

Article

A Network Intrusion Detection Method Based on Bagging Ensemble

Zichen Zhang ^{1,2} , Shanshan Kong ^{1,*}, Tianyun Xiao ^{1,2} and Aimin Yang ¹

¹ College of Science, North China University of Science and Technology, Tangshan 063210, China; aimin@ncst.edu.cn (A.Y.)

² The Key Laboratory of Engineering Computing in Tangshan City, North China University of Science and Technology, Tangshan 063210, China

* Correspondence: kongss@ncst.edu.cn

Abstract: The problems of asymmetry in information features and redundant features in datasets, and the asymmetry of network traffic distribution in the field of network intrusion detection, have been identified as a cause of low accuracy and poor generalization of traditional machine learning detection methods in intrusion detection systems (IDSs). In response, a network intrusion detection method based on the integration of bootstrap aggregating (bagging) is proposed. The extreme random tree (ERT) algorithm was employed to calculate the weights of each feature, determine the feature subsets of different machine learning models, then randomly sample the training samples based on the bootstrap sampling method, and integrated classification and regression trees (CART), support vector machine (SVM), and k-nearest neighbor (KNN) as the base estimators of bagging. A comparison of integration methods revealed that the KNN-Bagging integration model exhibited optimal performance. Subsequently, the Bayesian optimization (BO) algorithm was employed for hyper-parameter tuning of the base estimators' KNN. Finally, the base estimators were integrated through a hard voting approach. The proposed BO-KNN-Bagging model was evaluated on the NSL-KDD dataset, achieving an accuracy of 82.48%. This result was superior to those obtained by traditional machine learning algorithms and demonstrated enhanced performance compared with other methods.

Keywords: intrusion detection; ensemble learning; Bayesian optimization; extremely random tree



Citation: Zhang, Z.; Kong, S.; Xiao, T.; Yang, A. A Network Intrusion Detection Method Based on Bagging Ensemble. *Symmetry* **2024**, *16*, 850. <https://doi.org/10.3390/sym16070850>

Academic Editor: Jie Yang

Received: 22 May 2024

Revised: 29 June 2024

Accepted: 1 July 2024

Published: 5 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The increasing popularity of the Internet and the continuous development of information technology have led to an increase in the prominence of network security issues. This has given rise to a multitude of network attacks and malicious behaviors, which have the potential to pose a significant challenge to the security and stability of network systems [1]. IDSs represent a crucial network security protection measure. By monitoring network traffic and system activities, they can detect and respond to potential security threats in a timely manner. This makes them an invaluable tool in the protection of network security [2]. Network traffic can be categorized into normal and abnormal, and intrusion detection can be regarded as a binary classification problem. The accuracy of IDS can be significantly improved by improving the performance of the classifier [3]. There are three approaches to existing intrusion detection techniques: signature detection, anomaly detection, and statistical detection [4]. Signature detection is one of the most prevalent techniques employed in intrusion detection systems. Its fundamental premise is to identify known attack patterns and malicious behaviors through the utilization of a predefined feature library or rule set. Nevertheless, signature detection is subject to certain limitations in the context of addressing unknown and variant attacks. Anomaly detection represents the opposite approach, whereby potential attacks are identified by constructing a model of

normal behavior and detecting activities that deviate significantly from the norm. Although anomaly detection is an effective method for identifying unknown attacks, it is also prone to a higher false alarm rate. Statistical detection is an intrusion detection method (based on statistical principles) that identifies abnormal behavior by analyzing the statistical characteristics of network traffic or system activities. In comparison with signature detection and anomaly detection, statistical detection places greater emphasis on the analysis of data distribution and patterns, which can enhance the accuracy of detection to a certain extent.

In recent years, there has been a proliferation of intrusion detection methods, including machine learning (ML) [5], deep learning (DL) [6], ensemble learning (EL) [7], and others. These methods have been employed in research on intrusion detection, demonstrating enhanced performance. Machine learning models (e.g., decision trees [8], SVM [9], KNN [10]) can be utilized to identify pivotal features within network traffic and system logs, and to detect anomalous activities by learning patterns of normal network traffic and system behavior. Deep learning models (e.g., DNN [11], RNN [12], LSTM [13]) can automatically learn feature representations in data through the hierarchical structure of deep neural networks, and can provide better generalization capabilities when trained on large-scale data. Ensemble learning methods, including bagging [14], boosting [15], and stacking [16], are employed to enhance the model's robustness and generalizability by integrating diverse approaches to various base estimators.

Although intrusion detection systems are capable of detecting and responding to potential security threats in a timely manner to a certain extent, there are still significant challenges when faced with the real-world, large-scale, high-speed, and complex network environment. One such challenge is the asymmetry of information characteristics and redundancy features of datasets [17,18]. In practical data analysis and model training tasks, it is desirable for the features in the dataset to provide a substantial amount of information that can be leveraged to support model training and prediction. However, the presence of redundant features may result in increased computational complexity and a reduction in the model's generalization ability. Consequently, it is a significant challenge to accurately and efficiently identify redundant features for filtering. Secondly, there is the asymmetry of network traffic distribution [19]. In a realistic distribution of network traffic data, there is usually a large amount of normal traffic and a small amount of abnormal traffic. Consequently, the model may tend to predict the majority class, which may result in a decrease in model performance and generalization ability. Therefore, another challenge is to improve the stability and generalization ability of the model in the face of unbalanced samples.

Consequently, the conventional network malicious traffic detection techniques currently face significant challenges. In light of the asymmetry of dataset information features and redundant features, and the asymmetry of network traffic distribution in the field of network intrusion detection, there is an urgent need for research on detection methods with a high detection rate in order to enhance cybersecurity. To address the aforementioned issues, this paper employs feature engineering, integrated learning, optimization algorithms, and other methodologies to conduct in-depth research on the effective detection of anomalous traffic. This paper's contributions are as follows:

- The ERT method was employed to determine the significance of the Gini factors associated with the traffic data features. This involved the removal of irrelevant and superfluous features and the construction of an optimal subset of features.
- The BO algorithm was used to optimize the parameters of the KNN base estimators to improve the performance of the ensemble model.
- A bagging ensemble approach was proposed for intrusion detection. Based on feature selection and parameter optimization, training samples were randomly sampled using the bootstrap method to construct KNN base estimators for hard voting integration. This approach avoided the instability of a single base classifier due to the imbalance of data categories, effectively reducing the variance and improving the model's generalizability.

- A series of comprehensive experiments were conducted on the intrusion detection dataset NSL-KDD with the objective of comparing the performance of different machine learning models integrated as base estimators. This was performed to validate the effectiveness of the proposed model.

The rest of this paper is organized as follows: Section 2 presents related work. Section 3 explains our proposed model. Section 4 describes the dataset, shows the experimental results and analyzes them. Section 5 summarizes and provides an outlook.

2. Related Work

The existing body of the literature on intrusion detection techniques is primarily concerned with three principal areas of research. One area of investigation is the development of feature engineering techniques for intrusion detection datasets. In their study, Safaldin et al. [20] proposed an enhanced IDS by using the modified binary grey wolf optimizer with support vector machine (GWOSVM-IDS), which improves the accuracy and detection rate of intrusion detection and reduces the processing time by decreasing the rate of false alarms and the number of features generated by the IDS. Kumar et al. [21] proposed a penalized reward-based ant colony optimization (PRACO) feature selection method, which facilitates a more optimal exploration–exploitation trade-off by rewarding useful features and penalizing others. The proposed model achieved an accuracy of 81.682% on the NSL-KDD dataset. Ghosh et al. [22] proposed a new wormwood grouse mating (SGM) algorithm in 2022 and applied it to IDS by reducing the original 41 features to 14, achieving an average accuracy of 81.429%. Ye et al. [23] employed the meta-heuristic algorithm hybrid breeding optimization (HBO) to IDS and proposed an integrated framework for feature selection based on the improved HBO. The framework assigns a subpopulation to each feature space and identifies the optimal subset of features. The objective was to enhance the accuracy of intrusion detection through the integration of the subpopulations. Herrera et al. [24] conducted a comprehensive examination of existing feature selection algorithms to identify their shortcomings and developed novel multimetric feature selection algorithms to reduce the dimensionality of the training dataset by leveraging the qualitative information provided by multiple feature selection metrics, and through experimentation. This study demonstrated the efficacy of the proposed approach. Nazir et al. [25] applied feature selection algorithms to reduce the dimensionality of the data and improve the performance of classifiers by proposing a wrapper-based feature selection method—taboo search–random forest (TS-RF)—and testing it on the UNSW-NB15 dataset. The experimental results demonstrated that TS-RF enhanced classification accuracy while reducing the number of features and the false alarm rate. In 2024, Akhiat et al. [26] proposed an efficient integrated feature selection algorithm for intrusion detection, namely the intrusion detection efficient ensemble feature selection (IDS-EFS). This algorithm enhances the interpretability of the model by decreasing the network data dimensionality, thereby reducing resource requirements and improving generalization. Khammassi et al. [27] tested three different decision tree classifiers. The use of binomial logistic regression and multinomial logistic regression for binary and multiclassified datasets, respectively, has been demonstrated to successfully reduce the feature space and improve classification performance. Yang et al. [28] combined the feature selection technique with the integration method to propose an adaptive ensemble feature selection algorithm for intrusion detection systems (IDS-EFS). This combined approach resulted in the development of an adaptive integration model. Specifically, the neighborhood dependency of neighborhood rough set (NRS) was introduced into the bottle sea sheath algorithm (SSA) to propose a heuristic feature selection algorithm (NRS-SSA). Furthermore, SSA was introduced to optimize the weight matrix when setting the voting weights. The results demonstrated that the model achieved a state-of-the-art level of detection. Mohammad et al. [29] proposed an innovative feature selection algorithm called “the highest wins” (HW), which demonstrated advantages in a variety of evaluation metrics, including recall, precision, and error rate, compared with the well-known chi-square and information gain strategies.

The second area of research is the investigation of sample category imbalance in intrusion detection datasets. Qian et al. [30] identified the issue of sample category imbalance, which can result in the suboptimal detection performance of intrusion detection models. They proposed an improved hybrid sampling (IHS) method based on the chaotic particle swarm optimization (CPSO) algorithm as a sampling algorithm to address this issue, and a deep learning long short-term memory (DLSTM) model with long short-term memory (LSTM) functionality was proposed, which achieved high accuracy in classifying intrusion behaviors. This model outperformed other comparative models in terms of accuracy. Jiang et al. [31] employed a one-sided selection (OSS) method to reduce the influence of noisy samples in the majority class, subsequently augmenting the minority class samples through synthetic minority oversampling (SMOTE). The proposed method enabled the model to fully learn the features of the minority class samples while reducing the training time. In 2022, Jung et al. [32] proposed a hybrid resampling method that added minority classes and removed noisy data by using SMOTE and an edited neural network to generate a more balanced dataset. The proposed method was validated using two publicly available intrusion detection datasets, PKDD2007 and CSIC2012. The results demonstrated a clear advance on previous work. Zhang et al. [33] integrated deep learning methods with statistical techniques to address the challenge of detecting minority class samples. The authors proposed an intrusion detection method, ICVAE-BSM, which employed an improved conditional variational auto-encoder (ICVAE) and a boundary synthesis minority classes oversampling technique (BSM). This method was designed to enhance the efficiency of Internet of Things (IoT) attack detection in the presence of sample imbalance and accuracy issues. In 2024, Liu et al. [34] proposed a multiconstraint migration method with additional auxiliary domains and designed a multiscale and multilevel sample augmentation discriminator to accomplish final IoT intrusion detection under sample distribution imbalance. This approach achieved an average accuracy of 96.398% on four datasets and can be effectively used for intrusion detection in real IoT environments.

The third area of research concerns the improvement and fusion of intrusion detection model structures with the objective of enhancing detection performance. In their study, Esmaeili et al. [35] explored the potential of deep learning-based intrusion detection systems and demonstrated the superiority of LSTM and BiLSTM models over other models in terms of performance. Zaryn et al. [36] conducted a comparative study on the performance and efficiency of IoT anomaly detection models by using the NSL-KDD dataset, and the results indicated that the integrated model XGBoost represents the most accurate and efficient methodology. Lee et al. [37] proposed a two-stage fine-tuning algorithm based on the WGAN-GP model to enhance the recognition accuracy of sparse data probes by fine-tuning the classification algorithm and model parameters. The final experimental results demonstrated that the MLP classifier exhibited a notable improvement in accuracy, rising from 74% to 80% after fine-tuning. This level of performance is significantly superior to that of all other classifiers. Farooq et al. [5] proposed an intrusion detection scheme, IDS-FMLT, which incorporates machine learning techniques to detect intrusions in heterogeneous networks comprising disparate source networks and to safeguard the network from malevolent attacks. The scheme achieved a validation accuracy of 95.18% and a miss detection rate of 4.82%. Sarnovsky et al. [38] proposed a hierarchical intrusion detection system based on a symmetric combination of machine learning methods and knowledge-based approaches, combining several different machine learning models. The system is capable of predicting specific types of attacks and selecting the appropriate model to perform the prediction at a selected level. Alotaibi et al. [39] proposed a fusion of machine learning with an intelligent intrusion detection model that improves accuracy and decision making by combining the predictions of different models using a fuzzy inference system. Elnakib et al. [40] proposed an enhanced anomaly based intrusion detection deep learning multiclass classification model (EIDM), which is able to classify different traffic behaviors, including multiple attack types. The results demonstrated that the method achieved an accuracy of 95% on the CICIDS2017 dataset. In a separate study, Wang et al. [41] integrated deep learning models

and proposed two integrated deep learning models, SDAE-ELM and DBN-Softmax. A small batch gradient descent method was employed for network training and optimization, resulting in enhanced classification accuracy and the capacity for real-time response to intrusion behaviors. In their study, Praveena et al. [42] proposed a deep reinforcement learning technique based on the black widow optimization (DRL-BWO) algorithm and used an improved reinforcement learning-based deep belief network (DBN) for intrusion detection, achieving an accuracy of 98.5%.

Our approach is a simultaneous consideration of the three aforementioned aspects of the research. Firstly, the ERT method represents an efficacious solution to the issue of asymmetry between informative and redundant features. The method improves classification performance while reducing the dimensionality of the data, and the highly parallelized training approach is effective when dealing with large-scale data. Secondly, the use of a bagging ensemble method addresses the problem of asymmetry in sample distribution. This method improves the complexity of the model, avoids the instability of individual base estimators due to the imbalance of data categories, and enhances the generalization ability of the fused model. Finally, the parameters of the base estimators are tuned using the BO algorithm, which further improves the performance of the model.

3. Proposed Method

The proposed BO-KNN-Bagging framework is illustrated in Figure 1. It encompasses four principal stages: (1) Data preprocessing. Initially, the original attack categories are converted into binary classification, then the category-based labeled data are converted into numerical classification through label coding, and finally the data are scaled using the min-max normalization technique. (2) Feature selection. The ERT algorithm is employed to evaluate the importance of features and determine the weights of features, thereby identifying the most informative ones. (3) Construction and optimization of base estimators. A total of 5–200 base estimators are constructed, and multiple self-help samples are generated by extracting samples from the original dataset with putback based on the method of bootstrap. Tenfold cross-validation is performed, followed by tuning of the base estimators' parameters using BO. The model is then trained independently using each of these samples. (4) Model ensemble. In the binary classification problem, the independent prediction results of KNN base estimators are integrated using hard voting, i.e., majority voting, and the final classification result obtained is used as the output of the entire integrated model. The trained BO-KNN-Bagging model is employed for the purpose of detecting attacks.

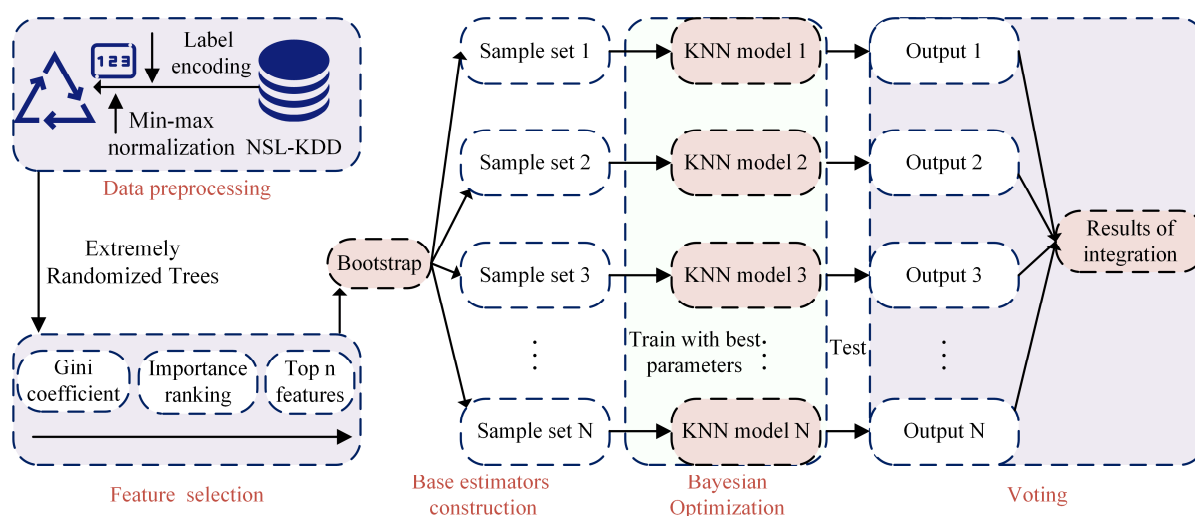


Figure 1. The BO-KNN-Bagging model framework.

3.1. Data Preprocessing

To facilitate the training and prediction of machine learning models, we employed label encoding to convert category-based label data into numerical form. Integer encoding maps each feature label to an integer value, which is more space efficient than binary encoding and performs well in many machine learning algorithms.

Concurrently, different features tend to have disparate sizes and ranges, which can result in some features exerting a considerable influence on the model while others have a relatively minor impact. To address this issue, all features are scaled to a uniform range using min-max normalization, also known as deviation normalization. This is a common data preprocessing technique employed to scale numerical data to a specific range, typically within the bounds of $[0, 1]$ or $[-1, 1]$. The method entails a linear transformation of the original data, whereby the minimum value is mapped to the desired minimum value and the maximum value is mapped to the desired maximum value. The calculation procedure for min-max normalization is presented below.

$$X_{norm} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (1)$$

where X represents each sample in the original dataset, X_{norm} is the normalized sample value, $X_{\min}$ and $X_{\max}$ are the minimum value and maximum value in the dataset, respectively. This method is suitable for processing features of different scales, ensuring that they have similar importance within the same range. It also helps to eliminate the dimensional effect between features, thereby improving the performance of the machine learning model. This makes the model more robust and accurate.

3.2. Bootstrap Aggregating

Bagging is an algorithm proposed by Breiman to train multiple base classifiers in parallel and perform ensemble learning based on bootstrapping [43]. As shown in Algorithm 1, bagging combines multiple models to improve the overall prediction performance and reduce the risk of overfitting. Among them, bootstrap means that a new sample matrix D' can be obtained by extracting one sample from the N sample matrix D with replacement, and extracting M ($M \leq N$) times:

$$D' = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_M, y_M)\} \quad (2)$$

where x and y are the feature matrix and label vector of the original samples, respectively. The number of samples is identical to that of the original sample set, and the same samples can be extracted from the new sample set. Although the sample distribution of each sample set differs to some extent, they all retain some of the information in the original training data, thus enabling each base estimator to learn slightly different content, thereby forming the diversity of the bagging ensemble model. Bootstrapping allows for the training of multiple models on different samples, thereby reducing the overall variance of the model and resulting in a more generalizable model. Additionally, by exposing the model to slightly different multisamples, the method ensures that the integrated model is better able to generalize to unseen data. This results in a model that is less sensitive to the idiosyncrasies of a single training dataset. Furthermore, the bootstrap method itself introduces randomness, which helps to deal with overfitting. Models trained on different samples produce different errors, and by combining them, these errors are averaged out, thus improving the model's generalization performance.

Aggregation represents the integration strategy, which in classification problems is typically accomplished through voting. This involves each model providing a prediction, with the final prediction being the category that receives the greatest number of votes among these models. In regression problems, bagging typically entails taking the average of these models as the final prediction. By combining multiple base estimators, the instability of a single base classifier due to imbalance of data categories can be effectively avoided, and

the variance can be effectively reduced. The application of bagging ensemble learning to a fused model enhances its generalizability compared with a single estimator [44,45]. In our study, for the KDDTrain+ training set in NSL-KDD, comprising 125,973 data points, a random sampling with putback was performed, with the same number of data points taken for each evaluator for the subsequent training step.

Furthermore, the probability that an individual is always not picked up in a training dataset containing N samples with putback sampling can be expressed as follows:

$$\lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N = \frac{1}{e} \approx 0.368 \quad (3)$$

This indicates that approximately 36.8% of the initial samples are not included in the newly constructed samples. Consequently, these samples can be utilized as out-of-bag data (OOB data) for evaluating the generalization capacity of the novel model. The out-of-bag error (OOB error) refers to the discrepancy between the observed values of the OOB data and the predicted values, which is essentially the error of predicting the new samples when the generalization error is considered rather than the training error on the training set. The lower the value of OOB error, the better the prediction performance of the model. In particular, the OOB error reflects the performance of the model on data that have not been seen during training. A lower OOB error indicates that the model has a greater capacity for generalization ability and is more accurate when applied to data that have not been used to inform its development. Consequently, we utilize the OOB error as a supplementary metric for evaluating the model.

Algorithm 1 Bagging

Input: D //The NSL-KDD dataset
 n //The number of models
 m //The number of instances in the training set
Output: R^* //Prediction results of the ensemble model

- 1: **Initialize:** $E = \emptyset$ //The initialization set is used to store the prediction results
- 2: **for** $i = 1$ to n
- 3: **for** $j = 1$ to m
- 4: $rnd = \text{random}(0, m)$ //Random sampling
- 5: $D_{ij} = D(rnd)$ //Bootstrap
- 6: **end for**
- 7: $E_i = \text{training}(D_{ij})$ //Train with optimal parameters
- 8: $E = E \cup E_i$ //Result integration
- 9: **end for**
- 10: $R^* = \text{voting}(E)$ //Aggregating
- 11: **Return** R^*

3.3. Extremely Randomized Trees

The ERT algorithm is a machine learning algorithm based on bagging ensemble models, initially proposed by Geurts et al. in 2006 [46]. It typically employs the CART algorithm to construct the base estimators, introducing greater randomness based on random forests. In contrast to the random forest algorithm, ERT randomly selects a subset of features when choosing node splits and randomly selects a threshold within that subset for splitting. This increases the diversity of the model and further reduces the variance of the model.

The execution of traditional decision tree algorithms in a serial manner often results in the suboptimal utilization of computational resources. In contrast, ERT has clear advantages in the feature selection task [47], which employs a highly parallelized training method that effectively utilizes multicore processors and distributed computing resources. This enables it to perform well when dealing with large-scale data and to be robust to noise and outliers in the data. Furthermore, since each tree is trained on a random subset, the degree

of overfitting of a single tree is relatively low, and the overall model is more capable of generalization. This enables ERT to maintain model stability and reliability when dealing more effectively with complex data.

The ERT algorithm assesses the significance of features by quantifying the impact of each feature on the model. The importance of a feature is typically gauged by measuring the influence of a feature on the model's performance when the decision tree is split, with each node split based on a specific feature. The reduction in impurity (typically calculated using the Gini coefficient) before and after splitting based on that feature is determined at the time of splitting. The cumulative value of this reduction is then used as a measure of the importance of that feature. The Gini coefficient is calculated using the following formula:

$$Gini(D) = 1 - \sum_{i=1}^C p_i^2 \quad (4)$$

where $Gini(D)$ is the Gini coefficient of the dataset D , C represents the total number of categories, and p_i^2 is the proportion of samples in the dataset belonging to category i . The Gini coefficient of the dataset is 0, which means that all samples belong to the same category. The Gini coefficient of the dataset is 0, indicating that all samples belong to the same category. A smaller Gini coefficient signifies a lower impurity of the dataset, with a Gini coefficient of 0 indicating that the dataset is pure, i.e., all samples belong to the same category. In our study, the importance of each feature is calculated, and after sorting them in descending order of importance, the selection of relevant features can be made according to the performance of the model.

3.4. K-Nearest Neighbor

KNN is a frequently utilized supervised learning algorithm, typically employed for classification and regression problems. The fundamental concept of KNN is to categorize unknown samples based on the most prevalent category among their nearest neighbors or to predict the mean value of the samples with respect to their nearest neighbors, utilizing the distance metric of the samples in the feature space. The primary steps of the KNN algorithm include determining the number of neighbors, calculating the distances between the unknown samples and all the training samples, selecting the K neighbors with the closest distances, and predicting the classification or regression based on their labels. In the KNN algorithm, Euclidean distance and Manhattan distance are commonly used as the distance index. The Euclidean distance between two n -dimensional vectors $\alpha(x_1, x_2, \dots, x_n)$ and $\beta(y_1, y_2, \dots, y_n)$ in the plane can be expressed as follows:

$$D(\alpha, \beta) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (5)$$

The Manhattan distance is:

$$D(\alpha, \beta) = \sum_{i=1}^n |x_i - y_i| \quad (6)$$

As a traditional machine learning algorithm, KNN is relatively straightforward to comprehend and implement, with commendable prediction outcomes. However, it is susceptible to limitations, particularly in the selection of the number of neighbors and distance metrics, and in its capacity to handle large datasets, which may result in inefficiencies and suboptimal performance in high-dimensional and imbalanced datasets [48].

3.5. Support Vector Machine

SVM is a supervised learning model for classification and regression analysis. It performs well in problems such as pattern recognition, classification, and regression, and has unique advantages, particularly for applications in high-dimensional spaces. The basic

idea is to find an optimal separating hyperplane to categorize the data into different classes. This hyperplane should not only maximize the margin between categories, but also avoid misclassification as much as possible. SVM achieves this goal by the following steps.

When linearly divisible, the SVM searches for a linear hyperplane that maximizes the spacing between the two classes of data points, which can be represented as a hyperplane:

$$w \cdot x + b = 0 \quad (7)$$

where w is the normal vector, which determines the orientation of the hyperplane, and b is the bias, which determines the distance of the hyperplane from the origin. For the optimization problem: maximize the margin, that is, minimize $\|w\|$, while satisfying the constraints on all data points:

$$y_i(w \cdot x_i + b) \geq 1 \quad (8)$$

When linearly indistinguishable, a slack variable ξ_i is introduced, allowing that some data points can be on the wrong side of the line but controlling the total error through a penalty term. For the optimization problem, minimize $\frac{1}{2}\|w\|^2 + C \sum_{i=1}^n \xi_i$, where C is the penalty coefficient, and control the trade-off between interval and error classification.

When nonlinearly differentiable, a kernel function is employed to map the data from a low-dimensional space to a high-dimensional space and to identify a linear hyperplane within the high-dimensional space. Commonly utilized kernel functions include the polynomial kernel, the rbf kernel, and the sigmoid kernel. The optimization problem then becomes the maximization of intervals within the high-dimensional space.

SVM is capable of efficiently processing data within a high-dimensional space, while overfitting can be effectively avoided by maximizing the interval. However, this approach is not well suited to large-scale datasets, as the training time and memory consumption may become excessive for very large amounts of data. In our study, the RBF kernel was employed as the kernel function for mapping, and the SVM was assembled using the bagging method to assess the performance of distinct integrated models.

3.6. Classification and Regression Trees

CART is a tree structure based on recursive partitioning that is employed for the purpose of partitioning a dataset and constructing predictive models. The fundamental concept of this method is the recursive partitioning of the dataset into progressively smaller subsets. This process continues until a given sample within a subset is classified into the same category as the subset itself. In the construction of a CART classification tree, the process commences with the root node and entails the selection of a feature and a threshold for the partitioning of the data. This is typically determined through the application of information gain, the Gini coefficient, or the sum of squared errors. The dataset is then divided into two subsets based on the selected feature and threshold. Subsequently, the same segmentation rules are recursively applied to each subset until a stopping condition is met. The stopping condition may be that the number of samples in a node is less than a certain threshold or that the purity of the node reaches a certain threshold (e.g., all samples belong to the same category).

The CART classification tree is capable of handling high-dimensional data and complex classification scenarios, and is relatively immune to outliers. However, it is susceptible to minor variations in the data, which may result in significant alterations to the structure of the generated tree. In our study, the Gini coefficient is employed to segregate the data, and the CART is assembled using the bagging ensemble method to assess the efficacy of distinct integrated models.

3.7. Bayesian Optimization

BO is a global optimization method based on Bayes theorem, which is suitable for the situation when the objective function is difficult to calculate or the computational cost is

high. The key idea is to establish a probability model of the objective function $f(x)$ to guide the search process, and the parameter configuration X^* that leads to the optimal value of the objective function is found:

$$X^* = \max_{x \in X} f(x), X \in R \quad (9)$$

In order to optimize the objective function, BO typically employs a Gaussian process as an a priori model to represent the unknown objective function. A Gaussian process characterizes the distribution of the data by means of a mean function and a covariance function, which is typically considered an approximation of the objective function. The covariance function is employed to represent uncertainty [49]. In contrast to traditional grid search or random search, BO is capable of identifying optimal solutions in high-dimensional spaces and complex objective functions with greater efficiency [50]. In our study, the optimization objective is to maximize the classification accuracy, and $f(x)$ is the accuracy of the model under the parameter. Ten-fold cross-validation is used to optimize the number of neighbors K and the type of distance metrics of the KNN.

4. Experiments Results and Discussion

This section delineates the methodology employed in the experiment, and the default parameter settings are presented in Table 1 below. Initially, the data and preprocessing are elucidated. Subsequently, the model evaluation metrics are explained. Finally, the obtained results are presented, analyzed, and compared with other models.

Table 1. Model parameter settings.

Model	Parameter Settings
ERT	'n_estimators': 100, 'criterion': 'gini'.
CART	'criterion': 'gini', 'splitter': 'best'.
SVM	'kernel': 'rbf'.
KNN	'n_neighbors': 5, 'weights': 'uniform', 'p': 2.
AdaBoost	'base_estimator': 'DecisionTreeClassifier', 'n_estimators': 50.
MLP	'hidden_layer_sizes': 100, 'activation': 'relu', 'solver': 'adam', 'max_iter': 200.
LSTM	'units': 50, 'activation': 'relu', 'recurrent_activation': 'sigmoid'.
BO	'cv': 10, 'n_iter': 10, 'n_neighbors': (1, 200), 'p': (1, 2).

4.1. Dataset Description

KDD-Cup99, a classic dataset for intrusion detection in computer networks, contains an analysis of simulated U.S. Air Force network traffic with the objective of identifying intrusions in the network [51]. However, it is notable that the dataset suffers from the problem of category imbalance with redundant features. The experiments used the NSL-KDD dataset, which processed and filtered the KDD-Cup99 to make the data distribution more balanced and eliminate some duplicate and unnecessary features, which was more suitable for evaluating the performance of machine learning models on network intrusion detection tasks [52].

The KDDTrain+ and KDDTest+ subsets of the NSL-KDD dataset were employed for training and testing purposes, comprising 125,973 and 22,544 instances, respectively. The original dataset comprised 43 columns, with the first 41 columns representing feature vectors, as illustrated in Table 2. The 42nd column is the attack type label, comprising the four attack subtypes of DoS, Probe, U2R, and R2L with the normal label. The normal label was not considered in this instance, as the specific type of attack was not a focus. Instead, the label of each instance was transformed to normal. The results of the transformation are presented in Table 3. Column 43 indicates the number of times the initial researchers correctly categorized the sample using 21 machine learning algorithms, with values between 0 and 21.

Table 2. Feature description.

Item	Feature Name	Item	Feature Name
1	duration	22	is_guest_login
2	protocol_type	23	count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_rerror_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login		

Table 3. Statistics of NSL-KDD.

Category	KDDTrain+	KDDTest+
Abnormal	58,630	9711
Normal	67,343	12,833
Total	125,973	22,544

4.2. Data Preprocessing

It should be noted that, although the NSL-KDD dataset represents an improved version, columns 2, 3, 4, and 42 are not numerical and therefore unsuitable for machine learning model training. The feature “protocol_type” comprises three types (icmp, tcp, udp), while the feature “service” encompasses 70 types. The “flag” contains 11 different types (OTH, S3, RSTOS0, S2, SH, S1, RSTO, RSTR, REJ, S0, SF), and the 42nd column is the attack label, which is divided into normal and abnormal categories. Consequently, further processing is required, as illustrated in Table 4 below. The data are transformed into numerical form using label encoding. Following this, the data are normalized and scaled to the range [0, 1].

Table 4. Label encoding results.

Feature Name	Types	Results
protocol_type	icmp	0
	tcp	1
	udp	2
service	http, private, domain_u, . . . , tftp_u, http_8001, http_2784	0–69
flag	OTH, S3, RSTOS0, S2, SH, S1, RSTO, RSTR, REJ, S0, SF	0–10
attack	normal	0
	abnormal	1

4.3. Evaluation Measures

The evaluation of the model's performance is conducted through the use of accuracy, precision, recall, and f1-score. These metrics are defined in Table 5, and the aforementioned metrics can be derived by calculating the confusion matrix for cyberattacks [53]. In this context, true positive (*TP*) denotes the number of samples in which abnormal traffic is correctly categorized as such; false negative (*FN*) denotes the number of samples in which abnormal traffic is incorrectly categorized as normal; false positive (*FP*) denotes the number of samples in which normal traffic is incorrectly predicted to be abnormal; true negative (*TN*) indicates the number of samples where the traffic predicted to be normal is actually normal. The formula for evaluating the metrics is shown below.

Table 5. Confusion matrix.

	Predicated Abnormal	Predicated Normal
Actual abnormal	<i>TP</i>	<i>FN</i>
Actual normal	<i>FP</i>	<i>TN</i>

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (10)$$

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

$$F1 - score = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (13)$$

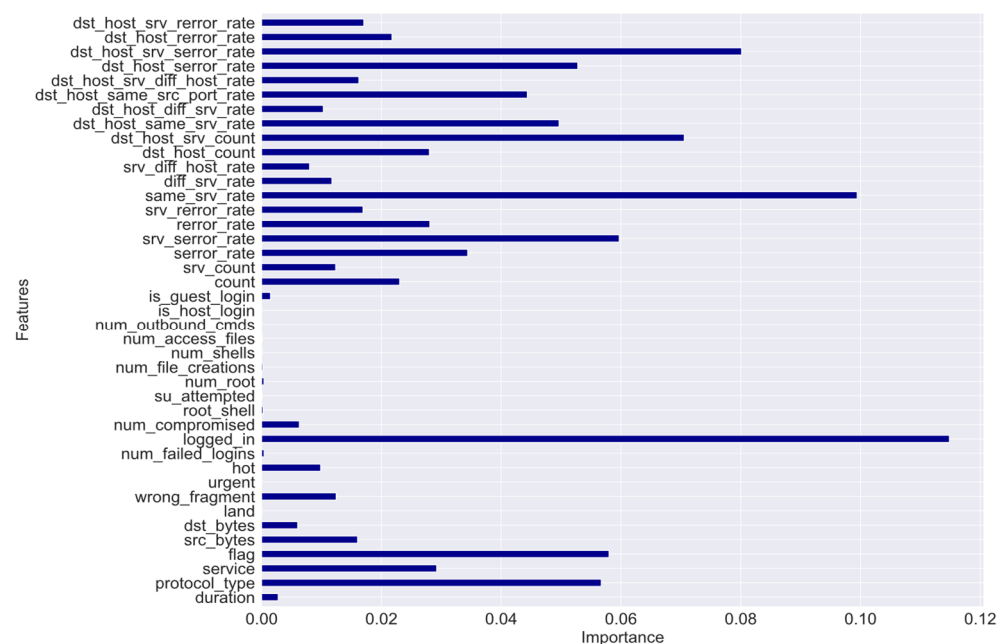
4.4. Feature Selection

The performance of various machine learning models (CART, SVM, KNN, AdaBoost, RF) was initially evaluated with default parameters. Table 6 presents the results. It can be observed that the majority of the models exhibit high recall values, indicating that these models are effective in detecting attack traffic. However, both accuracy and precision are lower due to the imbalanced nature of the NSL-KDD dataset, which contains a larger proportion of normal traffic and a smaller proportion of abnormal traffic. The aforementioned imbalance prompts the models to exhibit a proclivity towards predicting the majority class (normal traffic), thereby enhancing the recall value. However, this inclination concurrently results in a decline in accuracy and precision, attributable to the suboptimal performance of the minority class (abnormal traffic), in conjunction with the presence of a considerable number of redundant and noisy features, which may exert a detrimental influence on the training of the models. Rather than applying additional class-balancing techniques, such as undersampling and oversampling, we utilized feature engineering and ensemble methods to enhance the model's performance and generalizability. Our findings indicated that the SVM model based on the rbf kernel demonstrated the most favorable performance, with an accuracy rate of 78.26% and an f1-score of 79.48%. CART demonstrated the highest precision (68.91%) but the lowest recall (90.21%). The AdaBoost model exhibited poor performance, with all four indicators at a low level. Moreover, although the KNN model exhibited the highest recall at 97.74%, its accuracy, precision, and f1-score were 0.69%, 0.74%, and 0.87% lower, respectively, than the best performing SVM model.

Table 6. Performance with default parameters.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CART	78.25	68.91	90.21	78.13
RF	77.44	66.26	97.05	78.75
SVM (rbf)	78.26	66.98	97.70	79.48
KNN	77.57	66.24	97.74	78.97
AdaBoost	73.61	63.30	92.15	75.05

Subsequently, we employed the ERT method for the purpose of feature extraction and selection. Thereafter, we calculated the relative importance of the 41 features. The results of this process are presented in Figure 2. Feature importance is defined as the degree to which each feature contributes to the model prediction. Among the features, “logged_in” exhibited the greatest influence on the model, followed by “same_srv_rate” and “dst_host_srv_error_rate”. The features “is_host_login”, “num_outbound_cmds”, “num_access_files”, etc., had minimal impact on the model and can be considered redundant. Figure 3 illustrates the impact of 0–41 features selected by ERT on the accuracy. It can be observed that the model’s accuracy did not improve with the increase in the number of features, but fluctuated. This is due to the fact that the dimensionality of the data space increased with the increase in the number of features. In a high-dimensional space, the distance between data points becomes sparser, which impairs the model’s ability to learn and generalize. Additionally, the introduction of redundant features negatively affects the model’s training and prediction capabilities. Consequently, the accuracy of the model will reach its highest value at a specific point, necessitating the identification of optimal feature values that are applicable to a range of machine learning models. The overall accuracy of the CART model was optimal when the number of selected features was greater than 25. When the number of selected features was between 10 and 20, the KNN model had the highest accuracy, reaching a maximum of 81.95%, when the number was 17. This value was higher than the maximum accuracy of the other four machine learning models. When the number of selected features was between 0 and 10, the KNN model exhibited the most significant accuracy improvement.

**Figure 2.** Importance of different features.

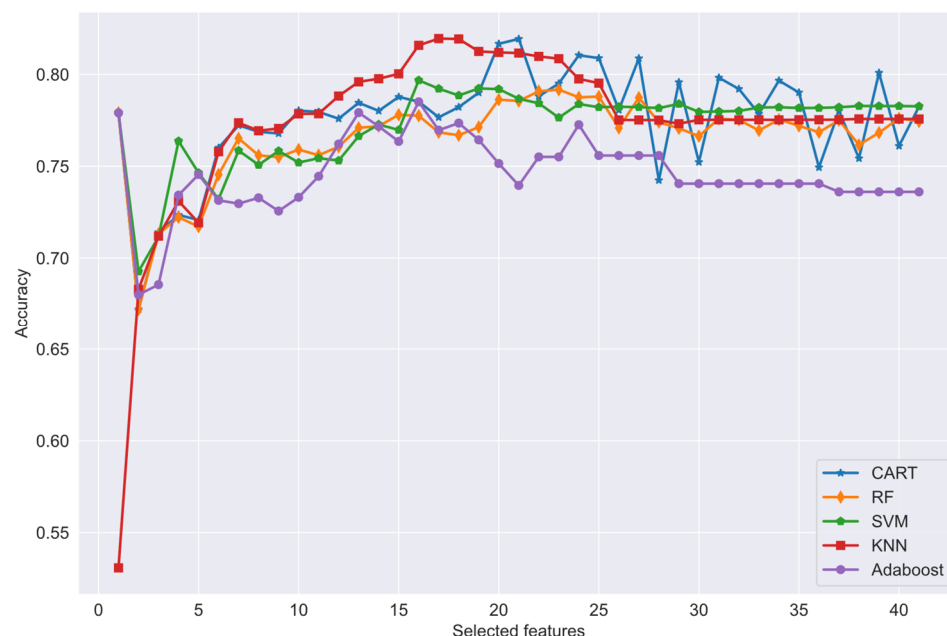


Figure 3. Effect of the number of selected features on accuracy.

The features were ordered according to their importance, with the most significant features listed first. The optimal number of features for different machine learning models was then selected in order of serial number. The specific features selected are presented in Table 7, and Table 8 illustrates the performance of the models after the features were selected. It can be observed that the features selected for different models were largely disparate, which is attributable to the differing learning abilities and complexities of the various classifiers. For instance, decision tree models (including CART and RF) tend to select features that can be most efficiently segmented, which may be directly related to the target variable or ability to achieve high information gain with a small number of segmentation steps. In contrast, SVM employs kernel functions to map the data, potentially leading to the selection of features that are most relevant to the decision boundary. Consequently, different models may select distinct features as the most crucial ones due to the distinctive characteristics of their learning algorithms. Following the application of the ERT feature selection method, there was a notable enhancement in the accuracy, precision, and f1-score of all models (in comparison with Table 6). Specifically, the CART model had the most significant improvement in recall, which increased by 6.35%. AdaBoost had the most improvement in the accuracy, precision, and recall, which went up by 4.9%, 5.49%, and 3.57%, respectively, followed by KNN (4.38%, 5.24%, and 3.22%). The RF model's performance also slightly improved to reach an accuracy of 79.17%, precision of 68.19%, and recall of 80.01%. The SVM model had the least performance improvement, with 1.42%, 1.78%, and 0.93% in accuracy, precision, and recall, respectively. The KNN model performed the best among the five models, achieving the highest accuracy (81.95%) and f1-score (82.19%).

Table 7. Selected feature number.

Model	Number
CART	11, 28, 38, 32, 25, 3, 1, 37, 33, 35, 24, 2, 26, 31, 22, 39, 40, 27, 36, 4, 7
RF	11, 28, 38, 32, 25, 3, 1, 37, 33, 35, 24, 2, 26, 31, 22, 39, 40, 27, 36, 4, 7, 23, 29
SVM (rbf)	11, 28, 38, 32, 25, 3, 1, 37, 33, 35, 24, 2, 26, 31, 22, 39
KNN	11, 28, 38, 32, 25, 3, 1, 37, 33, 35, 24, 2, 26, 31, 22, 39, 40
AdaBoost	11, 28, 38, 32, 25, 3, 1, 37, 33, 35, 24, 2, 26, 31, 22, 39

Table 8. Best performance for feature selection.

Model	Selected Top n Features	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CART	21	81.93	71.49	96.56	82.15
RF	23	79.17	68.19	96.79	80.01
SVM (rbf)	16	79.68	68.77	96.79	80.41
KNN	17	81.95	71.48	96.67	82.19
AdaBoost	16	78.51	68.79	91.71	78.62

4.5. Performance of Ensemble Models

To further improve the model performance, we used different machine learning models for integrated training, and explored the relationship between different number of estimators and accuracy, increasing the number of estimators in intervals of 5 from 5 to 200, i.e., the sample size of training increased from 5 times to 200 times; the results are shown in Figure 4. CART-Bagging, RF, Adaboost model accuracy fluctuates, KNN-Bagging, and SVM-Bagging are more stable, while KNN-Bagging model has higher accuracy. When increasing the number of estimators, with the increase in training samples, the initial number of some base estimators usually improves the accuracy significantly, this is because by integrating multiple models, the variance of a single model can be effectively reduced, and each base estimator can focus on a different aspect of the data during the training process, which makes the integrated model more robust. As the number of estimators continues to increase, the improvement in accuracy slows down, this is because as the number of estimators increases, the variance of the model on the training set has stabilized or decreased, and the contribution of additional estimators to reducing the variance becomes limited, a phenomenon often referred to as diminishing returns. After adding a certain number of estimators, the overall accuracy of the model stabilizes and reaches a convergence value. At this point, adding more estimators has little or no significant effect on the accuracy of the model. This is because the integration already benefits from sufficient diversity and generalization capabilities, and adding more estimators does not significantly improve the model's performance. Changes in the number of estimators reflect changes in the data samples, i.e., as the number of training samples increases or decreases, the overall model performance changes. Specifically, when the number of training samples is small, increasing the number of estimators significantly improves the accuracy of the model because more models can help capture different features and patterns in the data and reduce the variance of the model, thereby improving generalization. As the number of samples increases, the generalization ability of the model gradually increases, but the improvement in model performance from increasing the number of estimators gradually decreases. This is because, although increasing the estimators can continue to reduce the variance, further improvement becomes more limited as the integration is already relatively stable. When the training samples are very large, the model may already be sufficient to fit the complexity of the data, at which point increasing the number of estimators will have a very limited improvement in model accuracy because the model is already sufficiently generalized, and further increases in the number of estimators will not significantly reduce the variance of the model. In practical applications, it is necessary to balance the model performance and the use of computational resources in order to identify a number of estimators that can significantly improve accuracy without consuming excessive resources.



Figure 4. Effect of the number of estimators on accuracy.

In summary, we chose the optimal number of estimators for five integrated models, namely CART-Bagging, SVM-Bagging, KNN-Bagging, AdaBoost, and RF. The performance of these models was compared, and the results are shown in Table 9. The number of estimators chosen for the above five integrated models were 30, 30, 55, 20, and 5, respectively. When bagging ensemble was used, KNN was chosen as the evaluator with the best performance, achieving an f1-score of 82.22%, outperforming CART (80.41%) and SVM (80.53%). The AdaBoost model performed poorly, with the lowest accuracy (78.54%), precision (67.77%), and f1-score (79.35), and the CART-Bagging model had the lowest recall (94.93%) and highest accuracy (80.07%). Among all the integrated models, the KNN-Bagging model had the best performance, although accuracy was slightly lower than the CART-Bagging model (0.07%), achieving the highest precision (71.48%) and f1-score (82.22%), which was an improvement of 5.24% and 3.25%, respectively, with respect to the original KNN model, so we further tuned the parameters of KNN using BO.

Table 9. Performance comparison of ensemble models.

Model	Number of Estimators	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CART-Bagging	30	80.07	69.74	94.93	80.41
SVM-Bagging	30	79.84	68.96	96.77	80.53
KNN-Bagging	55	82.00	71.48	96.62	82.22
AdaBoost	20	78.54	67.77	95.71	79.35
RF	5	79.31	68.38	96.66	80.10

4.6. Performance of Ensemble Models after Parameter Optimization

Finally, we used BO to tune the number of neighbors K and the type of distance metrics (Euclidean and Manhattan distances) of the KNN. The range of BO parameters can be seen in Table 1 above, and the results show that the model performed best when the value of K was taken to be 2 and Euclidean distances were used. The accuracy, precision, and f1-score of the BO-KNN-Bagging model were improved by 0.48%, 0.74%, and 0.36%, respectively, relative to the untuned KNN-Bagging model. Table 10 investigates the effect of different number of estimators (5–200) on the accuracy of the BO-KNN-Bagging model, and the results show that when the number of estimators was small, the model performance was poor, achieving only 81.98% accuracy, which was even inferior to the accuracy of the KNN-

Bagging model without parameter tuning (82%). However, as the number of estimators increased, the accuracy stabilized and reached its highest value (82.48%) when the number of estimators was 100, and, as the number of estimators increased further, the accuracy fluctuated in a smaller range, which confirms the argument in Section 4.5. The performance comparison results of the proposed BO-KNN-Bagging with other machine learning models are shown in Figures 5 and 6. It can be observed that the optimized models all achieved the highest values of accuracy (82.48%), precision (72.22%), and f1-score (82.58%).

Table 10. Performance of ensemble models after parameter optimization.

Number of Estimators	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
5	81.98	71.62	96.35	82.17
50	82.46	72.21	96.35	82.55
100	82.48	72.22	96.40	82.58
150	82.46	72.20	96.40	82.56
200	82.47	72.21	96.40	82.57

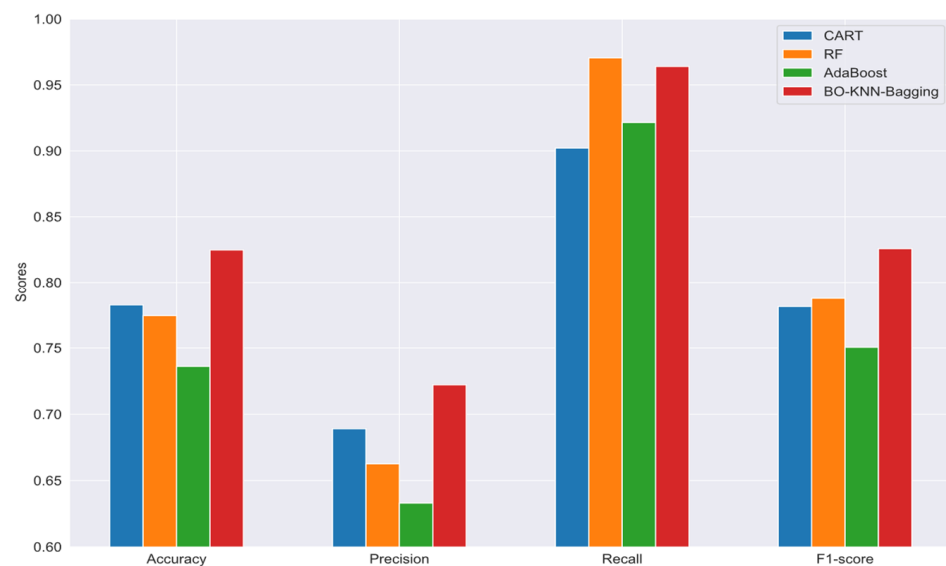


Figure 5. Performance comparison of different machine learning models.

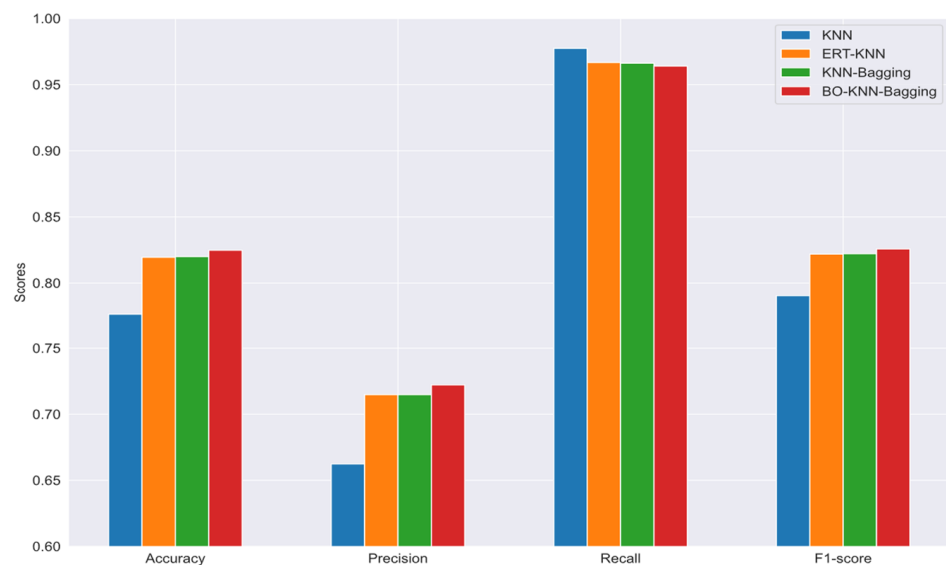


Figure 6. Performance comparison of different KNN models.

To further explore the generalization performance of the integrated model, we evaluated it using OOB error, which reflects the performance of the model on unseen data, and a lower error implies that the model has a higher accuracy on unseen data and a better generalization ability. The results are shown in Table 11, where the OOB score is the score of the model on OOB data, i.e., the prediction accuracy of the model on OOB data. It can be found that the proposed BO-KNN-Bagging model had the highest OOB score (0.9947) and the lowest OOB error (0.0053). The error was reduced by 0.0007 for the untuned KNN-Bagging model, which indicates that the proposed model achieved the highest accuracy with the lowest error on unknown data with good generalization performance.

Table 11. Comparison of generalization performance of ensemble models.

Model	OOB Score	OOB Error
CART-Bagging	0.9875	0.0125
SVM-Bagging	0.9825	0.0175
KNN-Bagging	0.9940	0.0060
BO-KNN-Bagging	0.9947	0.0053

Table 12 demonstrates the comparison of the proposed model with machine learning models and existing methods. Our model achieved 82.48% accuracy, 72.22% precision, and 82.58% f1-score, which was an improvement over the traditional KNN model by 4.91%, 5.98%, and 3.61%, respectively, and maintained a relatively high recall (96.4%). The accuracy also improved (4.64% and 7.38%) for the classical deep learning models (MLP and LSTM), which provides a significant advantage in the field of network intrusion detection. Table 13 shows the comparison of the accuracy of our model with the existing methods in the last five years, and our model had the highest accuracy, with an improvement of 3.48% and 2.18% compared with the SVM model studied by Safaldin et al. [20] and the integrated model XGBoost [36] studied by Zaryn et al. Compared with models that have been optimized for algorithmic feature selection, namely PRACO [21] and SGM [22], also demonstrated improved performance. Similarly, for some deep learning models, namely LSTM [35] and MLP [37], there was a 0.18% and 0.48% improvement in accuracy, respectively.

Table 12. Performance comparison of different models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CART	78.25	68.91	90.21	78.13
SVM (rbf)	78.26	66.98	97.70	79.48
Adaboost	73.61	63.30	92.15	75.05
MLP	77.84	66.65	97.17	79.07
LSTM	75.10	63.87	97.17	77.08
RF	77.44	66.26	97.05	78.75
KNN	77.57	66.24	97.74	78.97
ERT-KNN	81.95	71.48	96.67	82.19
KNN-Bagging	82.00	71.48	96.62	82.22
BO-KNN-Bagging	82.48	72.22	96.40	82.58

Table 13. Performance comparison of different methods.

Method	Year	Feature Selection	Accuracy (%)
SVM [20]	2021	YES	79.00
LSTM [35]	2022	YES	82.30
PRACO [21]	2022	YES	81.68
SGM [22]	2022	YES	81.73
XGBoost [36]	2023	NO	80.30
MLP [37]	2023	NO	80.00
BO-KNN-Bagging	2024	YES	82.48

5. Conclusions and Future Work

Aiming at the problem of asymmetry of information features and redundant features of datasets and asymmetry of network traffic distribution in the field of network intrusion detection, which leads to low accuracy and poor generalization of traditional machine learning detection methods in IDS, a network intrusion detection method based on bagging ensemble is proposed. Using ERT for feature selection, we explore the model performance of different machine learning models as bagging basis ensembles, and find that the KNN-Bagging ensemble model has the best performance, and then we use BO for parameter tuning of the basis ensembles KNN, and finally the basis ensembles are integrated. The results show that the BO-KNN-Bagging model achieves an accuracy of 82.48%, which is higher than traditional machine learning algorithms, and also has better performance compared with other methods. Nevertheless, there are still some limitations. The effectiveness of bagging depends on the variability among the base estimators, and although we performed random sampling to ensure diversity in the training data, the base estimators use the same model and parameters, and the performance gains of the integrated model may be limited. Also, bagging requires the training of multiple base estimators and therefore has a high computational overhead, especially for complex learning algorithms or large-scale datasets.

The next step is to construct more complex artificial neural network (ANN) models as base estimators for integrating different methods (e.g., boosting, stacking), and also to consider using oversampling techniques to make the model better learn the features of a few categories, so as to further improve the model's generalization performance and accuracy.

Author Contributions: Conceptualization, Z.Z. and S.K.; methodology, Z.Z.; software, T.X.; validation, Z.Z., S.K. and T.X.; formal analysis, Z.Z.; investigation, S.K.; resources, A.Y.; data curation, T.X.; writing—original draft preparation, Z.Z.; writing—review and editing, S.K.; visualization, T.X.; supervision, A.Y.; project administration, A.Y.; funding acquisition, A.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Hebei Province, grant number E2022209110.

Data Availability Statement: The datasets used in this paper are publicly available to everyone and can be accessed at <https://www.unb.ca/cic/datasets/nsl.html> (accessed on 15 April 2024) for NSL-KDD. Similarly, the remaining data will be available on request from the corresponding author.

Acknowledgments: We are very grateful to all the reviewers who helped to review the manuscript for valuable suggestions to improve the quality of the manuscript, and to the Symmetry Editorial department for completely maintaining a strict peer review work schedule and for timely publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wu, X.; Du, Y.; Fan, T.; Guo, J.; Ren, J.; Wu, R.; Zheng, T. Threat analysis for space information network based on network security attributes: A review. *Complex Intell. Syst.* **2023**, *9*, 3429–3468. [\[CrossRef\]](#)
2. Al-E'mari, S.; Anbar, M.; Sanjalawe, Y.; Manickam, S.; Hasbullah, I. Intrusion Detection Systems Using Blockchain Technology: A Review, Issues and Challenges. *Comput. Syst. Sci. Eng.* **2022**, *40*, 87. [\[CrossRef\]](#)
3. Tang, C.; Luktarhan, N.; Zhao, Y. SAAE-DNN: Deep learning method on intrusion detection. *Symmetry* **2020**, *12*, 1695. [\[CrossRef\]](#)
4. Alabdulwahab, S.; Moon, B. Feature selection methods simultaneously improve the detection accuracy and model building time of machine learning classifiers. *Symmetry* **2020**, *12*, 1424. [\[CrossRef\]](#)
5. Farooq, M.S.; Abbas, S.; Sultan, K.; Atta-ur-Rahman, M.A.; Khan, M.A.; Mosavi, A. A fused machine learning approach for intrusion detection system. *Comput. Mater. Continua* **2023**, *74*, 2607–2623. [\[CrossRef\]](#)
6. Tsimenidis, S.; Lagkas, T.; Rantos, K. Deep learning in IoT intrusion detection. *J. Netw. Syst. Manag.* **2022**, *30*, 8. [\[CrossRef\]](#)
7. Jemili, F.; Meddeb, R.; Korbaa, O. Intrusion detection based on ensemble learning for big data classification. *Clust. Comput.* **2024**, *27*, 3771–3798. [\[CrossRef\]](#)
8. Azam, Z.; Islam, M.M.; Huda, M.N. Comparative analysis of intrusion detection systems and machine learning based model analysis through decision tree. *IEEE Access* **2023**, *11*, 80348–80391. [\[CrossRef\]](#)

9. Zou, L.; Luo, X.; Zhang, Y.; Yang, X.; Wang, X. HC-DTTSVM: A network intrusion detection method based on decision tree twin support vector machine and hierarchical clustering. *IEEE Access* **2023**, *11*, 21404–21416. [\[CrossRef\]](#)
10. Liu, G.; Zhao, H.; Fan, F.; Liu, G.; Xu, Q.; Nazir, S. An enhanced intrusion detection model based on improved kNN in WSNs. *Sensors* **2022**, *22*, 1407. [\[CrossRef\]](#)
11. Khare, N.; Devan, P.; Chowdhary, C.L.; Bhattacharya, S.; Singh, G.; Singh, S.; Yoon, B. Smo-dnn: Spider monkey optimization and deep neural network hybrid classifier model for intrusion detection. *Electronics* **2020**, *9*, 692. [\[CrossRef\]](#)
12. Ramadan, R.A.; Emara, A.-H.; Al-Sarem, M.; Elhamahmy, M. Internet of drones intrusion detection using deep learning. *Electronics* **2021**, *10*, 2633. [\[CrossRef\]](#)
13. Donkol, A.A.E.-B.; Hafez, A.G.; Hussein, A.I.; Mabrook, M.M. Optimization of intrusion detection using likely point PSO and enhanced LSTM-RNN hybrid technique in communication networks. *IEEE Access* **2023**, *11*, 9469–9482. [\[CrossRef\]](#)
14. Louk, M.H.L.; Tama, B.A. Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. *Expert Syst. Appl.* **2023**, *213*, 119030. [\[CrossRef\]](#)
15. Saied, M.; Guirguis, S.; Madbouly, M. A comparative study of using boosting-based machine learning algorithms for IoT network intrusion detection. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 177. [\[CrossRef\]](#)
16. Shafieian, S.; Zulkernine, M. Multi-layer stacking ensemble learners for low footprint network intrusion detection. *Complex Intell. Syst.* **2023**, *9*, 3787–3799. [\[CrossRef\]](#)
17. Jaw, E.; Wang, X. Feature selection and ensemble-based intrusion detection system: An efficient and comprehensive approach. *Symmetry* **2021**, *13*, 1764. [\[CrossRef\]](#)
18. Aldallal, A.; Alisa, F. Effective intrusion detection system to secure data in cloud using machine learning. *Symmetry* **2021**, *13*, 2306. [\[CrossRef\]](#)
19. Yu, L.; Xu, L.; Jiang, X. A High-Performance Multimodal Deep Learning Model for Detecting Minority Class Sample Attacks. *Symmetry* **2023**, *16*, 42. [\[CrossRef\]](#)
20. Safaldin, M.; Otair, M.; Abualigah, L. Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 1559–1576. [\[CrossRef\]](#)
21. Kumar, P.; Shakti, S.; Datta, N.; Sinha, S.; Ghosh, P. Feature selection using PRACO method for IDS in cloud environment. *J. Intell. Fuzzy Syst.* **2022**, *43*, 5487–5500. [\[CrossRef\]](#)
22. Ghosh, P.; Alam, Z.; Sharma, R.R.; Phadikar, S. An efficient SGM based IDS in cloud environment. *Computing* **2022**, *104*, 553–576. [\[CrossRef\]](#)
23. Ye, Z.; Luo, J.; Zhou, W.; Wang, M.; He, Q. An ensemble framework with improved hybrid breeding optimization-based feature selection for intrusion detection. *Future Gener. Comput. Syst.* **2023**, *151*, 124–136. [\[CrossRef\]](#)
24. Herrera-Semenets, V.; Bustio-Martínez, L.; Hernández-León, R.; van den Berg, J. A multi-measure feature selection algorithm for efficacious intrusion detection. *Knowl.-Based Syst.* **2021**, *227*, 107264. [\[CrossRef\]](#)
25. Nazir, A.; Khan, R.A. A novel combinatorial optimization based feature selection method for network intrusion detection. *Comput. Secur.* **2021**, *102*, 102164. [\[CrossRef\]](#)
26. Akhiat, Y.; Touchanti, K.; Zinedine, A.; Chahhou, M. IDS-EFS: Ensemble feature selection-based method for intrusion detection system. *Multimed. Tools Appl.* **2024**, *83*, 12917–12937. [\[CrossRef\]](#)
27. Khammassi, C.; Krichen, S. A NSGA2-LR wrapper approach for feature selection in network intrusion detection. *Comput. Netw.* **2020**, *172*, 107183. [\[CrossRef\]](#)
28. Yang, Z.; Liu, Z.; Zong, X.; Wang, G. An optimized adaptive ensemble model with feature selection for network intrusion detection. *Concurr. Comput. Pract. Exp.* **2023**, *35*, e7529. [\[CrossRef\]](#)
29. Mohammad, R.M.A.; Alsmadi, M.K. Intrusion detection using Highest Wins feature selection algorithm. *Neural Comput. Appl.* **2021**, *33*, 9805–9816. [\[CrossRef\]](#)
30. Qian, H.; Zhang, X.; Zhang, C.; Jiang, C. A novel cyber intrusion detection model based on improved hybrid sampling. *Trans. Inst. Meas. Control* **2023**, *45*, 2903–2913. [\[CrossRef\]](#)
31. Jiang, K.; Wang, W.; Wang, A.; Wu, H. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access* **2020**, *8*, 32464–32476. [\[CrossRef\]](#)
32. Jung, I.; Ji, J.; Cho, C. EmSM: Ensemble mixed sampling method for classifying imbalanced intrusion detection data. *Electronics* **2022**, *11*, 1346. [\[CrossRef\]](#)
33. Zhang, Y.; Liu, Q. On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples. *Future Gener. Comput. Syst.* **2022**, *133*, 213–227. [\[CrossRef\]](#)
34. Liu, R.; Ma, W.; Guo, J. A multi-constraint transfer approach with additional auxiliary domains for IoT intrusion detection under unbalanced samples distribution. *Appl. Intell.* **2024**, *54*, 1179–1217. [\[CrossRef\]](#)
35. Esmaeili, M.; Goki, S.H.; Masjidi, B.H.K.; Sameh, M.; Gharagozlou, H.; Mohammed, A.S. ML-DDoSnet: IoT intrusion detection based on denial-of-service attacks using machine learning methods and NSL-KDD. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 8481452. [\[CrossRef\]](#)
36. Zaryn, G.; Waleed, F.; Xin-Wen, W.; Soundararajan, E.; Maria, B.; Franklin, M.; Alicia, D. Comparative Analysis of Machine Learning Techniques for IoT Anomaly Detection Using the NSL-KDD Dataset. *Int. J. Comput. Sci. Netw. Secur.* **2023**, *23*, 46–52. [\[CrossRef\]](#)

37. Lee, G.-C.; Li, J.-H.; Li, Z.-Y. A Wasserstein Generative Adversarial Network–Gradient Penalty-Based Model with Imbalanced Data Enhancement for Network Intrusion Detection. *Appl. Sci.* **2023**, *13*, 8132. [\[CrossRef\]](#)
38. Sarnovsky, M.; Paralic, J. Hierarchical intrusion detection using machine learning and knowledge model. *Symmetry* **2020**, *12*, 203. [\[CrossRef\]](#)
39. Alotaibi, F.M. Network Intrusion Detection Model Using Fused Machine Learning Technique. *Comput. Mater. Contin.* **2023**, *75*, 2479–2490. [\[CrossRef\]](#)
40. Elnakib, O.; Shaaban, E.; Mahmoud, M.; Emara, K. EIDM: Deep learning model for IoT intrusion detection systems. *J. Supercomput.* **2023**, *79*, 13241–13261. [\[CrossRef\]](#)
41. Wang, Z.; Liu, Y.; He, D.; Chan, S. Intrusion detection methods based on integrated deep learning model. *Comput. Secur.* **2021**, *103*, 102177. [\[CrossRef\]](#)
42. Praveena, V.; Vijayaraj, A.; Chinnasamy, P.; Ali, I.; Alroobaea, R.; Alyahyan, S.Y.; Raza, M.A. Optimal deep reinforcement learning for intrusion detection in UAVs. *Comput. Mater. Contin.* **2022**, *70*, 2639–2653. [\[CrossRef\]](#)
43. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [\[CrossRef\]](#)
44. Ngo, G.; Beard, R.; Chandra, R. Evolutionary bagging for ensemble learning. *Neurocomputing* **2022**, *510*, 1–14. [\[CrossRef\]](#)
45. Tüysüzoğlu, G.; Birant, D.; KIRANOĞLU, V. Temporal bagging: A new method for time-based ensemble learning. *Turk. J. Electr. Eng. Comput. Sci.* **2022**, *30*, 279–294. [\[CrossRef\]](#)
46. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [\[CrossRef\]](#)
47. Liang, Y.; Zhang, S.; Qiao, H.; Yao, Y. iPromoter-ET: Identifying promoters and their strength by extremely randomized trees-based feature selection. *Anal. Biochem.* **2021**, *630*, 114335. [\[CrossRef\]](#)
48. Zhang, S. Challenges in KNN classification. *IEEE Trans. Knowl. Data Eng.* **2021**, *34*, 4663–4675. [\[CrossRef\]](#)
49. Lahmiri, S. Integrating convolutional neural networks, kNN, and Bayesian optimization for efficient diagnosis of Alzheimer’s disease in magnetic resonance images. *Biomed. Signal Process. Control* **2023**, *80*, 104375. [\[CrossRef\]](#)
50. Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature* **2015**, *521*, 452–459. [\[CrossRef\]](#)
51. Tavallae, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
52. Yang, Y.; Zheng, K.; Wu, B.; Yang, Y.; Wang, X. Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. *IEEE Access* **2020**, *8*, 42169–42184. [\[CrossRef\]](#)
53. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2020**, *17*, 168–192. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.