applied sciences

MDPI

*Article*

# AI-Assisted Security Alert Data Analysis with Imbalanced Learning Methods

**Samuel Ndichu *, Tao Ban, Takeshi Takahashi and Daisuke Inoue**

Cybersecurity Research Institute, National Institute of Information and Communications Technology, Tokyo 184-8795, Japan
* Correspondence: ndichu@nict.go.jp

**Abstract:** Intrusion analysis is essential for cybersecurity, but oftentimes, the overwhelming number of false alerts issued by security appliances can prove to be a considerable hurdle. Machine learning algorithms can automate a task known as security alert data analysis to facilitate faster alert triage and incident response. This paper presents a bidirectional approach to address severe class imbalance in security alert data analysis. The proposed method utilizes an ensemble of three oversampling techniques to generate an augmented set of high-quality synthetic positive samples and employs a data subsampling algorithm to identify and remove noisy negative samples. Experimental results using an enterprise and a benchmark dataset confirm that this approach yields significantly improved recall and false positive rates compared with conventional oversampling techniques, suggesting its potential for more effective and efficient AI-assisted security operations.

**Keywords:** alert fatigue; intrusion analysis; intrusion detection; imbalanced learning



**Citation:** Ndichu, S.; Ban, T.; Takahashi, T.; Inoue, D. AI-Assisted Security Alert Data Analysis with Imbalanced Learning Methods. *Appl. Sci.* **2023**, *13*, 1977. https://doi.org/10.3390/app13031977

Academic Editors: Konstantinos Rantos, Konstantinos Demertzis and George Drosatos

Received: 31 December 2022
Revised: 30 January 2023
Accepted: 31 January 2023
Published: 3 February 2023



## 1. Introduction

An intrusion detection system (IDS) is a security system that monitors and detects malicious activity in a computer network [1]. It is designed to detect viruses, worms, Trojans, and other malicious code or software, as well as unauthorized access attempts, denial of service attacks, and other malicious activities to detect and prevent attacks. In addition, it alerts security analysts to suspicious activities. An IDS can be categorized as a signature- or anomaly-based IDS [2]. A signature-based IDS uses a database of known attacks to effectively detect malicious activity, but it cannot detect new, variant, or unknown attacks. An anomaly-based IDS identifies threats by detecting deviations from the expected system behavior. It is more effective at detecting new or unknown attacks, but is prone to false positives (FPs).

An IDS can be implemented as a host, application, or network-based system. A host-based IDS monitors the malicious activities of individual computers by analyzing system logs, system files, and other data. An application-based IDS analyzes the network traffic and application behavior to identify suspicious activities. These are typically used to protect web applications, databases, and other critical systems from malicious attacks. In addition, they can be used to detect insider threats, such as employees accessing sensitive data without authorization. Network-based IDSs (NIDSs) monitor networks for malicious activities and suspicious traffic. They analyze network traffic for known attack patterns and anomalies and issue security alerts when suspicious activity or traffic is detected.

Security analysts at security operation centers (SOCs) investigate real-time events presented as security alerts by NIDSs to identify critical incidents [3]. However, this task is often hampered by alert fatigue caused by an immense number and constant frequency of alerts issued by security appliances [4,5]. One effective way to deal with this problem is to train high-accuracy machine learning (ML) algorithms to automate security alert data analysis.

A dataset is considered imbalanced if the number of instances in each class is significantly disproportionate [6]. The class imbalance problem [7–12] in security alert data, where negative alerts vastly outnumber positive alerts, is well known. This imbalance can lead to conventional classifiers being biased towards trivial alerts, which is detrimental to the detection of crucial alerts associated with critical incidents. Therefore, developing a method to mitigate the class imbalance in security alert data is essential for ensuring a fast incident response at SOCs.

Our previous work [13] addressed the class imbalance problem in security alert data using a support vector machine synthetic minority oversampling technique (SVMSMOTE). The experimental results showed that adding synthetic positive samples to the training data can effectively improve the recall rate for detecting positive alerts. Nevertheless, owing to the strong skewness in the alert dataset, an extraordinarily high oversampling rate was chosen to generate sufficient synthetic positive alert samples to obtain a balanced training dataset. This extremely high sampling rate resulted in a distorted distribution at the class boundaries, resulting in an undesirable high false positive rate (FPR) in the detection.

In this study, we introduced a bidirectional approach to cope with excessive skewness in the data. First, an ensemble of three oversampling methods was implemented to generate an augmented set of high-quality variational synthetic positive samples. Subsequently, a data subsampling algorithm was utilized to identify and remove ambiguous, noisy, and redundant negative samples. Classification models obtained from the merged augmented set (MAS), composed of synthetic positive and cleaned negative samples, could simultaneously yield an improved recall rate and FPR. The proposed approach was evaluated on an enterprise dataset collected from SOC operations in a network and the UNSW-NB15 benchmark dataset. The proposed method had a recall rate of 99.519% and an FPR of 0.119% on the enterprise dataset and a recall rate of 96.772% and an FPR of 0.232% on the benchmark dataset. Hence, it can be concluded that it outperformed conventional oversampling approaches by a large margin. This finding casts light on solving highly skewed class distribution problems commonly encountered in cybersecurity scenarios and paves the way for more effective and efficient artificial-intelligence (AI)-assisted security operations.

The main contributions of this study are summarized as follows:

- We propose a bidirectional approach to address imbalanced classification problems with extremely high skewness in the class distribution.
- We present a highly reproducible implementation of the bidirectional approach with conventional oversampling and data subsampling methods and a benchmark dataset.
- As proof of concept, we show the effectiveness of the proposed approach in detecting critical alerts from security alert logs issued by multiple security appliances.

The remainder of this paper is organized as follows. Section 2 discusses the current literature on the topic. Section 3 details the proposed methodology for the security alert data analysis. Section 4 presents the experimental setup and comparison of the results. Finally, Section 5 concludes the study and highlights ideas for future work.

## 2. Related Work

FPs are inconvenient for security analysts since they do not represent actual intruder activities. A high FPR reduces the value of issued alerts [14,15]. Compared with conventional approaches that fine-tune the rule base of NIDSs [16–18], a more sophisticated approach to reduce the FPR for security alert data is to use an ML algorithm to adjust the configuration of the NIDS automatically. In this section, we review recent works on AI-based approaches for coping with imbalanced security alert data and intrusion detection and prioritization.

### 2.1. Combating Data Imbalance

The prevalence of benign network traffic events often causes the alert data generated by NIDS to be imbalanced, with more benign events than malicious ones. Studies have

investigated methods for handling imbalanced issues using resampling, cost sensitivity, and clustering techniques.

#### 2.1.1. Resampling

Various techniques have been proposed to address the imbalance in alert data, such as random undersampling [8], which involves selecting and deleting samples from the majority class, typically the negative class. Another common technique is overweighting the minority (positive) class instances [19,20] to increase the amount of data related to attack events by replication or duplication. The synthetic minority oversampling technique (SMOTE), Cluster-SMOTE, adaptive synthetic (ADASYN) algorithm, and generative adversarial network were adopted in [8,21–28] to present new information or variational data samples to the ML model. In addition, related works [8,29] explored a combination of undersampling and oversampling techniques to attain a balanced data distribution and improve the accuracy of classifiers.

#### 2.1.2. Cost-Sensitive Learning

Cost-sensitive approaches assign different weights to positive and negative instances. Previous studies [8,30] investigated the use of cost-sensitive learning to combat class imbalance through a weighted support vector machine (SVM) [31] and repeated incremental pruning to produce error reduction algorithms [32].

#### 2.1.3. Clustering

Clustering involves grouping samples into clusters to minimize the variance within each cluster [33]. In [34,35], clustering techniques were combined with active learning approaches, where random selection, *k*-means clustering, and *k*-means clustering with bagging and aggregation of observations were integrated to provide diverse samples to human experts. In their approach, active learning was adopted to enable human experts to label unlabeled data efficiently.

### *2.2. Intrusion Detection and Prioritization*

Intrusion detection and prioritization are commonly used for investigation; after alert identification, operators will only address alerts in the top 10% of the list. A detection and prioritization model to design an alert correlation system [19,36] based on the extended Dempster–Shafer theory was investigated to identify the top alerts for the analysis listed in ascending order of importance. The study in [37] leveraged a deep neural network [38] to prioritize and respond to intrusion alerts using an automated alert-triaging process. Important events in the logs were extracted and passed to security operators. Moreover, in [39,40], an unsupervised learning approach for alert prioritization was studied by adopting an isolation forest and day-forward-chaining analysis to detect anomalies and high-priority alerts. Recently, provenance graphs [41–44] have also been used for alert prioritization by performing automated attack triage using historical information to assign threat scores to the alerts.

Approaches using deep learning techniques for intrusion detection [15] have been proposed. Deep learning is a type of AI that uses multiple layers of nonlinear processing units to learn from data with automatic feature learning and is scalable for large volumes of data. These intrusion detection approaches are based on convolutional neural networks (CNNs) [45–47] and CNNs with feature extraction [48] using methods such as principal component analysis and auto-encoders for dimension reduction. Other approaches are based on deep neural networks (DNNs) [49], few-shot learning with a combination of a CNN and a DNN [50] for feature extraction, and DNNs with deep stacked auto-encoders [51]. Deep learning approaches are used for network traffic classification into normal or attack classes or multi-class classification of various network attack types.

### 2.3. Limitations

Random undersampling involves discarding the majority class data, and these may contain important information essential for constructing rule-based classifiers, potentially leading to failure or decreased performance. However, naive oversampling by replicating minority class samples can lead to overfitting and, thus, less effective models [6]. Other oversampling techniques, such as SMOTE for nominal (SMOTEN), assume that the data to be oversampled have only categorical features. We discovered that, when used for security alert data oversampling, SMOTEN produced poor results. In addition, some oversampling techniques do not consider neighboring examples from other classes, leading to increased class overlap and additional noise. Moreover, they have limitations when used in extremely imbalanced data scenarios. Cost-sensitive learning techniques are likely to fail if positive instances are sparse, whereas clustering algorithms are prone to overfitting.

Some intrusion detection and prioritization methods encompass manual time-consuming alert analysis procedures, where a report is sent to security operators with alerts listed in ascending order of importance. Deep learning approaches are characterized by complex data models, making them extremely expensive to train, and the lack of feature extraction in some of these approaches leads to a long training time. In addition, most of these approaches concentrate on evaluation metrics that are not well suited for highly imbalanced datasets, such as accuracy and recall, because most benchmark datasets are not highly imbalanced. In the proposed approach, an ensemble resampling technique is implemented to tackle the highly skewed data issue often observed in security alert data and to improve positive alert detection while keeping the FPR low.

## 3. Methodology

In this section, we describe the benchmark dataset and the proposed security alert data analysis method, as illustrated in Figure 1. The proposed framework consists of four modules: security alert generation, preprocessing, resampling, and classification and analysis. First, we provide the details of the benchmark dataset, followed by a detailed description of each module in the system framework.

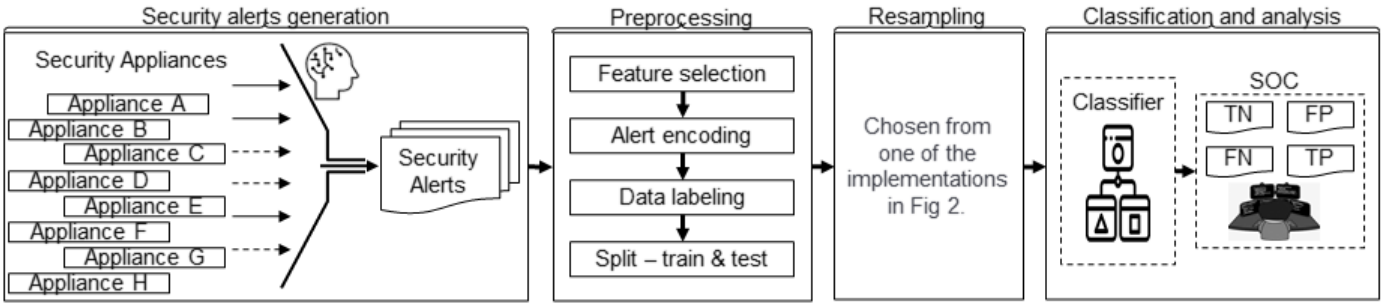



**Figure 1.** Proposed system framework for AI-assisted security alert data analysis.

### 3.1. Benchmark Dataset

The UNSW-NB15 dataset was created using the IXIA PerfectStorm tool in the Cyber Range Lab of UNSW Canberra to generate a hybrid of normal activities and attack behaviors. The dataset has nine types of attacks: fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode, and worms [52].

### 3.2. Security Alert Generation

In the proposed method, to generate the enterprise alert dataset, an SIEM system integrates multiple NIDSs deployed in a network, providing a central monitoring point to monitor the network status and detect cyber threats. Most security appliance vendors adopt the common event format (CEF) [53], an event interoperability standard introduced by ArcSight, Inc. Owing to the flexibility of the CEF format, alert logs issued by different security appliances often convey different types of information, making data integration

difficult. To address this, a data unification component synopsizes log files from different NIDSs into a single centralized view and outputs standard JavaScript object notation (JSON) [54] objects representing the enterprise dataset for further analysis.

### 3.3. Preprocessing

Preprocessing involves selecting relevant features, encoding alerts, labeling data, and splitting data into training and testing sets. The enterprise and benchmark datasets were preprocessed differently; the enterprise dataset went through all the preprocessing stages, while the benchmark dataset went through alert encoding and data splitting.

#### 3.3.1. Feature Selection

JSON objects represent the enterprise dataset that stores event data, including the appliance identification (ID), uniform resource locator (URL), Internet Protocol (IP) addresses, event description, event impact, downloaded file hash values, port numbers, and timestamps. Information entropy [55] was used to divide attributes into three types: numerical, categorical, and signature. Subsequently, security alert attributes with better generalization performance were carefully selected and used as features to represent alerts from the enterprise dataset.

#### 3.3.2. Alert Encoding

Categorical features for enterprise and benchmark datasets were represented in a binary format by encoding each alert message in a log file as a numerical vector using one-hot encoding [56]. Subsequently, the alerts were represented by binary vectors of the same length.

#### 3.3.3. Data Labeling

Security experts in SOC meticulously investigated the enterprise dataset alerts with deterministic evidence gathered to manually identify vital incidents. They examined the content of the communications captured in archived packet capture. They determined the criticality level of each alert after analyzing files using a sandbox, verifying the contents of the accessed URL, and comparing them with URL blacklists, among other methods. Owing to the alert fatigue that commonly occurs during security alert investigations, there is a substantial chance that security experts will miss or ignore highly critical alerts. Therefore, we hired AI experts to use visualization tools to review labeled alerts. Alerts with key incidents were labeled as highly critical and identified as positive alerts based on the gravity of the possible impact.

#### 3.3.4. Data Splitting

The enterprise and benchmark alert datasets were split into training and test sets, and subsequently, stratified $k$-fold cross-validation was applied. Stratified $k$-fold cross-validation is an extension of standard $k$-fold cross-validation, which ensures that the ratio between the target classes in each fold is the same as that in the original dataset. This method is particularly useful when dealing with imbalanced datasets, such as security alert data, in which the negative class is significantly larger than the positive class. Cross-validation results in more robust models, as it evaluates the performance of the model on unseen data, and it helps reduce the risk of overfitting by providing an estimate of the generalization performance of the model. In addition, it ensures that the model is not overly sensitive to small variations in the training data.

### 3.4. Resampling

The resampling module includes imbalanced, oversampling, and ensemble resampling for training data generation, as illustrated in Figure 2.

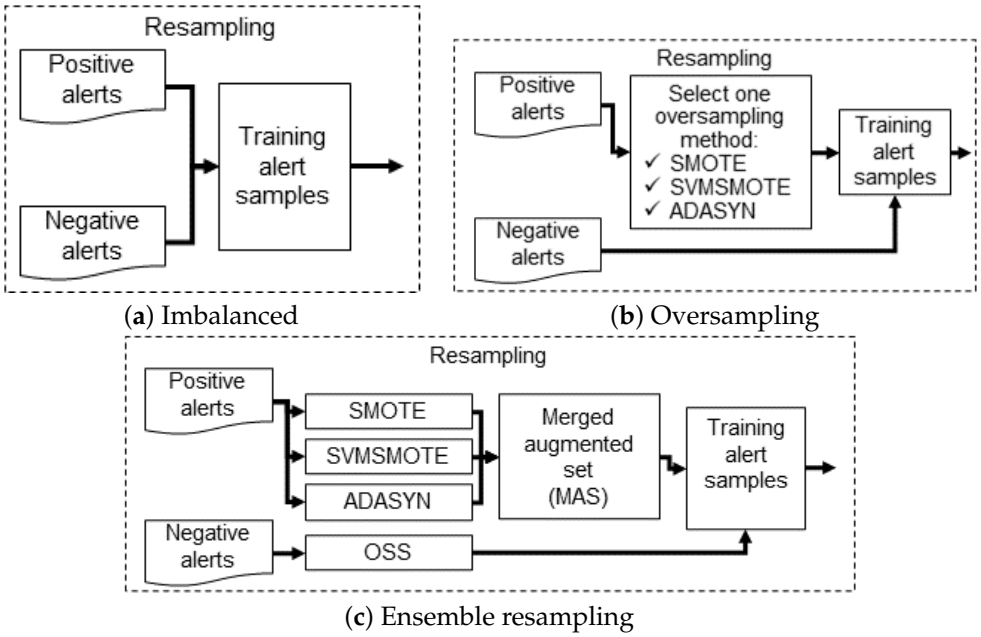



**Figure 2.** Security alert training samples generation.

#### 3.4.1. Imbalanced Data

Figure 2a outlines the process for obtaining imbalanced security alert training samples, which comprises preprocessed positive and negative alert samples. As a baseline setting, the original class ratio was maintained in the dataset.

#### 3.4.2. Oversampling

Figure 2b shows the training sample generation process, which leverages oversampling techniques. We selected the three most-popular data augmentation algorithms [57] for synthetic security alert data generation: SMOTE [22], ADASYN [23], and SVMSMOTE [58]. The selected algorithm was adopted to generate synthetic positive samples, which were subsequently combined with the corresponding negative samples to prepare the training samples. The oversampling algorithms are described in detail below:

- SMOTE utilizes the *k*-nearest neighbors (*k*-NN) algorithm to create synthetic samples. The method begins by choosing a random sample from the minority class. Subsequently, synthetic samples are generated by interpolating the line segments between the selected sample and its *k* nearest neighbors (NNs). It generates new synthetic samples close to the feature space between two sample pairs based on their local density and borders with the other class [59]. Synthetic samples were generated by taking the difference between the selected sample and its neighbor and multiplying this difference by a random number between 0 and 1, resulting in the random selection of a point in the line segment. This procedure broadens the decision region of the augmented class.
- ADASYN is a modification of SMOTE that uses a density distribution to determine the number of synthetic samples generated for each sample. A density distribution measure assigns more weight to samples that are more challenging to learn, forcing the learning algorithm to focus on these samples. In contrast to SMOTE, which generates the same number of synthetic samples for each point, ADASYN generates a different number of synthetic samples based on an estimate of the local distribution of chosen samples.
- SVMSMOTE is also an extension of SMOTE, which uses selective synthetic sample generation to determine which samples should be oversampled. In SVMSMOTE, the borderline area is approximated by support vectors after training an SVM classifier on the original training samples [60]. Synthetic samples are randomly created along

the lines joining each support vector in the minority class with a number of NNs [58]. This method focuses more on where the data are separated into different classes, and more samples are synthesized away from the overlapping regions of different classes.

The important parameters of the oversampling algorithms are listed in Table 1.

**Table 1.** Oversampling and undersampling parameters.

| Parameters | Algorithms | Definition |
| --- | --- | --- |
| *sampling_strategy* | SMOTE; ADASYN; SVMSMOTE | Sampling rate used to oversample the dataset |
| *k_neighbors* | SMOTE; SVMSMOTE | Number of NNs used to generate synthetic samples |
| *n_neighbors* | ADASYN | Number of NNs used to generate synthetic samples |
| *threshold_cleaning* | NCR | Determine condensed nearest neighbor application |
| *ss_n_neighbors* | OSS; NCR | Neighborhood size for a sample and its NNs |

#### 3.4.3. Ensemble Resampling

Figure 2c shows the training data generation process using ensemble resampling. First, each oversampling method generated a subset of synthetic positive samples. Second, the generated subsets were merged to form the MAS. Third, a data subsampling method was employed to obtain clean negative alert samples. Finally, the training data were obtained by combining the MAS and cleaned negative samples. MAS and data subsampling methods are described in detail below:

- Merged Augmented Set: We selected and implemented an ensemble of three of the most-effective data augmentation algorithms to generate synthetic security alert data: SMOTE, ADASYN, and SVMSMOTE. The ensemble resampling technique was used to generate an MAS of variational synthetic positive samples with high quality to mitigate the weaknesses arising from using a high oversampling rate, resulting in an improved FPR. The advantage of using an MAS includes a reduced sampling rate for each oversampling technique with less possibility of ill-formed synthetic samples and a more realistic distribution mimicking real data, as density-based methods are engaged.
- Subsampling Negative Security Alerts: Data subsampling methods select data samples to retain or delete [57,60]. We found that approaches that combine deletion and retention work best for subsampling security alert data. We compared the performance of two data subsampling methods for data deletion and retention: one-sided selection (OSS) [6,61] and the neighborhood cleaning rule (NCR) [62]. These two algorithms were used to identify and remove ambiguous, noisy, and redundant security alert samples from the negative class.
  - The OSS algorithm combines the condensed nearest neighbor algorithm [63] and the Tomek links [64]. The condensed nearest neighbor algorithm is an undersampling algorithm that identifies a minimal consistent set and removes redundant and ambiguous negative samples. Tomek links remove noisy and borderline negative samples. Data processing was performed as follows: First, using the condensed nearest neighbor, all positive samples and one randomly selected negative sample were added to a store. Second, the NN rule was used to classify training samples using the alerts in the store and to compare the assigned labels with the original ones. All misclassified samples were moved into the store, which is consistent with the training samples, while being smaller. Finally, all the negative samples participating in Tomek links were removed from the store to

remove borderline and noisy negative samples. All the positive samples were retained, and the resultant set was a set of clean samples.

– The NCR algorithm combines the condensed nearest neighbor and edited nearest neighbor (ENN) algorithms [65]. ENN is an undersampling algorithm that uses $k$ NNs to identify and remove misclassified or noisy negative samples before applying the $k$-NN classification algorithm. First, the algorithm selects all the positive samples. All noisy samples in the negative class are then identified and removed using the ENN rule. Finally, a one-step condensed nearest neighbor removes the remaining negative samples that are misclassified against the store, but only if the number of negative samples is larger than half the size of the positive class.

The crucial parameters of the OSS and NCR algorithms are listed in Table 1.

### 3.5. Classification and Analysis

This section covers the algorithms used to classify security alert data into positive and negative classes. We selected five classification algorithms that can efficiently handle high-dimensional sparse data: logistic regression (LR), $k$-NN, decision trees (DTs) [66,67], extreme gradient boosting (XGBoost) [68], and light gradient boosting machine (LightGBM) [69]. We used these algorithms to build prediction models for security alert data and subsequently evaluated them using stratified $k$-fold cross-validation.

LR models calculate the probability of a discrete outcome given an input variable. This is a robust and flexible method for predicting binary outcomes or states, making it a valuable analytical tool for classification problems. LR is particularly useful for alert analysis, because it can determine whether a new sample fits best into a given class. The algorithm predicts the probability that $y$ is associated with input variable $X$ using the logistic function Equation (1).

$$P(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{1}$$

where $\beta_0$ is the bias or intercept term and $\beta_1$ is the coefficient for the single input value ($x_i$).

$k$-NN is a simple, easy-to-implement supervised ML algorithm. It is assumed that similar samples exist in close proximity to each other. In a classification task, the algorithm determines the distance between a query sample and all samples in the data and selects the specified number of $k$ NNs, and subsequently, the NNs define the label of the query sample by a majority vote. $k$-NN classifiers have a fixed user-defined constant for the number of neighbors that must be determined. Neighbors-based methods are known as non-generalizing ML methods because they store all the training data. $k$-NN uses the Euclidean distance method to compute the closest neighbors to a given data point, as shown in Equation (2).

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

DTs are supervised ML algorithms that can predict the data class by learning simple decision rules inferred from training data. It has a hierarchical tree structure consisting of root nodes, branches, internal nodes, and leaf nodes. A DT starts with a root node that does not have incoming branches. The outgoing branches from the root node feed into the internal nodes, also known as decision nodes. Based on the available features, both node types conduct evaluations to form homogenous subsets denoted by leaf or terminal nodes. The leaf nodes represent all possible outcomes within the dataset. DT learning employs a divide-and-conquer strategy by conducting a greedy search to identify optimal split points within a tree. This process of splitting is repeated in a top-down recursive manner until all or the majority of the samples have been classified under specific class labels. Information gain and Gini impurity are popular splitting criteria for decision-tree

models. These methods are used to select the best attribute at each node and help evaluate the quality of each test condition and determine its ability to classify samples into a class. The Gini index measures the total variance across $N$ classes and is defined by Equation (3), where *pmk* represents the proportion of training variables in the $m$th segment that belong to the $k$th class.

$$G = -\sum_{k=1}^{k} P_{mk}(1 - P_{mk}) \tag{3}$$

LightGBM is a distributed high-performance gradient boosting framework that uses tree-based learning algorithms for classification tasks. The algorithm creates DTs that grow leafwise, i.e., given a condition, only a single leaf is split, depending on the gain, and the leaf with the maximum delta loss is chosen to grow. This is in contrast to other boosting algorithms that grow tree-levelwise. LightGBM improves the gradient-boosting algorithm by adding automatic feature selection and focusing on boosting examples with significant gradients. The algorithm comprises the gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) techniques. In GOSS, different samples play different roles in calculating the information gain, in which an instance with greater gradients can add more to the information gain. GOSS retains instances with high gradients and eliminates those with limited gradients at random to maintain the precision of information gain estimation [69,70]. The mathematical analysis of GOSS is shown in Equation (4):

$$\hat{V}_j(d) = \frac{1}{n}\left(\frac{(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i)^2}{n_l^j(d)} + \frac{(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i)^2}{n_r^j(d)}\right) \tag{4}$$

where $\hat{V}_j(d)$ is the estimated variance gain over the subset. $A \cup B$, $A_l = \{x_i \in A : x_{ij} \leq d\}$; $A_r = \{x_i \in A : x_{ij} > d\}$; $B_l = \{x_i \in B : x_{ij} \leq d\}$; $B_r = \{x_i \in B : x_{ij} > d\}$; the coefficient $\frac{1-a}{b}$ was used to normalize the sum of the gradients over $B$ to the size of $A^c$. Thus, the estimated $\hat{V}_j(d)$ was used over a smaller instance subset instead of the accurate $V_j(d)$ over all instances to determine the split point. Simultaneously, the EFB technique is used by LightGBM to minimize the model complexity by bundling exclusive features into a single feature.

XGBoost is a supervised learning algorithm that implements gradient-boosted DTs. Gradient boosting is a technique for predicting a target variable by combining the estimates of a set of simpler and weaker models. The weak models are DTs trained sequentially to minimize the loss function. New models are created that predict the residuals or errors of prior models and are subsequently added together to make the final prediction. The gradient descent algorithm minimizes the loss when adding new models. XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function and penalty term for model complexity. The loss function is based on the difference between the predicted and the target outputs. Gradient boosting is a powerful technique for achieving state-of-the-art results for several ML tasks. For a dataset with $n$ examples, where $i$ represents each sample, XGBoost uses a loss function to build trees by minimizing Equation (5):

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_t \Omega(f_t) \tag{5}$$

where the first part represents the loss function that calculates the pseudo-residuals of the predicted $\hat{y}_i$ and the true value $y_i$ in each leaf, and the second part $\Omega(f) = \gamma T + \frac{1}{2}\lambda \parallel w \parallel^2$. $\gamma$ represents the user-definable penalty, which encourages pruning, and $T$ represents the number of terminal nodes or leaves in a tree. $\lambda$ is the regularization term intended to reduce the insensitivity of the prediction to individual observations, and $w$ represents the leaf weights and output value for the leaf.

## 4. Experiments

This section presents the experimental results of applying the proposed approach to NIDS alert logs collected from the SOC operations in an existing enterprise network. In addition, the results obtained from the benchmark dataset are presented. Three data analysis processes were conducted in this study: imbalanced, oversampling, and ensemble resampling. The performance of the proposed approach was compared with that of existing state-of-the-art security alert data analysis approaches.

### *4.1. Dataset Preprocessing*

First, experiments were conducted using an enterprise dataset comprising security alert logs issued by six NIDSs that were used to monitor a Class B network comprising mobile devices, servers, and personal computers with over 1000 users and 30,000 hosts. The logs contained 131.11 million security alerts issued by the NIDSs, referred to as Appliance IDs A, B, C, D, E, and F, for security and ethical reasons, over ten months from 1 January to 31 October 2017. Table 2 presents the dataset statistics.

**Table 2.** Statistics of the enterprise dataset.

| Appliance ID | (#) Total Negative | (#) Total Positive | Imbalance Rate (%) | (#) Unique Negative | (#) Unique Positive | Imbalance Rate (%) |
|---|---|---|---|---|---|---|
| A | 11.841 | 16443 | 0.139 | 2581 | 90 | 3.370 |
| B | 66.209 | 3658 | 0.006 | 2018 | 22 | 1.078 |
| C | 0.558 | 6782 | 1.201 | 1581 | 195 | 10.980 |
| D | 5,112 | 1386 | 21.330 | 101 | 19 | 15.833 |
| E | 15.792 | 1783 | 0.011 | 6827 | 26 | 0.379 |
| F | 36.396 | 280073 | 0.764 | 12863 | 64 | 0.495 |
| Total | 130.801 | 310125 | **0.237** | 25971 | 416 | **1.577** |

The overall data imbalance rates are shown in bold font.

Among all the alerts, only 310,125 were labeled as positive, while the rest were labeled as negative, yielding an imbalance rate of 0.237%. Unique alerts that contained distinct feature values were sampled from the dataset, and 416 positive and 25,971 negative samples were obtained, resulting in an imbalance rate of 1.577%. This sampled dataset can be efficiently loaded into the memory for further analysis. Figure 3 presents a sample security alert message issued by an enterprise security appliance representing an anomalous network interaction—an unusual port scan.

```
CEF:0|xxxxxx|Enterprise|9.1|netflow-anomaly|Netflow Anomaly|2|act=LOG cat=Anomalous Network
Interaction/Unusual Port Scan cn1=25 cn1Label=impact cn2=21xxxx cn2Label=IncidentId cn3=25
cn3Label=IncidentImpact cnt=1 cs1=xxxxxxxx:30fbe7df:xxxxxxxx cs1Label=detectionId
cs2=https://xxx.xxx.xxx.xxx.xxx/portal#/event/1707xxxx/1899xxxxxx/591xxxx?event_time\=2017-06-18
cs2Label=EventDetailLink deviceExternalId=1707638505:1899xxxxxx dpt=35728 dst=xxx.xxx.232.135 end=Jun 18
2017 17:27:58 JST externalId=xxxx354 proto=TCP src=xxx.xxx.9.18 start=Jun 18 2017 16:58:55 JST
```

**Figure 3.** A security alert sample.

Second, the experiments were conducted using the UNSW-NB15 benchmark dataset. Table 3 presents the dataset statistics. The dataset contained 257,680 records of 164,675 attacks and 93,005 normal traffic. To obtain an imbalanced dataset, we sampled 37,000 and 4000 records from the normal and attack data, respectively, which resulted in an imbalance rate of 9.756%. A sampled benchmark dataset was used to conduct the experiments.

**Table 3.** Statistics of the UNSW-NB15 benchmark dataset.

| Traffic | (#) Records | (#) Sample | Imbalance Rate (%) |
|---|---|---|---|
| Attack | 164,675 | 4000 | |
| Normal | 93,005 | 37,000 | **9.756** |
| Total | 257,680 | 41,000 | |

The data imbalance rate is shown in bold font.

### *4.2. Security Alert Visualization*

*t*-distributed stochastic neighbor embedding (*t*-SNE) [71] is a dimensionality reduction technique that is particularly well suited for visualizing high-dimensional datasets. The *t*-SNE algorithm measures the similarity between data points in a high-dimensional space and projects them onto a lower-dimensional space, thereby allowing us to effectively reduce the dimensionality of the data while preserving the local structure of the data. We mapped the negative, positive, and synthetic security alert samples from the enterprise datasets in a two-dimensional layout.

Figure 4 presents a *t*-SNE visualization of the enterprise dataset used in the experiments. Disk and cross-markers represent negative and positive alert samples, respectively, and the letters A–F indicate the appliance ID. The alerts were color coded according to the issuing appliance, with the same color indicating positive and negative samples from the same appliance. This figure indicates that the dataset was highly imbalanced, with most alerts being negative. In addition, the alerts were largely clustered according to the issuing appliance, as indicated by the letters A–F, with appliances E and F issuing the most alerts.

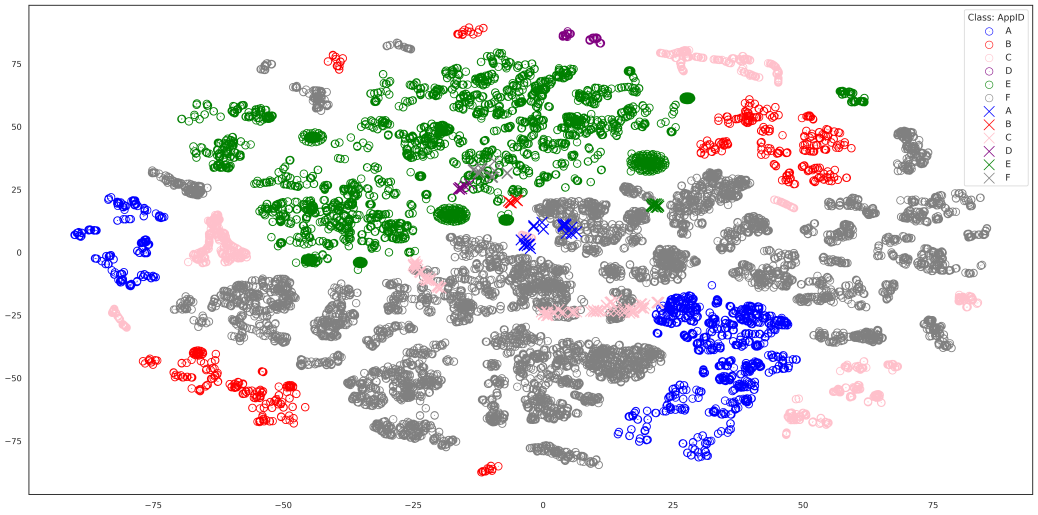
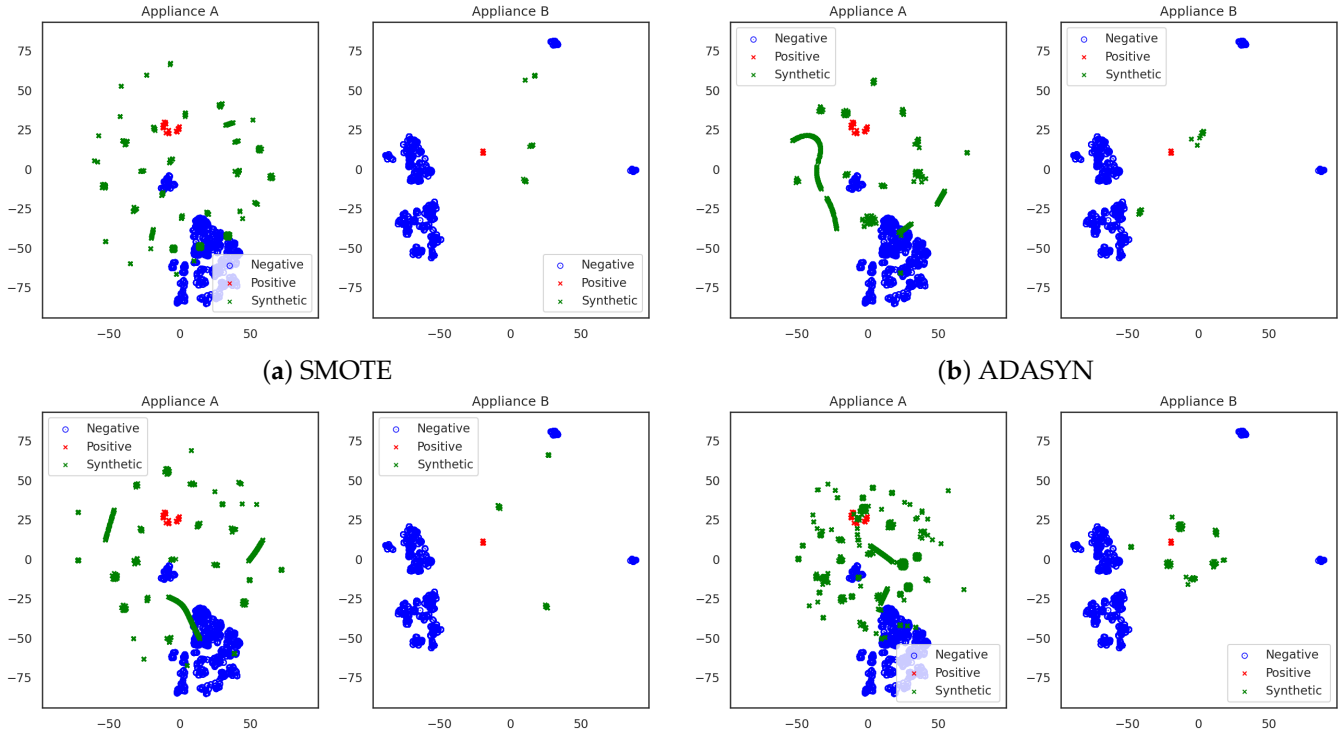



**Figure 4.** Visualizing unique security alert samples using *t*-SNE.

Figure 5 shows the distribution for the alert samples issued by Appliances A and B. The blue disks represent negative alert samples, and the red crosses represent positive samples. The figure indicates that negative samples comprise the majority class. Positive samples formed tight clusters and were highly separable from negative ones. Synthetic samples are shown as green crosses for SMOTE (Figure 5a), ADASYN (Figure 5b), SVMSMOTE (Figure 5c), and MAS (Figure 5d). As shown in Table 2, appliance A had more positive alert samples than Appliance B, resulting in more synthetic alert samples being generated for Appliance B by each oversampling method. Each method employs a different data generation scheme, resulting in a unique distribution of synthetic samples.

The alert distribution obtained using MAS (Figure 5d) alert samples resulted in high-quality synthetic alert samples clustered around the positive ones characterized by reduced class overlap. More data points were synthesized using MAS for Appliance B compared with the other methods, indicating that the method resulted in a more balanced data

distribution across all appliances. This suggests the potential for the improved performance of a security alert data analysis system trained on MAS alert samples.




(**a**) SMOTE (**b**) ADASYN

(**c**) SVMSMOTE (**d**) MAS

**Figure 5.** Visualizing positive, negative, and synthetic samples using *t*-SNE.

### *4.3. Evaluation Metrics*

The performance evaluation metrics included accuracy, recall, and precision. In addition, because the security alert data are highly imbalanced, we used the FPR, true negative rate (TNR), F-score, and balanced classification rate (BCR) for evaluating the performance of the proposed method. In this study, true positives (TPs) represent security alerts correctly classified as positive, FPs represent negative alerts incorrectly classified as positive, FNs represent positive alerts incorrectly classified as negative, and true negatives (TNs) represent correctly classified negative alerts. Precision can be more insightful than FPR because TNs are not used in its calculation; thus, they are not affected by class imbalances. We used precision–recall (PR) curves [72] and the PR area under the curve (PR-AUC) to evaluate the performance of the model. The PR-AUC is a model performance metric for binary responses appropriate for highly imbalanced datasets and is not dependent on model specificity. The PR curve plots the precision and recall values over the possible decision thresholds. The recall is plotted on the horizontal axis against the precision on the vertical axis. The performance of a classifier can then be assessed by computing the PR-AUC.

Accuracy is the percentage of security alert data samples correctly classified by a classifier, as shown in Equation (6).

$$Accuracy = \frac{TP + TN}{n} \tag{6}$$

The recall provides the proportion of correctly classified positive samples, as shown in Equation (7).

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Precision provides the proportion of correct positive class predictions, as shown in Equation (8).

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

The FPR is the number of FPs divided by the total number of actual negatives, as expressed using Equation (9).

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

The TNR is the number of correctly classified negative alerts divided by the total number of negative alerts, as expressed using Equation (10).

$$TNR = \frac{TN}{FP + TN} \tag{10}$$

The F-score can be interpreted as the weighted harmonic mean of the precision and recall, as shown in Equation (11).

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

BCR is the average of the recall and TNR, as shown in Equation (12).

$$BCR = \frac{Recall + TNR}{2} \tag{12}$$

### 4.4. Performance Evaluation

A security alert data analysis system should be evaluated based on its recall and FPR to ensure that it can effectively detect positive alert samples associated with critical events, ensuring that no malicious activity goes undetected while minimizing FPs [73]. A high recall is essential to guarantee the effectiveness of the model, whereas a low FPR is necessary to prevent alert fatigue.

The classifier algorithms mentioned in Section 3.5 were used for the performance evaluation. The experimental results for the evaluation metrics presented in this section were obtained using a stratified 10-fold cross-validation to validate the performance of the algorithms. The data were split into ten equally sized subsets, each containing an approximately equal proportion of the target class. Subsequently, the model was trained and tested on each of the ten subsets, and each subset was used as a testing set once and as a training set nine times. This process was repeated until all ten subsets were used as the testing set.

Next, the performance of the proposed system was evaluated. First, using the enterprise dataset, the baseline performance of the algorithms is presented using imbalanced data, followed by the results obtained with oversampled data. The results obtained using the ensemble resampled data are then provided. To validate the performance of the proposed approach, we present the evaluation results of the algorithms on the benchmark dataset for imbalanced, oversampled, and ensemble resampled data. In addition, the PR curves and their mean PR-AUC were calculated to assess the robustness of the algorithms. Finally, a performance comparison with existing state-of-the-art security data analysis approaches was conducted, the results of which are discussed in detail.

#### 4.4.1. Imbalanced Data

Table 4 shows the results obtained using imbalanced security alert data samples from the enterprise dataset. All classifiers achieved an accuracy rate of >99.341%. However, accuracy is not a reliable metric for measuring model performance when dealing with highly imbalanced datasets, as a model tends to be biased towards majority class samples. Therefore, metrics that are more appropriate for skewed data scenarios, such as recall,

FPR, and BCR, were adopted. LightGBM achieved the best recall and BCR of 98.317% and 99.095%, respectively, whereas XGBoost achieved the best FPR of 0.054%. Table 5 shows the results obtained using imbalanced data samples from the benchmark dataset. XGBoost had the best recall and BCR values of 95.940% and 97.881%, respectively, whereas LightGBM yielded the best FPR of 0.176%. Further performance improvement is desired to ensure that the system can accurately detect positive alert samples, thereby minimizing the risk of FNs while avoiding alert fatigue.

**Table 4.** Performance evaluation with different sampling settings on enterprise dataset.

| Algorithms | Accuracy (%) | Recall (%) | Precision (%) | FPR (%) | TNR (%) | F-Score (%) | BCR (%) |
|---|---|---|---|---|---|---|---|
| Imbalanced security alert data | | | | | | | |
| LR | 99.504 | 77.404 | 89.694 | 0.142 | 99.858 | 83.097 | 88.631 |
| *k*-NN | 99.341 | 91.346 | 73.359 | 0.531 | 99.469 | 81.370 | 95.407 |
| DT | 99.822 | 96.154 | 92.807 | 0.119 | 99.881 | 94.451 | 98.017 |
| LightGBM | 99.848 | **98.317** | 92.534 | 0.127 | 99.873 | 95.338 | **99.095** |
| XGBoost | **99.913** | 97.837 | **96.675** | **0.054** | **99.946** | **97.252** | 98.891 |
| Oversampled security alert data | | | | | | | |
| SMOTE | | | | | | | |
| LR | 98.950 | **99.760** | 60.058 | 1.063 | 98.937 | 74.977 | 99.348 |
| *k*-NN | 98.753 | 99.279 | 55.886 | 1.255 | 98.745 | 71.515 | 99.012 |
| DT | 99.538 | 99.519 | 77.528 | 0.462 | 99.538 | 87.158 | 99.529 |
| LightGBM | **99.883** | 98.798 | **94.050** | **0.100** | **99.900** | **96.366** | 99.349 |
| XGBoost | 99.814 | 99.519 | 89.805 | 0.181 | 99.819 | 94.413 | **99.669** |
| ADASYN | | | | | | | |
| LR | 99.000 | 92.067 | 62.378 | 0.889 | 99.111 | 74.369 | 95.589 |
| *k*-NN | 98.837 | **99.279** | 57.601 | 1.171 | 98.829 | 72.904 | 99.054 |
| DT | **99.860** | **99.279** | **92.394** | **0.131** | **99.869** | **95.713** | **99.574** |
| LightGBM | 99.841 | 98.798 | 91.741 | 0.142 | 99.858 | 95.139 | 99.328 |
| XGBoost | 99.818 | **99.279** | 90.175 | 0.173 | 99.827 | 94.508 | 99.553 |
| SVMSMOTE | | | | | | | |
| LR | 98.924 | **99.760** | 59.456 | 1.090 | 98.910 | 74.506 | 99.335 |
| *k*-NN | 98.833 | 98.077 | 57.627 | 1.155 | 98.845 | 72.598 | 98.461 |
| DT | 99.602 | 99.519 | 80.077 | 0.397 | 99.603 | 88.746 | 99.561 |
| LightGBM | **99.886** | 99.038 | **94.064** | **0.100** | **99.900** | **96.487** | 99.469 |
| XGBoost | 99.822 | 99.519 | 90.196 | 0.173 | 99.827 | 94.629 | **99.673** |
| Ensemble resampled security alert data | | | | | | | |
| MAS-NCR | | | | | | | |
| LR | 99.189 | **99.519** | 66.134 | 0.816 | 99.184 | 79.463 | 99.351 |
| *k*-NN | 99.208 | 98.558 | 66.884 | 0.782 | 99.218 | 79.689 | 98.888 |
| DT | 99.792 | 99.038 | 88.985 | 0.196 | 99.804 | 93.743 | 99.421 |
| LightGBM | 99.833 | **99.519** | 90.789 | 0.162 | 99.838 | 94.954 | 99.679 |
| XGBoost | **99.848** | **99.519** | **91.593** | **0.146** | **99.854** | **95.392** | **99.686** |
| MAS-OSS | | | | | | | |
| LR | 99.090 | 99.279 | 63.538 | 0.913 | 99.087 | 77.486 | 99.183 |
| *k*-NN | 99.420 | 98.317 | 73.694 | 0.562 | 99.438 | 84.243 | 98.878 |
| DT | 99.826 | 99.038 | 90.749 | 0.162 | 99.838 | 94.713 | 99.438 |
| LightGBM | 99.867 | 99.279 | 92.809 | 0.123 | 99.877 | 95.935 | 99.578 |
| XGBoost | **99.875** | **99.519** | **93.034** | **0.119** | **99.881** | **96.167** | **99.700** |

LR, *k*-NN, DT, LightGBM, and XGBoost stand for logistic regression, *k*-nearest neighbors, decision trees, light gradient boosting machine, and extreme gradient boosting algorithms, respectively. The highest evaluation performance scores are shown in bold font.

**Table 5.** Performance evaluation with different sampling settings on the UNSW-NB15 dataset.

| Algorithms | Accuracy (%) | Recall (%) | Precision (%) | FPR (%) | TNR (%) | F-Score (%) | BCR (%) |
|---|---|---|---|---|---|---|---|
| Imbalanced security alert data | | | | | | | |
| LR | 90.545 | 7.704 | 73.944 | 0.300 | 99.700 | 13.953 | 53.702 |
| *k*-NN | 96.753 | 71.313 | 94.768 | 0.435 | 99.565 | 81.384 | 85.439 |
| DT | 98.895 | 94.180 | 94.689 | 0.584 | 99.416 | 94.434 | 96.798 |
| LightGBM | 99.426 | 95.818 | **98.368** | **0.176** | **99.824** | 97.076 | 97.821 |
| XGBoost | **99.435** | **95.940** | 98.345 | 0.178 | 99.822 | **97.128** | **97.881** |
| Oversampled security alert data | | | | | | | |
| SMOTE | | | | | | | |
| LR | 85.205 | 73.955 | 37.621 | 13.551 | 86.449 | 49.872 | 80.202 |
| *k*-NN | 88.868 | 79.384 | 46.524 | 10.084 | 89.916 | 58.666 | 84.650 |
| DT | 98.866 | 95.158 | 93.556 | 0.724 | 99.276 | 94.350 | 97.217 |
| LightGBM | 99.392 | **96.674** | 97.197 | 0.308 | 99.692 | 96.935 | 98.183 |
| XGBoost | **99.438** | 96.625 | **97.700** | **0.251** | **99.749** | **97.160** | **98.187** |
| ADASYN | | | | | | | |
| LR | 44.725 | 96.209 | 14.850 | 60.965 | 39.035 | 25.729 | 67.622 |
| *k*-NN | 82.302 | 83.639 | 34.122 | 17.846 | 82.154 | 48.469 | 82.897 |
| DT | 98.642 | 95.500 | 91.260 | 1.011 | 98.989 | 93.332 | 97.245 |
| LightGBM | 99.323 | **97.188** | 96.060 | 0.441 | 99.559 | 96.620 | **98.374** |
| XGBoost | **99.384** | 96.870 | **96.941** | **0.338** | **99.662** | **96.905** | 98.266 |
| SVMSMOTE | | | | | | | |
| LR | 83.713 | 75.079 | 35.114 | 15.332 | 84.668 | 47.849 | 79.874 |
| *k*-NN | 89.289 | 77.867 | 47.665 | 9.449 | 90.551 | 59.133 | 84.209 |
| DT | 98.698 | 95.133 | 92.049 | 0.908 | 99.092 | 93.566 | 97.113 |
| LightGBM | 99.333 | 96.698 | 96.604 | 0.376 | 99.624 | 96.651 | 98.161 |
| XGBoost | **99.384** | **96.723** | **97.079** | **0.322** | **99.678** | **96.901** | **98.201** |
| Ensemble resampled security alert data | | | | | | | |
| MAS-NCR | | | | | | | |
| LR | 89.459 | 58.547 | 47.594 | 7.124 | 92.876 | 52.506 | 75.711 |
| *k*-NN | 90.883 | 78.234 | 52.832 | 7.719 | 92.281 | 63.072 | 85.258 |
| DT | 98.644 | 96.014 | 90.880 | 1.065 | 98.935 | 93.376 | 97.474 |
| LightGBM | 99.413 | **96.992** | 97.111 | 0.319 | 99.681 | 97.051 | **98.337** |
| XGBoost | **99.440** | 96.698 | **97.654** | **0.257** | **99.743** | **97.174** | 98.221 |
| MAS-OSS | | | | | | | |
| LR | 89.503 | 58.596 | 47.767 | 7.081 | 92.919 | 52.630 | 75.758 |
| *k*-NN | 92.716 | 76.889 | 60.555 | 5.535 | 94.465 | 67.751 | 85.677 |
| DT | 98.747 | 95.525 | 92.166 | 0.897 | 99.103 | 93.815 | 97.314 |
| LightGBM | 99.409 | 96.552 | 97.481 | 0.276 | 99.724 | 97.014 | 98.138 |
| XGBoost | **99.469** | **96.772** | **97.873** | **0.232** | **99.768** | **97.319** | **98.270** |

LR, *k*-NN, DT, LightGBM, and XGBoost stand for logistic regression, *k*-nearest neighbors, decision trees, light gradient boosting machine, and extreme gradient boosting algorithms, respectively. The highest evaluation performance scores are shown in bold font.

#### 4.4.2. Oversampled Data

To reduce the impact of the majority class on the detection performance of the security alert data analysis system, minority class data augmentation algorithms were implemented to balance the data distribution and improve the performance. Table 4 presents the results of applying the SMOTE, ADASYN, and SVMSMOTE oversampling techniques to the enterprise dataset. The positive alert detection performance was improved for all algorithms, with LR yielding the highest recall of 99.760% when SMOTE and SVMSMOTE were used for oversampling, but it had a high FPR of 1.063% and 1.090%, respectively. When

ADASYN was used for oversampling, LR achieved a recall of 92.067%. Meanwhile, when the data were left imbalanced, the algorithm achieved a recall of 77.404%. Two good models with high recall and low FPR were obtained using the LightGBM and XGBoost algorithms using SVMSMOTE oversampled security alert data. LightGBM yielded the best FPR of 0.100%. However, XGBoost outperformed it in terms of recall and BCR, achieving values of 99.519% and 99.673%, respectively. When imbalanced data were used, the XGBoost algorithm yielded an FPR of 0.054%. However, when a high oversampling rate of >40% was chosen to generate sufficient synthetic positive alert samples to combat data imbalance, the FPR degraded to 0.173%. Similar degradation in performance of the FPR was observed in the LR, *k*-NN, and DT algorithms.

Table 5 presents the results of applying SMOTE, ADASYN, and SVMSMOTE oversampling techniques to the benchmark dataset. Similar to the previous case, all algorithms achieved a performance improvement in the recall metric with LightGBM yielding the highest recall of 97.188% when ADASYN was used for oversampling. However, the algorithm exhibited a high FPR of 0.441%. For a security alert data analysis system to be effective, it is essential to maintain a low FPR such that security operators are not inundated with non-critical security alerts.

#### 4.4.3. Ensemble Resampled Data

The proposed approach leveraged MAS and a data-subsampling algorithm to maintain a positive alert detection performance while simultaneously improving the FPR. A low sampling rate for minority class data augmentation algorithms was selected to reduce the FPR caused by distorted distributions at the class boundaries. The sampling rate, represented by the *sampling_strategy* parameter, is considered high when it surpasses 40% and low when it is less than 40%. The *sampling_strategy* for each oversampling method was set to 30% to ensure a low sampling rate.

The results presented in Table 4 demonstrate that the proposed approach enhances the FPR reduction capabilities of the algorithms while maintaining positive alert detection performance. The improved FPR was reflected in most algorithms tested. XGBoost yielded the best FPR of 0.119% when using MAS-OSS. Table 5 presents the results on the ensemble resampled benchmark dataset, which further validates the improvement in the FPR performance brought about by the proposed approach. The FPR of XGBoost of 0.251% when using SMOTE oversampled data improved to 0.232% when using MAS-OSS. XGBoost excels because of its ability to leverage the power of ensembles of DTs to create more accurate and robust models with improved generalization performance. XGBoost-MAS-OSS was adopted in the proposed system, resulting in superior performance on all seven evaluation metrics for the enterprise and benchmark datasets. Notably, the high recall rate of the system indicates its effectiveness in detecting positive alerts. Simultaneously, the improved FPR suggests that the number of non-critical security alerts investigated by security analysts in the SOC has been reduced, thus addressing the alert fatigue issue, which has been a persistent challenge in the security industry for several years.

#### 4.4.4. Precision–Recall Curve

The PR curves for the experimentally evaluated algorithms of this study are shown in Figure 6, along with the mean PR-AUC score and standard deviation, which suggests the robustness of the model. LightGBM and XGBoost achieved the highest performance using imbalanced data (Figure 6a), with mean PR-AUC scores of 98.570% and 99.006%, respectively. The performance of LightGBM was further improved when oversampled data obtained using SVMSMOTE (Figure 6d) were used, resulting in a mean PR-AUC of 99.342%. Moreover, XGBoost achieved performance improvement with MAS-OSS (Figure 6f), yielding a mean PR-AUC of 99.315%. These results are consistent with the high performance of XGBoost-MAS-OSS in simultaneously detecting positive alerts with a low FPR, as demonstrated by other evaluation metrics.

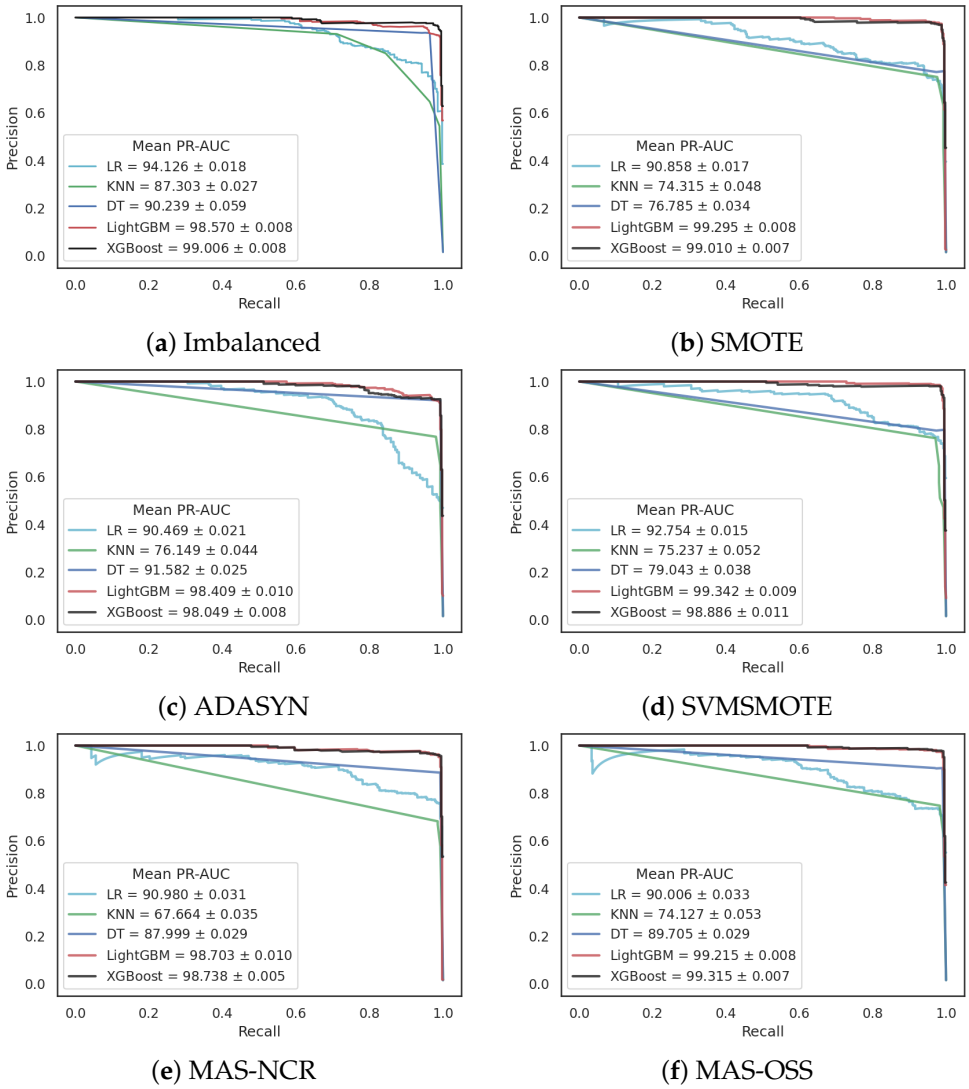



**Figure 6.** Precision–recall curves for different sampling settings on enterprise dataset.

#### 4.4.5. Performance Comparison

Table 6 presents a performance comparison of the proposed approach and the existing state-of-the-art security alert data analysis approaches. Unsupervised learning methods from [39,40], isolation forest (IF), and IF with day-forward chaining (IF-DC) were used for comparison. The results from [13], obtained using DT combined with SVMSMOTE (DT-SVMSMOTE), are also included for reference. Deep learning methods, CNN, and long short-term memory (LSTM), combined with MAS-OSS, were also used for comparison. Deep learning (CNN-MAS-OSS and LSTM-MAS-OSS) and unsupervised learning approaches (IF and IF-DC) yielded high FPRs, resulting in a large number of false alerts that require manual investigation. The proposed approach, XGBoost-MAS-OSS, yielded an FPR of 0.119%, outperforming all compared security alert analysis approaches by a large margin. Table 7 presents a performance comparison of the proposed approach with the deep learning approaches evaluated on the UNSW-NB15 dataset. The dataset is not highly skewed, and therefore, most of the cited approaches do not require techniques to handle class imbalance; hence, they did not employ such techniques. Therefore, the results are not directly comparable, as we sampled the attack data to obtain a class distribution closely representative of our highly skewed enterprise dataset. However, the proposed approach achieved superior scores in all evaluation metrics. These findings highlight the potential of the proposed approach in solving the highly skewed class distribution problems commonly

encountered in cybersecurity scenarios, thus paving the way for more effective and efficient AI-assisted security operations.

**Table 6.** Performance comparison with related work on the enterprise dataset.

| Algorithms | Accuracy (%) | Recall (%) | Precision (%) | FPR (%) | TNR (%) | F-Score (%) | BCR (%) |
|---|---|---|---|---|---|---|---|
| IF [39] | 90.705 | 100.000 | 0.551 | 9.300 | 90.700 | 1.097 | 95.350 |
| IF-DC [40] | 94.141 | 95.876 | 0.837 | 5.860 | 94.140 | 1.659 | 95.008 |
| DT-SVMSMOTE [13] | 99.596 | **99.524** | 79.771 | 0.403 | 99.597 | 88.559 | 99.560 |
| CNN-MAS-OSS | 83.685 | 73.558 | 6.798 | 16.153 | 83.847 | 12.447 | 78.703 |
| LSTM-MAS-OSS | 99.185 | 95.433 | 66.948 | 0.755 | 99.245 | 78.692 | 97.339 |
| XGBoost-MAS-OSS | **99.875** | 99.519 | **93.034** | **0.119** | **99.881** | **96.167** | **99.700** |

IF, IF-DC, DT-SVMSMOTE, CNN-MAS-OSS, LSTM-MAS-OSS and XGBoost-MAS-OSS stand for isolation forest, IF with day-forward chaining, decision tree with support vector machine synthetic minority oversampling technique, convolutional neural network with merged augmented set and one-sided selection, long short-term memory with MAS-OSS, and extreme gradient boosting with MAS-OSS.

**Table 7.** Performance comparison with related work on the UNSW-NB15 dataset.

| Algorithms | Accuracy (%) | Recall (%) | Precision (%) | F-Score (%) |
|---|---|---|---|---|
| DNN [49] | 78.400 | 72.500 | 94.400 | 82.000 |
| Few-shot learning [50] | 92.000 | – | – | 92.110 |
| Stacked autoencoder-softmax [51] | 89.134 | 63.270 | 89.134 | 90.850 |
| Autoencoder-CNN [47] | 93.400 | – | – | 95.290 |
| LSTM-MAS-OSS | 86.967 | 59.770 | 39.714 | 47.720 |
| CNN-MAS-OSS | 91.187 | 78.772 | 53.917 | 64.017 |
| XGBoost-MAS-OSS | **99.469** | **96.772** | **97.873** | **97.319** |

DNN, CNN, LSTM-MAS-OSS, CNN-MAS-OSS, and XGBoost-MAS-OSS stand for deep neural networks, convolutional neural networks, long short-term memory with merged augmented set and one-sided selection, CNN with MAS-OSS, and XGBoost with MAS-OSS.

## 5. Conclusions

This study proposed a bidirectional approach to cope with excessive skewness in security alert data analysis scenarios. The proposed approach implemented an ensemble of carefully selected oversampling methods to generate the MAS of variational synthetic positive alert samples with high quality. In addition, a data subsampling algorithm was employed to identify and remove ambiguous, noisy, and redundant negative alert samples. The results obtained using an enterprise and the UNSW-NB15 benchmark datasets demonstrate that classification models obtained using MAS-OSS can simultaneously yield improved recall and FPR, with the proposed system, XGBoost-MAS-OSS, achieving superior performance on all evaluation metrics. Notably, the high recall rate of the system indicates its effectiveness in detecting positive alerts. Moreover, the improved FPR suggests that the number of non-critical security alerts investigated by security analysts in the SOC has been reduced, thus addressing the alert fatigue issue, which has been a persistent challenge in the security industry for several years. This study focuses on data resampling methods to address the class imbalance problem in security alert data analysis. In future works, other methods for imbalanced security alert data classification, such as cost-sensitive kernel modification, active learning, and hybrid approaches, can be explored to enhance the performance of the proposed system. In addition, recently, research on intrusion analysis has used deep learning techniques such as CNN and DNN, where feature extraction approaches such as auto-encoders and deep stacked auto-encoders are used for data structure improvement and model complexity reduction. These approaches should be investigated in future studies.

**Author Contributions:** Conceptualization, S.N., T.B. and T.T.; methodology, S.N., T.B. and T.T.; software, S.N.; validation, T.B., T.T. and D.I.; formal analysis, S.N.; investigation, S.N.; resources, T.T. and D.I.; data curation, S.N. and T.B.; writing—original draft preparation, S.N.; writing—review and editing, S.N., T.B., T.T. and D.I.; visualization, S.N.; supervision, T.B., T.T. and D.I.; funding acquisition, D.I. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Ministry of Internal Affairs and Communications grant number JPJ000254.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The enterprise datasets generated and analyzed during the current study are not publicly available, and we cannot provide them in any sense. The benchmark UNSW-NB15 publicly available datasets were analyzed in this study. This data can be found here: [52]. The source code will be available upon request.

**Acknowledgments:** This research was conducted under a contract of "MITIGATE" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan.

**Conflicts of Interest:** The authors declare that they have no conflict of interest.

## References

1. Alkahtani, H.; Aldhyani, T. Botnet Attack Detection by Using CNN-LSTM Model for Internet of Things Applications. *Secur. Commun. Netw.* **2021**, *2021*, 3806459. [CrossRef]
2. Alkahtani, H.; Aldhyani, T. Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms. *Complexity* **2021**, *2021*, 9851. [CrossRef]
3. Zomlot, L.; Sundaramurthy, S.C.; Luo, K.; Ou, X.; Rajagopalan, S.R. Prioritizing Intrusion Analysis Using Dempster-Shafer Theory. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence (AISec '11), Chicago, IL, USA, 21 October 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 59–70.
4. Axelsson, S. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 186–205. [CrossRef]
5. Zhang, K.; Luo, S.; Xin, Y.; Zhu, H.; Chen, Y. Online Mining Intrusion Patterns from IDS Alerts. *Appl. Sci.* **2020**, *10*, 2983. [CrossRef]
6. Fernández, A.; García, S.; Galar, M.; Prati, R.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Springer International Publishing: New York, NY, USA, 2018. [CrossRef]
7. Chen, J.; Lalor, J.; Liu, W.; Druhl, E.; Granillo, E.; Vimalananda, V.; Yu, H. Detecting Hypoglycemia Incidents Reported in Patients' Secure Messages: Using Cost-sensitive Learning and Oversampling to Reduce Data Imbalance (Preprint). *J. Med. Internet Res.* **2018**, *21*, e11990. [CrossRef] [PubMed]
8. Cieslak, D.; Chawla, N.; Striegel, A. Combating imbalance in network intrusion datasets. In Proceedings of the 2006 IEEE International Conference on Granular Computing, Atlanta, GA, USA, 10–12 May 2006; pp. 732–737. [CrossRef]
9. Soe, Y.N.; Santosa, P.I.; Hartanto, R. DDoS Attack Detection Based on Simple ANN with SMOTE for IoT Environment. In Proceedings of the 2019 Fourth International Conference on Informatics and Computing (ICIC), Rome, Italy, 16–17 October 2019; pp. 1–5. [CrossRef]
10. Jadhav, A.; Mostafa, S.M.; Elmannai, H.; Karim, F.K. An Empirical Assessment of Performance of Data Balancing Techniques in Classification Task. *Appl. Sci.* **2022**, *12*, 3928. [CrossRef]
11. Rendón, E.; Alejo, R.; Castorena, C.; Isidro-Ortega, F.J.; Granda-Gutiérrez, E.E. Data Sampling Methods to Deal With the Big Data Multi-Class Imbalance Problem. *Appl. Sci.* **2020**, *10*, 1276. [CrossRef]
12. Oliveira, N.; Praça, I.; Maia, E.; Sousa, O. Intelligent Cyber Attack Detection and Classification for Network-Based Intrusion Detection Systems. *Appl. Sci.* **2021**, *11*, 1674. [CrossRef]
13. Ndichu, S.; Tao, B.; Takeshi, T.; Daisuke, I. A Machine Learning Approach to Detection of Critical Alerts from Imbalanced Multi-Appliance Threat Alert Logs. In *Proceedings of the Workshop on Cyber Threat Intelligence and Hunting with AI, IEEE International Conference on Big Data (IEEE BigData 2021)*; IEEE Xplore Digital Library: Hoboken, NJ, USA, 2021; pp. 1–8.
14. Lee, E.; Lee, Y.; Lee, T. Automatic False Alarm Detection Based on XAI and Reliability Analysis. *Appl. Sci.* **2022**, *12*, 6761. [CrossRef]
15. Vanin, P.; Newe, T.; Dhirani, L.L.; O'Connell, E.; O'Shea, D.; Lee, B.; Rao, M. A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning. *Appl. Sci.* **2022**, *12*, 1752. [CrossRef]
16. Ullah, I.; Mahmoud, Q.H. A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks. In Proceedings of the 2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC), Las Vegas, NV, USA, 11–14 January 2019; pp. 1–6. [CrossRef]

17. Zhao, Y.; Zheng, Z.; Wen, H. Bayesian Statistical Inference in Machine Learning Anomaly Detection. In Proceedings of the 2010 International Conference on Communications and Intelligence Information Security, Madrid, Spain, 13 October 2010; pp. 113–116. [CrossRef]
18. Roughan, M.; Griffin, T.; Mao, Z.M.; Greenberg, A.; Freeman, B. IP Forwarding Anomalies and Improving Their Detection Using Multiple Data Sources. In *NetT '04, Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality, Portland, OH, USA, 30 August 2004*; Association for Computing Machinery: New York, NY, USA, 2004; pp. 307–312.
19. Zomlot, L.; Chandran, S.; Caragea, D.; Ou, X. Aiding Intrusion Analysis Using Machine Learning. In Proceedings of the 2013 12th International Conference on Machine Learning and Applications, Miami, FL, USA, 4–7 December 2013; Volume 2, pp. 40–47. [CrossRef]
20. Zainel, H.; Koçak, C. LAN Intrusion Detection Using Convolutional Neural Networks. *Appl. Sci.* **2022**, *12*, 6645. [CrossRef]
21. Kumar, R.S.S.; Wicker, A.; Swann, M. Practical Machine Learning for Cloud Intrusion Detection: Challenges and the Way Forward. In *AISec '17, Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 81–90.
22. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority over-Sampling Technique. *J. Artif. Int. Res.* **2002**, *16*, 321–357. [CrossRef]
23. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008; pp. 1322–1328. [CrossRef]
24. Gonzalez-Cuautle, D.; Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Portillo-Portillo, J.; Olivares-Mercado, J.; Perez-Meana, H.M.; Sandoval-Orozco, A.L. Synthetic Minority Oversampling Technique for Optimizing Classification Tasks in Botnet and Intrusion-Detection-System Datasets. *Appl. Sci.* **2020**, *10*, 794. [CrossRef]
25. Sarhan, B.B.; Altwaijry, N. Insider Threat Detection Using Machine Learning Approach. *Appl. Sci.* **2023**, *13*, 259. [CrossRef]
26. Mohammadpour, L.; Ling, T.C.; Liew, C.S.; Aryanfar, A. A Survey of CNN-Based Network Intrusion Detection. *Appl. Sci.* **2022**, *12*, 8162. [CrossRef]
27. Yilmaz, I.; Masum, R.; Siraj, A. Addressing imbalanced data problem with generative adversarial network for intrusion detection. In Proceedings of the 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI), Online, 11–13 August 2020; pp. 25–30.
28. Ndichu, S.; Tao, B.; Takeshi, T.; Daisuke, I. Security-Alert Screening with Oversampling Based on Conditional Generative Adversarial Networks. In *Proceedings of the 2022 17th Asia Joint Conference on Information Security (AsiaJCIS)*; IEEE Xplore Digital Library: Hoboken, NJ, USA, 2022; pp. 1–7.
29. Bagui, S.; Li, K. Resampling imbalanced data for network intrusion detection datasets. *J. Big Data* **2021**, *8*, 6. [CrossRef]
30. Ban, T.; Samuel, N.; Takahashi, T.; Inoue, D. Combat Security Alert Fatigue with AI-Assisted Techniques. In *CSET '21, Proceedings of the Cyber Security Experimentation and Test Workshop, Vancouver, BC, Canada, 14 August 2021*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 9–16.
31. Chang, C.C.; Lin, C.J. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 1199. [CrossRef]
32. Arvin, A. *Medical Applications of Artificial Intelligence*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2013.
33. Duan, H.; Wei, Y.; Liu, P.; Yin, H. A Novel Ensemble Framework Based on K-Means and Resampling for Imbalanced Data. *Appl. Sci.* **2020**, *10*, 1684. [CrossRef]
34. McElwee, S.; Cannady, J. Cyber Situation Awareness with Active Learning for Intrusion Detection. In Proceedings of the 2019 SoutheastCon, Huntsville, AL, USA, 11 April 2019; pp. 1–7. [CrossRef]
35. Krishnakumar, A. *Active Learning Literature Survey*; Technical Reports; University of California: Santa Cruz, CA, USA, 2007; Volume 42.
36. Sundaramurthy, S.C.; Zomlot, L.; Ou, X. Practical IDS Alert Correlation in the Face of Dynamic Threats. In Proceedings of the 2011 International Conference on Security & Management, Las Vegas NV, USA, 18–21 July 2011; pp. 186–192.
37. McElwee, S.; Heaton, J.; Fraley, J.; Cannady, J. Deep learning for prioritizing and responding to intrusion detection alerts. In Proceedings of the MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM), Baltimore, MA, USA, 23–25 October 2017; pp. 1–5. [CrossRef]
38. Onchis, D.; Istin, C.; Hogea, E. A Neuro-Symbolic Classifier with Optimized Satisfiability for Monitoring Security Alerts in Network Traffic. *Appl. Sci.* **2022**, *12*, 1502. [CrossRef]
39. Aminanto, M.E.; Zhu, L.; Ban, T.; Isawa, R.; Takahashi, T.; Inoue, D. Combating Threat-Alert Fatigue with Online Anomaly Detection Using Isolation Forest. In *Proceedings of the Neural Information Processing*; Gedeon, T., Wong, K.W., Lee, M., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 756–765.
40. Aminanto, M.E.; Ban, T.; Isawa, R.; Takahashi, T.; Inoue, D. Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis. *IEEE Access* **2020**, *8*, 217977–217986. [CrossRef]
41. Hassan, W.U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; Bates, A. NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. In Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 24–27 February 2019; pp. 1–15.

42. Liu, Y.; Zhang, M.; Li, D.; Jee, K.; Li, Z.; Wu, Z.; Rhee, J.; Mittal, P. Towards a Timely Causality Analysis for Enterprise Security. In Proceedings of the 25th Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 18–21 February 2018; pp. 1–15.
43. Pasquier, T.; Han, X.; Moyer, T.; Bates, A.; Hermant, O.; Eyers, D.; Bacon, J.; Seltzer, M. Runtime Analysis of Whole-System Provenance. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1601–1616.
44. Milajerdi, S.; Eshete, B.; Gjomemo, R.; Venkatakrishnan, V.N. ProPatrol: Attack Investigation via Extracted High-Level Tasks. In *Proceedings of the Information Systems Security*; Ganapathy, V., Jaeger, T., Shyamasundar, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 107–126.
45. Zhang, X.; Chen, J.; Zhou, Y.; Han, L.; Lin, J. A Multiple-Layer Representation Learning Model for Network-Based Attack Detection. *IEEE Access* **2019**, *7*, 91992–92008. [CrossRef]
46. Chen, L.; Kuang, X.; Xu, A.; Suo, S.; Yang, Y. A Novel Network Intrusion Detection System Based on CNN. In Proceedings of the 2020 8th International Conference on Advanced Cloud and Big Data (CBD), Taiyuan, China, 5–6 December 2020; pp. 243–247. [CrossRef]
47. Andresini, G.; Appice, A.; Mauro, N.D.; Loglisci, C.; Malerba, D. Multi-Channel Deep Feature Learning for Intrusion Detection. *IEEE Access* **2020**, *8*, 53346–53359. [CrossRef]
48. Xiao, Y.; Xing, C.; Zhang, T.; Zhao, Z. An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks. *IEEE Access* **2019**, *7*, 42210–42219. [CrossRef]
49. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
50. Yu, Y.; Bian, N. An Intrusion Detection Method Using Few-Shot Learning. *IEEE Access* **2020**, *8*, 49730–49740. [CrossRef]
51. Khan, F.A.; Gumaei, A.; Derhab, A.; Hussain, A. A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection. *IEEE Access* **2019**, *7*, 30373–30385. [CrossRef]
52. Moustafa, N. The UNSW-NB15 Dataset. 2013. Available online: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 15 January 2023).
53. Marwaha, N. System and Method for Providing Common Event Format Using Alert Index. U.S. Patent 7,139,938, 24 July 2006.
54. ECMA, I. The JSON Data Interchange Format. 2013. Available online: https://www.ecma-international.org/wp-content/uploads/ECMA-404_1st_edition_october_2013.pdf (accessed on 1 May 2021).
55. Huang, K. *Statistical Mechanics*; John Wiley & Sons: Hoboken, NJ, USA, 2008.
56. Jackson, E.; Agrawal, R. Performance Evaluation of Different Feature Encoding Schemes on Cybersecurity Logs. In Proceedings of the 2019 SoutheastCon, Huntsville, AL, USA, 11 April 2019; pp. 1–9. [CrossRef]
57. Lemaître, G.; Nogueira, F.; Aridas, C.K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *J. Mach. Learn. Res.* **2017**, *18*, 1–5.
58. Nguyen, H.; Cooper, E.; Kamei, K. Borderline over-sampling for imbalanced data classification. *Int. J. Knowl. Eng. Soft Data Paradig.* **2011**, *3*, 4–21. [CrossRef]
59. Haibo, H.; Yunqian, M. *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st ed.; Wiley-IEEE Press: Piscataway, NJ, USA, 2013.
60. Brownlee, J. *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*; Machine Learning Mastery: Vermont, VC, Canada, 2020.
61. Kubat, M.; Matwin, S. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *Proceedings of the 14th International Conference on Machine Learning*; Morgan Kaufmann: New York, NY, USA, 1997; pp. 179–186.
62. Laurikkala, J. Improving Identification of Difficult Small Classes by Balancing Class Distribution. In *Proceedings of the Artificial Intelligence in Medicine*; Quaglini, S., Barahona, P., Andreassen, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 63–66.
63. Hart, P. The condensed nearest neighbor rule (Corresp.). *IEEE Trans. Inf. Theory* **1968**, *14*, 515–516. [CrossRef]
64. Tomek, I. Two Modifications of CNN. *IEEE Trans. Syst. Man, Cybern.* **1976**, *SMC-6*, 769–772. [CrossRef]
65. Wilson, D.L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Trans. Syst. Man, Cybern.* **1972**, *SMC-2*, 408–421. [CrossRef]
66. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
67. Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; et al. API design for machine learning software: Experiences from the scikit-learn project. In Proceedings of the ECML PKDD Workshop: Languages for Data Mining and Machine Learning, Prague, Czech Republic, 23–27 September 2013; pp. 108–122.
68. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
69. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 3149–3157.

70. Osman, M.; He, J.; Mokbal, F.M.M.; Zhu, N.; Qureshi, S. ML-LGBM: A Machine Learning Model Based on Light Gradient Boosting Machine for the Detection of Version Number Attacks in RPL-Based Networks. *IEEE Access* **2021**, *9*, 83654–83665. [CrossRef]
71. Laurens, v.d.M.; Geoffrey, H. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
72. Jesse, D.; Mark, G. The Relationship Between Precision-Recall and ROC Curves. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; Volume 6. [CrossRef]
73. Ndichu, S.; Tao, B.; Takeshi, T.; Daisuke, I. Critical-Threat-Alert Detection using Online Machine Learning. In *Proceedings of the Workshop on Big Data for Cybersecurity (BigCyber), IEEE International Conference on Big Data (IEEE BigData)*; IEEE Xplore Digital Library: Osaka, Japan, 2022; pp. 3007–3014. [CrossRef]