applied sciences

MDPI

Systematic Review

Feature Engineering and Model Optimization Based Classification Method for Network Intrusion Detection

Yujie Zhang and Zebin Wang *

School of Electrical and Control Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China
* Correspondence: 210611030@sust.edu.cn

Abstract: In light of the escalating ubiquity of the Internet, the proliferation of cyber-attacks, coupled with their intricate and surreptitious nature, has significantly imperiled network security. Traditional machine learning methodologies inherently exhibit constraints in effectively detecting and classifying multifarious cyber threats. Specifically, the surge in high-dimensional network traffic data and the imbalanced distribution of classes exacerbate the predicament of ideal classification performance. Notably, the presence of redundant information within network traffic data undermines the accuracy of classifiers. To address these challenges, this study introduces a novel approach for intrusion detection classification which integrates advanced techniques of feature engineering and model optimization. The method employs a feature engineering approach that leverages mutual information maximum correlation minimum redundancy (mRMR) feature selection and synthetic minority class oversampling technique (SMOTE) to process network data. This transformation of raw data into more meaningful features effectively addresses the complexity and diversity inherent in network data, enhancing classifier accuracy by reducing feature redundancy and mitigating issues related to class imbalance and the detection of rare attacks. Furthermore, to optimize classifier performance, the paper applies the Optuna method to fine-tune the hyperparameters of the Catboost classifier, thereby determining the optimal model configuration. The study conducts binary and multi-classification experiments using publicly available datasets, including NSL_KDD, UNSW-NB15, and CICIDS-2017. Experimental results demonstrate that the proposed method outperforms traditional approaches regarding accuracy, recall, precision, and F-value. These findings highlight the method's potential and performance in network intrusion detection.

Keywords: intrusion detection; machine learning; mRMR; SMOTE; catboost

check for updates

Citation: Zhang, Y.; Wang, Z. Feature Engineering and Model Optimization Based Classification Method for Network Intrusion Detection. *Appl. Sci.* **2023**, *13*, 9363. https://doi.org/10.3390/app13169363

Academic Editor: Mohamed Benbouzid

Received: 3 July 2023
Revised: 2 August 2023
Accepted: 12 August 2023
Published: 18 August 2023

Copyright: © 2023 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

With the network security risk increasing, implementing effective intrusion detection mechanisms has emerged as a critical strategy for safeguarding computer systems and network security [1–4]. Conventional intrusion detection techniques primarily depend on pre-established rules or recognized attack attributes [5], such as employing string pattern matching to identify attack signatures and ascertain the presence of an intrusion. Nonetheless, these traditional methods exhibit limitations in detecting novel and unidentified attacks, relying heavily on expert experience [6,7]. In contrast, machine learning techniques present notable benefits in intrusion detection. By acquiring knowledge from network data and attack samples, these methods enable the identification of unfamiliar malicious behaviors and signatures. Prior research [8,9] has employed intelligent intrusion detection systems founded on machine learning approaches, which can detect attacks without any previous information. Subsequent investigations [10–13] delve into the examination and comparison of various datasets frequently employed in network intrusion detection systems. The network data exhibits high dimensionality and category imbalance, posing challenges for traditional classification algorithms in extracting significant features. Moreover, the category imbalance further exacerbates the issue by causing low detection rates in models for certain categories.

To address the challenges posed by high-dimensional data and class imbalance, researchers have made extensive efforts. Among these approaches, a combination of PCA and LDA feature extraction methods was explored in literature [14] to map the original feature set onto a lower-dimensional space, subsequently applied to the classifier. However, an alternative feature extraction framework based on stratification and dynamics was proposed in the literature [15]. While these methods have shown improvements following feature extraction, it is essential to note that feature extraction primarily focuses on secondary features after dimensionality reduction. In network intrusion detection, preserving the physical meaning of features in the original dataset post-dimensionality reduction becomes imperative.

In comparison, feature selection exhibits certain advantages within the aforementioned domains. Reference [16] employed the linear correlation coefficient (FGLCC) algorithm and the cuttlefish algorithm (CFA) for data feature selection. Furthermore, a novel cross-correlation-based feature selection (CCFS) method was introduced in literature [17], which was validated across various classifiers. Conversely, to address the issue of dataset imbalance, literature [18] proposed a method that leverages the synthetic minority sampling algorithm (SMOTE) to balance the dataset. Additionally, literature [19] applied a combination of SMOTE, under-sampling clustering algorithms, and Gaussian mixture models for dataset manipulation. Numerous studies have indicated that performing feature engineering before data classification is essential for achieving optimal results.

Meanwhile, within intrusion detection classification, the GBDT classifier exhibits distinct advantages over other classifiers. However, literature [20] analyzed the current implementation of GBDT, highlighting its strengths and weaknesses. Moreover, the effectiveness of CatBoost in classification and regression tasks was revealed. Nevertheless, this analysis also raises concerns regarding hyperparameter sensitivity and the criticality of hyperparameter tuning.

Based on the analysis and study of references [21,22], this paper comprehensively examines various intrusion detection techniques, providing valuable references for the author's work. It demonstrates that in the current Internet environment, on the one hand, the increasing popularity of the Internet has led to a rise in the number and complexity of network attacks, making emerging threats critical [23]. On the other hand, various challenges and opportunities arise from applying AutoML in intrusion detection [24]. Considering the current state of scholarly research, to address the aforementioned issues, feature engineering and model optimization have become pivotal approaches [25]. The primary objective of this paper is to propose an efficient classification model tailored to handle high-dimensional and imbalanced data, thereby addressing the deficiencies in network intrusion detection research and ensuring the security and integrity of computer networks. The primary contributions can be summarized as follows:

Perform a comprehensive feature engineering process on the original dataset, encompassing data cleansing, one-hot encoding, and normalization techniques. Additionally, employ the mRMR algorithm, which utilizes mutual information, for effective feature selection. Mitigate the class imbalance issue by employing the SMOTE algorithm, thus enhancing the usability of the dataset.

Enhance the catboost model by leveraging the Optuna framework to optimize its hyperparameters, thereby improving the overall performance of the model.

Employ diverse machine learning algorithms to conduct binary and multi-classification for intrusion detection tasks on three datasets: NSL_KDD, UNSW-NB15, and CICIDS-2017. Thoroughly evaluate and analyze the resulting output to determine the classification model that demonstrates the most optimal performance.

2. Materials and Methods

This section provides an overview of the network intrusion detection classification method based on feature engineering and model optimization. The structural framework of the paper is divided into four main parts: data pre-processing, feature engineering,

training and validation of the classification models, and evaluation and analysis of the experimental results. The framework is depicted in Figure 1. The main objective of this study is to enhance the performance of network intrusion detection models through the adoption of feature engineering and model optimization methods. Firstly, we conduct data preprocessing on the NSL_KDD, UNSW-NB15, and CICIDS-2017 datasets, including data cleansing, one-hot encoding, and min-max normalization, to improve data usability and robustness. Subsequently, the preprocessed data undergoes a dual strategy of feature selection based on mrmr and feature engineering based on SMOTE. This approach enables the selection of relevant and non-redundant features and effectively addresses the issue of class imbalance, thereby bolstering the model's ability to handle the intricacies and diversities of network data. The feature-engineered data is then fed into the CatBoost model, which undergoes hyperparameter optimization using the Optuna framework, followed by training and classification. Evaluation metrics such as accuracy, recall, precision, and F1-score are employed to validate and assess the results.

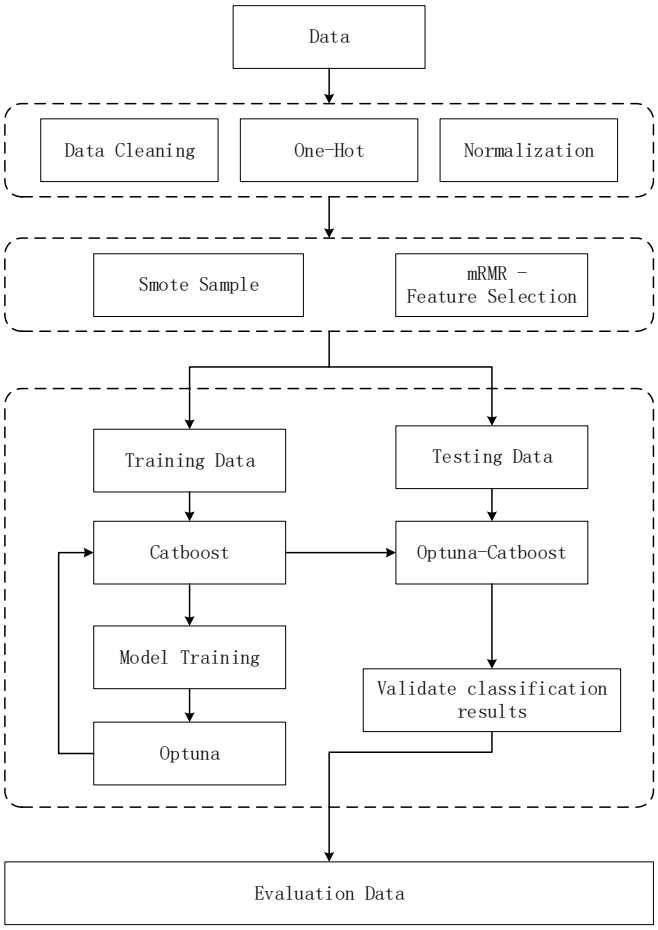

Figure 1. Schematic diagram of the structural framework of the study.

(1) The raw data is subjected to cleaning, normalization, and type conversion in the pre-processing stage to ensure data consistency and usability.
(2) The feature engineering process involves further processing the pre-processed dataset using the mRMR algorithm. This algorithm selects highly relevant features, reduces redundant information, and enhances the representation of the classification model. Additionally, the SMOTE algorithm is employed to generate new minority class samples, addressing the issue of class imbalance, and improving the overall class distribution of the dataset.
(3) The next step involves training and validating the classification model using the feature-engineered dataset. The dataset is divided into a training set and a test set, with

a ratio of 7:3. The Catboost model is optimized using the Optuna framework, which selects the best hyperparameter configuration to enhance the model generalization ability and performance.

(4) Finally, the results are evaluated and analyzed. Performance metrics such as accuracy, precision, recall, and F-value are calculated to assess the classification performance of the model on the test set. These metrics provide insights into the effectiveness of the proposed approach.

2.1. Data Preprocessing

Due to the presence of missing data, duplicate data, and character data in the dataset, it is essential to address these factors to mitigate their adverse impact and improve the overall data quality. This can be achieved through pre-processing techniques to reduce noise and anomalies, rendering the data more suitable for training and analysis with machine learning models. The following steps were implemented to accomplish this:

(1) Data cleansing: Data cleansing is essential in the data preprocessing pipeline. It involves scrutinizing the raw data to identify and rectify errors, missing values, duplicates, and inconsistencies [26]. By eliminating low-quality data, this process ensures that the analysis results remain unaffected by irrelevant or flawed information. In this study, specific cleansing procedures were undertaken, as outlined in Table 1, which comprehensively captures the different aspects targeted during the data cleansing process.

Table 1. Data to be cleaned.

Dataset	Feature Column	Meaning
NSK-KDD	num_outbound_cmds	The feature has zero values in the dataset and provides no information for the classification task
	is_host_login	Indicates whether the host login is successful or not, no contribution to the classification task
	land	This feature indicates whether the source IP address and the destination IP address are the same, and this feature has no effect on the classification task
UNSW-NB15	id	The feature is only an identifier and has no practical meaning for the classification task
	Srcip dstip	These features represent the source and target IP addresses, and for the classification task of intrusion detection, the IP addresses alone have no direct impact
CICIDS2017	Source IP Destination IP	These features represent source and target IP addresses and have no impact on the classification task of intrusion detection
	Timestamp	This feature represents the timestamp of the stream and has no effect on the classification task
	There are dirty Nan and Inf data in the dataset (distributed in columns 15 and 16) that need to be removed	

(2) Feature one-hot encoding: The purpose of feature one-hot encoding is to transform non-numeric categorical features into a numerical representation that can be processed by computer-based algorithms [27]. Machine learning algorithms rely on mathematical models and statistical methods, which typically operate on numeric data. However, categorical features consist of discrete symbols or labels that lack direct numerical significance. To address this, we employ one-hot encoding to expand the discrete feature values into a Euclidean space, thereby enabling the encoding of data features. Each possible feature value corresponds to a new binary variable in this process. A new binary variable is created for each feature, wherein the binary bit associated with the feature is set to 1, while the remaining binary bits are set

to 0. Consequently, the transformation of categorical features into numeric values is accomplished.

(3) Data normalization: This technique converts the entire range of values of a set of features into a predetermined range. Often, the range of data values can vary greatly between features, which can cause the training process of machine learning algorithms to suffer [28]. For example, one feature may have a range of values of [0, 1000], while another may only have a range of [0, 1]. In this case, without normalization, the feature with the larger range of values will take on more weight in the training of the algorithm, so min-max normalization is introduced to map all the data to the interval [0, 1], thus speeding up the convergence of the model and improving the accuracy of the classification results, as follows:

$$X' = \frac{X - Min}{Max - Min} \quad (1)$$

where Min is the minimum value of the feature, Max is the maximum value of the feature and X′ is the normalized feature value.

The pre-processing steps mentioned above, including data cleansing, one-hot encoding, and data normalization, were applied separately to the three datasets used in this paper: NSL-KDD, UNSW-NB15, and CICIDS-2017. Each dataset underwent these pre-processing steps to ensure consistency and improve the data quality before further analysis and model training.

- Numerical processing of NSL-KDD datasets

The features in columns 2, 3, and 4 of the NSL-KDD dataset (corresponding to "protocol_type", "service," and "flag", respectively) underwent unique encoding in this study. Specifically, the "protocol_type" feature was mapped as a 3-dimensional feature, the "service" feature was mapped as a 70-dimensional feature, and the "flag" feature was encoded accordingly. These encoded features were appended to the original data. As a result, the dimensionality of the original data increased from 41 to 118 after numerical processing, considering the removal of num_outbound_cmds, is_host_login, and land features, as they were irrelevant to determining abnormal data. This resulted in a total of 115 dimensions for the processed dataset. Table 2 illustrates the outcome of the numerical processing of the dataset.

Table 2. NSL-KDD after data processing.

	0	**1**	**2**	...	**113**	**114**	**115**	**Label**
0	3.47×10^{-5}	2.07×10^{-4}	0	...	0	1	0	normal
1	0	3.18×10^{-7}	0	...	0	1	0	saint
2	1.73×10^{-5}	0	1.11×10^{-5}	...	0	0	0	mscan
...	...	...	...	...	...	...	...	...
125,968	0	8.12×10^{-7}	6.22×10^{-3}	...	0	1	0	phf
125,969	3.47×10^{-5}	6.33×10^{-6}	2.88×10^{-3}	...	0	1	0	sqlattack

- UNSW_NB15 Numerical processing of data sets

The UNSW_NB15 dataset underwent unique coding for the features in columns 3, 4, and 5 (corresponding to "proto", "service", and "state", respectively). In this study, these three columns of features were uniquely encoded. The "protocol type" feature was mapped as a 3-dimensional feature, the "proto" feature was mapped as a 131-dimensional feature, the "service" feature was mapped as a 12-dimensional feature, and the "state" feature was mapped as a 12-dimensional feature. The "srcip" and "dstip" columns in the feature dataset represent IP addresses, but since the judgment of data abnormality does not

depend on IP addresses, they were not included in the coding process. Table 3 illustrates the outcome of the numerical processing of the dataset.

Table 3. UNSW-NB15 after data processing.

	0	**1**	**2**	...	**185**	**186**	**187**	**Label**
0	1.67×10^{-7}	9.39×10^{-5}	0	...	1.69×10^{-2}	3.27×10^{-2}	0	Normal
1	5.00×10^{-8}	9.39×10^{-5}	0	...	1.69×10^{-2}	1.63×10^{-2}	0	Normal
2	4.85×10^{-3}	8.45×10^{-4}	7.26×10^{-4}	...	2.54×10^{-1}	0	0	DoS
...	...	...	...	...	...	...	...	...
175,337	0	1.84×10^{-2}	1.78×10^{-3}	...	3.38×10^{-2}	1.63×10^{-2}	0	Normal
175,338	3.47×10^{-5}	2.39×10^{-2}	8.45×10^{-4}	...	0	0	0	Normal

- CICIDS2017 Numerical processing of datasets

In the CICIDS2017 dataset, there are a total of 84 features and 1 tag. The features include information such as "Source IP", "Destination IP", and "Timestamp." However, based on experience and domain knowledge, it is determined that these three features are not relevant to the final detection target. Therefore, they were processed and removed from the dataset. Table 4 illustrates the outcome of the numerical processing of the dataset.

Table 4. CICIDS2017 after data processing.

	0	**1**	**2**	...	**72**	**73**	**74**	**Label**
0	1.22×10^{-3}	6.93×10^{-1}	3.33×10^{-5}	...	0	6.92×10^{-1}	6.92×10^{-2}	DoS Hulk
1	8.08×10^{-4}	5.08×10^{-4}	0	...	0	0	0	BENIGN
2	6.97×10^{-1}	2.13×10^{-6}	4.76×10^{-6}	...	0	0	0	BENIGN
...	...	...	...	...	...	...	...	...
1,977,047	9.16×10^{-2}	3.83×10^{-7}	0	...	0	0	0	PortScan
1,977,048	1.22×10^{-3}	1.51×10^{-4}	4.76×10^{-6}	...	0	0	0	DoS Hulk

2.2. Mutual Information-Based Maximum Feature Minimum Redundancy (mRMR) Feature Selection

The pre-processed dataset encounters the challenge of high dimensionality in the feature space, which can significantly increase algorithmic training and prediction time complexity and susceptibility to overfitting [29]. Considering the intricate non-linear relationships in network traffic feature data and the diverse patterns exhibited by different types of intrusions across multiple features, the mRMR algorithm, based on mutual information, effectively captures these non-linear relationships and quantifies the correlation between features and target variables. By utilizing information theory concepts such as information entropy, information gain, and mutual information, the MRMR algorithm assesses the informative value of features for classification purposes. It evaluates the classification performance by measuring the magnitude of the mRMR value between features and class labels [30]. The calculation of the mRMR value is as follows:

According to the definition of mutual information, when variables X and Y are completely statistically independent, there is no information common between the two variables. Conversely, when the degree of dependence between the two variables is higher, the value of mutual information is greater. The amount of information shared is greater. The value of mutual information $I(X, Y)$ is obtained according to Equation.

$$I(X, Y) = \sum_{x,y} p(x, y) log \frac{p(x, y)}{p(x)p(y)} \quad (2)$$

Maximum Relevance: The mRMR algorithm computes the level of correlation between each feature and the target variable. It assigns higher priority to features with stronger correlations with the target variable.

$$\max D(S, c), D = \frac{1}{|S|}\sum_{f_i \in S} I(f_i, c) \tag{3}$$

Minimum Redundancy: The mRMR algorithm evaluates the level of redundancy among features. It emphasizes selecting features that have minimal redundancy with the already selected features. This ensures that the selected feature subset exhibits a low correlation among its features and avoids including redundant information.

$$\min R(S), R = \frac{1}{|S|^2}\sum_{f_i f_j \in S} I(f_i, f_j) \tag{4}$$

The final mRMR value is computed by considering both maximum relevance and minimum redundancy criteria. Features are ranked in descending order of their scores, and features with high scores are progressively added to the feature subset. This approach ensures that features with high relevance to the target variable and minimal redundancy with the existing feature set are included in the subset.

$$max J(D, R), J = D - R \tag{5}$$

where S is the feature set; f_i is the feature; c is the target category; $I(f_i, c)$ is the mutual information between the feature and the target category C; $I(f_i, f_j)$ is divided into the information between the feature and the feature.

The mRMR value combines the correlation between features and the target variable with the degree of redundancy between features. Its purpose is to select a subset of features that are both highly informative and unique. The coefficient ranges from 0 to 1, where a higher absolute value indicates a stronger correlation between the two variables. The strength of the correlation is typically determined based on the following Table 5.

Table 5. Correlation Strength Correspondence Table.

Range of mRMR Thresholds	Correlation Strength
0.0~0.2	Very weak or no correlation
0.2~0.4	Weak correlation
0.4~0.6	Moderate correlation
0.6~0.8	Strongly related
0.8~1.0	Extremely strong correlation

The mRMR values of each feature were calculated by averaging the mRMR values obtained from multiple extractions of the dataset used in the experiments conducted for this paper. The results are presented in the following Table 6.

Table 6. Characteristic columns for each threshold interval.

Dataset	Features Extracted from Each Threshold mRMR Feature Column				
	0.0~0.2	**0.2~0.4**	**0.4~0.6**	**0.6~0.8**	**0.8~1.0**
NSL-KDD	98, 40, 7, 57, 112, 42, 86, 83, 111, 19. . .	65, 35, 34, 36, 22, 108, 88, 41, 23	21, 116, 33, 29, 8, 10, 28, 14, 13	37, 27, 26, 39, 38, 5, 25, 6, 1, 32	2, 3, 31, 30, 20, 4, 24, 9
UNSW-NB15	39, 46, 118, 35, 34, 137, 48, 130, 154. . .	15, 23, 2, 16, 12, 11, 18, 19, 17, 20, 27, 36, 26	29, 8, 10, 28, 33, 14, 7, 31, 3, 30, 13, 22, 21, 37	5, 25, 6, 1, 32, 38	9, 24, 9
CICIDS2017	56, 60, 31, 32, 58, 57, 44, 30, 71, 48, 43. . .	27, 24, 77, 74, 69, 76, 11, 62, 2, 26, 25, 64, 3, 28, 22, 9, 17, 13	21, 16, 35, 37, 15, 20, 36, 53, 8, 34, 55, 23, 1, 14	18, 6, 67, 39, 10, 4, 63, 12, 66, 54, 5, 65	42, 40, 41, 52

2.3. Smote-Based Data Equalization

In the case of class imbalance, where the number of samples from the minority class is significantly smaller than the majority class, there is a risk of classifier bias towards the majority class, leading to performance issues [31]. To address this, minority class oversampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique) can balance the class distribution and improve classifier performance.

SMOTE sampling operates by interpolating between samples from the minority class to generate new samples. The underlying principle of SMOTE is as follows: For each sample in the minority class, its K nearest neighbors (typically K = 5) are identified using a distance metric such as Euclidean distance [32]. A new sample is then created by randomly selecting a point along the line connecting the minority class sample and one of its K nearest neighbors. The principle is shown in Figure 2. The formula used for generating the new sample is as follows:

$$\begin{aligned} X_{new} &= X + rand(0,1) \times (X_{near} - X) \\ &= [1 - rand(0,1)] \times X + rand(0,1) \times X_{near} \end{aligned} \tag{6}$$

where X_{new} denotes a final synthetic sample, X denotes a minority class sample of the input, X_{near} denotes a nearest neighbour sample of the selected X, rand(0,1) is a random number between 0 and 1.

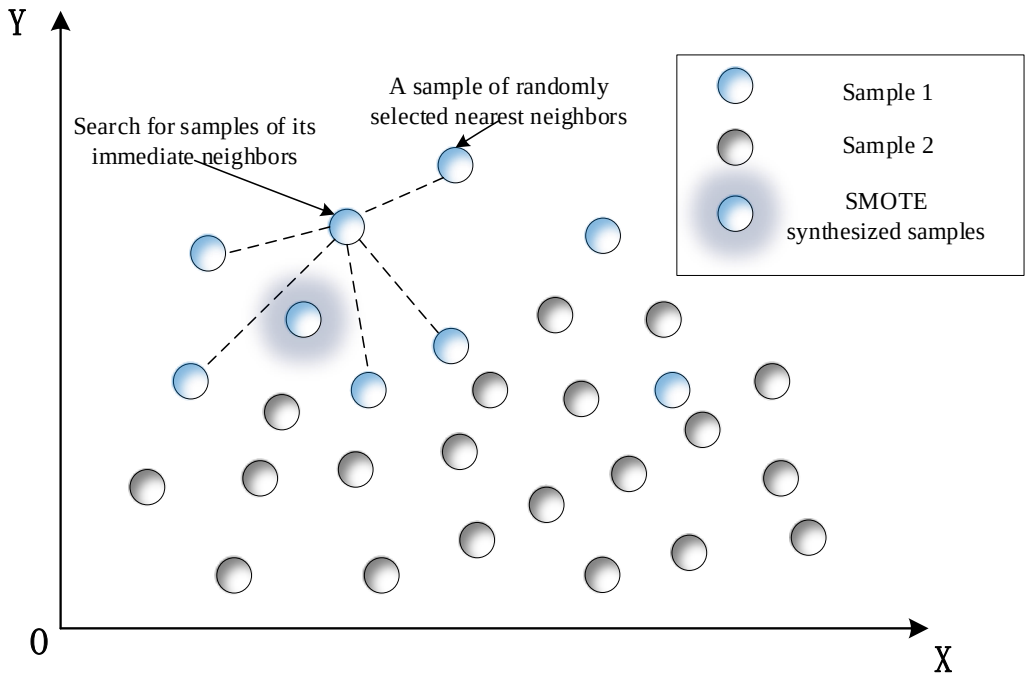

Figure 2. Schematic diagram of SMOTE algorithm.

In SMOTE sampling, the selection of the number of nearest neighbor samples is crucial. If the number is too small, it may generate overly similar synthetic samples, thereby impacting the classifier's generalization ability. On the other hand, selecting too many nearest neighbor samples may lead to the generation of complex synthetic samples, increasing the computational complexity of the classifier. Hence, in practical applications, choosing an appropriate number of nearest neighbor samples is necessary based on specific circumstances.

This paper applies the SMOTE algorithm to the three datasets used in the experiments to achieve balanced calculation. Consequently, the proportions of each category in the datasets are altered, as depicted in Figures 3–5.

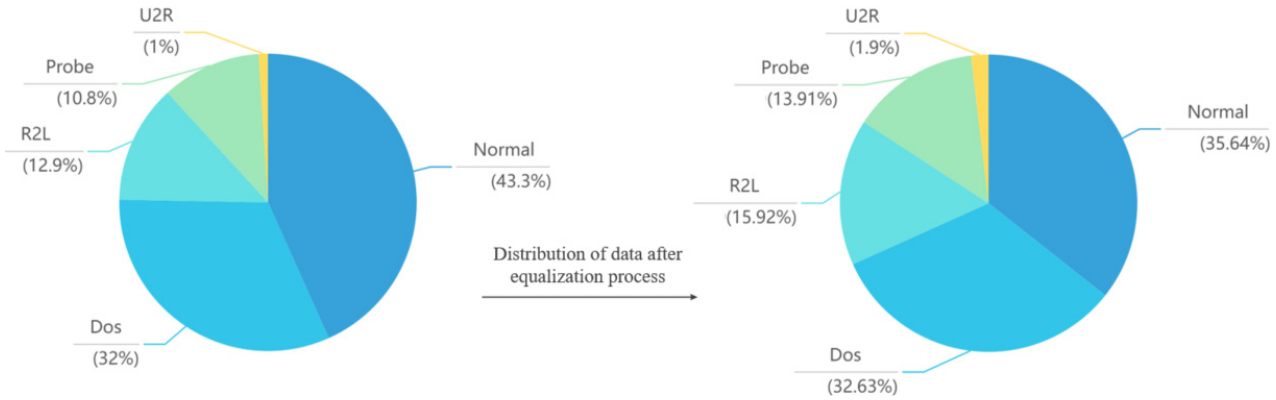

Figure 3. Comparison of data distribution before and after NSL-KDD balancing.

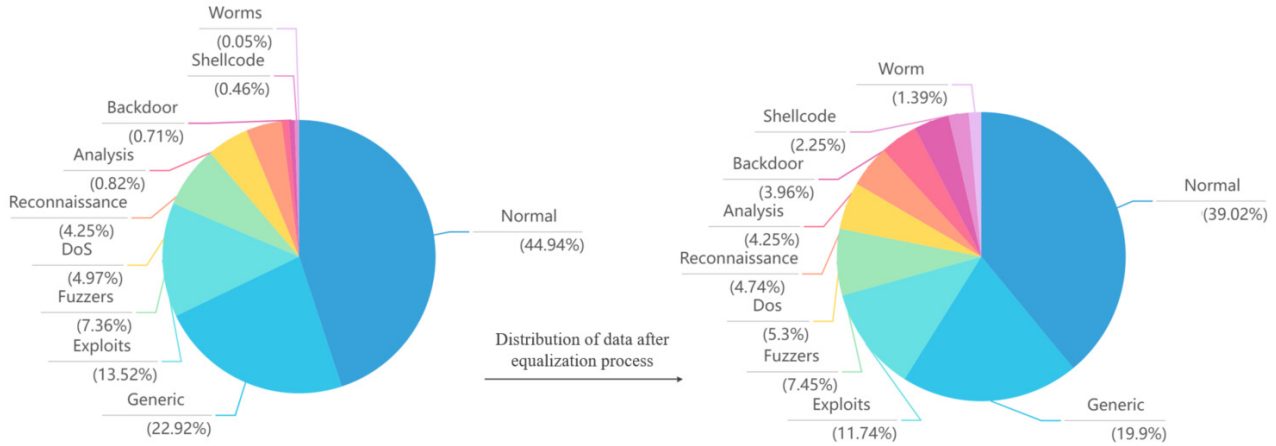

Figure 4. Comparison of data distribution before and after UNSW-NB15 balancing.

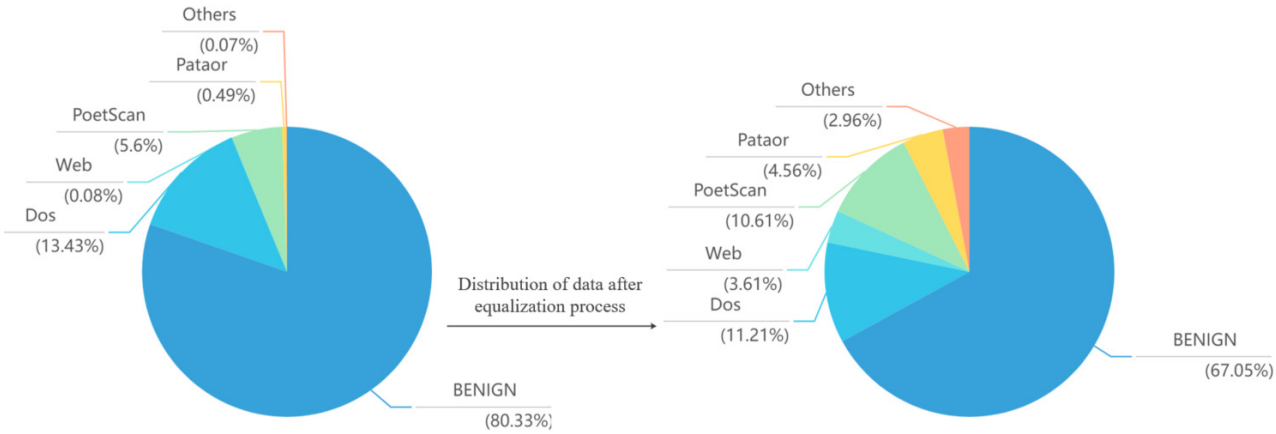

Figure 5. Comparison of data distribution before and after CICIDS2017 balancing.

2.4. Catboost Model Based on Optimisation of Optuna Hyperparameters

CatBoost is an advanced machine learning algorithm based on Gradient Boosting Decision Trees (GBDT). It effectively addresses the challenges posed by large feature spaces and class imbalance through gradient optimization techniques. The algorithm utilizes a fully symmetric tree as its base learner, as illustrated in Figure 6 [33]. Unlike a typical decision tree, a fully symmetric tree ensures that internal nodes at the same depth employ identical features and feature thresholds for splitting. Consequently, a fully symmetric tree can be represented as a decision table with 2^d entries, where d represents the number of

levels in the tree [34]. This balanced structure enhances processing speed and improves feature handling compared to a standard decision tree.

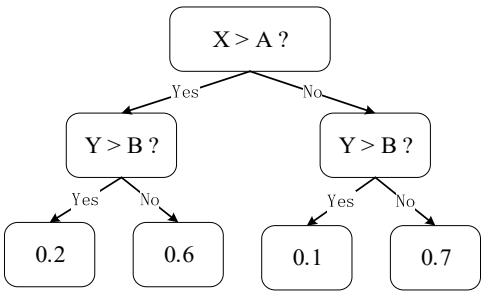

Figure 6. Schematic diagram of the structure of the tree.

In the domain of network intrusion detection, certain classes (intrusions) often suffer from underrepresentation. When confronted with large-scale datasets in network intrusion detection, employing the CatBoost algorithm as a classifier offers inherent advantages in handling high-dimensional data, addressing the class imbalance, managing classification features, facilitating automatic feature scaling, and ensuring computational efficiency. CatBoost excels in mitigating sample imbalance through the utilization of weighted loss functions and class weight adjustment, thereby enhancing classification accuracy for minor classes. It mitigates the risk of overfitting by introducing random statistical alignment during model training, improving generalization performance. By adeptly tackling the challenges associated with network intrusion detection datasets and delivering efficient classification capabilities, CatBoost proves to be a valuable choice for this task.

CatBoost is built upon the foundation of a gradient-boosting tree. However, it deviates from the conventional approach by eschewing a greedy target statistics-based method for splitting nodes when handling category features. Instead, CatBoost considers the prior distribution term during the calculation of node gain. This approach effectively mitigates the impact of low-frequency features and noise in the category variables on the construction of decision trees.

$$\mathrm{x}_{i,k} = \frac{\sum_{j=1}^{P-1}\left[x_{\sigma_{j,k}} = x_{\sigma_{p,k}}\right] \times Y_j + a \times P}{\sum_{j=1}^{P-1}\left[x_{\sigma_{j,k}} = x_{\sigma_{p,k}}\right] + a} \tag{7}$$

where σ_j is the j th data; $\mathrm{x}_{i,k}$ denotes the , k th column of discrete features of the i th row of data in the training set; a is a prior weight; and P is the prior distribution term.

Optuna is a framework for automated hyperparameter optimization, which efficiently explores the hyperparameter space to discover the optimal combination of hyperparameters [35]. When it comes to the CatBoost algorithm, there are several hyperparameters that can be fine-tuned, including learning rate, number of trees, and depth. Manually tuning these hyperparameters can be time-consuming and requires expertise in the field. By leveraging Optuna, the model's robustness can be enhanced through the optimization of hyperparameters, resulting in a selection of hyperparameter combinations that exhibit excellent performance across different environments [36]. The utilization of Optuna to optimize CatBoost facilitates improved model performance, generalization, and interpretability, ultimately yielding superior hyperparameters for intricate tasks like intrusion detection. This process unfolds through the following steps:

1. Define hyperparameter space:

Before employing Optuna for hyperparameter optimization, it is essential to establish the search space for the hyperparameters. This entails specifying a range or distribution of feasible values for each hyperparameter. The following Table 7 presents an overview of the hyperparameters in CatBoost and their respective details:

Table 7. Parameters of the search and related information.

Hyperparameters	Type and Value Range
learning_rate	Floating point type, [0, 1]
max_depth	Integer type, [1, 16]
iterations	Integer type
min_data_in_leaf	Integer type, [1, 100]
l2_leaf_reg	Floating point type, [0, 10]

2. Define the objective function:

Define an objective function that inputs hyperparameters and outputs an evaluation metric. The output of the objective function will be utilized to assess the quality of the hyperparameter combination and conduct an optimization search. In this study, accuracy is employed as the objective function for parameter optimization.

3. Hyperparametric Optimization Search:

Optuna employs a data structure called 'Trials' to record each iteration's hyperparameter values and evaluation outcomes. During each iteration, a fresh hyperparameter combination is chosen for evaluation. For each selected combination, the objective function is invoked to assess its performance and store the evaluation results in the Trials data structure. The following hyperparameter combination attempt is determined based on the historical hyperparameter values and the evaluation outcomes of the objective function.

4. Iterative process:

Multiple iterations are conducted by systematically selecting new hyperparameter combinations, evaluating the objective function, and updating the model and strategy. This iterative process aims to identify the optimal combination of hyperparameters that minimizes or maximizes the value of the objective function. The schematic flow is shown in Figure 7.

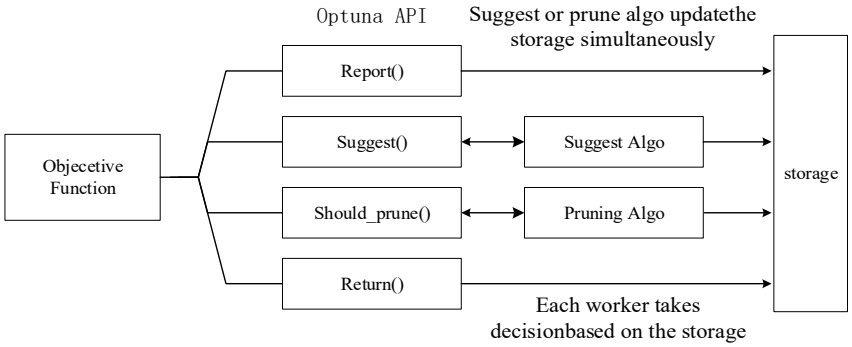

Figure 7. Optuna principle schematic.

3. Experiments

3.1. Experimental Environment

All experiments in this paper were implemented in a Windows 10 Intel (R) Core™ i7-7700HQ CPU (2.80 GHz), 16.00 GB RAM environment. The algorithms were implemented using the Sklearn library in Python, etc.

3.2. Introduction to the Data Set

When choosing datasets for intrusion detection classifier performance comparison experiments, it is essential to consider the following factors:

Diversity of datasets: Select datasets from various types and sources to ensure the results are comprehensive and applicable to different scenarios.

Size of the dataset: Choose datasets with an adequate sample size to ensure the reliability and statistical significance of the experimental findings.

The authenticity of the dataset: Prefer real-world datasets whenever possible, as they reflect the complexities and characteristics of actual intrusion scenarios more accurately than synthetic or artificially generated datasets.

Based on these considerations, the following three datasets are recommended for intrusion detection classifier performance comparison experiments.

(1) NSL-KDD dataset:

The NSL-KDD dataset is a revised version of the KDD-99 dataset, removing redundant and unnecessary data while retaining both normal and attack connections. The NSL-KDD dataset "https://www.unb.ca/cic/datasets/nsl.html (accessed on 1 May 2023)" [37] offers improved data quality and a more balanced distribution of classes. Consequently, this study selects the files KDDTrain+ and KDDTest+ from the NSL-KDD dataset as the training and testing sets for the classifier. Each record in the dataset consists of 43 features, with the last column representing four different types of access (DOS, PROBE, R2L, U2R) and one normal access. The specific composition and scale of the dataset are detailed in Table 8.

Table 8. Composition of the NSL-KDD dataset.

	KDDTrain+	KDDTest+
Normal	67,343	9711
Dos	45,927	7460
Probe	11,656	2421
R2L	995	2885
U2R	52	67
Total	125,973	22,544

(2) UNSW-NB15 dataset:

The UNSW-NB15 dataset is a network intrusion detection dataset developed by the University of New South Wales "https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 1 May 2023)" [38–42]. It comprises real network traffic records, encompassing normal connections and various types of attacks, thus offering a more precise reflection of the threats and attacks prevalent in modern network environments. Utilizing the UNSW-NB15 dataset facilitates the assessment of classifiers in highly realistic and intricate network scenarios. The dataset includes nine types of attacks, namely Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Each flow data within the dataset is characterized by 48 features. The training set in the CSV file encompasses 175,341 records, while the test set contains 82,332 records. The specific composition and scale of the dataset are outlined in Table 9.

Table 9. Composition of the UNSW-NB15 dataset.

	Training Set	Testing Set
Normal	56,000	37,000
Anallysis	2000	677
Backdoor	1746	583
Dos	12,264	4089
Exploits	33,393	11,132
Fuzzers	18,184	6062
Generic	40,000	18,871
Reconnaissance	10,491	3496
Shellcode	1133	378
Worms	130	44
Total	175,341	82,332

(3) CICIDS2017 dataset:

The CICIDS2017 dataset https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 1 May 2023) [43], is developed by the Canadian Department of National Defence's Research and Development Canada (DRDC) for network intrusion detection. It comprises data traffic records from real-world network environments, including normal connections and various types of attacks. The dataset offers a substantial number of samples and diverse attack scenarios, featuring five major categories of attack accesses and one category for normal access, with each data entry containing 78 features. This dataset is suitable for evaluating classifiers' performance in large-scale and complex network environments. However, it is essential to note that the CICIDS2017 dataset is derived from real-world data collected over five days, specifically from 3 July 2017 to 7 July 2017. Each day's abnormal flow traffic may differ, so the entire dataset is segregated and saved based on dates. To address this, the current study integrates the data from these five days and randomly splits it into a training set and a test set in a 7:3 ratio. After partitioning, the training set comprises 1,980,111 records, while the test set contains 850,632 records. The specific composition and scale of the dataset are provided in Table 10.

Table 10. Composition of the CICIDS2017 dataset.

	Total_Data	Train	Test
Benign	2,273,097	1,591,168	681,929
Patator	13,835	9685	4151
Dos	380,688	266,482	114,206
Web	2180	1526	654
PortScan	158,930	111,251	47,679
Others	2013	1409	604
Total	2,830,743	1,980,111	850,632

3.3. Model Performance Evaluation Indicators

For classification problems, the classification result is uncertain, and all possible results can be divided into four cases in the following table. TP indicates the number of attack flows detected by the model and the detection result is correct; FN indicates the number of attack flows detected, but the detection result is incorrect, and the flows are normal; TN indicates the number of normal flows detected and the detection result is correct; FP indicates the number of normal flows detected, but the detection result is incorrect, and the flows are attack flows; FP indicates the number of flows detected as normal, but the detection result is incorrect, and the traffic is attack traffic. Where FP and FN are called false positives, as shown in Table 11, based on the four parameters, four metrics can usually be derived to measure the actual performance of a model.

Table 11. Confusion matrix.

Status	Judged as Attack Flow	Judged as Normal Flow
Attack Flow	TP	FP
Normal flow	FN	TN

Accuracy, the number of predicted pairs as a proportion of the total number of samples, is mathematically expressed as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Precision, the mathematical expression for how many of the samples predicted to be positive are positive, is as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

Recall the mathematical expression for how many positive cases in a sample are correctly predicted, is as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

F1 score. Precision and recall are conflicting metrics, and in training, it is often necessary to find the equilibrium point between precision and recall, which is represented by the summed average of precision and recall, F, whose mathematical expression is as follows:

$$\text{F-measure} = \frac{2 \times (\text{Precision} + \text{Recall})}{\text{Precision} \times \text{Recall}} \tag{11}$$

3.4. Experimental Analysis

3.4.1. Experiment 1 Accuracy of Feature Extraction within Each Threshold

The choice of features directly impacts the classification outcome, making feature selection a crucial aspect of intrusion detection. The objective of feature selection is accomplished by calculating the mRMR value coefficient for each feature individually. This study discarded features exhibiting no correlation or weak correlation based on the feature correlation strength correspondence table. Subsequently, classification experiments are conducted within a weak correlation strength threshold of 0.0 to 1.0, as shown in Table 12.

Table 12. Results of classification with features of different thresholds.

Range of mRMR Thresholds		NSL-KDD	UNSW-NB15	CICIDS2017
0.4~0.6	Accuracy	98.15%	96.87%	98.51%
0.2~0.8		98.86%	96.60%	99.01%
0.0~1.0		97.40%	96.06%	98.91%

In the classification experiments, the CatBoost model is utilized as an example to determine the threshold value. The results depicted in the figure indicate that the recognition rate is higher when the coefficients fall within the range of 0.2 to 0.8 compared to other values. Moreover, the effectiveness diminishes when the mRMR value is too small or too large. Consequently, this study adopts the threshold value of 0.2 to 0.8 for the classification experiment and proceeds with the feature selection process.

3.4.2. Experiment 2 Optuna Optimization Effects

In conjunction with the selected model, the optimal hyperparameters (learning rate, maximum tree depth, number of trees, number of samples per leaf node, and regularization parameters) were determined for each dataset using CatBoost with Optuna search. The iterative results of this search process are presented in Figure 8. (a) is the iterative optimization on the NSL-KDD dataset, (b) is the iterative optimization on the UNSW-NB15 dataset, and (c) is the iterative optimization on the dataset CICIDS2017.

The optimal hyperparameters of catboost for different data sets were obtained after several iterations, and the results are shown in Table 13:

Table 13. Optimal hyperparameter results.

	Learning_Rate	Max_Depth	Iterations	Min_Data_in_Leaf	l2_Leaf_Reg
NSL-KDD	0.0949	8	124	5	2
UNSW-NB15	0.0941	10	183	11	3
CICIDS2017	0.0979	9	186	8	2

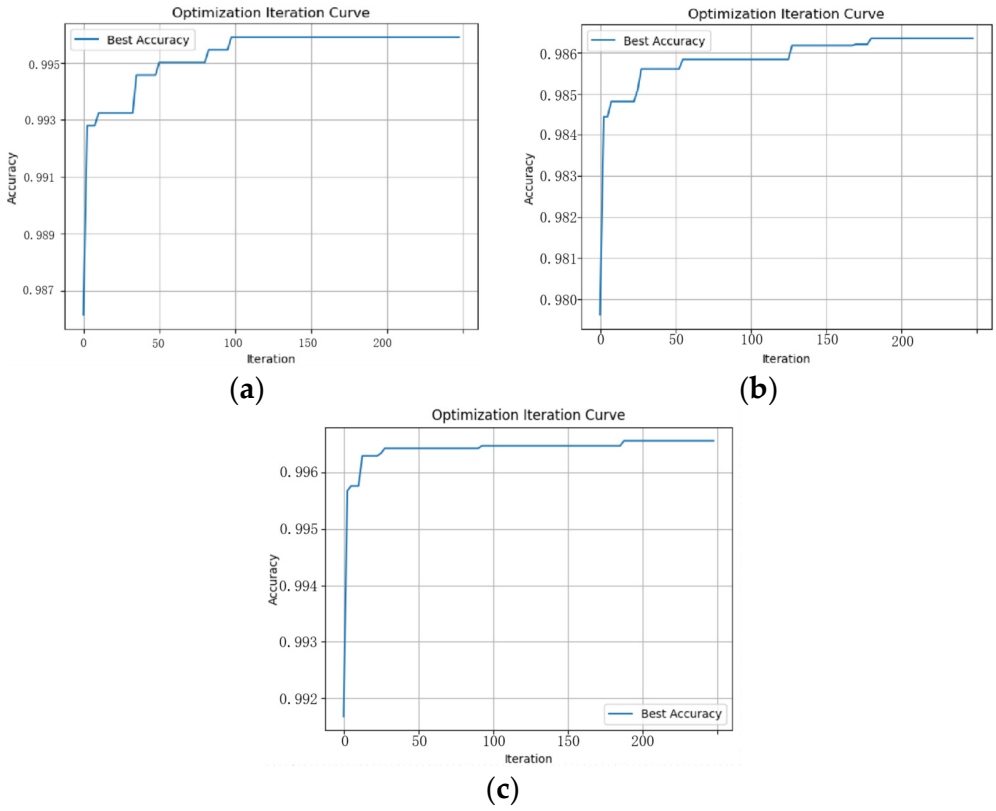

Figure 8. Optuna Optimization Search Process Diagram.

3.4.3. Experiment 3 Comparison of Multiple Optimization Algorithms

The classical machine learning algorithms and the models developed in this study were utilized to perform binary and multiclassification experiments on three distinct datasets. Subsequently, a comprehensive comparison and analysis were conducted based on the results.

- Binary Classification

Using the NSL-KDD dataset to train the model, it can be seen from Table 14 that the accuracy, recall, precision and F1 of the Mrmr-SMOTE-Optuna-Catboost algorithm designed in this paper for the detection of the NSL-KDDtest+ dataset are 99.2623%, 99.6350%, 99.2219% and 99.4280% respectively. All the metrics are higher than those of several machine learning models. Figure 9 is its confusion matrix.

Table 14. Comparison results of the proposed model with traditional methods in multiple classifications of the NSL-KDD dataset.

NSL-KDD Dataset Performance Comparison				
Method	**Accuracy**	**Precision**	**Recall**	**F-Measure**
SVM [44–46]	93.4780%	94.2598%	95.5015%	94.8766%
LR [47–49]	96.1362%	96.3387%	97.6362%	96.9831%
XGBoost [50–52]	97.9111%	98.1793%	98.5726%	98.3756%
LightGBM [53,54]	98.2312%	98.5115%	98.7397%	98.6254%
Catboost [55–57]	98.5390%	99.0700%	98.6616%	98.8654%
Algorithm of this paper	99.2623%	99.6350%	99.2219%	99.4280%

Using the UNSW-NB15 dataset to train the model, Table 15 shows that the t algorithm designed in this paper is also superior to the comparison model in all indicators, which is better than most machine learning methods. Figure 10 is its confusion matrix.

TP	FP
17472	64
FN	TN
137	9574

Figure 9. Confusion matrix of the NSL-KDD dataset.

Table 15. Comparison results of the proposed model with traditional methods in multiple classifications of the UNSW-NB15 dataset.

UNSW-NB15 Dataset Performance Comparison				
Method	**Accuracy**	**Precision**	**Recall**	**F-Measure**
SVM [44–46]	94.1382%	95.6972%	94.6674%	95.1795%
LR [47–49]	95.7758%	97.3158%	95.7704%	96.5369%
XGBoost [50–52]	97.3700%	98.8201%	96.8995%	97.8503%
LightGBM [53,54]	97.7769%	99.1607%	97.2270%	98.1843%
Catboost [55–57]	98.0843%	98.9901%	97.8886%	98.4363%
Algorithm of this paper	98.7050%	99.5037%	98.3908%	98.9441%

TP	FP
57536	287
FN	TN
941	36059

Figure 10. Confusion matrix of the UNSW-NB15 dataset.

For all attack types in the CICIDS2017 dataset, as shown in Table 16, the intrusion detection rate after feature engineering and model optimization has been significantly improved. The detection performance is better than traditional intrusion detection methods, and Figure 11 shows its confusion matrix.

Table 16. Comparison results of the proposed model with traditional methods in multiple classifications of the CICIDS-2017 dataset.

CICIDS-2017 Dataset Performance Comparison				
Method	**Accuracy**	**Precision**	**Recall**	**F-Measure**
SVM [44–46]	96.3273%	95.4190%	93.6821%	94.5426%
LR [47–49]	96.7872%	95.6917%	94.7427%	95.2149%
XGBoost [50–52]	97.8146%	97.0793%	96.3794%	96.7281%
LightGBM [53,54]	98.9606%	98.5077%	98.3618%	98.4347%
Catboost [55–57]	98.9606%	98.7985%	98.0737%	98.4347%
Algorithm of this paper	99.6612%	99.8893%	99.0898%	99.4879%

TP	FP
334745	371
FN	TN
3075	678854

Figure 11. Confusion matrix of the CICIDS2017 dataset.

The integrated model proposed through the analysis of the results surpasses alternative strategies in the context of binary classification. It demonstrates remarkable generalization capabilities, making it suitable for datasets of varying complexity. The model consistently outperforms others across multiple datasets, underscoring its versatility and adaptability. Notably, this work emphasises feature selection and hyperparameter optimization more than previous studies. Our method was rigorously tested during validation on various benchmark datasets, yielding averaged and reliable results for a comprehensive comparison.

- Multi classification

Intrusion detection entails the accurate detection of intrusion behaviors and the precise identification of specific attack types. While some existing intrusion detection algorithms exhibit strong performance in binary classification tasks, they encounter challenges in multi-classification tasks, particularly in accurately categorizing the minority classes within abnormal data, resulting in decreased accuracy. This study leveraged information from three datasets to consolidate the smaller categories into larger categories. We then classified the categorical items into 5, 10, and 6 categories, including normal traffic. Tables 17–19 present a performance comparison between the intrusion detection model based on feature extraction and various machine learning algorithms for multi-classification tasks on the three datasets [58–60].

Table 17. The proposed model is compared with existing methods in multiple classifications on the NSL-KDD dataset.

NSL-KDD Dataset Performance Comparison Accuracy						
	SVM [44–46]	LR [47–49]	XGBoost [50–52]	LightGBM [53,54]	Catboost [55–57]	Algorithm of this paper
Normal	60.37%	83.61%	97.80%	98.07%	98.83%	99.84%
Dos	96.67%	66.39%	98.66%	98.59%	97.71%	99.12%
Probe	98.16%	96.19%	98.93%	98.84%	99.21%	99.89%
R2L	39.35%	27.94%	77.81%	84.50%	82.59%	94.63%
U2R	0	0	12.25%	42.47%	48.21%	67.52%

Table 18. The proposed model is compared with existing methods in multiple classifications on the UNSW-NB15 dataset.

UNSW-NB15 Dataset Performance Comparison Accuracy						
	SVM [44–46]	LR [47–49]	XGBoost [50–52]	LightGBM [53,54]	Catboost [55–57]	Algorithm of this paper
Normal	59.37%	84.03%	89.57%	95.42%	93.87%	98.85%
Fuzzers	35.57%	39.87%	63.94%	67.24%	68.91%	75.63%
Analysis	0	0	2.00%	26.14%	28.67%	36.46%
Backdoors	9.46%	6.27%	6.59%	22.52%	18.47%	31.28%
DoS	5.83%	23.04%	19.94%	32.33%	36.05%	54.29%
Exploits	25.87%	32.76%	54.31%	77.42%	81.65%	89.42%
Generic	83.42%	86.52%	89.77%	96.62%	97.87%	98.43%
Reconnaissance	10.05%	39.85%	91.83%	93.17%	93.00%	97.81%
Shellcode	0	0	18.77%	25.19	26.74%	46.32%
Worms	0	5.26%	19.81%	26.77%	26.54%	36.84%

Table 19. The proposed model is compared with existing methods in multiple classifications on the CICIDS2017 dataset.

CICIDS2017 Dataset Performance Comparison Accuracy						
	SVM [44–46]	LR [47–49]	XGBoost [50–52]	LightGBM [53,54]	Catboost [55–57]	Algorithm of this paper
BENIGN	67.82%	91.50%	98.79%	99.24%	99.36%	99.72%
PortScan	44.87%	48.06%	45.60%	62.74%	59.24%	89.81%
Patator	88.79%	87.64%	97.25%	98.58%	97.83%	99.24%
DoS	52.29%	52.78%	94.95%	97.54%	96.23%	99.91%
Web	0	0	90.43%	95.71%	93.87%	96.92%
Others	0	0	63.13%	59.77%	65.92%	69.84%

It can be inferred from the table that the proposed integrated model outperforms other strategies in terms of overall performance metrics after completing feature engineering on the dataset, as well as hyperparameter tuning, overcoming the challenges of high latitude and class imbalance encountered in intrusion detection, and efficiently detecting intrusions on the three most well-represented datasets.

3.4.4. Experiment 4 Optimization Algorithm Evaluation

In the preceding multiple sets of experiments, a comparative study was conducted between our proposed algorithm and traditional classical machine learning algorithms in binary and multiclass tasks. The results confirmed that our algorithm significantly outperforms the traditional methods in various evaluation metrics for binary and multiclass experiments. The analysis based on evaluation metrics demonstrates the algorithm's capability to effectively address the challenges posed by high dimensionality and class imbalance. Moreover, the optimized algorithm exhibits strong generalizability, as evidenced by its excellent performance across diverse datasets. To ensure the scientific rigor of the experimental results, a comprehensive performance comparison was further conducted between our algorithm and a series of state-of-the-art network intrusion detection methods. Given the deployment environment of practical Internet of Things (IoT) systems, the focus was placed on analyzing the comparative performance of multiple algorithms in the binary classification scenario, where the higher cost lies in correctly distinguishing between normal and intrusive traffic data rather than classifying various types of intrusions. The detailed comparison results are presented in Table 20.

Through comprehensive comparative analysis of various indicators, the proposed algorithm in this study demonstrated significant superiority over NSL-KDD, UNSW-NB15, and CICIDS-2017 datasets. Given the high dimensionality and data imbalance of the datasets, we first selected features that could best represent the original data, reducing the spatial complexity and dimensionality of the dataset. Subsequently, by oversampling the minority class data, we successfully addressed the issue of data class imbalance. After effective feature engineering, we input the reduced dataset into subsequent classification algorithms, significantly reducing the spatial complexity and the number of features for each data point, thereby reducing the overall dataset size and alleviating the computational burden of the algorithm.

Considering the intrusion detection application environment, where most dataset represents normal class samples, we focused on the minority class representing the attack category. Hence, the model combined SMOTE algorithm to enhance the proportion of the minority class, thereby increasing the algorithm's sensitivity to the minority class data and improving the accuracy of data classification. Consequently, our algorithm exhibits significant superiority in binary classification tasks for network intrusion detection.

Table 20. Results of the comparison of the proposed model with the state-of-the-art methodology.

	Algorithm Model	Dataset	Accuracy	Precision	Recall	F-Measure
Selvapandian et al. [61]	Deep learning	NSL-KDD	96.28	94.41	97.51	-
Kishwar Sadaf et al. [62]	Auto-IF		95.40	94.81	97.25	96.01
Samira Sarvari et al. [63]	Clustering Method		98.81	-	97.25	-
Algorithm of this paper			99.2623	99.6350	99.22	99.43
Sydney M.Kasongo et al. [64]	XGBoost-DT	UNSW-NB15	94.12	80.33	98.38	88.45
Pengzhen Zhou et al. [65]	KNN-DDAE-DNN		89.34	85.52	-	90.94
Moutaz Alazab et al. [66]	Moth-Flame-OPT		92.4	-	92.3	94.2
Algorithm of this paper			98.70	99.50	98.39	98.94
Shruti Patil et al. [67]	XAI-LIME	CICIDS-2017	96.26	89.00	89.00	89.00
Abdulaziz Fatani et al. [68]	TSODE-CNN		99.73	99.76	99.66	99.48
Algorithm of this paper			99.66	99.89	99.09	99.49

However, it is essential to note that despite demonstrating significant advantages in experiments, our method still has some limitations. For instance, the algorithm may have limited generalization performance and feature representation capabilities, as its performance could be influenced by the quality and quantity of features extracted from network data. Therefore, further research on feature engineering may be necessary to improve its effectiveness. Additionally, the algorithm's performance is optimized for specific datasets mentioned in the paper, which may affect its generalization ability on other untested data, posing certain challenges in practical applications. Addressing these limitations and further research to enhance the algorithm's robustness and adaptability will be crucial in advancing its practical application capabilities and contributing to the overall research.

3.4.5. Actual Deployment Testing

In recent years, the Internet of Things (IoT) has developed rapidly, bringing convenience to people. However, it also harbors numerous hidden risks, and security issues cannot be overlooked. In the future development and application of IoT, intrusion detection is worthy of attention and exploration. Through various experimental comparisons presented earlier, the theoretical superiority of our algorithm has been demonstrated. In the following sections, we will attempt to apply this algorithm to real-world scenarios and analyze its strengths and limitations under resource constraints or specific network conditions.

Currently, the algorithm faces certain challenges when applied in different scenarios. Intrusion detection algorithms require high real-time capabilities in network environments, but traditional cloud-based intelligent controls fall short due to bandwidth limitations and delays, failing to meet the demands. Therefore, this study adopts an intelligent gateway controller based on edge computing to replace the traditional cloud-based framework, enabling the construction of an intrusion detection model. Edge computing provides services closer to smart devices at edge nodes, offering advantages like rapidity, real-time processing, low power consumption, and low bandwidth costs. Deploying the intrusion detection model at the edge can meet the real-time security needs of smart homes. In line

with the "cloud-edge-device" model, a technological system combines edge computing, IoT, cloud computing, and big data technologies. This results in a three-dimensional, efficient intrusion detection and multi-device networking capability, forming an intelligent IoT system framework, as depicted in Figure 12. The framework enables real-time data aggregation and management, enhancing the security measures for IoT.

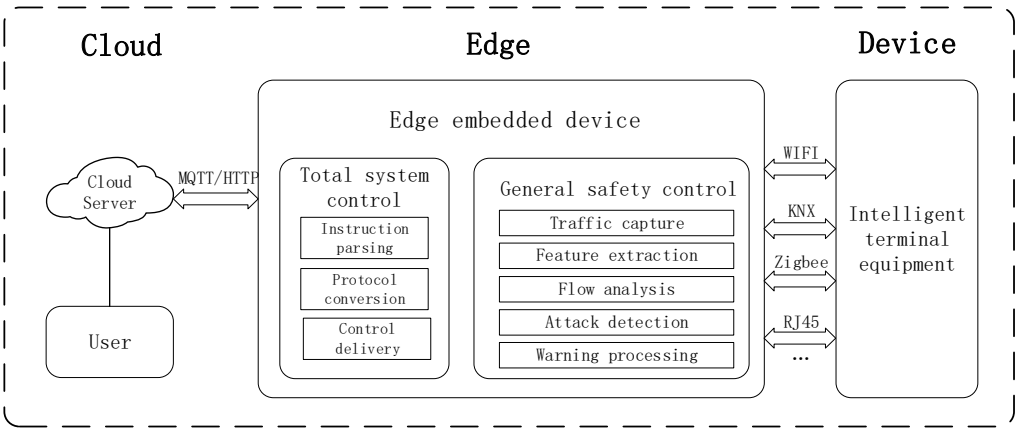

Figure 12. Smart iot intrusion detection system framework.

In the above system framework, the research aims to enable the cloud server to receive instructions from administrators or users and relay these commands to the intelligent gateway of edge computing through the cloud server. Serving as the central hub of all information, the intelligent gateway controls commands from the cloud server. It receives sensor information from the device side or transmits control commands to the terminal controller. The intrusion detection algorithm is applied to identify and detect the status of the information, determining whether it is in a normal state.

In different application scenarios, the number of devices in the intelligent gateway may vary. For this study, we have chosen Allwinnertech H6 as the core SoC of the edge intelligent gateway platform. This chip has a quad-core Cortex-A53 CPU and Mali T720 GPU, supporting dual-channel DDR4 and EMMC5.0 high-speed flash memory. Such selection enables us to better test the algorithm's efficiency and versatility under limited computational resources.

Within the framework of edge computing, implementing an intrusion detection system heavily relies on the data capture module to acquire flow data and parse data packets. Simultaneously, the data processing module aids in preprocessing flow data to conform to the model's input requirements. Lastly, the intrusion detection module classifies and detects the flow, presenting the detection results. The specific deployment process design is illustrated in Figure 13.

After completing the deployment, we conducted real-world testing and a comprehensive evaluation of the deployed model using the multiple evaluation criteria described earlier. In the Internet of Things (IoT) network environment, Denial-of-Service (DoS) attacks are currently one of the most common and highly destructive attacks. These attacks overwhelm the target by flooding it with many data packets, causing excessive resource consumption and rendering the service unavailable. To assess the algorithm's ability to detect such attacks, we chose the XOIC tool and utilized UDP and ICMP messages as the two DoS attack methods for testing.

Within the IoT framework described in the document, end-users issue control commands through the cloud server, which then relays the information to the edge intelligent gateway device. At the gateway, the information is subjected to real-time detection. Considering practical constraints, each normal connection was repeated ten times, lasting for 60 s. This comprised seven connections between the cloud server and the edge intelligent gateway and three connections between terminal devices and the edge intelligent

gateway. Subsequently, the two attack types were initiated, each lasting for 10 s. This process was repeated 100 times, and the average values were calculated based on the actual detection results.

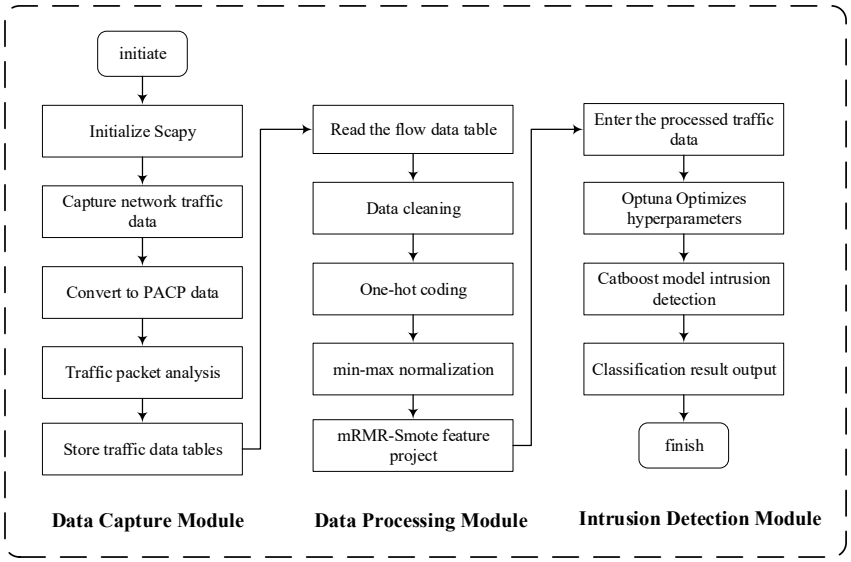

Figure 13. Smart IoT Intrusion Detection System Realization Process.

Moreover, to minimize interference from other traffic passing through the gateway, we set up filtering rules in the data collection and analysis module. Specifically, we collected and analyzed attack traffic between attacking machines and the target and normal traffic between the intelligent gateway and the servers or terminal devices. Throughout the testing process, we observed the statistical information and logs from the local detection module to analyze the data. The actual intrusion detection results based on our proposed algorithm are presented in Table 21.

Table 21. Comparison of actual application before and after algorithm improvement.

	Catboost	Algorithm of This Paper
Accuracy/%	94.5992	98.7448
Precision/%	96.4023	99.2053
Recall/%	95.1788	98.8556
F-measure/%	95.7866	99.0301
Test time/s	0.772	0.368

In the edge intelligent gateway application with limited computing resources, we implemented an enhanced model to perform feature engineering based on Minimum Redundancy, Maximum Relevance and Synthetic Minority Over-sampling Technique. This resulted in a reduction of data space complexity and a decrease in computational burden. Compared to the unprocessed algorithm, the improved algorithm demonstrated a decrease of 0.404 s in testing time under similar network conditions. This enhancement enables the algorithm to handle a greater volume of traffic data within constrained computing resources and time limitations, meeting the intrusion detection system controller's requirements for identifying network data flows. Furthermore, the improved algorithm exhibited notable improvements in accuracy, precision, recall, and F-measure values, with increases of 4.15%, 2.80%, 3.68%, and 3.24%, respectively, leading to more accurate data classification.

Based on the aforementioned test results, deploying the enhanced intrusion detection model on edge nodes enables real-time monitoring of traffic data. It allows timely response to intrusion behavior, effectively safeguarding the security of the IoT platform. However, to further enhance the algorithm's robustness and adaptability, we should focus on providing

more relevant details and conducting rigorous evaluations of the algorithm's performance in various scenarios.

4. Discussion and Conclusions

In the rapidly expanding world of the Internet of Things, network attacks are rising. Therefore, enhancing the performance of intrusion detection systems is of paramount importance to safeguard IoT applications and connected devices. Many researchers are devoted to developing secure, lightweight IDS frameworks using machine learning techniques. When ensuring the quality of machine learning-based IDS, two critical factors are data feature handling and model generalization.

Research findings indicate that data preprocessing techniques like data cleaning, One-hot encoding, and normalization can enhance dataset availability. Through feature selection and oversampling in feature engineering, the proportion of minority class samples can be increased while reducing data dimensionality and computational overhead, thereby improving the algorithm model's effective representation of data and accurate classification. Moreover, introducing the Optuna framework for hyperparameter optimization of the CatBoost algorithm further enhances its accuracy, leading to superior classification results. Looking towards the future, we can focus on the following key aspects:

1. Utilizing random undersampling: Consider employing random undersampling to reduce the number of majority class instances and optimize dataset balance. This reduces the demand for computational resources and maintains dataset representativeness.
2. Lightweight model framework: Strive to reduce model complexity and computational burden to enhance efficiency and real-time performance. Exploring simplified model structures, feature selection algorithms, or model compression techniques can achieve lightweight objectives.
3. Real-time environment testing: Besides evaluating model performance in the laboratory, conducting comprehensive tests in real-time environments will provide more realistic scenarios and data to further validate the model's reliability and practicality.

By conducting further research and improvements in these areas, we can elevate the model's performance and make it more suitable for real-world network intrusion detection scenarios. These endeavors will strengthen the security and reliability of the IoT environment, effectively countering the ever-growing network threats. Our goal is to protect IoT platforms, enabling them to monitor traffic data in real time and respond promptly to intrusion behavior, ensuring the safety and stable operation of the Internet of Things.

Author Contributions: Conceptualization, Z.W.; methodology, Y.Z.; writing—original draft preparation, Z.W.; supervision, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The experimental data used to support the findings of this study are available from the corresponding author upon request.

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

References

1. Florackis, C.; Louca, C.; Michaely, R.; Weber, M. Cybersecurity Risk. *Rev. Financ. Stud.* **2022**, *36*, 351–407. [CrossRef]
2. Insua, D.R.; Couce-Vieira, A.; Rubio, J.A.; Pieters, W.; Labunets, K.; Rasines, D.G. An Adversarial Risk Analysis Framework for Cybersecurity. *Risk Anal.* **2019**, *41*, 16–36. [CrossRef]
3. Mills, R.; Marnerides, A.K.; Broadbent, M.; Race, N. Practical Intrusion Detection of Emerging Threats. *IEEE Trans. Netw. Serv. Manag.* **2021**, *19*, 582–600. [CrossRef]
4. Maseno, E.M.; Wang, Z.; Xing, H. A Systematic Review on Hybrid Intrusion Detection System. *Secur. Commun. Netw.* **2022**, *2022*, 9663052. [CrossRef]

5. Zipperle, M.; Gottwalt, F.; Chang, E.; Dillon, T. Provenance-based Intrusion Detection Systems: A Survey. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef]
6. Hawkar, K.; Shaikha, A.; Wafaa, M.; Abduallah, B. A Review of Intrusion Detection Systems. *Acad. J. Nawroz Univ.* **2017**, *6*, 101–105. [CrossRef]
7. Om, H.; Kundu, A. A hybrid system for reducing the false alarm rate of anomaly intrusion detection system. In Proceedings of the 2012 1st International Conference on Recent Advances in Information Technology (RAIT), Dhanbad, India, 15–17 March 2012; pp. 131–136. [CrossRef]
8. Hsu, C.-Y.; Wang, S.; Qiao, Y. Intrusion detection by machine learning for multimedia platform. *Multimed. Tools Appl.* **2021**, *80*, 29643–29656. [CrossRef] [PubMed]
9. Zhang, C.; Jia, D.; Wang, L.; Wang, W.; Liu, F.; Yang, A. Comparative research on network intrusion detection methods based on machine learning. *Comput. Secur.* **2022**, *121*, 102861. [CrossRef]
10. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *J. Big Data* **2019**, *86*, 147–167. [CrossRef]
11. Bagui, S.; Li, K. Resampling imbalanced data for network intrusion detection datasets. *Rev. Financ. Stud.* **2021**, *8*, 351–407. [CrossRef]
12. Yang, Z.; Liu, X.; Li, T.; Wu, D.; Wang, J.; Zhao, Y.; Han, H. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Comput. Secur.* **2022**, *116*, 102675. [CrossRef]
13. Yousefnezhad, M.; Hamidzadeh, J.; Aliannejadi, M. Ensemble classification for intrusion detection via feature extraction based on deep Learning. *Soft Comput.* **2021**, *25*, 12667–12683. [CrossRef]
14. Reddy, G.T.; Reddy, M.P.K.; Lakshmanna, K.; Kaluri, R.; Rajput, D.S.; Srivastava, G.; Baker, T. Analysis of Dimensionality Reduction Techniques on Big Data. *J. Mag.* **2020**, *8*, 54776–54788. [CrossRef]
15. Li, Y.; Qin, T.; Huang, Y.; Lan, J.; Liang, Z.; Geng, T. HDFEF: A hierarchical and dynamic feature extraction framework for intrusion detection systems. *Comput. Secur.* **2022**, *121*, 102842. [CrossRef]
16. Mohammadi, S.; Mirvaziri, H.; Ghazizadeh-Ahsaee, M.; Karimipour, H. Cyber intrusion detection by combined feature selection algorithm. *J. Inf. Secur. Appl.* **2019**, *44*, 80–88. [CrossRef]
17. Farahani, G. Feature Selection Based on Cross-Correlation for the Intrusion Detection System. *Secur. Commun. Netw.* **2020**, *2020*, 8875404. [CrossRef]
18. Tan, X.; Su, S.; Huang, Z.; Guo, X.; Zuo, Z.; Sun, X.; Li, L. Wireless Sensor Networks Intrusion Detection Based on SMOTE and the Random Forest Algorithm. *Sensors* **2019**, *19*, 203. [CrossRef]
19. Zhang, H.; Huang, L.; Wu, C.Q.; Li, Z. An Effective Convolutional Neural Network Based on SMOTE and Gaussian Mixture Model for Intrusion Detection in Imbalanced Dataset. *Comput. Netw.* **2020**, *177*, 107315. [CrossRef]
20. Hancock, J.T.; Khoshgoftaar, T.M. CatBoost for big data: An interdisciplinary review. *J. Big Data* **2020**, *7*, 94. [CrossRef]
21. Abbood, Z.A.; Khaleel, I.; Aggarwal, K. Challenges and Future Directions for Intrusion Detection Systems Based on AutoML. *Mesopotamian J. CyberSecur.* **2021**, *2021*, 16–21. [CrossRef]
22. Alajanbi, M.; Ismail, M.A.; Hasan, R.A.; Sulaiman, J. Intrusion Detection: A Review. *Mesopotamian J. CyberSecur.* **2021**, *2021*, 1–4. [CrossRef]
23. Zaib, R.; Zhou, K.-Q. Zero-Day Vulnerabilities: Unveiling the Threat Landscape in Network Security. *Mesopotamian J. CyberSecur.* **2022**, *2022*, 57–64. [CrossRef]
24. Nassreddine, G.; Younis, J.; Falahi, T. Detecting Data Outliers with Machine Learning. *Al-Salam J. Eng. Technol.* **2023**, *2*, 152–164. [CrossRef]
25. Khan, N.; Khaleel, I.; Daghighi, E. Improved feature selection method for features reduction in intrusion detection systems. *Mesopotamian J. CyberSecur.* **2021**, *2021*, 9–15. [CrossRef]
26. Chan, P.P.K.; He, Z.M.; Li, H.; Hsu, C.C. Data sanitization against adversarial label contamination based on data complexity. *Int. J. Mach. Learn. Cyber.* **2018**, *9*, 1039–1052. [CrossRef]
27. Shen, C.; Wang, Q.; Priebe, C.E. One-Hot Graph Encoder Embedding. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 7933–7938. [CrossRef]
28. Huang, H.-C.; Qin, L.-X. Empirical evaluation of data normalization methods for molecular classification. *PeerJ* **2018**, *6*, e4584. [CrossRef]
29. Özyurt, F. A fused CNN model for WBC detection with MRMR feature selection and extreme learning machine. *Soft Comput.* **2020**, *24*, 8163–8172. [CrossRef]
30. Singh, P.; Borgohain, S.K.; Sharma, L.D.; Kumar, J. Minimized feature overhead malware detection machine learning model employing MRMR-based ranking. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6992. [CrossRef]
31. Ma, X.; Shi, W. AESMOTE: Adversarial Reinforcement Learning with SMOTE for Anomaly Detection. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 943–956. [CrossRef]
32. Douzas, G.; Bacao, F. Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. *Inf. Sci.* **2019**, *501*, 118–135. [CrossRef]
33. Nayak, J.; Naik, B.; Dash, P.B.; Vimal, S.; Kadry, S. Hybrid Bayesian optimization hypertuned catboost approach for malicious access and anomaly detection in IoT nomalyframework. *Sustain. Comput. Inform. Syst.* **2022**, *36*, 100805. [CrossRef]

34. Chen, R.; Zhou, L.; Xiong, C.; Xu, H.; Zhang, Z.; He, X.; Dong, Q.; Wang, C. Islanding detection method for microgrids based on CatBoost. *Front. Energy Res.* **2022**, *10*, 1016754. [CrossRef]
35. Shekhar, S.; Bansode, A.; Salim, A. A Comparative study of Hyper-Parameter Optimization Tools. *arXiv* **2021**, arXiv:2201.06433.
36. Lai, J.-P.; Lin, Y.-L.; Lin, H.-C.; Shih, C.-Y.; Wang, Y.-P.; Pai, P.-F. Tree-Based Machine Learning Models with Optuna in Predicting Impedance Values for Circuit Analysis. *Micromachines* **2023**, *14*, 265. [CrossRef] [PubMed]
37. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A. A Detailed Analysis of the KDD CUP 99 Data Set. In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), Ottawa, ON, Canada, 8–10 July 2009.
38. Nour, M.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015.
39. Nour, M.; Slay, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset. *Inf. Secur. J. Glob. Perspect.* **2016**, *25*, 18–31.
40. Moustafa, N.; Slay, J.; Creech, G. Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. *IEEE Trans. Big Data* **2017**, *5*, 481–494. [CrossRef]
41. Moustafa, N.; Creech, G.; Slay, J. Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models. In *Data Analytics and Decision Support for Cybersecurity*; Springer: Cham, Switzerland, 2017; pp. 127–156.
42. Sarhan, M.; Layeghy, S.; Moustafa, N.; Portmann, M. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings*; Springer Nature: Berlin/Heidelberg, Germany, 2020; p. 117.
43. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Portugal, 22–24 January 2018.
44. Shawe-Taylor, J.; Sun, S. A review of optimization methodologies in support vector machines. *Neurocomputing* **2011**, *74*, 3609–3618. [CrossRef]
45. Mohammadpour, L.; Hussain, M.; Aryanfar, A.; Raee, V.M.; Sattar, F. Evaluating Performance of Intrusion Detection System using Support Vector Machines: Review. *Int. J. Secur. Appl.* **2015**, *9*, 225–234. [CrossRef]
46. Alqarni, A.A. Toward support-vector machine-based ant colony optimization algorithms for intrusion detection. *Soft Comput.* **2023**, *27*, 6297–6305. [CrossRef]
47. Bulso, N.; Marsili, M.; Roudi, Y. On the Complexity of Logistic Regression Models. *Neural Comput.* **2019**, *31*, 1592–1623. [CrossRef]
48. Wang, Y. A multinomial logistic regression modeling approach for anomaly intrusion detection. *Comput. Secur.* **2005**, *24*, 662–674. [CrossRef]
49. Sperandei, S. Understanding logistic regression analysis. *Biochem. Medica* **2014**, *24*, 12–18. [CrossRef] [PubMed]
50. Zhang, P.; Jia, Y.; Shang, Y. Research and application of XGBoost in imbalanced data. *Int. J. Distrib. Sens. Netw.* **2022**, *18*, 15501329221106935. [CrossRef]
51. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. *arXiv* **2016**, arXiv:1603.02754.
52. Dhaliwal, S.S.; Nahid, A.-A.; Abbas, R. Effective Intrusion Detection System Using XGBoost. *Information* **2018**, *9*, 149. [CrossRef]
53. Ke, G.; Meng, Q.; Finley, T.; Wang, T. LightGBM: A highly efficient gradient boosting decision tree. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 3149–3157. Available online: https://proceedings.neurips.cc/paper_files/paper/2017 (accessed on 1 May 2023).
54. Liu, J.; Gao, Y.; Hu, F. A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. *Comput. Secur.* **2021**, *106*, 102289. [CrossRef]
55. Dorogush, A.V.; Ershov, V.; Gulin, A. CatBoost: Gradient boosting with categorical features support. *arXiv* **2018**, arXiv:1810.11363.
56. Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A.V.; Gulin, A. CatBoost: Unbiased boosting with categorical features. *arXiv* **2017**, arXiv:1706.09516.
57. Leevy, J.L.; Hancock, J.; Zuech, R.; Khoshgoftaar, T.M. Detecting cybersecurity attacks across different network features and learners. *J. Big Data* **2021**, *8*, 38. [CrossRef]
58. Ngueajio, M.K.; Washington, G.; Rawat, D.B.; Ngueabou, Y. Intrusion Detection Systems Using Support Vector Machines on the KDDCUP'99 and NSL-KDD Datasets: A Comprehensive Survey. *arXiv* **2022**, arXiv:2209.05579.
59. Kilincer, I.F.; Ertam, F.; Sengur, A. A comprehensive intrusion detection framework using boosting algorithms. *Comput. Electr. Eng.* **2022**, *100*, 107869. [CrossRef]
60. Poornima, R.; Elangovan, M.; Nagarajan, G. Network attack classification using LSTM with XGBoost feature selection. *J. Intell. Fuzzy Syst.* **2022**, *43*, 971–984. [CrossRef]
61. Selvapandian, D.; Santhosh, R. Deep learning approach for intrusion detection in IoT-multi cloud environment. *Autom. Softw. Eng.* **2021**, *28*, 19. [CrossRef]
62. Sadaf, K.; Sultana, J. Intrusion Detection based on Autoencoder and Isolation Forest in Fog Computing. *IEEE Access* **2020**, *8*, 167059–167068. [CrossRef]

63. Sarvari, S.; Sani, N.F.M.; Hanapi, Z.M.; Abdullah, M.T. An Efficient Anomaly Intrusion Detection Method With Feature Selection and Evolutionary Neural Network. *IEEE Access* **2020**, *8*, 70651–70663. [CrossRef]
64. Kasongo, S.M.; Sun, Y. Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *J. Big Data* **2020**, *7*, 105. [CrossRef]
65. Zhou, P.; Zhang, H.; Liang, W. Research on hybrid intrusion detection based on improved Harris Hawk optimization algorithm. *Connect. Sci.* **2023**, *35*, 2195595. [CrossRef]
66. Alazab, M.; Abu Khurma, R.; Awajan, A.; Camacho, D. A new intrusion detection system based on Moth–Flame Optimizer algorithm. *Expert Syst. Appl.* **2022**, *210*, 118439. [CrossRef]
67. Patil, S.; Varadarajan, V.; Mazhar, S.M.; Sahibzada, A.; Ahmed, N.; Sinha, O.; Kumar, S.; Shaw, K.; Kotecha, K. Explainable Artificial Intelligence for Intrusion Detection System. *Electronics* **2022**, *11*, 3079. [CrossRef]
68. Fatani, A.; Elaziz, M.A.; Dahou, A.; Al-Qaness, M.A.A.; Lu, S. IoT Intrusion Detection System Using Deep Learning and Enhanced Transient Search Optimization. *IEEE Access* **2021**, *9*, 123448–123464. [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.