applied sciences

MDPI

Article

Clustering-Assisted Classification: A DBSCAN-FNN Model for Anomaly Detection

Danijela Protic [1], Jordan Atanasijevic [1] and Nenad Stojanovic [1,2,*]

[1] Center for Applied Mathematics and Electronics, 11000 Belgrade, Serbia; danijelaprotic318@gmail.com (D.P.); atanasijevicjordan@gmail.com (J.A.)
[2] Military Academy, University of Defence, 11000 Belgrade, Serbia
* Correspondence: nenad.m.stojanovic@vs.rs

Abstract

This study investigates the integration of a density-based spatial clustering of application with noise (DBSCAN) and a feedforward neural network (FNN) for anomaly detection in computer network traffic. The Kyoto 2006+ dataset, containing real network traffic records, was used for model development and evaluation. Feature selection and scaling methods were applied to reduce redundancy, mitigate multicollinearity, and improve the stability of the models. DBSCAN was employed to identify clusters and noise within the traffic data, and the resulting cluster-based information was incorporated as an additional input feature for FNN classification. To evaluate the individual and combined effects of the DBSCAN-generated feature and network architecture, four configurations were investigated: FNN (8-8-1), FNN (8-9-1), DBSCAN-FNN (9-8-1), and DBSCAN-FNN (9-9-1). Experimental results showed that increasing network capacity alone produced only marginal changes in classification performance, while DBSCAN-generated feature improved recall and reduced the number of false negatives. DBSCAN-FNN (9-9-1) architecture achieved the highest accuracy, recall and an F_1. The results indicate that observed performance improvements cannot be attributed exclusively to DBSCAN-generated feature or the network architecture. Instead, the findings suggest that clustering-derived information can complement conventional input features and, when combined with an appropriate network configuration, enhance FNN-based anomaly detection.

Keywords: clustering-assisted classification; computer networks; Artificial Intelligence; machine learning; anomaly detection; DBSCAN algorithm; feedforward neural network; Kyoto 2006+ dataset

Check for updates

Academic Editor: Andrea Prati

Received: 7 October 2026
Revised: 30 October 2026
Accepted: 30 October 2026
Published: 5 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Artificial Intelligence (AI) is the technology with arguably the greatest transformative potential of the 21st century. It is an interdisciplinary field of research encompassing a range of subdisciplines—including machine learning (ML), deep learning (DL), natural language processing (NLP), computer vision (CV), and robotics—each contributing to the development of intelligent systems capable of learning, reasoning, and making decisions [1]. Recent rapid advancements in AI have also impacted data protection by introducing advanced mechanisms that go beyond traditional cybersecurity practices [2]. To enhance layered defense, Security Information and Event Management (SIEM) systems integrate ML to automatically detect anomalies, identify zero-day vulnerabilities, and continuously analyze potential risks [3]. ML models are widely used to learn from new data, thereby enabling

security systems to evolve alongside emerging threats. The authors of [2,4] classify ML into four categories: (1) supervised learning, (2) unsupervised learning, (3) semi-supervised learning and (4) reinforcement learning. Supervised ML is based on labeled datasets, where each input is paired with a known output [2]. Unsupervised ML, in contrast, uses unlabeled data to discover hidden structures or groupings [4]. Semi-supervised ML utilizes a small amount of labeled data alongside a larger amount of unlabeled data, thereby reducing the need for vast amounts of data [1]. Reinforcement learning focuses on sequential decision-making, where an agent interacts with an environment and learns optimal strategies through a system of rewards and penalties [2]. In recent years, ML-based intrusion detection has become a widely adopted security practice for blocking attacks and mitigating information security risks. This reactive measure identifies and mitigates risks in real time using Intrusion Detection Systems (IDSs); these systems are classified by operational characteristics such as host-based [5], network-based [6], application-based [7], or sensor-based [7] and employ distinct detection approach as signature -based [8], anomaly-based [9], or hybrid [7].

In this paper, we present a clustering-assisted classification (CAC) approach that combines the density-based spatial clustering of applications with noise (DBSCAN) algorithm and a single-hidden-layer feedforward neural network (FNN) for anomaly detection in computer network traffic. DBSCAN is utilized to group data points into clusters of varying shapes and sizes according to local density while identifying points that do not belong to any cluster as spatial noise [10]. The FNN performs the binary classification of network traffic into normal (Class 1) and anomaly (Class −1) categories based on prediction probability [7]. The proposed approach was evaluated using the eight selected features from the real Kyoto 2006+ dataset, with experiments conducted on the daily subset 20080907.txt (7 September 2008), containing 49,790 instances. Prior to model development, feature selection, dimensionality reduction, and data scaling were performed to reduce redundant information, mitigate multicollinearity, and improve model stability. These preprocessing steps support robust model training and reduce the risk of overfitting. In the proposed work, DBSCAN was applied to generate an additional feature representing clustering information, which was then appended to the selected features as an additional input to the FNN. To evaluate the individual and combined effects of the DBSCAN-generated feature and network architecture, four configurations were investigated: FNN (8-8-1), FNN (8-9-1), DBSCAN-FNN (9-8-1), and DBSCAN-FNN (9-9-1). The results showed that neither increasing network capacity nor incorporating the DBSCAN-generated feature alone consistently improved all evaluation metrics. The highest accuracy, recall, and F_1 were achieved by the DBSCAN-FNN (9-9-1) architecture, indicating that both the clustering-derived information and network configuration influence anomaly detection performance. The main contributions of this work are threefold: (1) the development of a clustering-assisted feature engineering strategy in which DBSCAN-derived information is incorporated as an additional input feature for FNN-based anomaly detection; (2) a systematic experimental evaluation that separately examines the effects of the DBSCAN-generated feature and hidden-layer size through four network configurations; and (3) an empirical analysis demonstrating that classification performance is influenced by both feature augmentation and network architecture rather than by either factor alone.

2. Related Work

Research into anomaly detection systems encompasses various approaches, including data collection practices, feature definition, data preprocessing, and hybrid models. Recent studies highlight the importance of reliable data, adequate feature selection methods to avoid redundancy, robust scaling methods ensuring stable model performance, and clustering-

assisted classification methods combining the advantages of unsupervised and supervised learning. These studies provide a solid foundation for developing ML-based IDSs designed for complex network topologies, offering higher precision and greater applicability across diverse scenarios.

2.1. Data Collection

Previous research on IDS—utilizing approaches based on both anomaly and signature detection, as well as various datasets for model evaluation—faced challenges due to the large volume of data required for model training. The datasets were largely simulated (generated)—meaning they were not collected from actual network traffic—or contained diverse types of network traffic. Furthermore, the number of features varied significantly across datasets; some had a high number of dimensions, while others did not. The timeframes for data generation or collection were are also inconsistent, and the types varied considerably. For instances, the CAIDA and Kyoto 2006+ datasets contain data collected from actual network traffic [11,12]; ISDX 2012, CIC-IDS-2017, and CSE-CIC-2018 contain data from five- to ten-day simulations of small-network data that includes signature-based attacks [13,14]; DARPA98, KDD Cup '99, and NSL-KDD contain approximately 40 features simulating network traffic in a packet-based format [12,15]; AWID contains 156 features extracted from each packet of a simulated small-scale WLAN [16]; ADFA was created during the evaluation of a host-based IDS relying on system calls [17]; CICIDS2017 and UNSW-NB15 cover a wide range of attacks [18]; and Bot-IoT and CTU-13 [19,20] are primarily used in research related to the Internet of Things (IoT) and botnet environments, among others. The primary objective of our research was to evaluate machine learning models that classify computer network traffic into the two classes: normal traffic and anomaly. The core idea was to demonstrate that DBSCAN clustering improves classification based on an FNN. We utilized the publicly available Kyoto 2006+ dataset, created by recording daily instances of real network traffic over a ten-year period (2006–2015) [21]. The full Kyoto 2006+ dataset incorporates 14 statistical features inherited from the KDD Cup 99 dataset and extends them with 10 additional network traffic attributes. The features are grouped into basic, traffic statistical, and security-related features (Table 1).

Table 1. Feature groups within the Kyoto 2006+ dataset.

Group	Description	Attribute
Basic	Connection duration [s], service type, and connection state at termination.	Duration, Service, Flag
Security-related	Indicators of intrusion detection alerts, malware detection, and traffic label (normal or attack).	IDS_detection, Malware_detection, Label
Traffic statistical	Quantitative measures of traffic volume, errors, and connection distributions.	Source_bytes, Destination_bytes, Count, Same_srv_rate, Serror_rate, Srv_serror_rate, Dst_host_count, Dst_host_srv_count, Dst_host_same_src_port_rate Dst_host_serror_rate, Dst_host_srv_serror_rate

Based on the design of the dataset collection process, certain features reflect different aspects of network traffic behavior [11]. This overlap illustrates multicollinearity—a situation where multiple features describe the same event from different perspectives [22,23]. In the context of ML modeling, such overlaps are treated as sources of distortion or data redundancy. Redundant features can disrupt distance calculations, assigning disproportionate weight to specific behavior patterns [24]. To avoid issues such as overlap, redundancy, and multicollinearity (and to mitigate overfitting while enhancing robustness to noise, predictive power, and model generalization capability), various data preprocessing techniques can be applied, including feature selection, dimensionality reduction, regularization, and feature scaling.

2.2. Feature Selection, Dimensionality Reduction, and Regularization

Feature selection, dimensionality reduction, and regularization are data preprocessing methods that reduce the size of a dataset. Although each of these methods serves a different purpose, together, they contribute to improved results regarding accuracy, efficiency, and generalization. Feature selection offers a range of advantages during the data processing stage [25]. This process can reduce the dimensionality of a dataset containing a large number of (mostly redundant) features, which would otherwise hinder model training [21]. Feature selection can also improve classification model accuracy by removing noise and irrelevant features [24]. Furthermore, it helps avoid overfitting, as fewer unnecessary features lead to better generalization and more stable performance on test data [23]. Dimensionality reduction is typically performed after the removal of duplicates, missing values, and noise, but before model training. Common dimensionality reduction techniques include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and correlation analysis used to eliminate highly correlated features [26]. Regularization is frequently used in predictive modeling to penalize superfluous variables, thereby preventing the model from "learning noise" and losing its generalization capability [27]. The balance between model accuracy and simplicity is controlled by the regularization hyperparameter, leading to improved generalization on test data. The most well-known regularization techniques are Lasso (L1) [28], Ridge (L2), ElasticNet (a combination of L1 and L2), and Dropout [29].

2.3. Feature Scaling

To prevent the dominance of attributes with large values over those with smaller ones and to ensure the stability of ML models, features can be normalized/scaled to carry equal weight during model evaluation. In [30–36], the authors discussed numerous feature-scaling techniques, including: min–max scaling (mapping data to a predefined range), Z-score standardization (for data following a normal distribution), L2 normalization (based on Euclidean or L2 norm; after normalization, the attribute vector is scaled to norm equals 1), decimal scaling (shifting the decimal point to scale data within a specific range), log transformation (for data spanning a wide range or following an exponential distribution), robust scaling (based on the median and interquartile range), Box–Cox (applied only to positive values to make the data approximate a normal distribution), Yeo-Johnson (a generalization of the Box–Cox that handles both positive and negative values and is used for highly asymmetric distributions), and others.

2.4. Clustering-Assisted Classification

Clustering-assisted classification is a multi-step hybrid approach in ML for anomaly detection in computer networks [37]. Network traffic instances are initially grouped into subsets of similar instances using clustering algorithms [38]. The clustering result is then used as an additional feature, enabling the classifier to recognize hidden structures within

the data. This approach offers several advantages: CAC significantly reduces data complexity by grouping data into multiple clusters; classifiers operate on homogeneous data subsets, resulting in higher model accuracy; the risk of overfitting is also reduced because model training focuses on relevant data groups; and processing smaller segments accelerates the training process [39]. CAC also contributes to the interpretability of results, as the discovered clusters reveal latent structures (e.g., patient types in medical analytics or customer segments in marketing). However, classification depends directly on clustering—poor clustering can degrade classifier performance. Combining clustering and classification can be computationally demanding, and the choice of clustering algorithm significantly impacts the final results [40]. An approach combining DBSCAN and FNN can be considered clustering-assisted classification. This approach combines not only two algorithms but also two fundamentally different philosophical approaches to solving the same data security problem. DBSCAN is an unsupervised ML algorithm that analyzes the density and structure of a dataset when attack instances are not known in advance, searching for noise outside the dense clusters of normal traffic. The algorithm operates without assumptions regarding the number of clusters and identifies points representing noise as anomalies [10,41]. The DBSCAN algorithm offers numerous advantages: robustness to noise, flexibility in identifying various cluster shapes, and the fact that it does not require a predefined number of clusters. It has proven effective on data with irregular cluster shapes and varying densities, and models trained on real high-dimensional and heterogeneous computer network data demonstrate high accuracy. On the other hand, FNN is a supervised ML model trained using labels; it learns precise geometric boundaries between computer network traffic instances [7,42]. Neural network models typically have good predictive accuracy and can be used for multiclass classification. FNN is highly accurate in detecting known attack types or anomalies. In this approach, a DBSCAN was employed as a clustering-based preprocessing stage to generate cluster labels, which were subsequently incorporated as an additional input feature for the FNN classification.

3. Proposed Work

The proposed work advances previous research in the field of intrusion detection by addressing three key issues: redundant feature, scaling variability, and the lack of robust hybrid models for intrusion detection in complex computer networks. Prior to conducting experiments, feature selection was performed to eliminate overlapping features, and min-max scaling was applied to prevent ML model instability and false detections; CAC was employed to enhance classification accuracy and the classifier's generalization capability. Our approach combines DBSCAN clustering with FNN classification to leverage the advantages of both unsupervised and supervised learning for more reliable anomaly detection. The DBSCAN algorithm is used to identify noise and refine class boundaries within the dataset, flagging suspicious samples in the process. The output of the DBSCAN algorithm serves as an additional feature for training the FNN. The FNN processes this enriched, consolidated dataset and learns non-linear relationships through two hidden layers, thereby classifying network traffic as either normal or anomalous.

3.1. The Kyoto 2006+ Dataset Feature Selection

The Kyoto 2006+ dataset contains a diverse set of traffic and statistical features intended to comprehensively characterize network behavior. The presence of correlated and redundant features has been widely recognized in the literature, motivating the selection of the most informative features for model training and evaluation [43,44]. Since the Kyoto 2006+ dataset is designed to present similar aspects of network traffic from different perspectives [11], several feature pairs such as SYN error statistics, destination–host activity,

service concentration, and connection frequency were found to describe similar network behaviors at different aggregation levels, indicating the presence of multicollinearity [22,23]. Serror_rate, Srv_serror_rate, Dst_host_serror_rate, and Dst_host_srv_serror_rate all characterize unsuccessful TCP connection attempts from different perspectives. Similarly, Count and Dst_host_count represent connection frequency over different observation scopes, while Same_srv_rate and Dst_host_diff_srv_rate provide complementary information regarding service utilization patterns. To minimize redundancy and reduce potential multicollinearity effects, only one representative feature from highly correlated groups was retained. In addition, the attributes IDS_detection and Malware_detection were excluded because they represent outputs of existing security mechanisms rather than intrinsic network traffic characteristics, potentially introducing information leakage into the classification process (Table 2).

Table 2. Feature redundancy map.

Feature Pair	Measurement Scope	Description
Serror_rate vs. Srv_serror_rate	SYN errors	Host-level vs. service-level SYN error rates. May be identical when the host runs only one service.
Count vs. Dst_host_count	Connection frequency	Short window (2 s) vs. long window. Both may increase during fast automated attacks.
Same srv rate vs. Dst_host_diff_srv_rate	Service distribution	Can display complementary behavior.
Srv_serror_rate vs. Dst_host_srv_serror_rate	Service-specific SYN errors	Both measure SYN errors per service. One local, one destination-wide.
Dst_host_count vs. Dst_host_srv_count	Destination host activity	One counts all connections to host, the other only per service; Can exhibit similar trends.
Same_srv_rate vs. Dst_host_same_src_port_rate	Service vs. port concentration	Both indicate traffic concentration and may attain high values simultaneously.
Serror_rate vs. Dst_host_serror_rate	SYN errors	Both describe related aspects of SYN error behavior.
Network statistics vs. IDS_detection	Raw anomalies vs. IDS alerts	Pronounced attacks may be associated with both statistical anomalies and IDS alerts.
IDS_detection vs. Malware_detection	Security system outputs	Both may indicate malicious activity and can overlap for known attacks.
Label vs. IDS_/Malware_detection	Ground truth vs. system detection	Labels may be consistent with IDS or Malware detection indicators for known attacks.

To avoid the negative impact of overlap, the final input set consisted of eight attributes: Duration, Source_bytes, Destination_bytes, Count, Same_srv_rate, Serror_rate, Dst_host_count, and Dst_host_same_src_port_rate. The selected attributes were chosen because they capture complementary aspects of network behavior while maintaining low redundancy. Specifically, Duration represents temporal characteristics of a connection, Source_bytes and Destination_bytes describe traffic volume, Count and Dst_host_count capture connection frequency and destination-oriented activity, Same_srv_rate characterizes service usage patterns, Serror_rate reflects TCP SYN anomalies frequently associated with scanning and denial-of-service activities, and Dst_host_same_src_port_rate provides information regarding source-port concentration. Together, these features provide a compact yet comprehensive representation of network traffic behavior, preserving the most relevant information for intrusion detection while reducing feature-space complexity (Table 3). Feature Label is used to train supervised models, representing the target variable. This feature is crucial in determining the values of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) [45,46]. The Label values are categorical and numerically encoded: 1 → normal traffic (legitimate user/system activity), −1 → known attack/anomaly (malicious activity, scanning, abuse), and −2 → unknown behavior. The original label values of the Kyoto 2006+ dataset were transformed into a binary classification scheme. Class 1 represents normal network traffic (legitimate user/system activity), whereas Class −1 represents abnormal traffic, including known attacks, anomalies (e.g., malicious activity, scanning, and abuse), and unknown behavior. This transformation enables the distinction between benign and potentially malicious network events.

Table 3. Reduced Kyoto 2006+ dataset.

Feature	Description
Duration	Tracks the length of network session. Normal traffic is a mix of session durations ranging from short to medium values. An attack generates extreme values, either zero in connection failures, or very high in malicious sessions.
Source_bytes	Measures the amount of data sent by the client. Reveals both normal user actions (uploads, emails) and attack fingerprints (scans, buffer overflows, brute force).
Destination_bytes	Measures total number of data bytes (excluding protocol headers) that the server sends back to the client during the entire connection, expressed as an integer ≥ 0.
Count	Counts fast connections to the same host within the last 2 s. Expressed as an integer, ranging from 1 to several hundreds.
Same_srv_rate	Percentage of those fast connections targeting the same service/port relative to the total Count, ranging from 0% to 100%. One of the most reliable indicators of attack concentration.
Serror_rate	Main indicator of SYN errors/DDoS attacks. Calculates the percentage of incomplete or broken TCP handshakes within the 2 s window defined by Count.

Table 3. *Cont.*

Feature	Description
Dst_host_count	Broader counter of total number of connections from the same source address to the same destination host over a long historical window than Count. Values are typically capped at 100 or 255.
Dst_host_same_src_port_rate	Detects whether attacks are automated, repeatedly coming from the same source port. Distinguishes real human traffic (randomized ports) from automated bots (fixed ports).

3.2. Min–Max Feature Scaling

To avoid ML model instability and false detections, the features should be scaled due to their variability and extreme ranges [47]. In this study, we applied min–max scaling to the numerical features as follows:

$$x_{scaled} = a + \frac{(x - x_{min})(b - a)}{x_{max} - x_{min}}, \quad (1)$$

where x, x_{scaled}, x_{max} and x_{min}, represent original, scaled, maximum, and minimum feature values, respectively, while $[a, b]$ defines the scaling interval [7,30,32,33,36].

For the purpose of this work, two independent scaling procedures were applied. First, the original eight input features were scaled to the [0, 1] interval and used as input to the DBSCAN algorithm. This scaling was adopted because DBSCAN relies on Euclidean distance, which is sensitive to differences in feature scales. In parallel, the same original features were scaled to the [–1, 1] interval for training the FNN models, as this range was more suitable for the hyperbolic tangent activation functions and contributed to faster convergence during training. The baseline FNN model containing eight inputs, eight neurons in the hidden layer, and one output neuron (8-8-1) was trained using the normalized input features. For the proposed DBSCAN-FNN model, the DBSCAN-generated feature was also scaled to the [–1, 1] interval and added to the original feature set.

3.3. DBSCAN-FNN Model

The DBSCAN-FNN model represents a CAC approach in which DBSCAN is used to generate an additional clustering-derived feature, while the final decision regarding normal or anomalous traffic is made by the FNN classifier. In this way, clustering information is incorporated into the classification process without using DBSCAN as a standalone detector.

DBSCAN groups data by density, identifies clusters of irregular shapes and sizes, and detects data points representing noise [48]. This method does not require the user to specify the number of clusters in advance, making it suitable for analyzing data where the structure is not predefined. Instead, it identifies clusters by grouping points within a specific radius, locates high-density regions, and detects outliers [10,41]. The algorithm relies on two parameters: the maximum distance between neighboring points (ε) and the minimum number of points required to form a dense region (*MinPts*) [49]. Based on these parameters, points can be classified as core points, border points, or outliers (noise) [50].

The FNN classifier produces results based on prediction probability and a classification threshold. The FNN comprises layers of interconnected neurons—inputs are passed to the neurons of the input layer, while all subsequent layers are connected to the preceding layer [7]. Since nonlinear optimization is used for training, stopping criteria (e.g., a selected number of iterations or a specified value for the relative change in the error function) must be determined in advance.

The DBSCAN-FNN model achieves a balance between discovering data structure and learning complex patterns, resulting in superior performance and a more stable model for network traffic anomaly detection. Another advantage is that the model's concept can be applied to other CAC combinations, utilizing datasets with varying attributes, feature selection methodologies, and instance scaling. The DBSCAN-FNN CAC is illustrated in Figure 1.

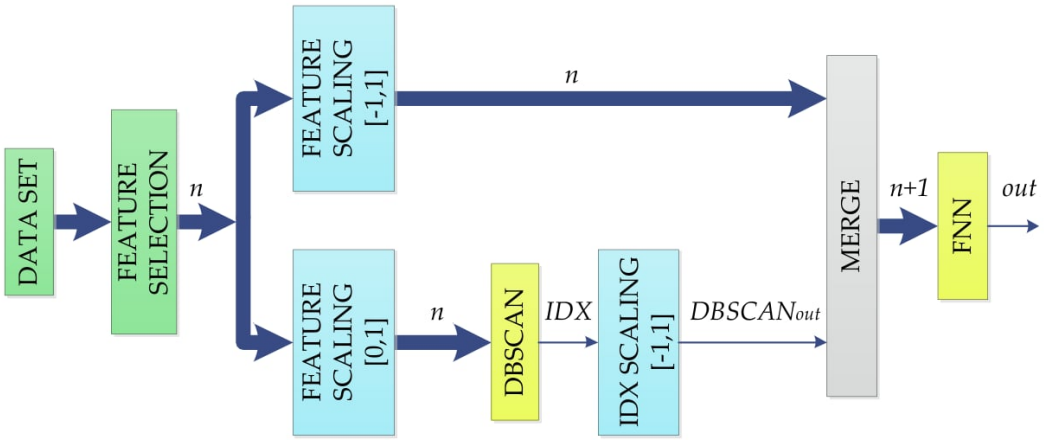


Figure 1. Density-based spatial clustering of application with noise − feedforward neural network (DBSCAN-FNN) clustering−assisted classification.

The CAC process comprises a preprocessing phase (feature selection and scaling), DBSCAN clustering, and FNN classification. The DBSCAN algorithm was employed solely to generate an additional feature representing cluster membership. This feature was appended to the selected input attributes and used as an additional for the FNN training. The final classification was performed exclusively by the neural network.

The classification process consists of the following steps: (1) n numerical features containing m instances are selected; (2) the features are scaled to the $[0, 1]$ range for DBSCAN clustering, and to the $[-1, 1]$ range for FNN classification; (3) DBSCAN generates the vector *IDX* containing cluster labels assigned to individual instances; (4) the cluster labels are transformed into a binary indicator where all clustered instances are assigned the value 1 and noise instances retain the value -1; (5) the resulting binary vector is used as an additional feature (*DBSCANout*) for training the FNN; (6) $n + 1$ features are used to train a fully connected single-hidden-layer FNN with a *tanh* activation function applied to all neurons; and (7) the output is classified as normal traffic if $out \geq 0$; otherwise, an anomaly is detected.

4. Experimental Results and Discussion

The experimental evaluation was conducted using one-day instances from 7 September 2008, comprising 49,790 instances in total. Prior to model training, a data cleaning procedure was performed to remove all instances containing Not-a-Number (NaN) values, resulting in a final dataset of 46,963 samples. MATLAB 2023b functions were used for feature scaling, nearest neighbor determination, plot k-distance graphs, model evaluation, generating confusion matrices, and plotting Receiver Operating Characteristic (ROC) curves [51]. The metrics used to compare the results are overall Accuracy (A_{cc}), False Positive Rate (FPR), False Negative Rate (FNR), True Negative Rate (TNR), Precision (PR)/Positive Predictive Value (PPV), Recall (RC)/True Positive Rate (TPR)/Sensitivity, Negative Predictive Value (NPV) and F1_score (F_1). In this study, normal traffic (Label = 1) was considered the

positive class, while attack traffic (Label = −1) was considered the negative class. Therefore, *TP* and *FN* refer to normal traffic samples, whereas *TN* and *FP* refer to attack traffic samples.

$$A_{cc} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2}$$

$$FPR = \frac{FP}{FP + TN}, \tag{3}$$

$$FNR = \frac{FN}{FN + TP}, \tag{4}$$

$$TNR = Specificity = \frac{TN}{TN + FP}, \tag{5}$$

$$PR = PPV = \frac{TP}{TP + FP}, \tag{6}$$

$$RC = TPR = Sensitivity = \frac{TP}{TP + FN}, \tag{7}$$

$$NPV = \frac{TN}{TN + FN}, \tag{8}$$

$$F_1 = \frac{2 \cdot RC \cdot PR}{RC + PR} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \tag{9}$$

A_{cc} represents the ratio of correctly classified instances to total number of instances and should be as high as possible; it refers to errors from both classes. Unlike conventional A_{cc}, a Balanced Accuracy (BA_{cc}) evaluates the classifier performance on both classes equally and is less sensitive to class imbalance.

$$BA_{cc} = \frac{RC + TNR}{2} = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2}. \tag{10}$$

In this study, BA_{cc} provides an additional measure of the classifier's ability to correctly recognize both normal and attack traffic. The interpretation of *FPR* depends on the selected positive class. Since the normal traffic was treated as the positive class, *FPR* represents the proportion of attack samples incorrectly classified as normal traffic. As normal traffic was treated as the positive class, *FNR* represents the proportion of normal traffic samples incorrectly classified as attacks. *NPR* measures the proportion of attack predictions that are correct. *PR* measures the proportion of correctly predicted positive instances among all instances predicted as positive. *RC* measures the proportion of positive instances that are correctly classified. Since the normal traffic was treated as the positive class, *RC* represents the proportion of normal network connections that are correctly identified as normal by the classifier. The *FNR* is complementary to *RC* and can be expressed as:

$$FNR = 1 - TPR, \tag{11}$$

A high *RC* value indicates a lower number of *FN* classifications. A low *PR* value indicates that a considerable proportion of instances predicted as positive are actually *FP* cases, whereas a low *RC* value combined with a high *PR* value indicates that a large number of positive instances were missed, although the instances classified as positive are predominantly correct. To balance these two metrics, the harmonic mean F_1 is introduced. It provides a balanced measure of classification performance by simultaneously considering both *PR* and *RC*. However, it does not distinguish whether performance limitations originate primarily from low *PR* or low *RC*. Therefore, the F_1 should be considered alongside other confusion matrix-derived measures to provide a more comprehensive assessment of classifier performance. As noted by authors of [52], classifier evaluation should rely on multiple complementary measures, since individual metrics may exhibit different sensitivities to class distributions and error types.

4.1. DBSCAN Clustering

The parameters ε and *MinPts* were determined using a *k*-distance graph before the DBSCAN model evaluation. The procedure was as follows: Euclidean distances [7] were calculated for the reduced and scaled dataset, and the *knnsearch* function was used to identify the *k*-nearest neighbors ($k = MinPts$); the *k*-distance values were plotted in ascending order (since the resulting matrix had more than one column, only the last column was displayed), and the "elbow" on the graph was identified to determine the value of ε. Given the large number of instances, the *MinPts* value was set to 28 to avoid creating an excessive number of clusters, and for ε, a value of 0.1 was selected (Figure 2).

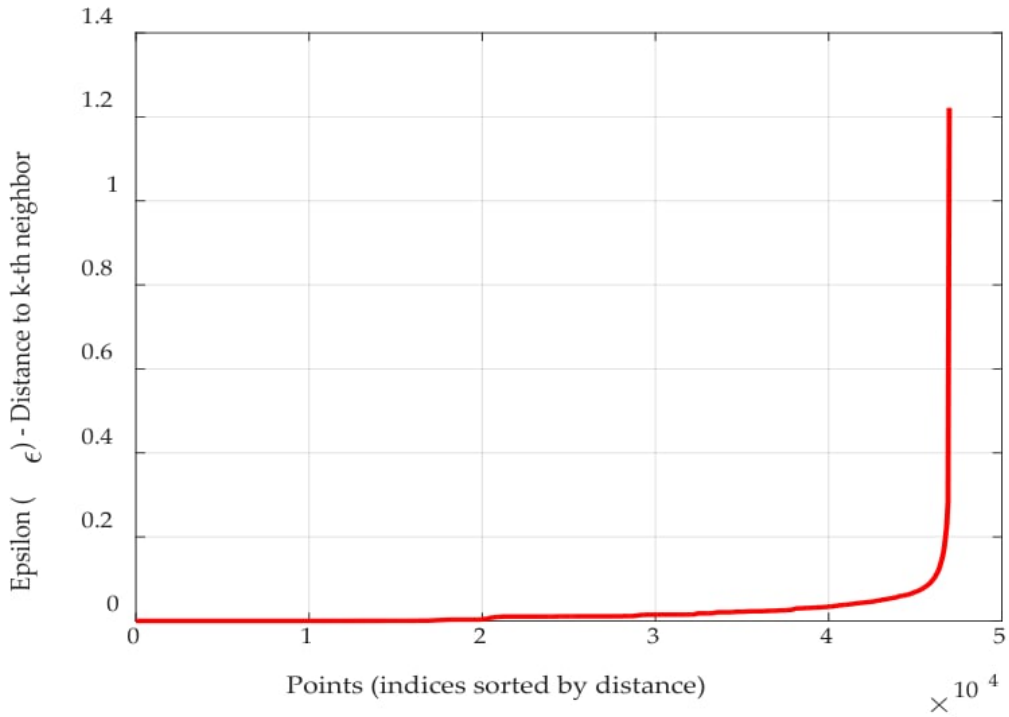


Figure 2. *k*-distance graph.

Since DBSCAN was employed solely to generate an additional feature for the subsequent neural network classifier, the parameter selection was aimed at obtaining stable and informative cluster assignments rather than optimizing clustering performance itself. The DBSCAN-derived feature was generated prior to cross-validation using the complete reduced and scaled dataset.

After selecting *MinPts* and ε, MATLAB's built in *dbscan* function is used to train the model as follows:

$$[IDX, CorePts] = dbscan(\mathbf{IN}_{DBSCAN}, \varepsilon, MinPts), \quad (12)$$

where $\mathbf{IN}_{DBSCAN}$ represents a matrix of features scaled to the $[0, 1]$ range, *CorePoints* is a vector indicating the instances identified by DBSCAN as core points, and *IDX* contains the cluster assignments produced by DBSCAN. In the experiments, DBSCAN identified nine clusters. However, the cluster labels themselves were not used as inputs to the FNN. Instead, *IDX* was transformed into a binary indicator *DBSCANout*, where all clustered instances were assigned the value 1, and noise instances retained the value −1. As DBSCAN was employed solely for feature generation, only the performance of the final neural network classifier is reported and analyzed in the following sections.

4.2. FNN Classification

The FNN transfer function is represented by the formula:

$$y_i(\mathbf{w}, \mathbf{W}) = F_i\left(\sum_{j=1}^{q} W_{ij} f_j\left(\sum_{l=1}^{m} w_{jl} z_l + w_{j0}\right) + W_{f0}\right), \quad (13)$$

where z_l, y_i, $\mathbf{w}$ and $\mathbf{W}$ represent inputs, outputs, and weight matrices, respectively; m determines the number of inputs; q represents the number of neurons in the hidden layer; f_j and F_i are activation functions of the hidden and output layer, respectively; and w_{j0} and W_{f0}, represent biases [7]. In our work, f_j and F_i are hyperbolic tangent ($tanh$) activation functions. The network weights were randomly initialized within the range $[-1, 1]$ using MATLAB's default initialization procedure, and no fixed random seed was specified during the experiments. The classifier was implemented in MATLAB Classification Learner using an FNN architecture. To investigate the influence of network capacity on classification performance, two baseline architectures were evaluated: an 8-8-1 network and an 8-9-1 network. Both architectures used the same eight selected input features, differing only in the number of neurons in the hidden layer. Network training was performed using the MATLAB *trainlm* function based on the Levenberg–Marquardt (LM) optimization algorithm, which combines the advantages of the Gradient Descent (GD) and Gauss–Newton (GN) methods for weight adjustment. A more detailed description of the LM algorithm can be found in [53–55]. Training was carried out with randomly selected weights (within the $[-1, 1]$ range), a maximum of 1000 iterations and a regularization parameter $\lambda = 0$, while all remaining stopping criteria were set to their default MATLAB values as specified in [56]. To ensure a robust and unbiased performance assessment, the neural network models were evaluated using 5-fold cross-validation within the MATLAB Classification Learner environment. The reported *TP*, *TN*, *FP*, and *FN* values correspond to the overall cross-validated confusion matrix generated by the software [57]. The results are presented in Table 4.

Table 4. Results of feedforward neural network (FNN) classification.

Metric	8-8-1	8-9-1
TP	45,013	45,002
TN	1544	1558
FP	157	143
FN	249	260
A_{cc}	99.14%	99.14%
FPR	9.23%	8.41%
FNR	0.55%	0.57%
TNR	90.77%	91.59%
PR	99.65%	99.68%
RC	99.45%	99.43%
NPV	86.11%	85.71%
F_1	99.55%	99.55%
BA_{cc}	95.11%	95.51%

As shown in Table 4, both baseline architectures achieved identical A_{cc} and F_1 values. Increasing the number of hidden neurons from eight to nine slightly improved *TNR* and BA_{cc} while causing a small reduction in *RC*. These changes indicate a minor shift in the trade-off between majority- and minority-class recognition rather than a substantial improvement in overall classification performance. Therefore, increasing network capacity alone does not consistently improve all evaluation metrics. Overall, both baseline FNN architectures demonstrated stable and highly accurate classification performance and pro-

vide a suitable reference for evaluating the effect of the DBSCAN-generated feature in the subsequent experiments.

4.3. DBSCAN-FNN Classification

The DBSCAN-generated feature was incorporated into the input layer of the FNN, resulting in DBSCAN-FNN architectures with nine input neurons. To evaluate the influence of both the clustering-derived feature and network capacity, two DBSCAN-FNN architectures were considered: 9-8-1 and 9-9-1. The confusion matrix counts, obtained directly from the MATLAB Classification Learner, are shown in Table 5.

Table 5. Results of DBSCAN-FNN classification.

Metric	9-8-1	9-9-1
TP	45,029	45,029
TN	1529	1555
FP	172	146
FN	233	233
A_{cc}	99.14%	99.19%
FPR	10.11%	8.58%
FNR	0.51%	0.51%
TNR	89.89%	91.42%
PR	99.62%	99.68%
RC	99.49%	99.49%
NPV	86.78%	86.97%
F_1	99.55%	99.58%
BA_{cc}	94.69%	95.45%

Both models achieved high classification performance across all evaluation metrics. When the number of hidden neurons was increased from eight to nine, A_{cc} improved from 99.14% to 99.19%, while F_1 increased from 99.55% to 99.58%. The 9-9-1 architecture also achieved higher *TNR* and BA_{cc} values, indicating improved recognition of the minority class. These results suggest that, within the evaluated DBSCAN-FNN configurations, increasing the hidden layer size from eight to nine neurons improves minority-class recognition and overall classification performance.

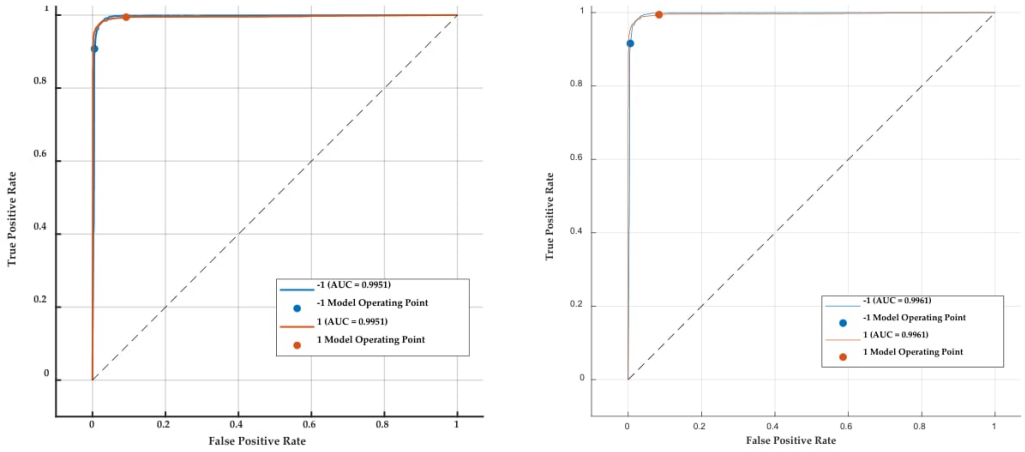
4.4. Comparative Analysis of FNN and DBSCAN-FNN Models

To investigate whether the observed performance improvements were associated with the DBSCAN-generated feature, the network architecture, or a combination of both, four configurations were evaluated: FNN (8-8-1), FNN (8-9-1), DBSCAN-FNN (9-8-1), and DBSCAN-FNN (9-9-1). The comparative results presented in Table 6 provide additional insight into the influence of both the DBSCAN-generated feature and the neural network architecture on classification performance.

Table 6. Comparative results of the investigated architectures.

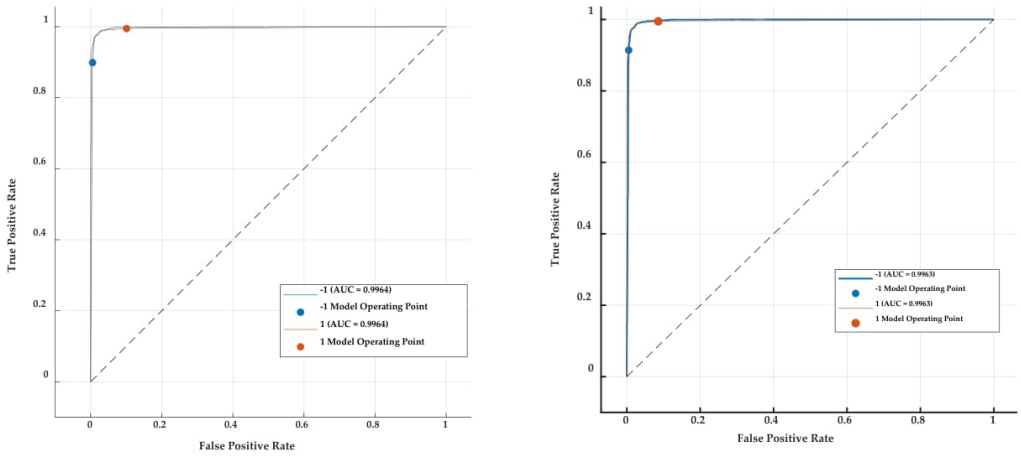
Metric	8-8-1	8-9-1	9-8-1	9-9-1
A_{cc}	99.14%	99.14%	99.14%	99.19%
TNR	90.77%	91.59%	89.89%	91.42%
RC	99.45%	99.43%	99.49%	99.49%
F_1	99.55%	99.55%	99.55%	99.58%
BA_{cc}	95.11%	95.51%	94.69%	95.45%

Increasing the number of hidden neurons from eight to nine in the baseline FNN slightly improved *TNR* (from 90.77% to 91.59%) and BA_{CC} (from 95.11% to 95.51%), while producing only marginal changes in the remaining performance metrics. This indicates that increasing network capacity alone does not substantially improve overall classification performance. Incorporating the DBSCAN-generated feature into the 9-8-1 architecture increased *RC* (from 99.45% to 99.49%) and reduced the number of false negatives (from 249 to 233), indicating improved detection of the majority-class instances. However, this improvement was accompanied by a decrease in *TNR* and BA_{cc}, suggesting a trade-off between majority- and minority-class recognition. The DBSCAN-FNN (9-9-1) architecture achieved the highest A_{cc} (99.19%), *RC* (99.49%), and F_1 (99.58%) among the evaluated models. In contrast, the FNN (8-9-1) architecture achieved the highest BA_{cc} (95.51%). These results indicate that neither increasing network capacity nor incorporating the DBSCAN-generated feature alone consistently improves all evaluation metrics. Therefore, the observed performance gains cannot be attributed exclusively to either factor. Instead, the results suggest that the classification performance is influenced by the combined effect of the DBSCAN-generated feature and the network architecture configuration. Figures 3 and 4 complement the numerical results presented in Table 6 by illustrating the ROC curves of the evaluated FNN and DBSCAN-FNN architectures.



(**a**) (**b**)

Figure 3. (**a**) Receiver operating characteristic (ROC) curve FNN (8-8-1); (**b**) ROC curve FNN (8-9-1).



(**a**) (**b**)

Figure 4. (**a**) ROC curve for DBSCAN-FNN (9-8-1); (**b**) ROC curve for DBSCAN-FNN (9-9-1).

5. Conclusions and Further Work

In this paper, we present an experimental evaluation of a clustering-assisted anomaly detection framework based on the DBSCAN algorithm and FNN. The DBSCAN-generated feature was incorporated into the FNN input layer and was evaluated through four configurations: FNN (8-8-1), FNN (8-9-1), DBSCAN-FNN (9-8-1), and DBSCAN-FNN (9-9-1). The experimental results showed that increasing network capacity alone produced only limited changes in classification performance, while the inclusion of the DBSCAN-generated feature improved *RC* and reduced the number of false negatives. However, neither factor alone consistently improved all evaluation metrics. Among the evaluated architectures, DBSCAN-FNN (9-9-1) achieved the highest A_{cc} (99.19%), *RC* (99.49%), and F_1 (99.58%), while FNN (8-9-1) achieved the highest BA_{cc}. These findings suggest that anomaly detection performance is influenced by both the clustering-generated feature and the network architecture configuration. The results further indicate that clustering-assisted feature generation can provide complementary information that supports FNN-based classification of network traffic.

A limitation of this study is that the proposed method was evaluated solely on the Kyoto 2006+ dataset. Therefore, although the obtained results demonstrate promising performance on this benchmark, additional validation using other intrusion detection datasets is required before broader conclusions regarding robustness and generalization across diverse network environments can be drawn. Furthermore, feature scaling and DBSCAN clustering were performed on the selected features prior to cross-validation. Consequently, the DBSCAN-generated feature may have been influenced by samples later assigned to the test folds. In addition, the present study did not include independent repeated experiments, formal statistical significance testing, or a dedicated analysis of computational cost. As a result, the reproducibility, statistical significance, and runtime efficiency of the observed performance differences remain to be further investigated.

Future work will focus on addressing these limitations by performing evaluations on additional datasets and conducting repeated experimental runs accompanied by appropriate statistical significance tests. Further research will also investigate the computational efficiency and scalability of the proposed approach under larger traffic volumes and real-time operating conditions. In addition, further studies will explore enhanced preprocessing techniques, optimizing CAC parameters, dimensionality reduction techniques, and more advanced neural architectures beyond the single-hidden-layer FNN employed in this work. These extensions are expected to provide further insight into the robustness, scalability, and practical applicability of the proposed framework.

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

Funding: This research received no external funding.

Data Availability Statement: The proposed approach was evaluated using the features from the Kyoto 2006+ dataset, with experiments conducted on the daily subset 20080907.txt (7 September 2008), Available at: https://www.kaggle.com/datasets/harshwardhanbhangale/kyoto-2006.

Acknowledgments: Danijela Protic, Jordan Atanasijevic and Nenad Stojanovic acknowledge the support of the Center for Applied Mathematics and Electronics under the project CPME/O1/25-27 (6/25-27).

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

References

1. Razzaq, K.; Shah, M. Machine learning and deep learning paradigms: From techniques to practical applications and research frontiers. *Computers* **2025**, *14*, 93. [CrossRef]
2. Tufail, S.; Riggs, H.; Tariq, M.; Sarwat, A.I. Advancements and challenges in machine learning: A comprehensive review of models, libraries, applications, and algorithms. *Electronics* **2023**, *12*, 1789. [CrossRef]
3. Nurusheva, A.; Abdiraman, A.; Satybaldina, D.; Goranin, N. Machine learning algorithms in SIEM systems for enhanced detection and management of security events Bull. L N Gumliyov Eurasian. *Natl. Univ. Math. Comput. Sci. Mech. Ser.* **2024**, *148*, 6–17. [CrossRef] [PubMed]
4. Sadeghiamirshahidi, N.; Kamali, S.E.; Dere, B.R.R. Applications of machine learning across smart manufacturing, healthcare, finance, computer vision, robotics, and environmental & sustainability: A systematic literature review. *Appl. Sci.* **2026**, *16*, 6574. [CrossRef]
5. Rehman, R.U. *Intrusion Detection Systems with Snort: What is Intrusion Detection*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2003; pp. 5–10.
6. Kajal, R.; Devi, S. Intrusion detection systems: A review. *J. Netw. Inf. Secur.* **2013**, *1*, 15–21.
7. Protic, D. XOR-Based Detector of Conflicting Decisions on Anomalies in the Computer Networks. Ph.D. Thesis, Singidunum University, Belgrade, Serbia, 29 May 2023.
8. Rigaki, M. *Adversarial Deep Learning Against Intrusion Detection Classifiers*; Lulea University of Technology: Lulea, Sweden, 2017; pp. 1–33.
9. Mittal, N.K. A survey on wireless sensor network for community intrusion detection systems. In Proceedings of the 3rd International Conference on Recent Advances in Information Technology (RAIT), Dahanbad, India, 3–5 March 2016; pp. 107–111.
10. Wang, D.; Huang, Y.; Cai, Z. A two-phase clustering approach for traffic accident black spots identification: Integrated GIS-based processing and HDBSCAN model. *Int. J. Inj. Control. Saf. Promot.* **2023**, *30*, 270–281.
11. Song, J.; Takakura, H.; Okabe, Y. Description of Kyoto University Benchmark Data. Available online: https://www.takakura.com/Kyoto_data/BenchmarkData-Description-New.pdf (accessed on 28 October 2026).
12. Protic, D. Review of KDD CUP '99, NSL-KDD and Kyoto 2006+ datasets. *Mil. Tech. Cour.* **2018**, *66*, 580–595. [CrossRef]
13. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *IEEE Trans. Dependable Secur. Comput.* **2012**, *9*, 453–467. [CrossRef]
14. Thakkar, A.; Lohiya, R. A review of the advancement in the intrusion detection datasets. *Procedia Comput. Sci.* **2020**, *167*, 636–645. [CrossRef]
15. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009), Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6. [CrossRef]
16. Kolias, C.; Kambourakis, G.; Stavrou, A.; Gritzalis, S. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 184–208. [CrossRef]
17. Creech, G.; Hu, J. Generation of a new IDS test data set: Time to retire the KDD collection. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2013), Shanghai, China, 7–10 April 2013; pp. 4487–4492. [CrossRef]
18. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167. [CrossRef]
19. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network intrusion detection Future. *Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
20. García, S.; Grill, M.; Stiborek, J.; Kreibich, C. An empirical comparison of botnet detection methods. In Proceedings of the 2014 IEEE International Conference on Computer Communications (INFOCOM 2014), Toronto, ON, Canada, 27 April–2 May 2014; pp. 1–9. [CrossRef]
21. 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 (BADGERS@EuroSys 2011), Salzburg, Austria, 10 April 2011; pp. 29–36. [CrossRef]
22. Chan, J.Y.L.; Leow, S.M.H.; Bea, K.T.; Cheng, W.K.; Phoong, S.W.; Hong, Z.W.; Chen, Y.L. Mitigating the multicollinearity problem and its machine learning approach: A review. *Mathematics* **2022**, *10*, 1283. [CrossRef]
23. Hoque, M.A.; Kibria, B.M.G.; Bursac, Z. New two-parameter ridge estimators for addressing multicollinearity in linear regression: Theory, simulation, and applications. *Stats* **2026**, *9*, 15. [CrossRef]
24. Zhang, Y.; Chen, X.; Jin, Y.; Wang, J.; Li, J.; Liu, Y.; Zhao, Y. Feature selection for network intrusion detection using correlation analysis. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 4531–4543.
25. Protic, D.; Gaur, L.; Stankovic, M.; Rahman, A.M. Cybersecurity in smart cities: Detection of opposing decisions on anomalies in the computer network behavior. *Electronics* **2022**, *11*, 3718. [CrossRef]

26. Wang, Y.; Li, J.; Zhang, H.; Chen, X. Dimensionality reduction techniques for intrusion detection systems: A comparative study. *J. Netw. Comput. Appl.* **2020**, *165*, 102694.
27. Pearson Sheppert, A. Techniques for mitigating overfitting in machine learning: A comprehensive review, taxonomy, and practical guide. *Front. Artif. Intell.* **2026**, *9*, 1794271. [CrossRef] [PubMed]
28. Zou, H.; Hastie, T.; Tibshirani, R. On the "degrees of freedom" of the lasso. *Ann. Stat.* **2023**, *35*, 2137–2192. [CrossRef]
29. Singthongchai, J.; Wangkhamhan, T. Adaptive normalization and Dropout regularization in deep learning for medical imaging. *J. Imaging* **2026**, *12*, 14. [CrossRef] [PubMed]
30. Izonin, I.; Tkachenko, R.; Shakhovska, N.; Ilchyshyn, B.; Singh, K.K. A two-step data normalization approach for improving classification accuracy in the medical diagnosis domain. *Mathematics* **2022**, *10*, 1942. [CrossRef]
31. Tokareva, A.; Starodubtseva, N.; Frankevich, V.; Silachev, D. Minimizing cohort discrepancies: A comparative analysis of data normalization approaches in biomarker research. *Computation* **2024**, *12*, 137. [CrossRef]
32. Mand, M.; Singh, B.; Kukreja, V.K. Evaluating the impact of data normalization on probability distributions. *Appl. Sci.* **2026**, *16*, 5603. [CrossRef]
33. Protic, D.; Stankovic, M.; Prodanovic, R.; Vulic, I.; Stojanovic, G.M.; Simic, M.; Ostojic, G.; Stankoviski, S. Numerical feature selection and hyperbolic tangent feature scaling in machine learning-based detection of anomalies in the computer network behavior. *Electronics* **2023**, *12*, 4158. [CrossRef]
34. Protic, D.; Stankovic, M. A hybrid model for anomaly-based intrusion detection in complex computer networks. In Proceedings of the 2020 21st Arab Conference on Information Technology, Giza, Egypt, 28–30 November 2020; pp. 1–8.
35. Ahmad, T.; Aziz, M.N. Data preprocessing and feature selection for machine learning intrusion detection systems. *ICIC Express Lett.* **2019**, *13*, 93–101.
36. Roszkowska, E. Entropy and normalization in MCDA: A data-driven perspective on ranking stability. *Entropy* **2026**, *28*, 114. [CrossRef] [PubMed]
37. Basu, S.; Banerjee, A.; Mooney, R.J. Semi-supervised clustering by seeding. In Proceedings of the 19th International Conference on Machine Learning (ICML), Sydney, Australia, 18–12 July 2002; pp. 27–34.
38. Xu, R. Clustering algorithms in biomedical research: A review. *IEEE Rev. Biomed. Eng.* **2010**, *3*, 120–154. [CrossRef] [PubMed]
39. Ghosh, J.; Acharya, A. Cluster ensembles. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2011**, *1*, 305–315. [CrossRef]
40. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **2010**, *31*, 651–666. [CrossRef]
41. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR, USA, 2–4 August 1996; pp. 226–231.
42. Protic, D.; Stankovic, M. Anomaly-based intrusion detection: Feature selection and normalization instance to the machine learning model accuracy. *Eur. J. Eng. Form. Sci.* **2018**, *1*, 43–48. [CrossRef]
43. Avila, R.; Khoury, R.; Pere, C.; Khanmohammadi, K. Employing Feature Selection to Improve the Performance of Intrusion Detection Systems. In *Foundations and Practice of Security (FPS 2021)*; LNCS 13291; Springer: Berlin/Heidelberg, Germany, 2022; pp. 93–112. [CrossRef]
44. Zhao, F.; Zhao, J.; Niu, X.; Luo, S.; Xin, Y. A Filter Feature Selection Algorithm Based on Mutual Information for Intrusion Detection. *Appl. Sci.* **2018**, *8*, 1535. [CrossRef]
45. Powers, D.M.W. Evaluation: From Precision, Recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63. [CrossRef] [PubMed]
46. Erbani, J.; Portier, P.-.É.; Egyed-Zsigmond, E.; Nurbakova, D. Confusion Matrices A Unified Theory. *IEEE Access* **2024**, *12*, 1. [CrossRef]
47. Zhang, Y.; Chen, X.; Jin, L.; Wang, X.; Guo, D. Network Intrusion Detection: Based on Deep Hierarchical Network and Original Flow Data. *IEEE Access* **2019**, *7*, 37004–37016. [CrossRef]
48. Liu, J.; Qin, H.; Liu, Z.; Wang, S.L.; Zhang, Q.; He, Z.; Lei, S. A density-based spatial clustering of application with noise algorithm and its empirical research. *Highlights Sci. Eng. Technol.* **2022**, *7*, 174–179. [CrossRef]
49. A Guide to the DBSCAN Clustering Algorithm. Available online: https://www.datacamp.com/tutorial/dbscan-clustering-algorithm (accessed on 27 July 2026).
50. Atanasijevic, J.; Protic, D.; Kolasinac, D.; Ivanovic, S. A modified DBSCAN algorithm for traffic accident clustering. In Proceedings of the 52nd International Symposium of Operational Research, Palic, Serbia, 7–10 September 2025; pp. 74–79.
51. MATLAB 2023b Release Highlights. Available online: https://www.mathworks.com/products/new_products/release2023b.html (accessed on 4 August 2026).
52. Sokolova, M.; Lapalme, G. A Systematic Analysis of Performance Measures for Classification Tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]
53. Silva, L.M.; de Sa, J.M.; Alexandre, L.A. Data classification with multilayer perceptrons using generalized error function. *Neural Netw.* **2008**, *21*, 1302–1310. [CrossRef] [PubMed]

54. Levenberg, K. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Q. Appl. Math.* **1944**, *2*, 164–168. [CrossRef]
55. Marquardt, D.W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Ind. Appl. Math.* **1963**, *11*, 431–441. [CrossRef]
56. Trainlm Levenberg-Marquardt Backpropagation. Available online: https://www.mathworks.com/help/deeplearning/ref/trainlm.html (accessed on 26 October 2026).
57. Ni, L.; Zhang, S.; Huang, K.; Wang, Y. Multi-level screening method for network security alarms based on DBSCAN algorithm and Rete Rule Inference. *Sci. Rep.* **2026**, *16*, 36369. [CrossRef] [PubMed]

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