the set $\{(s(x_1), y_1), \cdots, (s(x_{n-1}), y_{n-1})\}$ that maximizes the likelihood

$$\coprod_{i=1}^{n} p_i, p_i = \begin{cases} f(s(x_i)), & \text{if } y_i = 1, \\ 1 - f(s(x_i)), & \text{if } y_i = 0. \end{cases} \quad (1)$$

Such function $f()$ is indeed unique and it can be easily found using the "pair-adjacent violators algorithm" (PAVA), described in detail in the summary of [34]. The Venn-Abers predictor corresponding to the given scoring classifier is the multiprobabilistic predictor that is defined as follows. Try the two different labels 0 and 1, for the test object x_n, where s_0 is the scoring function for $(z_1, z_2, z_3, \cdots, z_{n-1}, (x_n, 0))$, s_1 refers to the scoring function for $(z_1, z_2, z_3, \cdots, z_{n-1}, (x_n, 1))$, f_0 means the isotonic calibrator for

$$((s_0(x_1), y_1), \cdots, (s_0(x_{n-1}), y_{n-1}), (s_0(x_n), 0)), \tag{2}$$

and f_1 describes the isotonic calibrator for

$$((s_1(x_1), y_1), \cdots, (s_1(x_{n-1})y_{n-1}), (s_1(x_n), 1)). \tag{3}$$

The multiprobabilistic prediction output by the Venn-Abers predictor is (p_0, p_1), where $p_0 = f_0(s_0(x))$ and $p_1 = f_1(s_1(x))$. The Venn-Abers predictor is described as Algorithm 1.

3. Evaluation Methods

Venn-Abers predictors are compared with known probabilistic predictors using standard loss functions. Since Venn-Abers predictors output pairs of probabilities rather than point probabilities, it is necessary to fit them (somewhat artificially) in the standard framework generating one probability p from the pair: p_0 and p_1.

3.1. The Validity of Probabilistic Predictions. Probabilistic prediction can provide reliability estimate on the prediction. However, the estimated probability should be valid. In this paper, loss function is used to examine the validity of probabilistic predictions; besides, square loss is applied. Supposing y is the probability value for predicted label of testing example x and y is equal to 1 if the prediction is abnormal; otherwise, the value of y is 0. The square loss function is described as

$$\lambda_{sq}(p, y) = (y - p)^2. \tag{4}$$

As to the characteristics of the loss function, while the prediction is correct, the larger the predicted probability value is, the smaller the loss function is; while the prediction is wrong, the smaller the predicted probability value is, the greater the loss function is.

Firstly, supposing that loss function is λ_{sq} and given a multiprobabilistic prediction (p_0, p_1), it needs to find the corresponding minimax probabilistic prediction p [35]. If the true outcome is $y = 0$, it can replace p_0 when p is equal to $p^2 - {p_0}^2$. If $y = 1$, it can replace p_1 when p is equal to $(1-p)^2 - (1-p_1)^2$. The first regret as a function of $p \in [0, 1]$ strictly increases from a nonpositive value to 1 while p changes from 0 to 1. The second regret as a function of p strictly decreases from 1 to a nonpositive value while p changes from 0 to 1. Therefore, the minimax regret is the solution to

$$p^2 - p_0^2 = \left(1 - p^2\right) - (1 - p_1)^2, \tag{5}$$

which is

$$p = p_1 + \frac{p_0^2}{2} - \frac{p_1^2}{2}. \tag{6}$$

While calculating the loss function of Venn-Abers' prediction results, the above formula is substituted into λ_{sq} to calculate the loss function. The smaller the loss function is, the higher the effectiveness of the prediction is. Therefore, the index of the effectiveness of the probability prediction or multiprobability prediction based on the loss function is defined. Given n testing examples, for different methods, the root mean square error (d_{sq}) is calculated and compared:

$$d_{sq} = \frac{\sum_{1=1}^{N} \lambda_{sq}(p_i, y_i)}{N}. \tag{7}$$

For each method, the smaller the d_{sq} is, the better the validity of the method is.

3.2. The Accuracy of Probabilistic Predictions. In order to measure the accuracy of log anomaly detection, for the labeled dataset, this paper takes the test set with a true label of 0 as a positive result; otherwise, it is a negative result. In this experiment, precision, recall, F-measure, and accuracy are used as the evaluation of each prediction result metrics. The calculation about these four values depends on the following values:

(i) TP: the number of prediction results is positive, but the number is actually positive

(ii) FP: the prediction result is positive, but the number is actually negative

(iii) TN: the prediction result is negative, and the number is actually negative

(iv) FN: the number of predictions is negative, but the number is actually positive

The precision rate formula is shown in (8). It represents the proportion of positive data predicted by the model as correct to all positive data predicted. At the same time, the ability of the model to reduce the false alarm rate can be measured by it.

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{8}$$

The recall formula is shown in (9). It means the proportion of the data measured by the model are positive examples to the actual data of the positive examples, and it can describe the size of the coverage of the positive examples identified by the model.

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{9}$$

The accuracy formula is shown in (10). It describes the

Input: training sequence $(z_1, z_2, z_3, \cdots, z_{n-1})$
Input: test object x_n
Output: multiprobabilistic prediction (p_0, p_1)
  for $y \in \{0, 1\}$ **do**
    set s_y to the scoring function for $(z_1, z_2, z_3, \cdots, z_{n-1}, (x_n, y))$
    set f_y to the isotonic calibrator for $((s_1(x_1), y_1), \cdots, (s_1(x_{n-1}), y_{n-1}), (s_y(x_n), y)$
    $p_y = f_y(s_y(x_n))$
end for

Algorithm 1: Venn-Abers predictor.

proportion of data judged correctly by the model to the total data. It can also describe the model's ability to correctly identify log anomalies.

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{10}$$

The formula of F-measure is shown in (11). It refers to the harmonic average of precision and recall and is used to comprehensively measure the precision and recall of the model.

$$F\text{-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{11}$$

4. Results and Discussion

The experiments were conducted on the following platform: Intel(R) Core(TM) i5-4200U CPU @1.60 GHz, 8 GB RAM, and Windows operating system. The dataset comes from the Hadoop cluster deployed by Amazon on EC2 nodes [10–12]. It runs the sample Hadoop map-reduce jobs for almost 39 hours and generates HDFS log data. In particular, the HDFS logs have well-established anomaly labels, each of which indicates whether or not a request for a data block operation is an anomaly. The labels are made based on domain knowledge, which are suitable for these evaluations on anomaly detection with different log parsers. Specifically, the dataset with 11,175,629 raw log messages records 575,061 operation requests with 29 total event types. Among all the 575,061 requests, 16,838 of them are marked as anomalies, which are used as ground truth in the evaluation.

This dataset only contains logs of events such as adding, moving, deleting, and their exceptions. HDFS uses a series of file blocks as its storage unit, and each file block has its own ID [36]. During the experiment, the original data was firstly processed into event templates, with a total of 29 message types. Event templates with the same blk_ID are grouped together in order to form a vector. The dimension of each vector corresponds to a different event template, and the value of the dimension represents the number of times the event of the template occurs, so the event count matrix has a dimension of $575{,}062 \times 29$. But during the actual experiment, many vectors were found to be exactly the same. In fact, there are only 580 different vectors; that is, most file blocks go through a common execution action. Figures 3 and 4 show F_1 and F_0, respectively, corresponding to all vectors and deduplicated vectors drawn by Venn-Abers. In Figure 3 of all vectors, because there are a large number of repeated vectors, the degree of dispersion of the F value is poor, and in Figure 4 after deduplication, the distribution of the F value is obvious. So in the following experimental results, a dataset of 580 feature vectors is used.

4.1. Single Model Performance Comparison

4.2. Comparison of the Validity. The average loss function value is the average sum of squares of the differences between the true category and the probability of the predicted result in all cases. The true category must be 1 or 0 (true or false), while the prediction result probability is a value between 0 and 1. For a set of predicted values, the lower the average loss function value, the better the prediction calibration is. The data is divided into four datasets of different sizes according to the different feature vectors for testing the loss value. As shown in Table 1, the loss values obtained by the two Venn-Abers predictors SVM-VA and LR-VA, developed based on SVM and LR, have all declined to varying degrees, which demonstrate the ability of Venn-Abers predictors to improve the classification performance.

The box plots of the square loss values of LR, SVM, LR-VA, and SVM-VA in all datasets are shown in Figure 5. Box plots are mainly used to display the statistical distribution of data. The figure generation method is used to sort the upper edge, lower edge, median, and two quartiles of a group of data and connect the two quartiles to form a box. The median, the top, and bottom edges are all connected. From the median loss value, the LR-VA model drops up to 3.6% compared with the LR model, and the SVM-VA model is 6% lower than the SVM model. Looking at the overall distribution, the quartiles of LR-VA, LR, SVM-VA, and SVM are 0.085, 0.088, 0.035, and 0.06, respectively. It can be easily seen that the LR-VA and SVM-VA models are mainly distributed in lower areas and have small spans. The model with the lowest loss value is evaluated as SVM-VA; the SVM model is second; the LR-VA model is third; the worst performing is the LR model. The assessment criteria of validity of Venn-Abers predictors are all smaller than that of corresponding underlying methods, which indicated that the probabilistic prediction conducted by Venn-Abers predictors is more valid than corresponding underlying machine learning methods.

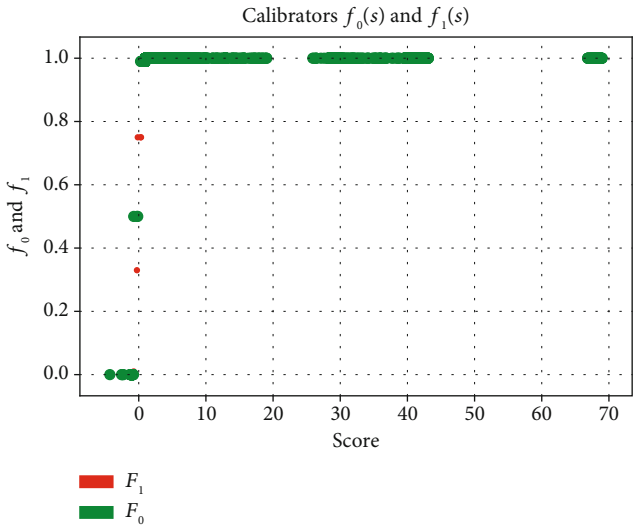

Figure 3: Results of all vector mapping.

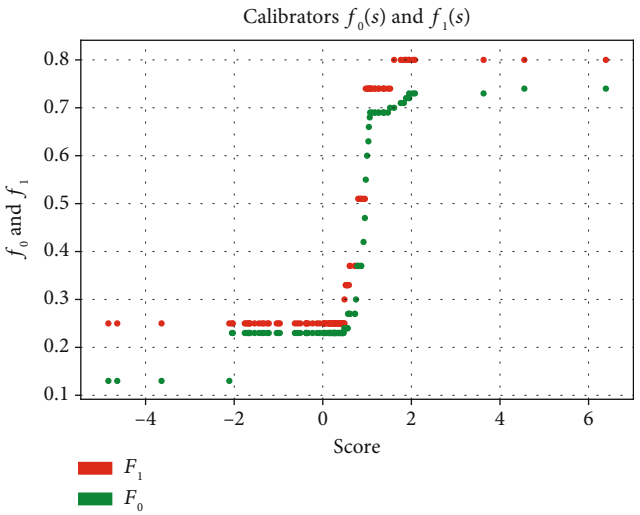

Figure 4: Vector mapping results after deduplication.

Table 1: The square loss function values obtained by two Venn-Abers predictors (LR-VA and SVM-VA) and two underlying algorithms (LR and SVM) in datasets of different sizes.

Dataset	LR		SVM	
	LR	LR-VA	SVM	SVM-VA
HDFS_116	0.300	0.178	0.100	0.080
HDFS_232	0.200	0.155	0.150	0.120
HDFS_348	0.130	0.070	0.160	0.040
HDFS_464	0.080	0.060	0.100	0.090
HDFS_580	0.112	0.094	0.163	0.115

4.3. Comparison of the Accuracy. In Section 3.1, the probability pair output by the Venn-Abers predictor is fit to one probability p, which is easy to compare with other classifiers. Because the probability p represents the probability that the predicted label is 1, the larger the p value, the higher the probability that the predicted label is 1; the smaller the p value, the lower the probability that the predicted label is 1 and the higher probability that the label is 0. In the process of system log anomaly detection in this paper, a label of 0 indicates that the prediction log is normal, and a label of 1 indicates that the prediction log is abnormal. The scatter plot of the distribution between the predicted labels (0 and 1) and the probability value p is shown in Figures 6 and 7. If the probability p results obtained by the Venn-Abers predictor are polarized in the [0, 1] interval, it means that the quality of the Venn-Abers predictor is well, as shown in Figures 6(d) and 7(d). However, during the experiment, the distribution of probability p is not limited to the vicinity of 0 and 1. Then, according to the statistical distribution of the probability p in the [0, 1] interval, the label of the test object will be predicted again. We exercise some amount of control over these metrics in Section 3.2 by setting a threshold value$c$, where $p > c$ means that the label corresponding to the test object is abnormal [37]. During the experiment, the threshold c is adjusted so as to make the prediction accuracy as close to 1 as possible.

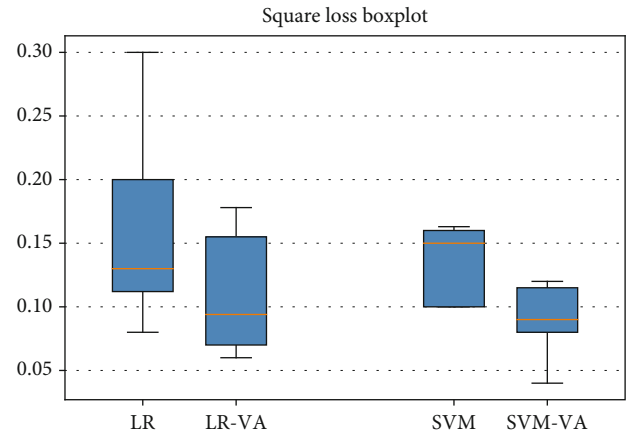

Figure 5: The loss value of different algorithms in all datasets.

In this way, the probability value can be transformed into a prediction result, and then, three groups of experimental results based on Venn-Abers are referred to as VA_0.6, VA_0.72, and VA_0.8 separately. Compared with the underlying threshold-based classification method as shown in Figure 8, the Venn-Abers predictor-based classification method has not only been successfully applied but also improved the accuracy of anomaly detection to a certain extent. Compared with the underlying threshold-based classification method in the LR algorithm as shown in Figure 8, the experimental results obtained by the Venn-Abers predictor are the best while $c = 0.72$, with accuracy and recall increased by 2% and 3% precision and F1 values increased by 1%. While $c = 0.8$ and 0.6, the experimental results obtained by the Venn-Abers predictor show more false positives and false negatives, and the results make it inferior to the underlying threshold-based classification method.

In the SVM algorithm, the same method is also used for judgment. As shown in Figure 7, while the thresholds $c =$ 0.6, 0.72, and 0.8 separately, the distribution of the probability p and the label. The experimental results of the underlying

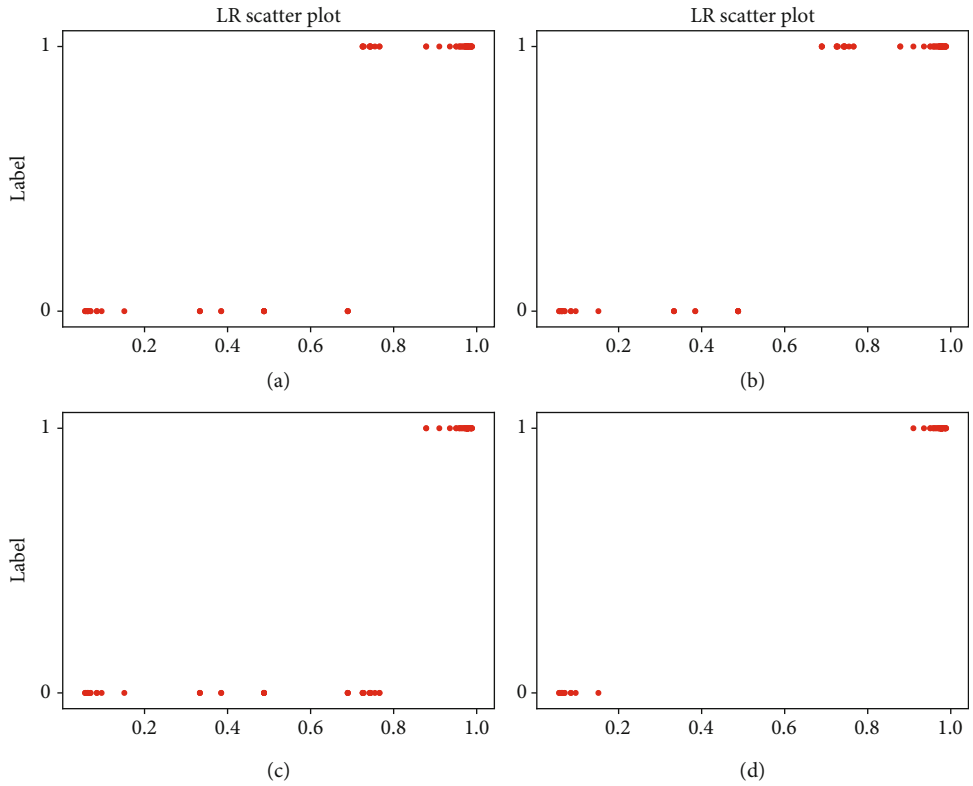

Figure 6: Scatterplot of the distribution of predicted probabilities p and test labels (0 and 1) in the LR model under different threshold values. (a) The distribution of probability p and label when threshold = 0.72. (b) The distribution of probability p and label when threshold = 0.6. (c) The distribution of probability p and label when threshold = 0.8. (d) The distribution of probability p and label in ideal situation.

threshold-based classification method SVM and Venn-Abers predictor are as shown in Figure 9. It can be seen that while $c = 0.72$, the accuracy and F1 values of the Venn-Abers predictor increase by nearly 7%, and the recall increases 13%; besides, the false alarm rate was reduced from 12% to 3%. An increase in the number of false positives at $c = 0.8$ causes the values of accuracy, precision, and F1 to decrease. While $c = 0.6$, the number of false positives decreases but the effect is not as well as that of $c = 0.72$. Therefore, it is important to choose the right threshold. Through the above experiments, it will be found that compared with the underlying threshold-based method, the Venn-Abers predictor method will dynamically change the value of c to capture more abnormal data; thereby, it can have better judgments.

The distribution of the probability p in the interval [0, 1] is analyzed. If the probability p is distributed at the poles of the interval [0, 1], the detection effect will be the best. While analyzing the distribution of probability p in the LR model, the result is that $p = 0.487805$ occurs 91 times, $p = 0.689441$ occurs 3,190 times, and $p = 0.765957$ occurs 12,665 times. Tracing the feature vector corresponding to the probability p, the result shows that the feature vector corresponding to the same probability p has a high degree of similarity, and only one dimension is different. For example, the feature vector corresponding to $p = 0.689441$ is only different from 6 occurrences of event11 and 2 occurrences of event12. The same situation occurs in the SVM model, the $p = 0.738409$ occurs 13,258 times, and the feature vector corresponding to the probability p is either repeated or highly similar. The detection of a single model cannot make good judgments on highly similar feature vectors. If model fusion can be used to take advantage of different algorithms, then the log data can be further judged so as to obtain better detection results, which will be discussed in the future.

4.4. Multimodel Performance Comparison. The integrated learning framework Stacking constructed in Section 2.5 was used to analyze the performance of multiple models. First of all, the comparison of classification performance of models is built separately from the base classifiers (SVM, KNN, DT, RF, and GBDT) and integrated models. Then, the classification performance of Stacking and Stacking-VA in the log anomaly detection data is compared.

4.5. Underlying Model Analysis. First, using the constructed feature data and all labels, 80% of the total data is divided into the training set, and the rest are used as the test set. The five models (SVM, KNN, DT, RF, and GBDT) are conducted performance tests on log data separately as shown in Table 2.

4.6. Stacking Model Fusion Analysis. The Stacking model fusion process is as follows: the five underlying classifiers of SVM, KNN, DT, RF, and GBDT are exerted in order to, respectively, perform five-fold cross-validation on the training data. For the first base classifier, the four-fold training data is used as the training set, and the other is employed to predict another one-fold validation data, and $trainPre_{1-n}$ $(n = 5)$ is obtained. At the same time, the prediction dataset

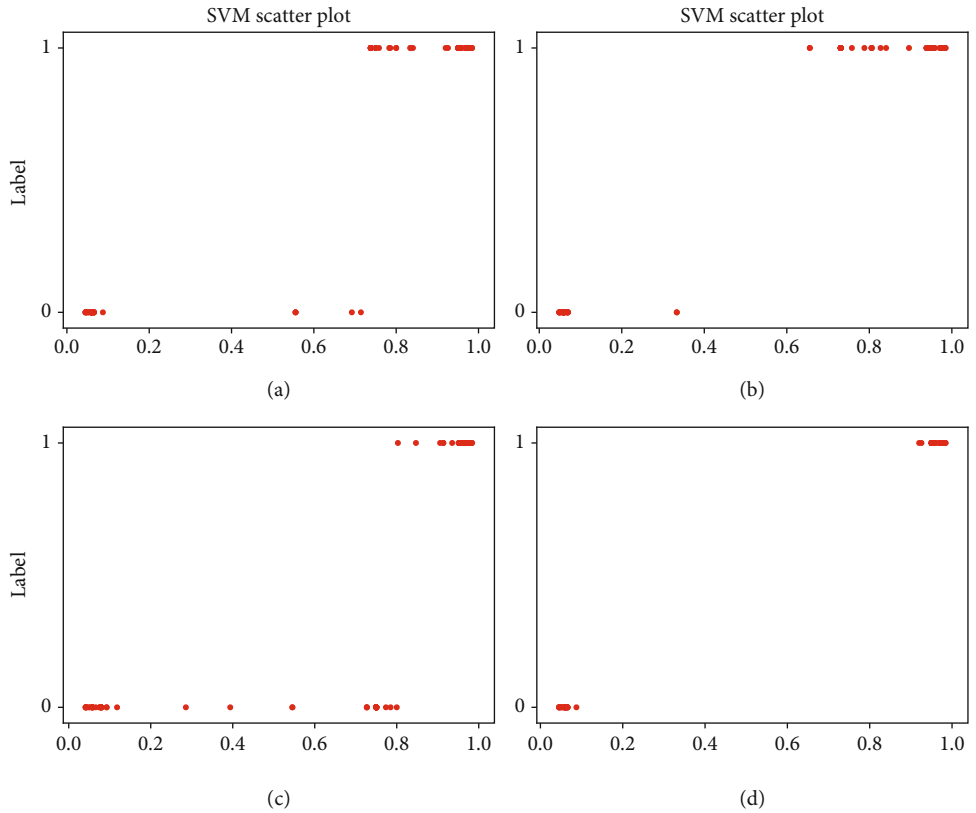

Figure 7: Scatterplot of the distribution of predicted probabilities p and test labels (0 and 1) in the SVM model under different threshold values. (a) The distribution of probability p and label when threshold = 0.72. (b) The distribution of probability p and label when threshold = 0.6. (c) The distribution of probability p and label when threshold = 0.8. (d) The distribution of probability p and label in ideal situation.

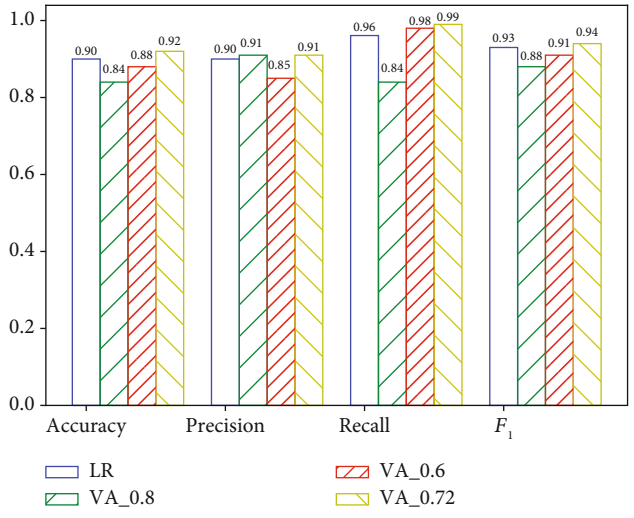

Figure 8: Accuracy of LR model and Venn-Abers predictors under different thresholds.

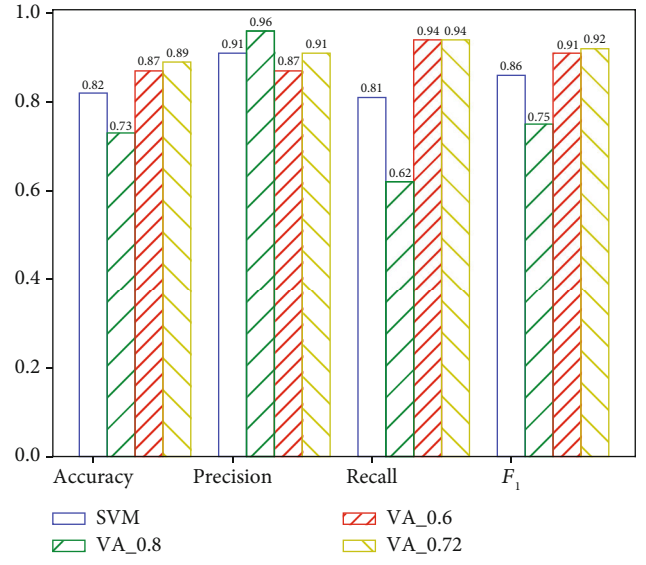

Figure 9: Accuracy of SVM model and Venn-Abers predictors under different thresholds.

does not include five-fold cross-validation, so each time the model is used to predict the corresponding test data, and $testPre_{1-n}(n = 5)$ is given. After the training set is five-fold cross-validated, the average of $testPre_n$ output from each test set is taken as $testPre_{1-\text{mean}}$, $testPre_1$, and $testPre_{1-\text{mean}}$ which are spliced together as new feature data $[trainPre_{1-1}, trainPre_{1-2}, trainPre_{1-3}, trainPre_{1-4}, trainPre_{1-5}, trainPre_{1-\text{mean}}]$.

Table 2: Model accuracy comparison.

Model	Accuracy	Precision	Recall	F_1 score
SVM	0.82	0.91	0.81	0.86
KNN	0.88	0.92	0.90	0.91
DT	0.71	0.83	0.70	0.76
RF	0.88	0.92	0.91	0.92
GBDT	0.91	0.91	0.96	0.94

Table 3: The model fusion results.

Model	Accuracy	Precision	Recall	F_1 score
SVM	0.82	0.91	0.81	0.86
KNN	0.88	0.92	0.90	0.91
DT	0.71	0.83	0.70	0.76
RF	0.88	0.92	0.91	0.92
GBDT	0.91	0.91	0.96	0.94
Stacking	0.93	0.94	0.95	0.97

Next, the other four underlying classifiers adopt the above method to generate new feature data $[trainPre_{2-1}, trainPre_{2-2}, trainPre_{2-3}, trainPre_{2-4}, trainPre_{2-5}, trainPre_{2-\text{mean}}] \cdots [trainPre_{5-1}, trainPre_{5-2}, trainPre_{5-3}, trainPre_{5-4}, trainPre_{5-5}, trainPre_{5-\text{mean}}]$. Finally, the new feature data generated by the first five base classifiers are used as new training set data to train the LR classification model. The model fusion results are shown in Table 3.

In the experiment, a Stacking model fusion framework was constructed and compared with the effects of other five methods. According to the comprehensive analysis, the effects of the six methods from low to high are DT, SVM, KNN, RF, GBDT, and Stacking. DT, RF, and GBDT all combine multiple tree models. DT is the underlying classifier, and one tree determines the prediction result. Its effect is not as good as DT + Boosting = GBDT and DT + Bagging = RF. Multiple trees together determine the prediction result. The GBDT algorithm is an addition model composed of k trees; thus, the effect is better than DT, and RF greatly increases the diversity of trees due to the addition of random attribute selection; thereby, it can achieve better results; Stacking is the best, Stacking is combined with the underlying classifier which adopts the relearning method to construct a complex learning process, and then, it can learn more information. Therefore, Stacking is based on different algorithms and secondary learning so as to achieve the optimal generalization effect.

4.7. Stacking and Stacking-VA Analysis. The Stacking model fusion algorithm has a score function decision_function(), which can be used as the input of the Venn-Abers predictor, thereby constructing the Venn-Abers predictor Stacking-VA based on the Stacking algorithm. The score function of the Stacking model and the label corresponding to the test set are adopted as the input of the Venn-Abers predictor and the multiprobability sequence values $[p_0, p_1]$ as output. p_0 and p_1 are fused according to formula (6) to obtain an accurate prediction result p, which can be compared with the experimental index of Stacking fusion.

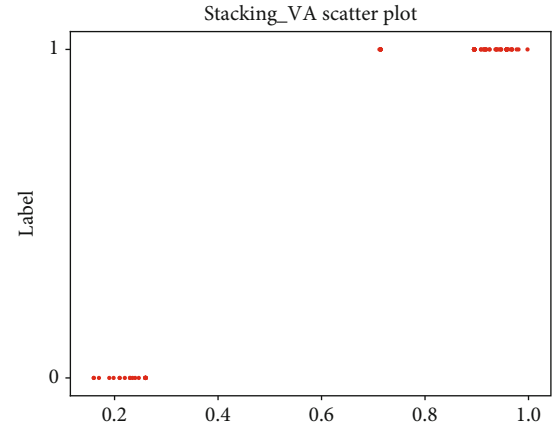

Figure 10: Distribution of p values and test labels.

Table 4: Stacking and Stacking-VA performance comparison.

Model	Accuracy	Precision	Recall	F_1 score
Stacking	0.93	0.94	0.95	0.97
Stacking-VA	0.95	0.96	1.0	0.96

It can be seen from Figure 10 that it is easy to choose a dynamic threshold of 0.5 to distinguish the prediction result of Stacking-VA. The performance comparison of Stacking and Stacking-VA is shown in Table 4. The performance of the multimodel fusion algorithm based on Venn-Abers predictor is better than that of Stacking.

The reason why Stacking-VA predictor is superior to Stacking can be analyzed theoretically. Stacking-VA uses dynamic thresholds to predict results based on the distribution of multiple probability values. Unlike Stacking-integrated learning using static thresholds, the judgment of abnormal logs will be more accurate. For example, as shown in Figure 10, if the dynamic threshold is set to 0.7, the probability value p is not less than 0.7; then the prediction is normal; or otherwise, it is abnormal. Stacking through the decision_tree() function is greater than 0, and it is predicted to be normal; otherwise, it is predicted to be abnormal, so that Stacking-VA can capture the labels that Stacking itself predicts through the dynamic threshold; thereby, it can reduce the number of false positives.

Integrated learning compares the prediction effect of a single model. Because the Stacking model makes full use of the advantages of each algorithm, it effectively reduces the risk of poor generalization performance of a single model and makes the distribution of predicted label data closer to the distribution of real label data. For example, for the single model, because the feature vectors corresponding to part of the data are highly similar, the score function cannot be effectively calculated to further make the distribution of probability p more dispersed. And this situation will be avoided in Stacking. In the process of Stacking model fusion, combining the characteristics of multiple models and relearning can get

an effective score function, so that the distribution of probability p is close to the two extreme values: 0 and 1, which can reduce the existence of intermediate values.

5. Conclusion

In this work, HDFS is really a dataset to detect abnormal system logs. A flexible machine learning framework, Venn-Abers, was introduced to make precise and valid probabilistic prediction for the log data. Instead of predicting only a single label for the unknown object, Venn-Abers predictors are able to calculate the label probability distribution of a set of samples and provide evaluation of the validity of predictive labels with a degree of certainty. This paper attempts to exploit the Venn-Abers predictor on a single model such as logistic regression, support vector machine, and integrated learning algorithm Stacking for log anomaly detection. Two Venn-Abers predictors are developed and compared with two underlying classification methods in the aspect of the validity of probabilistic predictions. The results show that the validity of Venn-Abers predictors holds all the way as the samples increased. In terms of probability prediction accuracy, the Venn-Abers predictors are developed from a single model and integrated learning methods, and then, the probability values of the predicted labels are calculated separately. The probability values and the distribution of the predicted labels are determined through statistics, and the optimal threshold is set to more accurately detect log abnormalities.

The experimental results show that the Venn-Abers predictor under integrated learning can take advantage of different algorithms to obtain the best anomaly detection results. It proves that the Venn-Abers predictor can be effectively applied in the field of system log anomaly detection from multiple aspects. The accuracy of machine learning prediction log data results can be evaluated and classified accurately and validly.

In the future, more diverse scoring classifiers will be considered to be integrated into this prototype platform for anomaly detection of system logs. Different underlying algorithms are selected according to the degree of correlation, which can maximize the advantages of different algorithms. On the other hand, the Venn-Abers predictor will analyze the probability interval to further improve the accuracy of log anomaly detection.

Data Availability

The research data supporting the results of this study can be available from https://zenodo.org/record/3227177#.XvVGE20zbIU.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Science Foundation of China under Grants 61872202, 61601467, and U1533104; the Civil Aviation Safety Capacity Building Foundation of China under Grants PESA2018079, PESA2019073, and PESA2019074; the Natural Science Foundation of Tianjin under Grant 19JCYBJC15500; Key Research Program of the Chinese Academy of Sciences under Grant No. KFZD-SW-440; 2019 Tianjin New Generation AI Technology Key Project under Grant 19ZXZNGX00090; and Tianjin Key Research and Development Plan under Grant 20YFZCGX00680. The authors thank the Chinese University of Hong Kong for providing HDFS log data. The authors appreciate the valuable comments provided by the anonymous reviewers.

References

[1] L. Martí, N. Sanchez-Pi, J. Molina, and A. Garcia, "Anomaly detection based on sensor data in petroleum industry applications," *Sensors*, vol. 15, no. 2, pp. 2774–2797, 2015.

[2] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS) a fast unsupervised anomaly detection algorithm," in *KI-2012: Poster and Demo Track*, pp. 59–63, German Research Center for Artificial Intelligence (DFKI), 2012.

[3] A. Grover, *Anomaly Detection for Application Log Data*, Master's Projects, 2018.

[4] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 427–438, 2000.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[6] T. Kohonen, *Self-Organizing Maps*, Springer Science & Business Media, 2012.

[7] D. Ö. Faruk, "A hybrid neural network and Arima model for water quality time series prediction," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 586–594, 2010.

[8] I. Goodfellow, "NIPS 2016 Tutorial: Generative adversarial networks," 2016, https://arxiv.org/abs/1701.00160.

[9] J. Zhu, S. He, and J. Liu, "Tools and benchmarks for automated log parsing," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 121–130, Montreal, QC, Canada, 2019.

[10] W. Xu, L. Huang, and A. Fox, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles - SOSP '09*, pp. 117–132, Haifa, Israel, 2009.

[11] J. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection," in *USENIX Annual Technical Conference*, pp. 1–14, Boston, MA, USA, 2010.

[12] S. He, J. Zhu, P. He et al., "Experience report: system log analysis for anomaly detection," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 207–218, Ottawa, ON, Canada, 2016.

[13] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, Dallas, TX, USA, 2017.

[14] H. Li and X. Wu, "Time series anomaly detection method based on frequent pattern discovery," *Journal of Computer Applications*, vol. 38, no. 11, pp. 3204–3210, 2017.

[15] B. Xia, J. Yin, J. Xu, and Y. Li, "LogGAN: a sequence-based generative adversarial network for anomaly detection based on system logs," in *Science of Cyber Security. SciSec 2019*, pp. 61–76, Springer, 2019.

[16] B. Xia, Y. Bai, and J. Ying, "Generative adversarial network based log-level anomaly detection approach for system logs," *Journal of Computer Applications*, pp. 1–7, 2020.

[17] R. Jordaney, K. Sharad, and S. K. Dash, "Transcend: detecting concept drift in malware classification models," *USENIX Security Symposium*, pp. 625–642, USENIX, 2017.

[18] W. Zhi, H. Gao, and Y. Zhang, "Fortifying botnet classification based on Venn-Abers prediction," in *DEStech Transactions on Computer Science and Engineering*, pp. 721–728, Guilin, China, 2017.

[19] Y. Ren, Z. Gu, Z. Wang et al., "System log detection model based on conformal prediction," *Electronics*, vol. 9, no. 2, p. 232, 2020.

[20] H. Papadopoulos, "Reliable probabilistic classification with neural networks," *Neurocomputing*, vol. 107, pp. 59–68, 2013.

[21] A. Lambrou, H. Papadopoulos, and I. Nouretdinov, "Reliable probability estimates based on support vector machines for large multiclass datasets," in *Artificial Intelligence Applications and Innovations. AIAI 2012*, pp. 182–191, Springer, 2012.

[22] C. Zhou, I. Nouretdinov, Z. Luo et al., "A comparison of Venn machine with Platt's method in probabilistic outputs," in *Artificial Intelligence Applications and Innovations*, pp. 483–490, Springer, 2011.

[23] I. Nouretdinov, D. Devetyarov, V. Vovk et al., "Multiprobabilistic prediction in early medical diagnoses," *Annals of Mathematics and Artificial Intelligence*, vol. 74, no. 1-2, pp. 203–222, 2015.

[24] H. Papadopoulos and G. Anastassopoulos, "Vesicoureteral reflux detection with reliable probabilistic outputs," *Information Sciences*, vol. 308, pp. 113–124, 2015.

[25] I. Nouretdinov, D. Volkhonskiy, and P. Lim, "Inductive Venn-Abers predictive distribution," *Conformal and Probabilistic Prediction and Applications*, pp. 15–36, Springer, 2018.

[26] V. Vovk and I. Petej, *Venn-Abers Predictors*, Computer Science, 2012.

[27] L. Pan, Z. Gu, Y. Ren, C. Liu, and Z. Wang, "An anomaly detection method for system logs using Venn-Abers predictors," in *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, pp. 362–368, Hong Kong, 2020.

[28] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[29] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[30] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[31] J. Sheng, L. Yue, and Y. Chengyu, "Detection method of Android malware based on multi-feature and Stacking algorithm," *Computer Systems & Applications*, vol. 27, no. 2, pp. 197–201, 2018.

[32] L. Breimen, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[33] U. Johansson, T. Löfström, and H. Boström, "Calibrating probability estimation trees using Venn-Abers predictors," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 28–36, Hyatt Regency Calgary | Calgary, Alberta, Canada, 2019.

[34] M. Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and E. Silverman, "An empirical distribution function for sampling with incomplete information," *The Annals of Mathematical Statistics*, vol. 26, no. 4, pp. 641–647, 1955.

[35] V. Vovk, I. Petej, and V. Fedorova, "Large-scale probabilistic predictors with and without guarantees of validity," *Advances in Neural Information Processing Systems*, pp. 892–900, Computer Science, 2015.

[36] D. Borthakur, "The hadoop distributed file system: architecture and design," *Hadoop Project Website*, vol. 11, 2007.

[37] J. Peck, B. Goossens, and Y. Saeys, "Calibrated multi-probabilistic prediction as a defense against adversarial attacks," in *Benelux Conference on Artificial Intelligence and Belgian Dutch Conference on Machine Learning*, pp. 1–11, BenelearnAt: Brussels, Belgium, 2019.