electronics

Article

Cybersecurity in Smart Cities: Detection of Opposing Decisions on Anomalies in the Computer Network Behavior

Danijela Protic [1,*], Loveleen Gaur [2], Miomir Stankovic [3] and Md Anisur Rahman [4]

[1] Center for Applied Mathematics and Electronics, 11000 Belgrade, Serbia
[2] Amity International Business School, Amity University, Noida 201303, India
[3] Mathematical Institute of SASA, 11000 Belgrade, Serbia
[4] School of Computing, Mathematics, and Engineering, Charles Sturt University, Bathurst, NSW 2795, Australia
* Correspondence: adanijela@ptt.rs

Abstract: The increased use of urban technologies in smart cities brings new challenges and issues. Cyber security has become increasingly important as many critical components of information and communication systems depend on it, including various applications and civic infrastructures that use data-driven technologies and computer networks. Intrusion detection systems monitor computer networks for malicious activity. Signature-based intrusion detection systems compare the network traffic pattern to a set of known attack signatures and cannot identify unknown attacks. Anomaly-based intrusion detection systems monitor network traffic to detect changes in network behavior and identify unknown attacks. The biggest obstacle to anomaly detection is building a statistical normality model, which is difficult because a large amount of data is required to estimate the model. Supervised machine learning-based binary classifiers are excellent tools for classifying data as normal or abnormal. Feature selection and feature scaling are performed to eliminate redundant and irrelevant data. Of the 24 features of the Kyoto 2006+ dataset, nine numerical features are considered essential for model training. Min-Max normalization in the range [0,1] and [−1,1], Z-score standardization, and new hyperbolic tangent normalization are used for scaling. A hyperbolic tangent normalization is based on the Levenberg-Marquardt damping strategy and linearization of the hyperbolic tangent function with a narrow slope gradient around zero. Due to proven classification ability, in this study we used a feedforward neural network, decision tree, support vector machine, k-nearest neighbor, and weighted k-nearest neighbor models Overall accuracy decreased by less than 0.1 per cent, while processing time was reduced by more than a two-fold reduction. The results show a clear benefit of the TH scaling regarding processing time. Regardless of how accurate the classifiers are, their decisions can sometimes differ. Our study describes a conflicting decision detector based on an XOR operation performed on the outputs of two classifiers, the fastest feedforward neural network, and the more accurate but slower weighted k-nearest neighbor model. The results show that up to 6% of different decisions are detected.

Keywords: anomaly detection; binary classification; feature scaling; machine learning

Citation: Protic, D.; Gaur, L.; Stankovic, M.; Rahman, M.A. Cybersecurity in Smart Cities: Detection of Opposing Decisions on Anomalies in the Computer Network Behavior. *Electronics* **2022**, *11*, 3718. https://doi.org/10.3390/electronics11223718

Academic Editor: Byung-Gyu Kim

Received: 15 October 2022
Accepted: 11 November 2022
Published: 13 November 2022

1. Introduction

The rapid development of smart cities reveals computer network connectivity and interoperability issues and highlights the problems that can arise in large-scale heterogeneous data processing. These issues are obstacles to organic efforts to improve urban intelligence and environmental sustainability while offering significant potential for key technologies and engineering practices in data-driven smart city systems. Understanding data management is important for unlocking smart cities [1,2]. Leveraging real-time data improves the operational efficiency, connectivity, decision-making, and overall performance of Internet of Things (IoT)-based computer networks and communications platforms for data collection, device management, and cloud solutions. The authors of [3] provide a realistic view of how organizations can evolve to the next level of maturity and how the forces

driving this transition can adopt and benefit from IoT. Cloud services enable excessive connectivity between various IoT devices and sensors, resulting in billions of connected devices and massive amounts of data. Three principles must be followed: data must be sent over multiple channels dynamically, it must be secure, and it must be scalable. With the development of smart city technology, network security threats have become an important obstacle to maximizing the benefits of data-driven technologies, and intrusion detection has become an important prerequisite for protecting sensitive data [4,5].

The intrusion detection system (IDS) is derived from the human immune system (HIS), which consists of humoral immunity that protects the body from pathogens from outside the body (similar to detecting malicious attacks). Similarly, cell-mediated immunity reacts to self cells deviation (a negative selection process related to detecting abnormalities) [6].

The primary purpose of an intrusion detection system is to monitor network traffic to detect patterns (signatures) of malicious attacks or deviations from standard network functionality. A signature-based IDS compares the anonymous network data patterns against a known set of attack signatures. It is the simplest and most effective method against various common attacks. However, the performance of the signature-based IDS is limited to known attacks, i.e., the detector cannot identify unknown attacks. On the other hand, to detect changes in network behavior anomaly-based IDS monitors the state of the network traffic and generates alerts when abnormalities are detected [7,8]. Anomaly detection's main benefit is identifying previously unknown suspicious behavior or known malicious activity. The biggest challenge is determining what is considered normal computer network behavior. Developing statistical models for normal network behavior is difficult because model evaluation requires a large amount of data, which takes time and storage [9].

Binary classifiers based on supervised machine learning (ML) are good candidates for detecting "normality", although they require large amounts of data [10]. The authors of [11] provide publication citation statistics for various ML techniques collected from 2005 to 2020. The results show that the most cited articles are related to Support Vector Machines (SVMs), followed by publications on neural networks, Decision Trees (DTs), and nearest-neighbor models.

In this study, we present five standard binary classifiers: the k-Nearest Neighbor (k-NN), weighted k-NN (wk-NN), DT, SVM, and Feedforward Neural Network (FNN). A k-NN is the most well-known distance-based algorithm that assigns a new instance to a class to which most of its k nearest neighbors belong [12,13]. A k-NN model with k = 10 and a similarity measure is based on Euclidean distance because of its robustness to noisy data, flexibility, and easy implementation [14]. The wk-NN model is used because it extends the k-NN model to improve the accuracy by heavily weighting neighbors in the decision who are closer to the new instance than neighbors who are more distant [15]. The weights are calculated as the inverse square of the Euclidean distance [16]. Medium Gaussian SVM provides high prediction speed in binary classification [15]. The model classifies instances in n-dimensional space using a hyperplane. The model uses a hyperplane to classify instances in n-dimensional space [17]. DT models predict the class label in input data based on decisions from the root to the leaf nodes [18,19]. Due to its high prediction speed and low memory costs, medium DT (Iterative Dichotomiser 3 algorithm) with 20 splits is used [16]. A feedforward neural network (FNN) with one hidden layer (nine input, nine hidden, and one output neuron) is used due to its fast processing speed and generalization ability [14,15]. It is one of the simplest and quickest models that rely on backpropagation to produce results based on the predicted probabilities and classification thresholds. The calculation is done by transferring the data from the input to the output and then propagating the error of the cost function backward to adjust the weights [20].

The Kyoto 2006+ dataset was used as a benchmark for the experiments because it was created for anomaly detection and contained records of more than ten years of actual network traffic data collected from honeypots on five computer networks within and outside Kyoto University [21]. Of 24 features, 14 statistical features are extracted from the KDD Cup '99 dataset, and 10 additional features were added by the authors for

further network analysis and evaluation of the other network-based intrusion detection systems [22]. The Kyoto 2006+ dataset provides labeled instances that do not describe exact attack-specific details, but give a separation between normal and abnormal network traffic [23].

All irrelevant features of the Kyoto 2006+ dataset are removed using feature selection and feature scaling techniques. Feature selection is used to remove all categorical features, connection duration features, statistical features and features intended for further analysis. After identifying the nine relevant numerical features, the Min-Max normalization in the ranges [0,1] and [−1,1], Z-score standardization, and a novel tangent-hyperbolic (TH) normalization are used for scaling.

The idea of TH normalization is to scale the features to a fixed range $[-0.7616, +0.7616]$, i.e., $[tanh(-1), tanh(1)]$, and then use a Levenberg-Marquardt (LM) damping strategy to speed up training and improve model performance [24,25]. The results show that TH scaling provides clear benefits in reducing processing time, which improves the efficiency of IoT and cloud computing operations [26].

The performance of the above models is compared in terms of processing time, accuracy, F1-score, false alarms, and true positive rate. Each classifier needs to make very accurate decisions about anomalies. However, regardless of the overall accuracy of the classifiers, when working in parallel, one may consider network traffic normal while the other detects anomalies, and vice versa. This paper proposes an XOR-based detector to detect unusual conflicting decisions in computer networks.

In our previous work [14,15], we introduced the concept of XOR detection. Since the result of the XOR operation is 1 if the two bits are different, if the decisions are other (for non-zero bits), the total number of different decisions can be calculated as the sum of all the results. The XOR-based detector is designed to compare the outcomes of two binary classifiers: FNN (eager learner) and wk-NN (lazy learner). The results show a small percentage of conflicting decisions that are not affected by record size, model accuracy, or processing time. However, it can be used to warn not only of anomalies but also of potentially harmful activities in smart city computer networks concerning privacy, data breaches, and more.

The remainder of this paper is organized as follows: Section 2 presents a systematic review of relevant references on anomaly-based intrusion detection. The Kyoto 2006+ dataset is compared with 13 other datasets most commonly used for IDS experiments. Feature selection and feature scaling are briefly introduced and the TH normalization is described in detail. Section 3 introduces the concept of an XOR-based conflicting decisions detector. Section 4 presents the experimental results. Section 5 concludes this paper.

2. Related Work

ML models are proven to be essential and efficient in detecting risks and threats to computer networks. Many ML algorithms attempt to find data trends by comparing the features of data points. Supervised ML algorithms are widely used in computer network traffic analysis. The performance evaluation and comparative analysis of supervised ML used in classification are presented in [11,27–32].

The authors of [1] present a literature review of machine learning techniques used in various smart city applications. Research methods include the classification, estimation, and performance of machine learning algorithms classified into one of four categories: DT, SVM, artificial neural network (ANN), and advanced machine learning methods, including deep learning (DL), ensembles, and hybrid approaches. The study found that ensembles and hybrid methods perform better due to their high accuracy and low total cost as compared to DL. However, these methods take more time to process than single methods. Furthermore, it has been shown that SVM and DT outperform ANN in terms of accuracy and various other metrics. However, since the differences were small, the authors concluded that either model could be used. Performance evaluation and comparative analysis of FNN, SVM, DT, k-NN and wk-NN classifiers can be found in [3,17,28,31–40].

The authors of [41] propose a set of tests to investigate the effectiveness of supervised ML algorithms in anomaly detection. As a result of the survey findings, the model's evaluation is impeded by two major issues. First, accurate classification requires a large amount of labeled data, which takes time. Second, different feature scales necessitate feature scaling. If the input contains different data, the model may diverge, overestimate, underestimate or ignore some parameters, reducing the estimation efficiency [42,43].

The goal of feature selection is to find a subset of features that can reduce model complexity, minimize generalization errors, provide better predictive power, and provide fast model evaluation without significantly affecting its performance. Feature selection can be supervised [43,44], unsupervised [45,46] or semi-supervised [47,48], depending on whether the training set is labeled or not. In [17], the authors propose a feature selection method to save storage space and allow feature selection to speed up classification algorithms. The authors of [49] and [50] use feature selection to remove irrelevant features from the Kyoto 2006+ dataset. In [51], the authors propose an FPA algorithm, compare it with three other IDSs with 15 relevant features from the Kyoto 2006+ dataset, and show that Service, Flag, and Srv_serror_rate are the most essential features. The PDS method for feature selection has been introduced in [49]. The authors found 18 features relevant for k-NN. Service-aware partitioning of datasets is used to handle enormous data flexibility and improve classification accuracy and processing time as described in [50]. Eighteen features were found to be essential for anomaly detection.

In addition to feature selection, feature scaling is often used to reduce the mutual influence of features and their negative impact on model evaluation if they are on drastically different scales. The two most well-known feature scaling methods are normalization, which shifts and rescales features to a fixed range and ensures the consistency of all features, and Z-score standardization, which shifts the data according to a Gaussian distribution [17]. The authors in [52] and [53] describe Min-Max normalization for network intrusion using ML models on the selected Kyoto 2006+ dataset. Various feature selection methods have been proposed. In their study, the authors investigated the effects of Z-score and Min-Max normalization on the accuracy of the J48 classifier [54]. In [30], the authors presented a comparative analysis of ML-based classifiers for anomaly-based intrusion detection based on Z-score standardization as a pre-processing step. In [16], the authors summarize research on the effect of feature scaling within ± 1 range on DT, SVM, FNN, k-NN and wk-NN models. The results show that all classifiers have high accuracy and a short processing time.

Table 1 summarizes the most relevant research on feature selection (estimation efficiency reduction, feature relevance, storage and training time minimization), feature scaling (Min-Max normalization, Z-score standardization), and ML-based classification (FNN, DT, SVM, k-NN and wk-NN models).

Table 1. Summary of the related work.

Authors	Year	Feature Selection/Feature Scaling	Classifiers
Band et al. [34]	2022	The most informative feature. Min-Max.	ANN, DT, SVM
Lin et al. [13]	2022	Point-biserial selection. Cluster-center scaling.	k-NN
Shresta et al. [28]	2021	Estimation efficiency reduction. Feature relevance.	DT, k-NN
Al-Imran and Ripon [53]	2021	85 network flow features. Min-Max.	DT, k-NN
Kousis and Tjottjis [40]	2021	PCA. Normalization.	ANN, DT, k-NN, SVM
Pai et al. [31]	2021	Sequential search. Standardization	DT, SVM
Kumar et al. [11]	2021	PCA.	ANN, DT, k-NN, SVM
Protic and Stankovic [16]	2021	Numerical features selection. Min-Max.	DT, FNN, k-NN, SVM, wk-NN
Kumar et al. [30]	2020	ANOVA F-test, Z-score.	DT

Table 1. *Cont.*

Authors	Year	Feature Selection/Feature Scaling	Classifiers
Protic and Stankovic [15]	2020	Numerical features selection. Min-Max.	FNN, wk-NN
Obaid [54]	2019	PCA, Min-Max, Z-score.	DT
Ruggieri [18]	2019	Exact enumeration.	DT
Abiodun et al. [33]	2018	Estimation efficiency reduction.	ANN, FNN
Maza and Touharia [39]	2018	Incremental, decremental, random feature selection.	DT, SVM
Nawi et al. [42]	2013	Min-Max, Z-score.	ANN

2.1. The Kyoto 2006+ Dataset

Over the years, researchers have conducted intrusion detection experiments on a variety of data sets including ADFA, AWID, CAIDA, CIC-IDS-2017, CIDDS-001, CSE-CIC-2018, DARPA, IRSC, ISCX 2012, KDD Cup '99, Kyoto 2006+, NSL-KDD, UGR'16, and UNSW-NB15 [11,35,53,55–59]. Table 2 displays the most frequently used datasets.

Table 2. Description of the most frequently used datasets.

Dataset	Year of Creation	Types of Attacks	Number of Features	Type of Traffic	Content of the Dataset
ADFA	2014	Brute force, Java/Linux meterpreter, C100 webshell	26	Hybrid	Linux/Windows OS system call.
AWID	2015	Wi-fi 802.11 attacks	156	Emulated	Wireless LAN traffic.
CAIDA	2007	Distributed Denial of Service	Not used	Hybrid	Recorded on commercial backbone links from high speed monitors.
CIC-IDS-2017	2017	Botnets, DDoS, Goldeneye, Hulk, HTTP	80+	Emulated	5-day packet-based network traffic.
CIDDS-001	2017	DoS, Bruteforce, Ping/Port Scan	14	Emulated	4 weeks traffic form OpenStack and Ext. servers.
CSE-CIC-2018	2018	FTP/SSH potator, Dos, DDoS, Web attacks, 1st/2nd level infiltration, botnet.	80+	Emulated	10 days computer network traffic.
DARPA	1998–1999	DoS, R2L, U2R, probe	41	Emulated	7 weeks of packet-based traffic.
IRSC	2015	DoS, R2U, surveillance	Not available	Hybrid	Sudans university network.
ISCX 2012	2012	Infiltrating, DDoS, HTTP, SSH	20	Emulated	Packet-based traffic (7 days).
KDD Cup '99	1998	Denial of Service, R2L, U2R, probing	42	Emulated	5 weeks of packet-based traffic.
Kyoto 2006+	2006–2015	Port scan, malware, shellcode, DoS	24	Real	10 years of real network traffic.
NSL-KDD	1998	Denial of Service, R2L, U2R, probing	42	Emulated	KDD-Cup '99 dataset with redundant and duplicate records excluded.

Table 2. *Cont.*

Dataset	Year of Creation	Types of Attacks	Number of Features	Type of Traffic	Content of the Dataset
UGR'16	2016	Denial of Service, Portscans Botnet	41	Hybrid	Network traces were captured in tier-3 ISP for four months.
UNSW-NB15	2015	Contemporary attacks behavior,.	49	Hybrid	tcpdump traces over 31 h.

The experiments in this study are based on the Kyoto 2006+ dataset for several reasons. First, most datasets are emulated or hybrid, except for the Kyoto 2006+ dataset. The Kyoto 2006+ dataset was collected over the ten years from various computer networks inside and outside the University of Kyoto. The dataset's first version was created by collecting real network traffic data from 2006 to 2009 from ~350 honeypots including two darknet sensors with ~300 unused IP addresses and various other IDS. It contains about 1 billion instances of normal and abnormal data. A new dataset version includes ~20 GB of additional data collected from 2009 to 2015. In addition, although the Kyoto 2006+ dataset includes DoS, exploits, port scans, malware, and shellcode attacks, any other details about types of attacks are not given; there is no information about payload or packet traces. [36]. Table 3 describes the Kyoto 2006+ dataset.

Table 3. The Kyoto 2006+ dataset.

No	Feature	Description
1	Duration	Connection duration [s].
2	Service	Type of connection service.
3	Source bytes	# B sent by source IP address.
4	Destination bytes	# B sent by destination IP address.
5	Count	# of connections with same source/destination IP addresses to those of current connection in past 2s.
6	Same_srv_rate	% of connections to the same service in the feature Count
7	Serror_rate	% of connections that have 'SYN' errors in the feature Count.
8	Srv_error_rate	% of connections that have 'SYN' errors in Srv_count in past 2s.
9	Dst_host_count	Source/destination IP addresses are the same as the current connection (among past 100).
10	Dst_host_srv_count	The number of connections whose service type is also the same to that of the current connection.
11	Dst_host_same_src_port_rate	% of connections whose source port is the same to that of the current connection in Dst_host_count.
12	Dst_host_serror_rate	% of connections that have 'SYN' errors in Dst_host_count.
13	Dst_host_srv_serror_rate	% of connections that have 'SYN' errors in Dst_host_srv_count.
14	Flag	The state of the connection at the time of connection was written.
15	IDS_detection	Reflects if IDS triggered an alert for the connection.
16	Malware_detection	Indicates malware.
17	Ashula_detection	Shellcode and exploit codes were in the connection.
18	Label	Indicates an attack.
19	Source_IP_Address	Source IP address used in the session.

Table 3. *Cont.*

No	Feature	Description
20	Source_Port_Number	Session's source port number.
21	Destination_IP_Address	Also sanitized.
22	Destination_Port_Number	Session's destination port number.
23	Start_time	Start of the session.
24	Duration	Session duration.

IDS Bro is used to convert packet traffic into a session format. It is a network-based analytic system focusing on high-performance network security monitoring [22]. Bro's event engine receives Internet Protocol packets and converts them into events. The policy script interpreter (PCI) then generates the output. Table 4 describes an example of a session-based format.

Table 4. An instance from a daily record.

No	Type	Value
1	Statistical	0.52
2	Categorical	smtp
3	Statistical	3333
4	Statistical	244
5	Numeric	1.00
6	Numeric	1.00
7	Numeric	0.00
8	Numeric	0.00
9	Numeric	6.00
10	Numeric	99.00
11	Numeric	0.00
12	Numeric	0.00
13	Numeric	0.00
14	Statistical	SF
15	For further analysis	0
16	For further analysis	0
17	For further analysis	0
18	Numeric	1
19	Categorical	fdfd:c3e9:3c9c:264d:052b:4470:1f85:3407
20	Categorical	41339
21	Categorical	fdfd:c3e9:3c9c:9f52:7d2e: 27ee:079e:0f3f
22	Categorical	25
23	Categorical	00:00:36
24	For further analysis	0.523710

The main problem related in evaluating anomaly detectors on the Kyoto 2006+ dataset is the large amount of recorded data. The authors of [49] propose a PDS method for feature selection from the Kyoto 2006+ dataset to obtain 18 relevant features. The discarded features are prediction labels indicating the type of attack and the source and destination IP addresses. The authors have also discarded the Start_time feature due to it having many different values. In [50], the authors propose a technique for partitioning datasets with a service that can handle big data flexibly. The authors use features indexed 1–14, 20, 22, and 24 (18 features in total), because the Kyoto 2006+ dataset contains session-specific

information. The authors of [51] discuss classification problems caused by unwanted large features. They proposed the FPA algorithm and compared its performance with three other IDS algorithms using 15 features from the Kyoto 2006+ dataset. The results show that Service, Flag and Srv_serror_rate are the three most essential features.

In the experiments presented here, the problem of the dataset size issue is addressed using a feature selection step. Categorical features, statistical features on connection duration, and features for further analyses are removed from the data set. Nine numerical features remained for model evaluation. Feature *Label* (18) is used to identify sessions as normal or abnormal.

2.2. Evaluation Processes and Performance Metrics

Pre-processing, binary classification, and detection of opposing decisions are three mutually related experiments organized to follow the process from feature selection to the detection of the differences in the decisions about anomalies. This structured framework aims to show how novelties in the pre-processing step affect Accuracy, False alarms, True positive rate, F1-score, and processing time.

2.2.1. Pre-Processing Steps

The purpose of pre-processing is to clear the data set of irrelevant features, since the large amount of data required for model evaluation makes this process time-consuming. The authors of [60] argue that the main goal of collecting data from smart city computer networks is to obtain accurate and comprehensive data. These data are raw and noisy, coming from various sources, and must be pre-processed. Data pre-processing is needed because entering raw data to the training model will not produce acceptable results. The pre-processing shown here consists of two steps: (1) feature selection; and (2) feature scaling.

According to the authors of [61] and [62], the literature implies that users who are knowledgeable about their dataset can select features that match some criteria based on their knowledge and experience. Following this, the feature selection proposed in this paper is performed to remove all categorical features, connection duration features, statistical features, and features used for further analysis, as follows: (1) Remove all categorical features (17 features are left for model training: 1, 3–17, 24); (2) All statistical features and features intended for further analyses are cut. Finally, features 5–13 are used to evaluate the model. The feature *Label* is used to indicate the presence of an attack. The original data set has three labels: 1 for standard sessions, -1 for known attacks, and -2 for unknown attacks. However, since unknown attacks are sporadic in the dataset, we also assign label -1 to unknown attacks.

The second stage in the pre-processing step is to implement feature scaling, since the scale of these features varies extensively. We show the results for Z-score standardization, Min-Max normalization in range $[0, 1]$, and Min-Max normalization in range $[-1,1]$ to show that different feature scales can produce different results. We also introduce a novel TH feature scaling methodology inspired by the tangent-hyperbolic function (*tanh*), its sharp gradient (*tanh'*), and the damping strategy of the LM algorithm applied to the quasi-linear part of the *tanh* function.

The tangent hyperbolic $tanh(x)$ is an *S*-shaped, zero-centered function, with a very sharp gradient $anh(x)' = 1 - tanh(x)^2$, $x = 0$, i.e., $tanh(x)\prime_{x\to\pm 0} \approx 1$. Because a portion of the *tanh* function corresponding to the values $tanh(\pm 1) \approx \pm 0.7616$ can be considered (quasi) linear, this property can be used to constrain the instances to the same symmetrical fixed range $[-0.7616,\ 0.7616]$. The TH methodology scales features in such a way that n instances $x(i),\ i = 1, \ldots,\ n$ in the feature-vector can be determined as follows (Equation (1)):

$$x(i)_{TH} = tanh\left(\frac{x(i) - \frac{x_{Max}+x_{Min}}{2}}{\frac{x_{Max}-x_{Min}}{2}}\right), \quad (1)$$

where $x(i)_{TH}$ represents the scaled instance, and x_{Max} and x_{Min} represent maximum and minimum values of unscaled features, respectively. The nonlinear iterative LM algorithm combines the first-order gradient descent (GD) with the second-order Gauss-Newton (GN) algorithms to find the global minimum of the cost function $f : R \to R$. The GD minimizes f according to the search direction determined by the negative value of the gradient and the step size. It is accurate but slows down near the optimum. When this happens, the damping strategy switches from the GD to the much faster GN algorithm based on the second-order derivative. The LM algorithm approximates the calculation of the Hessian matrix (**H**) with the matrix products of the Jacobians (**J**) so that $\mathbf{H} \approx \mathbf{J}x\mathbf{J}^{\mathrm{T}}$ assuming that the error function is approximately quadratic near the optimal solution, following Taylor's truncated formula [62,63]. Assume that the iterates of the algorithm are $x^{(1)}$, $x^{(2)}$, ..., $x^{(m)}$ and p is the optimal solution of the algorithm. The LM algorithm finds the iterate $x^{(m+1)}$ by minimizing the first and second terms in the expression given in Equation (2).

$$\left|\left|\hat{f}\left(x, x^{(m)}\right)\right|\right|^2 + \lambda^{(m)}\left|\left|x, x^{(m)}\right|\right|, \ \lambda^{(m)} > 0, \tag{2}$$

such that $\hat{f}(x, p) \approx f(x)$, and $x \approx p$. $\lambda^{(m)}$ denotes the adaptive damping parameter, which varies with the step size. Therefore, the iterate $x^{(m+1)}$ can be determined as follows (see Equation (3)):

$$x^{(m+1)} = x^{(m)} - \left(\mathbf{J}^{\mathbf{T}}\mathbf{J} + \lambda^{(m)}\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}} f\left(x^{(m)}\right), \tag{3}$$

where **I** represents the identity matrix. When $\lambda^{(m)} \to \infty$ the LM algorithm works as a GD algorithm because $\mathbf{H} + \lambda^{(m)}\mathbf{I} \to \mathbf{I}$. Otherwise, if $\lambda^{(m)} \to 0$ the LM algorithm behaves like the GN algorithm because $x^{(m)}$ is close to the optimal solution.

2.2.2. Performance Metrics

A two-step classification scheme is used to determine which of two classes a new instance belongs to. First, the classifiers are trained using 70% of instances, and the remaining 30% is used to test the models. A binary confusion matrix is used to describe the measurement performance of classifiers given that the true values of the data set are known and the results consist of two classes. True negative (TN) and true positive (TP) values identify negative and correctly classified positive results. The false positive (FP, false alarm) value indicates the misclassification of normal data, whereas the false negative (FN) value denotes an incorrectly assigned anomaly. The false alarm exists when the observation is negative with positive prediction, i.e., indicates the value which is the number of actual negative examples classified as positive. The true positive rate (TPR), also known as Recall or Sensitivity, is the ratio of the correctly identified positive classes, as shown in Equation (4).

$$\mathrm{TPR} = \mathrm{Recall} = \mathrm{Sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{4}$$

The TPR indicates how well the model recognizes a positive class. It measures all possible classes (how many are correctly predicted) and must be as high as possible. It is useful when FN dominates FP. Also note the Positive predicted value (PPV), known as Precision, given in Equation (5).

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{5}$$

Precision describes how many possible predicted classes are positive and measures the probability that the positive class is correct. Since Precision shows the accuracy of the TP class, it should be as high as possible. When it is difficult to compare the models with

high Recall and low Precision, their harmonic mean, also known as F1-score or F1, can be used to indicate the similarity of the two results (See Equation (6)):

$$\mathrm{F1} = \frac{1}{\frac{1}{\mathrm{Recall}} + \frac{1}{\mathrm{Precision}}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}. \quad (6)$$

However, because the F1-score is difficult to interpret, it is unknown whether the classifier minimizes Precision or Recall. Therefore, a full picture of the results is provided when the F1-score is used in combination with other metrics. The accuracy of the classifier (ACC) given by Equation (7) determines how many classes are correctly predicted.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \quad (7)$$

Accuracy should be as high as possible. It represents the ratio of correct classifications to the total number of instances. It explains how often the model predicts the right result. In the results presented in this paper, classification performance is discussed in terms of ACC, FP, TPR, F1-score, and processing time ($tp = t_{train} + t_{test}$).

3. Proposed Work

In machine learning, classification refers to predicting a class label for a given instance of input data. A supervised ML model learns from the training set and its true labels and then makes predictions on the test set. Binary classification is used when a binary label is assigned to an unknown data instance. Figure 1 depicts a diagram of the classification process for five binary classifications presented in this work.

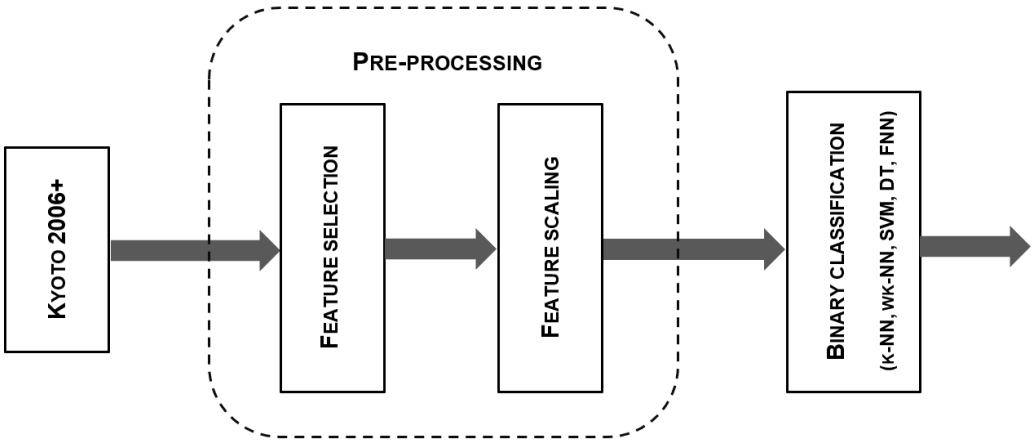

Figure 1. Binary classification process.

First, the Kyoto 2006+ dataset has been relieved of all irrelevant features. The feature scaling is then done. The classifier is trained using a known set of data instances in the training phase. The classifier is then tested on an unknown data set. Each classifier is expected to be highly accurate in decision-making.

However, regardless of the accuracy of the classifiers, when working in parallel, one may detect anomalies while the other considers the network data normal, and vice versa. We propose an XOR-based detector of conflicting decisions designed to compare the outcomes of two binary classifiers. The basic idea of this detector is to apply an XOR bitwise operation to the classification results. Figure 2 shows the conceptual design of a detector that makes a decision based on the outputs of the FNN (eager learner) and wk-NN (lazy learner) [15].

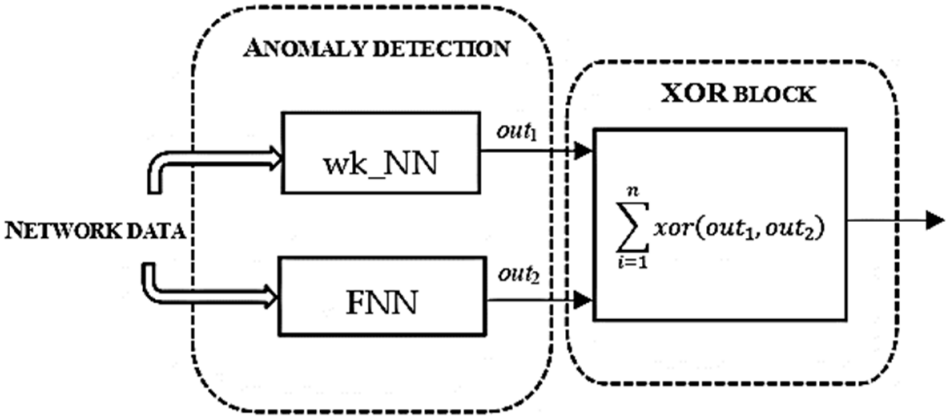

Figure 2. Detector of the conflicting decisions.

In conflicting decision detection, we consider it insignificant whether the outcomes of the classifiers are both true or false. What is considered necessary is that the results differ. If the classifiers make different decisions, the XOR logical operation performed on their outputs results in a total of one; otherwise, the result is zero. The number of different decisions can then be determined, as in Equation (8):

$$sum_{xor} = \sum_{i=1}^{n} xor(out_{1i}, out_{2i}), \tag{8}$$

where sum_{xor} represents the cumulative sum of n decisions, out_{1i} and out_{2i} are outcomes of the classifiers for $i = 1, \ldots, n$, and $xor(out_{1i}, out_{2i})$ is logically true (1) if the decisions differ, otherwise the result is false (0). If highly sensitive data must be protected, detection of opposing decisions can help raise additional alarms, related not only to the anomalies in the computer network but also to the potentially harmful non-standard challenges in 5G networks concerning privacy centered around location tracking, semantic information attacks, leakage from access points, etc. In [64,65], the authors discuss security in terms of alert range and accuracy of criteria selection.

4. Results and Discussion

The MATLAB classification learner is used to compare the effects of the feature selection on binary classification. Initially, the features are free of Not-a-Number (NaN) values that MATLAB does not recognize. In the first part of the experiments, accuracy and processing time for 17 and nine features were compared. Table 5 displays ACC and t_p for four daily records, with different numbers of instances in the data sets.

For all other models, the processing time was significantly shorter when nine features were used for model evaluation, compared to the time when 17 features were used. As expected, the processing time is as long as the number of instances increases. At the same time, the accuracy of the SVM model decreased to ~0.2%, followed by the k-NN and wk-NN models at ~0.8%, and the DT model at ~2%. For these reasons, nine features are assumed to be sufficient for use in experiments on the effects of feature scaling on accuracy and processing time and on XOR-based opposing decision detection. In this part of the experiments, the feedforward neural network was not tested because it does not deal with non-numerical features.

In the second part of the experiment, the effect scaling feature is presented. A daily record of the Kyoto 2006+ dataset of approximately 60,000 instances was used as a benchmark. The experiments were performed as follows. First, the features are freed of NaN values and a subset of 57,300 instances is used as a benchmark for the experiments. Second, all irrelevant features are removed. Feature *Label* is left for the decision about anomalies; when *Label* = 1, the network traffic is considered normal; otherwise, anomalies are detected (*Label* = -1). Third, four feature scaling methods are used: TH, Min-Max in the range

[0,1] and [−1,1] and Z-score standardization. The Z-score standardization was first used for training.

Table 5. ACC and t_p for four daily records.

Number of Instances	Model	ACC (9 Features) [%]	t_p (9 Features) [s]	ACC (17 Features) [%]	t_p (17 Features) [s]
158,570	k-NN	98.3	275.72	99.0	1000.8
	wk-NN	98.4	277.32	99.1	1019.15
	DT	97.2	3.8452	98.4	14.241
	SVM	98.1	449.35	98.4	467.7
127,740	k-NN	98.2	193.82	98.6	682.07
	wk-NN	98.1	194.81	98.8	690.58
	DT	97.2	3.3033	99.8	9.5367
	SVM	97.8	280.82	97.9	379.61
80,807	k-NN	98.8	91.25	99.4	285.77
	wk-NN	98.8	91.267	99.5	285.25
	DT	98.9	2..2615	99.4	6.2339
	SVM	97.9	227.28	98.1	125.25
57,280	k-NN	99.4	43.734	99.6	129.99
	wk-NN	99.5	43.272	99.6	130.88
	DT	99.4	1.7489	99.7	4.4535
	SVM	99.2	30.239	99.3	37.894

The features are not scaled to the same fixed range, and the model is trained using normally distributed instances. The results show that the decisions for all models are very accurate except for the DT classifier. In addition, the processing time of the two nearest neighbor models is significantly longer than that of all other models. Min-Max scaling in the range $[0, 1]$ is then used to solve the problem of different scales. Compared to the previous results, it can be seen that feature scaling has no positive effect on any classifier.

Furthermore, the Min-Max scaling in the range $[-1, 1]$ is used to avoid problems caused by very long or very small derivatives, but does not affect the classification results. Although the results showed very accurate models, there is still a problem with long processing times. Finally, the TH scaling is used. All features are all scaled into the same symmetrical range of ± 0.7616. Compared to other scaling methods, the results show a significant reduction in processing time, which is more than half that of the nearest neighbor models. The results are presented in terms of Accuracy, processing time, FP, TPR and F1-score. The results support the assumption that using the TH scaling to accelerate training is reasonable. All models except SVM have high accuracy and F1-scores. The results are given in Table 6.

Table 6. ACC and t_p, F1-score, FP and TPR.

	TH					Min-Max[0,1]					Min-Max[−1,1]					Z-Score				
	ACC [%]	tp [s]	F1 [%]	FP	TPR	ACC [%]	tp [s]	F1 [%]	FP	TPR	ACC [%]	tp [s]	F1 [%]	FP	TPR	ACC [%]	tp [s]	F1 [%]	FP	TPR
FNN	99.36	5	98.89	49	0.989	99.53	11	99.33	47	0.989	99.31	12	99.04	49	0.983	99.43	12	99.17	37	0.986
DT	99.40	2.5	99.18	36	0.992	99.47	2.8	99.12	38	0.984	99.41	6.3	99.14	41	0.985	98.89	2.2	99.15	41	0.985
SVM	99.10	26.9	98.69	64	0.980	99.15	36.9	98.93	63	0.981	99.17	43.4	98.89	65	0.989	99.22	36.3	98.88	62	0.981
k-NN	99.30	56.1	99.01	54	0.984	99.45	107.4	99.21	53	0.992	99.41	103.2	99.20	53	0.986	99.43	102.4	99.11	58	0.985
wk-NN	99.40	56.3	99.16	60	0.989	99.48	102.7	99.26	56	0.993	99.58	105.3	99.29	47	0.989	99.48	103.2	99.26	58	0.986

Overall, the results show that the TH scaling has a significant positive effect on processing time at the expense of a slight decrease in accuracy and F1-score. Scaling the

features within ± 0.7616 ensures that each feature is equally important in the decision and does not influence the others. The results also show that the damping strategy speeds up model training. The results show that feature scaling does not affect the number of false positive results. The percentage of FPs compared to the total number of instances varies from 0.065%, when the Z-score standardization is used for scaling and FNN is the classifier, to 0.11% when Min-Max normalization in the range $[-1,1]$ is used for scaling and SVM is the classifier. Low false positive values and high TPR for all the models indicate the applicability of the proposed feature selection method. The wk-NN model showed the highest classification accuracy in all cases except when the FNN is trained with the data scaled in the range $[-1,1]$. In this case, there is a 0.05% accuracy difference between the wk-NN model and feedforward neural network. In addition, the wk-NN model has the best F1-score other than FNN trained on data scaled in the range [0,1].

To demonstrate the functionality of the XOR-based model, we ran experiments on 3-day records, containing 57,270, 57,280, and 58,300 instances. First, we divided each daily record so that two-thirds of the instances are used to evaluate the models, and one-third is used to calculate the sum of the detected decisions. According to the findings, it was expected that classifiers detected network behavior equally. To investigate this expectation, the fastest FNN and the most accurate wk-NN models' decisions are compared (See Table 7).

Table 7. Detection of different decisions.

Instances	Different Decisions	Different Decisions [%]
57,270	1160	6.1
57,280	460	2.4
58,300	100	0.5

The results show that the number of conflicting decisions between the weighted k-NN model and the feedforward neural network are independent of the number of instances. Uncertainty in the results can be caused by data errors, residual errors in the model, unidentified malicious attacks, etc. It should be noted that the XOR-based detector of conflicting decisions cannot predict the specific conflict in the decision. It only provides additional warnings to the authorities in such cases. The decision criteria can be chosen in different ways depending on the sensitivity of the data, the technology used, legal practice, etc. [66].

5. Conclusions

As technology advances, the number of cyber-attacks has increased exponentially. As a result, detecting and predicting cyber-attacks is essential for any system that processes sensitive data. Detecting network behavior anomalies is a relatively simple process of determining what is "normal" and what is an "anomaly". With the rapid growth of computer networks and increasingly faster data processing, the classifiers need to improve the predictability. The studies presented in this paper can serve as a reference for researchers who want to use new methods for feature selection and scaling, or to choose the appropriate algorithms based on their application scenarios and available resources.

The authors often use datasets that are simulations of the network traffic. In such cases, the impact of duplicate and redundant records on model estimation can lead to low processing power and reduce the model's overall accuracy. The Kyoto 2006+ dataset is a publicly available 10-year data set of real network traffic designed for anomaly detection. The issue of the data set size is solved by feature selection and scaling. The nine numerical features are scaled using TH, Min-Max[0,1] and Min-Max$[-1,1]$ normalization and Z-score standardization. Five ML-based binary classifiers, namely: FNN, k-NN, wk-NN, DT, and SVM, are used to determine whether the network is working properly. The classifiers' performance are compared using accuracy, processing time, number of false alarms, TPR, and F1-score.

This paper proposes an XOR-based model to detect conflicting decisions in abnormal computer network behavior. The outputs of the fastest FNN and the most accurate wk-NN are compared. It has been demonstrated that their decisions sometimes differ. The sum of the non-zero bits determines the number of opposite conclusions after the classifiers' results are XORred. The results show that dataset size, model accuracy, and processing time do not affect the number of decisions.

Author Contributions: Conceptualization, D.P. and M.S.; methodology, D.P.; software, D.P.; validation, D.P., L.G., M.S. and M.A.R.; formal analysis, D.P.; investigation, D.P. and M.S.; resources, D.P. and M.S.; data curation, D.P.; writing—original draft preparation, D.P. and L.G.; writing—review and editing, L.G. and M.S.; visualization, D.P.; supervision, L.G., M.S. and M.A.R.; project administration, D.P., L.G., M.S. and M.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: http://www.takakura.com/Kyoto_data/ (accessed on 11 May 2020).

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

References

1. Fang, Y.; Shan, Z.; Wang, W. Modeling and key technologies of a data driven smart cities. *IEEE Access* **2021**, *9*, 91244–91258. [CrossRef]
2. Rahman, A.; Al-Saggaf, Y.; Zia, T. A data mining framework to predict cyber attack for cyber security. In Proceedings of the 15th IEEE Conference on Industrial Electronic and Applications, Kristiansand, Norway, 9–13 November 2020; pp. 207–212.
3. Ramakrishnan, R.; Gaur, L. *Internet of Things: Approach and Applicability in Manufacturing*; Chapman and Hall: London, UK; CRC: Boca Raton, FL, USA, 2019. [CrossRef]
4. Kaularachchi, Y. Implementing data driven smart city applications for future cities. *Smart Cities* **2022**, *5*, 455–474. [CrossRef]
5. Mohamed, N.; Al-Jaroodi, J.; Jawhar, I. Opportunities and challenges of data-driven cybersecurity for smart cities. In Proceedings of the 2020 IEEE Systems Security Symposium, Crystal City, VA, USA, 1 July–1 August 2020; pp. 1–7. [CrossRef]
6. Aliyu, F.; Sheltami, T.; Deriche, M.; Nasser, N. Human immune-based intrusion detection and prevention system for fog computing. *J. Netw. Syst. Manag.* **2020**, *30*, 11. [CrossRef]
7. Sen, J.; Methab, S. Machine Learning Applications in Misuse and Anomaly Detection. 2009. Available online: https://arxiv.org/ftp/arxiv/papers/2009/2009.06709.pdf (accessed on 18 July 2022).
8. Bialas, A.; Michalak, M.; Flisiuk, B. Anomaly detection in network traffic security assurance. In *Engineering in Dependability of Computer Systems and Networks*; Zamojski, W., Mayurkiewicy, J., Sugier, J., Walkowiak, T., Kacprzyk, J., Eds.; Springer: Cham, Switzerland, 2020; p. 987. [CrossRef]
9. Almomani, O. A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms. *Symmetry* **2020**, *12*, 1046. [CrossRef]
10. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network anomaly detection: Methods systems and tools. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 303–336. [CrossRef]
11. Kumar, S.; Gupta, S.; Arora, S. Research trends in network-based intrusion detection systems: A review. *IEEE Access* **2021**, *9*, 157761–157779. [CrossRef]
12. Bohara, B.; Bhuyan, J.; Wu, F.; Ding, J. A survey on the use of data clustering for intrusion detection system in cybersecurity. *Int. J. Netw. Secur. Its Appl.* **2020**, *12*, 1. [CrossRef]
13. Lin, I.-C.; Chang, C.-C.; Peng, C.-H. An anomaly-based IDS framework using centroid-based classification. *Symmetry* **2022**, *14*, 105. [CrossRef]
14. Protic, D.; Stankovic, M.; Antic, V. WK-FNN design for detection of anomalies in the computer network traffic. *Facta Univ. Ser. Electron. Energetics* **2022**, *35*, 269–282. [CrossRef]
15. Protic, D.; Stankovic, M. A hybrid model for anomaly-based intrusion detection in complex computer networks. In Proceedings of the 21st International Arab Conference on Information Technology (ACIT), Giza, Egypt, 28–30 November 2020; pp. 1–8. [CrossRef]
16. Protic, D.; Stankovic, M. Detection of anomalies in the computer network behaviour. *Eur. J. Eng. Form. Sci.* **2021**, *4*, 10–17. [CrossRef]
17. Ahmed, I.; Shin, H.; Hong, M. Fast content-based file type identification. In *Advances in Digital Forensics VII*; Springer: Berlin/Heidelberg, Germany, 2011. [CrossRef]
18. Ruggieri, S. Complete search for feature selection decision trees. *J. Mach. Learn. Res.* **2019**, *20*, 1–34.
19. Pham, B.T.; Jaafari, A.; Avand, M.; Al-Ansari, N.; Du, T.D.; Yen, H.P.H.; Phong, T.V.; Nguyen, D.H.; Le, H.V.; Mafi-Gholami, D.; et al. Performance evaluation of machine learning methods for forest fire modeling and prediction. *Symmetry* **2020**, *12*, 1022. [CrossRef]

20. Hardesty, L. Explained: Neural networks. *MIT News*, 14 April 2017. Available online: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (accessed on 11 July 2021).
21. Yusof, N.N.M.; Sulaiman, N.S. Cyber attack detection dataset: A review. *J. Phys. Conf. Ser.* **2022**, *2319*, 1–6. [CrossRef]
22. Song, J.; Takakura, H.; Okabe, Y.; Eto, M.; Inoue, D.; Nakao, K. Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, Austria, 10–13 April 2011; pp. 29–36. [CrossRef]
23. Mills, R.; Marnerides, A.K.; Broadbent, M.; Race, N. Practical Intrusion Detection of Emerging Threat. Available online: https://eprints.lancs.ac.uk/id/eprint/156068/1/TNSM_Paper_Accepted_Version.pdf (accessed on 5 September 2022).
24. Levenberg, K. A method for the solution of certain problems in least squares. *Q. Appl. Math.* **1944**, *5*, 164–168. [CrossRef]
25. Marquardt, D. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.* **1963**, *11*, 431–441. [CrossRef]
26. Su, P.; Chen, Y.; Lu, M. Smart city information processing under internet of things and cloud computing. *J. Supercomput.* **2022**, *78*, 3676–3695. [CrossRef]
27. Raza, S.; Wallgren, L.; Voigt, T. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Netw.* **2013**, *11*, 2661–2674. [CrossRef]
28. Shrestha, R.; Omidkar, A.; Ahmadi Roudi, S.; Abbas, R.; Kim, S. Machine-learning enabled intrusion detection system for cellular connected UAV Networks. *Electronics* **2021**, *10*, 1549. [CrossRef]
29. Alsheikh, M.A.; Lin, S.; Niyato, D.; Tan, H.P. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1996–2018. [CrossRef]
30. Kumar, Y.V.; Kamatchi, K. Anomaly based network intrusion detection using ensemble machine learning technique. *Int. J. Res. Eng. Sci. Manag.* **2020**, *3*, 290–297.
31. Pai, V.; Devidas, B.; Adesh, N.D. Comparative analysis of machine learning algorithms for intrusion detection. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1013*, 1–7. [CrossRef]
32. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A.A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167. [CrossRef]
33. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
34. Band, S.S.; Ardabili, S.; Sookhak, M.; Chronopoulos, A.T.; Elnaffar, S.; Moslehpour, M.; Csaba, M.; Torok, B.; Pai, H.T.; Mosavi, A. When smart cities get smarter via machine learning: An in-depth literature review. *IEEE Access* **2022**, *10*, 60985–61015. [CrossRef]
35. SIGKDD-KDD Cup. KDD Cup 1999: Computer Network Intrusion Detection. 2018. Available online: www.kdd.org (accessed on 21 September 2022).
36. McCarthy, R. Network Analysis with the Bro Security Monitor. 2014. Available online: https://www.admin-magazine.com/Archive/2014/24/Network-analysis-with-the-Bro-Network-Security-Monitor (accessed on 21 September 2022).
37. Ambusaidi, M.A.; He, X.; Nanda, P.; Tan, Z. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans. Comput.* **2016**, *65*, 2986–2998. [CrossRef]
38. Bistron, M.; Piotrowsk, Z. Artificial intelligence applications in military systems and their influence on sense of security of citizens. *Electronics* **2021**, *10*, 871. [CrossRef]
39. Maza, S.; Touahria, M. Feature selection algorithms in intrusion detection system: A survey. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 5079–5099. [CrossRef]
40. Kousis, A.; Tjortjis, C. Data mining algorithms for smart cities: A bibliometric analysis. *Algorithms* **2021**, *14*, 242. [CrossRef]
41. Cheong, Y.G.; Park, K.; Kim, H.; Kim, J.; Hyun, S. Machine learning based intrusion detection systems for class imbalanced datasets. *J. Korea Inst. Inf. Secur. Cryptol.* **2017**, *27*, 1385–1395. [CrossRef]
42. Nawi, N.M.; Atomi, W.H.; Rehman, M.Z. The effect of data preprocessing on optimizing training on artificial neural network. *Procedia Technol.* **2013**, *11*, 23–39. [CrossRef]
43. Weston, J.; Elisseff, A.; Schoelkopf, B.; Tipping, M. Use of the zero norm with linear models and kernel methods. *J. Mach. Learn. Res.* **2003**, *3*, 1439–1461.
44. Song, L.; Smola, A.; Gretton, A.; Borgwardt, K.; Bedo, J. Supervised feature selection via dependence estimation. In Proceedings of the International Conference on Machine Learning, 2007, Corvallis, OR, USA, 20–24 June 2007; Available online: http://www.gatsby.ucl.ac.uk/~{}gretton/papers/SonSmoGreetal07.pdf (accessed on 7 August 2022).
45. Dy, J.G.; Brodley, C.E. Feature selection for unsupervised learning. *J. Mach. Learn. Res.* **2005**, *5*, 845–889.
46. Mitra, P.; Murthy, C.A.; Pal, S. Unsupervised feature selection using feature similarity. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 301–312. [CrossRef]
47. Zhao, Z.; Liu, H. Semi-supervised feature selection via spectral analysis. In Proceedings of the SIAM International Conference on Data Mining, Minneapolis, MN, USA, 26–28 April 2007; pp. 641–646. [CrossRef]
48. Xu, Z.; Jin, R.; Ye, J.; Lyu, M.; King, I. Discriminative semi-supervised feature selection via manifold regularization. *IEEE Trans. Neural Netw.* **2010**, *21*, 1033–1047. [PubMed]
49. Swathi, K.; Rao, B.B. Impact of PDS based kNN classifiers on Kyoto dataset. *Int. J. Rough Sets Data Anal.* **2019**, *6*, 61–72. [CrossRef]
50. Uhm, Y.; Pak, W. Service-aware two-level partitioning for machine learning-based network intrusion detection with high performance and high scalability. *IEEE Access* **2020**, *9*, 6608–6622. [CrossRef]

51. Singh, A.P.; Kaur, A. Flower pollination algorithm for feature analysis of Kyoto 2006+ dataset. *J. Inf. Optim. Sci.* **2019**, *40*, 467–478.
52. Garcia, S.; Luengo, J.; Herera, F. Data preparation basic models. In *Data Preprocessing in Data Mining*; Intelligent System Reference Library; Springer: Berlin/Heidelberg, Germany, 2015; Volume 72, pp. 39–57. [CrossRef]
53. Al-Imran, M.; Ripon, S.H. Network Intrusion Detection: An analytical assessment using deep learning and state-of-the-art machine learning models. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 1–20. [CrossRef]
54. Obaid, H.S.; Dheyab, S.A.; Sabry, S.S. The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. In Proceedings of the 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON), Jaipur, India, 13–15 March 2019; pp. 279–283. [CrossRef]
55. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1–22. [CrossRef]
56. Ferryian, A.; Thamrin, A.H.; Takeda, K.; Murai, J. Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Appl. Sci.* **2021**, *11*, 7868. [CrossRef]
57. Soltani, M.; Siavoshani, M.J.; Jahangir, A.H. A content-based deep intrusion detection system. *Int. J. Inf. Secur.* **2022**, *21*, 547–562. [CrossRef]
58. Tsai, C.-F.; Hsu, Y.-F.; Lin, C.-Y.; Lin, W.-Y. Intrusion detection by machine learning. *Expert Syst. Appl.* **2009**, *36*, 11994–12000. [CrossRef]
59. Serkani, E.; Gharaee, H.; Mohammadzadeh, N. Anomaly detection using SVM as classifier and DT for optimizing feature vectors. *ISeCure* **2019**, *11*, 159–171.
60. Rahman, A.; Islam, Z. AWST: A novel attribute weight selection technique for data clustering. In Proceedings of the 13th Australasian Data Mining Conference (AusDM 2015), Sydney, Australia, 8–9 August 2015; pp. 51–58.
61. Rahman, M.A.; Islam, M.Z. CRUDAW: A novel fuzzy technique for clustering records following user defined attribute weights. In Proceedings of the Tenth Australasian Data Mining Conference (AusDM 2012), Sydney, Australia, 5–17 December 2012; Volume 134, pp. 27–42.
62. Lampton, M. Damping-undamping strategies for Levenberg-Marquardt least-squares method. *Comput. Phys.* **2019**, *11*, 110–115. [CrossRef]
63. Dinov, I.D. *Data Science and Predictive Analytics*; Springer: Ann Arbor, MI, USA, 2018.
64. Allier, S.; Anquetil, N.; Hora, A.; Ducasse, S. A framework to compare alert ranking algorithms. In Proceedings of the 19th Working Conference on Reverse Engineering, 2012, Kingston, ON, Canada, 15–18 October 2012.
65. Zhao, N.; Jin, P.; Wang, L.; Yang, X.; Liu, R.; Zhang, W.; Sui, K.; Pei, D. Automatically and adaptively identifying severe alerts for online service systems. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020.
66. Gaur, L.; Solanki, A.; Jain, V.; Khazanchi, D. *Handbook of Research on Engineering Innovations and Technology Management in Organizations*; ICI Global: Brussels, Belgium, 2020.