TYPE Original Research
PUBLISHED 03 November 2025
DOI 10.3389/fcomp.2025.1676362

Check for updates

OPEN ACCESS

EDITED BY
Dalius Navakauskas,
Vilnius Gediminas Technical University,
Lithuania

REVIEWED BY
Poongodi Chinnasamy,
Christ University, India
Darius Plonis,
Vilnius Gediminas Technical University,
Lithuania

*CORRESPONDENCE
Evita Roponena
evita.roponena@rtu.lv

RECEIVED 30 July 2025
ACCEPTED 14 October 2025
PUBLISHED 03 November 2025

CITATION
Roponena E, Poļaka I and Grabis J (2025) Anomaly detection in netflow traffic: workflow for dataset preparation and analysis. *Front. Comput. Sci.* 7:1676362. doi: 10.3389/fcomp.2025.1676362

Anomaly detection in netflow traffic: workflow for dataset preparation and analysis

Evita Roponena*, Inese Poļaka and Jānis Grabis

Institute of Information Technology, Riga Technical University, Riga, Latvia

Information and communication technology (ICT) is crucial for maintaining efficient communications, enhancing processes, and enabling digital transformation. As ICT becomes increasingly significant in our everyday lives, ensuring its security is crucial for maintaining digital trust and resilience against evolving cyber threats. These technologies generate a large amount of data that should be analyzed simultaneously to detect threats to an ICT system and protect the sensitive information it may contain. NetFlow is a network protocol that can be used to monitor network traffic, collect Internet Protocol (IP) addresses, and detect anomalies in NetFlow. The article follows the design science research (DSR) methodology to reach an objective of providing a methods for developing a set of features for NetFlow analysis with a machine learning. The sets of features were analyzed and validated by implementing anomaly detection with the K-means clustering algorithm and time-series forecasting using the long short-term memory (LSTM) method. The study provides two separate sets of features for both machine learning methods (24 features for clustering and 14 for LSTM), an overview of the anomaly detection methods used in this research and a method to combine both machine learning approaches. Furthermore, this study introduces a method that integrates the outputs of both models and evaluates the reliability of the final decision based on Bayes' theorem and previous performance of the models.

KEYWORDS

anomaly detection, Bayes theorem, clustering, feature engineering, machine learning, NetFlow, time-series

1 Introduction

Today, most institutions face cybersecurity risks due to the extensive use of information and communication technology (ICT) devices. A large part of business and everyday processes are held in a virtual environment, which creates a possibility for both outside and inside threats. These challenges affect all institutions, as most institutions rely on ICT to process data and provide online services. This creates possible security breaches affecting business quality and data integrity. Cyber Incident Response Institution CERT.LV summarized statistics on Latvian cyberspace during the first quarter of 2025 (CERT.LV, 2025) showing that the number of cybersecurity incidents has increased by 11% and the percentage of denial-of-service attacks has also risen compared to the previous quarter. Unfortunately, insufficient cybersecurity knowledge among employees also affects cybersecurity. For example, identity theft, lost devices, insecure third-party email attachments, and malicious websites are some of the usual causes of these incidents.

Therefore, fast identification of cybersecurity threats is essential to ensure information security at any institution. Monitoring the data flow for known threats enables taking immediate action after the identification. However, anomaly detection can identify previously unknown threats by recognizing changes in data patterns. NetFlow is a CISCO network protocol that has the capability to collect Internet Protocol (IP) network traffic that can be used to monitor tasks and analyze network traffic (Cisco Systems, 2011). Currently, a variety of network intrusion detection system (NIDS) datasets are available that contain benign traffic and synthetic attack scenarios (Ring et al., 2019). In these datasets, each data sample is represented by a set of features (SoF), and these features are used in machine learning (ML) tasks to train and evaluate models. However, a lack of a standardized set of features leads to every dataset having its own unique set of features (Sarhan et al., 2021). Therefore, the datasets contain many irrelevant features to specific contexts, and it is hard to evaluate ML models used to identify anomalies on the basis of these datasets.

The researchers in Sarhan et al. (2021) propose a method for converting the available NIDS dataset into a standardized NetFlow dataset with 12 features: source and destination IP address and port number, protocol, Transmission Control Protocol (TCP) flags, layer 7 protocol, incoming bytes and packets, outgoing bytes and packets, and flow duration. The NetFlow dataset proposed in Komisarek et al. (2023) additionally contains features such as flow ID, time of the first and the last packet of the flow, protocol map, total flows, anomaly category, and anomaly. The results of the experiments in Fraihat et al. (2023) provided satisfactory threat identification using only seven characteristics: protocol, TCP flags, server TCP flags, largest packet length, source to destination bytes/second, maximum flow Time-To-Live (TTL), and maximum TCP window (source to destination). The aggregated NetFlow feature dataset consisting of 39 descriptive features was proposed in Minkevics and Kampars (2021). Despite contributions made in the aforementioned studies, the absence of a standardized feature set for the NetFlow dataset highlights the need for continued research in this area.

A literature review on the ML techniques for cybersecurity (Roponena et al., 2021) showed that network traffic analysis is the most popular cybersecurity task in the reviewed literature. Although supervised machine learning methods were mostly used, it is an inadequate approach for real-life, unlabeled flow analysis. Unsupervised methods proposed in the literature were clustering methods that group data into clusters based on their similarities. Several clustering techniques were mentioned in the literature for various tasks: X-means (Gu et al., 2008), Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Dias et al., 2020), K-means (Dias et al., 2020), Self-Organizing Map (SOM) (Fraihat et al., 2023), Agglomerative Clustering (Komisarek et al., 2023), and Spectral Clustering (Chiou et al., 2014).

However, unsupervised learning can lead to a high rate of false positives and false negatives, negatively influencing the performance of cybersecurity operations. Even if an ML model concludes that the specific data point is an anomaly, it does not necessarily indicate that it is an actual anomaly or malicious. Usually, ML models are tested in artificial or constrained scenarios, which limits the relevance of the results in real-life settings, even though the model shows high precision during testing (Doshi-Velez and Kim, 2017). Robust decision-making in high-stakes scenarios, such as cybersecurity management, necessitates either demonstrable trust in model predictions, for example, explainable results, or a reliable estimation of the likelihood that a given output is correct.

There are several techniques available to combine the outputs of multiple models into one final result. Ensemble learning combines several baseline models to build a single model by fusing baseline model outputs using voting with assigned weights (Mohammed and Kora, 2023). This is proven to improve prediction performance in many studies, including our previous research on supervised NetFlow analysis (Roponena and Polaka, 2022); however, ensemble learning introduces complexity and increases computation time and resource requirements. Statistical methods such as Bayes' Theorem (Theodoridis, 2015), Bayesian Model Averaging (BMA) (Fragoso et al., 2018), and the frequentist approach are applicable to answer the question: how reliable is the combined answer from several models? However, BMA is useful when probabilistic predictions are used, meaning that in a binary classification case BMA average might not correspond to any feasible model prediction (Hinne et al., 2020). The frequentist approach does not incorporate prior knowledge into the analysis and is based solely on given data, whereas Bayesian approaches are performed additionally, incorporating prior belief (Jun, 2016).

This research complements the previously published article (Roponena et al., 2024) on *How to prepare a suitable machine learning dataset for NetFlow anomaly analysis?* where dataset preparation refers to a method of data collection, preprocessing, and analysis. This study provides additional material for previously defined questions and includes a new research question: *How to combine and evaluate the reliability of several unsupervised ML models' results?* Thus, the objective of this study is to provide a method for developing a set of features (SoF) for NetFlow analysis, as well as defining common methods for NetFlow anomaly detection and a method for assessing the reliability of ML output to help define the SoF and the process of data analysis. This study contributes to the field of network anomaly detection by defining a SoF specifically designed for NetFlow records and developing methods for preparing these features for ML tasks. While the underlying dataset is not published, the feature definitions and preparation steps are made explicit to support reproducibility and application in other contexts.

Section 2 details used methods in this study. Section 3 provides the SoF and reliability assessment. The results are discussed in Section 4, which also provides limitations and future research directions.

2 Materials and methods

The research was conducted following the design science research methodology (vom Brocke et al., 2020). The research was divided into four design cycles, as shown in Figure 1. Each design cycle contributes to the development of the main artifacts of this research: NetFlow anomaly detection set of features (SoF), methods for anomaly detection, and the ML output reliability assessment method. The purpose of the SoF is defined as a NetFlow dataset for anomaly analysis with unsupervised ML. Various steps were initially chosen to prepare the dataset, which included NetFlow data preprocessing, feature dimensionality reduction, and possible feature synthesis. However, several changes were made during the research depending on the chosen ML methods (clustering and time-series forecasting) and the findings during the analysis of the NetFlow patterns, including dividing the NetFlow anomaly SoF

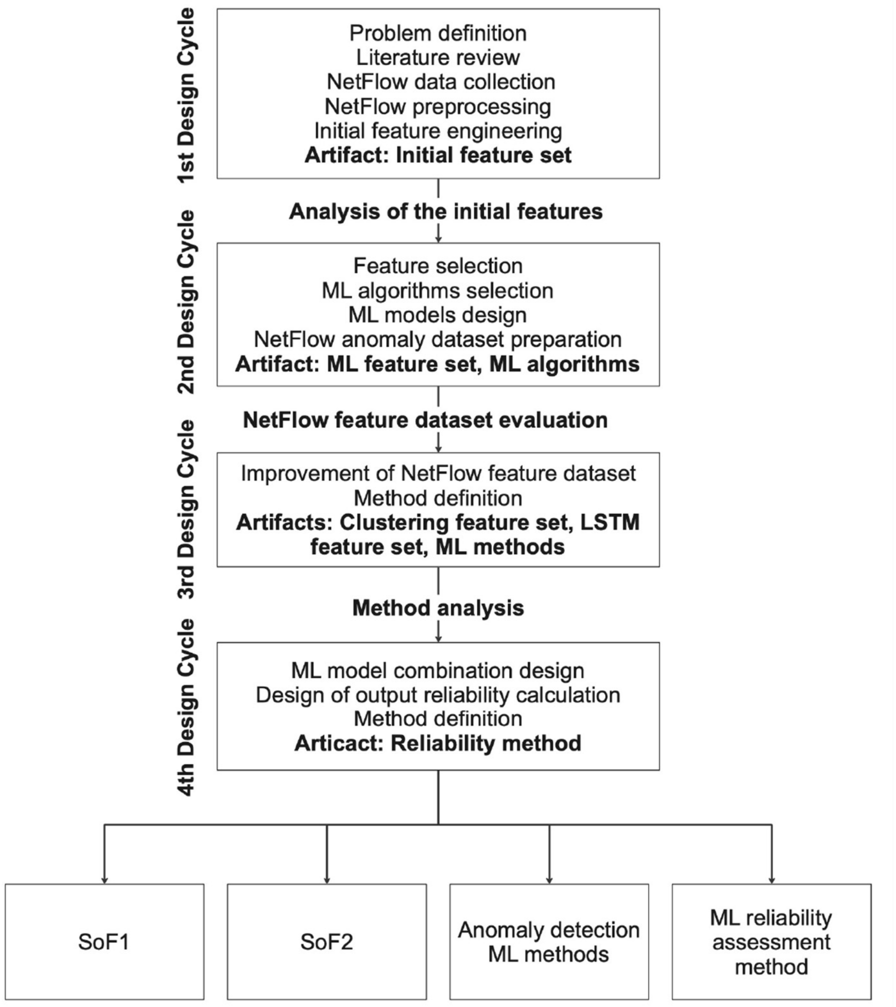

1st Design Cycle
Problem definition
Literature review
NetFlow data collection
NetFlow preprocessing
Initial feature engineering
Artifact: Initial feature set
Analysis of the initial features
2nd Design Cycle
Feature selection
ML algorithms selection
ML models design
NetFlow anomaly dataset preparation
Artifact: ML feature set, ML algorithms
NetFlow feature dataset evaluation
3rd Design Cycle
Improvement of NetFlow feature dataset
Method definition
Artifacts: Clustering feature set, LSTM feature set, ML methods
Method analysis
4th Design Cycle
ML model combination design
Design of output reliability calculation
Method definition
Articact: Reliability method
SoF1
SoF2
Anomaly detection ML methods
ML reliability assessment method

FIGURE 1
NetFlow feature dataset preparation and anomaly detection methods development research steps.

artifact into two SoF based on the ML method. The final research steps are described in this section.

The first design cycle defines the research problem and a review of related work provided in Section 1. The technology stack was chosen to design the solution, consisting of *Apache Spark* (The Apache Software Foundation, 2024) for big data processing and Python programming language (Rossum and Drake, 2009) for the development of ML models. Real-life raw NetFlow data collection was performed for the initial analysis, feature engineering, and preparation of SoF. The Kolmogorov-Smirnov test (KS test), the initial feature correlation analysis, and other tests were performed for the initial feature analysis after the first cycle to evaluate features before their inclusion in SoF.

The analysis of initial SoF allowed us to select the most relevant NetFlow features for subsequent analysis in the second design cycle. It enabled the preparation of a second SoF containing manually induced NetFlow anomalies, generated by subjecting ICT devices to stress testing. Anomaly detection methods were analyzed to select ML algorithms, considering the chosen technologies in the first cycle. The K-means clustering algorithm was chosen as an anomaly detection method, while the long short-term memory (LSTM) algorithm was selected for detecting anomalies in time-series. In addition, the second cycle includes the development of ML models and anomaly detection algorithms. The ML models were used to evaluate SoF suitability for anomaly detection to conclude this cycle.

The third design cycle included the refinement of the SoF necessary for the improvement of NetFlow anomaly detection with the selected ML algorithms. This cycle included the final design of the NetFlow anomaly detection feature datasets. Additionally, the designed anomaly detection algorithm was conceptualized into a method.

The fourth and last design cycle provides a method for combining ML model outputs and reliability assessment of the results. The last cycle yields the final versions of the set of NetFlow features for K-means (SoF1) and LSTM (SoF2), the ML methods for anomaly detection, and the ML reliability assessment method.

TABLE 1 Selected NetFlow fields.

NetFlow field	Field description
SrcAddr	Source IP address
DstAddr	Destination IP address
TimeFlowEnd	Time the flow ended
TimeFlowStart	Time the flow started
Bytes	Number of bytes in the flow
Packets	Number of packets in the flow
TCPFlags	TCP flag number that contains all TCP flags used in the flow

2.1 Netflow collection and analysis

The *NetFlow v.9* protocol was used to collect the NetFlow data from several switches and routers within the Riga Technical University network in *JSON* format. The data were pseudonymized to protect personal information, such as IP addresses. However, it is possible to return data to its initial state for validation purposes. Therefore, the NetFlow data used in this research are not publicly available. Not all collected NetFlow fields were recognized as useful for anomaly detection purposes, and only the fields presented in Table 1 were selected.

Data were validated to review their quality; for example, *NaN* values were excluded because they indicate errors during the data collection process. All data fields were transformed to the right data type and format with *Apache Spark*, for example, numeric fields as numeric. The *TCPFlags* numeric values were transformed to a binary value for the specific *TCP flag* extraction. For example, if the numeric value of the *TCPFlags* is *27*, the corresponding binary value is *0001 1011*, which indicates that the record contains four TCP flags: Acknowledgment (ACK), Push (PSH), Synchronization (SYN), and Finish (FIN).

The literature review concluded that there is no standard SoF for NetFlow analysis. Therefore, the NetFlow features were selected based on the features proposed in Minkevics and Kampars (2021). The initial SoF was created by dividing the data into 10-min time-windows using the *TimeFlowEnd* field. Afterward, the data were grouped by the *TimeFlowEnd* field and the source IP address to perform data aggregation within the created groups, obtaining the maximal, minimal, average, and total values of the features, as well as the bytes and packets variance of each group. Feature extraction by aggregation within a time-window helps to reduce the size of the dataset and obtain statistics about the data flow.

An additional data transformation was performed for the analysis of the initial SoF. Non-numeric data fields, for example, IP addresses, were removed from the SoF. The data were normalized using the Python built-in method *FunctionTransformer* to transform values to their logarithmic values from 0 to 1. Normalization was needed to ensure that the values in the SoF are within a similar range for the feature analysis.

The initial SoF was analyzed to find useful data patterns. The boxplots were created to identify which features have visible outliers. The boxplot diagram showed that the TCP flags, namely, Urgent (URG), Explicit Congestion Notification (ECE), and Congestion Windows Reduced (CWR), had only the value 0. The presence of the TCP flags URG (not frequently used nowadays), ECE, and CWR (recommended to be blocked by the firewall) in the future network flow could indicate an anomaly. The features *AverageBytes*, *AveragePackets*, *MinPackets*, *MaxPackets*, *MinFlowTime*, *PacketsVariance*, and *BytesVariance* had values outside their usual amplitude range. This indicates that the observed NetFlow contained outlier values, but does not explicitly conclude the presence of the anomalies. The data used to create the boxplots were not filtered by the device. An in-depth analysis of each device's NetFlow patterns could improve the findings.

The KS test allows us to compare the distribution of a sample against a given distribution, including the normal distribution (Virtanen et al., 2020). The null hypothesis of the test is that the distribution of the observed NetFlow is normal. The Python library *SciPy* (Virtanen et al., 2020) function *kstest* returns two values *statistic* and *pvalue* that are used to confirm or reject the null hypothesis. The statistic value measures the maximum difference between the empirical distribution function of the sample and the specified distribution function. If the *pvalue* is less than a specified significance level (e.g., 0.05), then the null hypothesis can be rejected. After applying the *kstest* for the values of each feature, it was concluded that the *pvalue* is lower than 0.05 for each feature, and the data do not have a normal distribution.

Feature correlation was calculated using Python class *corr*. The highest correlation was between the aggregated features of bytes and packets, flow time and packets, and TCP flags that come in pairs, for example, ACK and PSH. This implies that excluding a feature from a highly correlating pair would not be beneficial for NetFlow analysis, as each feature describes different characteristics of the flow. However, changes in the correlation coefficient could indicate an anomaly in the traffic.

After the initial analysis, it was decided to include all previously described features for anomaly detection with the ML models and include the IP addresses for the preprocessing and evaluation steps.

2.2 Design of machine learning models and anomaly detection

The first design cycle provided the initial SoF that was used and improved during the second design cycle. It started with a review of the ML algorithms used for NetFlow analysis. The NetFlow dataset used in this study did not contain labeled data that would indicate whether the data flow was benign or if it contained an anomaly or malicious traffic. Therefore, unsupervised learning was chosen as the most suitable way to identify anomalies in the dataset that could indicate malicious activity in the network. The dataset was supplemented with anomalous traffic created by overloading analyzed web servers with data flow in a fixed time. The open-source load testing tool *Apache JMeter* was used to overload the selected servers with network traffic from a known IP address. This ensured that there were known anomalies to evaluate the performance of the anomaly detection algorithm.

All clustering algorithms mentioned in the introduction are built in the Python programming language; however, this does not guarantee they can process big data fast enough. *Apache Spark* provides a possibility to solve this problem. Therefore, the K-means clustering algorithm was chosen as it is built into *Apache Spark*. This algorithm is also suitable for outlier detection by adding additional steps.

The K-means algorithm is an iterative algorithm that tries to group a dataset into non-overlapping K clusters (Jain, 2010). The data points inside the clusters are similar, and each cluster is different from the other clusters. The *Apache Spark* K-means implementation supports two distance measurement functions: Euclidean distance and Cosine similarity. The Cosine similarity uses angular distances, which are often used to measure document similarity in text analysis (Han et al., 2012), but in the case of the NetFlow data analysis, not only angle, but also magnitude is significant; therefore, the Euclidean distance was chosen for our dataset analysis.

In high-dimensional spaces, the Euclidean distances are prone to the curse of dimensionality. The dimension of a dataset corresponds to the number of features it contains (in our case, 24 clustering features). The curse of dimensionality refers to difficulties during data analysis and visualization in identifying patterns (Chiou et al., 2014). A dimensional reduction algorithm, for example, principal component analysis (PCA), is recommended to overcome this problem. The dimensions in our ML dataset were reduced from 24 to 2 using the PCA after feature normalization. As a result, it was possible to visualize data using two-dimensional (2D) plots and calculate the outliers in each created cluster.

The scatterplot and elbow method were used to determine the initial number of clusters for a specific web server. While performing K-means on the NetFlow datasets, it was concluded that the amount of NetFlow and the number of clusters changed during different times of the day, and during weekdays and weekends. Therefore, it was decided to divide data into eight subdatasets of the NetFlow dataset: workdays and weekends morning ($h \geq 6$ a.m. and $h < 10$ a.m.), day ($h \geq 10$ a.m. and $h < 5$ p.m.), evening ($h \geq 5$ p.m. and $h < 9$ p.m.), and night ($h \geq 9$ p.m. and $h < 6$ a.m.). Additionally, a new feature *WeekDay* was added to the dataset solely for division purposes, and *UniqueIPfromSource* was included to determine if the analyzed web server has a higher or lower number of connection requests than usual.

The K-means algorithm by itself is not able to identify outliers; therefore, additional steps were added after performing the clustering. It was decided to identify outliers in the dataset based on the Euclidean distance between a data point and the center of the cluster to which it belongs. The maximum distance threshold was calculated by dividing the sum of the maximum and minimum distances by 2 (Barai and Dey, 2017). Each datapoint that exceeded this threshold was considered an outlier. Time-series dataset preparation is required to transform already obtained features (Table 1) into the time-series. The LSTM model was developed using the deep learning API Keras (2015), which is written in Python and runs on top of the open-source machine learning platform *TensorFlow*. The core data structures of *Keras* are layers and models. A *Sequential Model* was used to create an LSTM model structure for NetFlow analysis. The *Sequential model* is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. It allows us to add the necessary layers for the planned neural network. Model architecture consisted of six layers: two LSTM layers, two dense or fully connected layers, one dropout layer, and one batch normalization layer.

Analysis of forward (sent to the ICT device) and backward (sent from the ICT device) traffic flow of one ICT device provides the understanding of the usual pattern of the input and output flow and its changes during or after an attack. The combination of forward and backward NetFlow data provides a more comprehensive picture of the bidirectional communication flow between network endpoints.

The set of aggregated NetFlow features (SoF1) was used to test the suitability of the LSTM model for NetFlow forecasting (Figure 2A); however, the model was unable to identify data patterns using the SoF1. Therefore, a set of non-aggregated NetFlow features (SoF2) was used to train and test the LSTM model (Figure 2B). However, it is important to mention that the SoF2 provided worse results for the clustering and significantly increased the clustering time. The SoF2 contained a number of packets, bytes, flow duration, TCP flag count, weekday, and time delta. The *Timedelta* feature refers to a time interval between current and preceding data samples and was included because of the irregular temporal distribution of the data. The number of packets and bytes was forecasted for the next step by the LSTM model.

Additional outlier detection steps were added to find the outliers based on the value forecasted by the model. Outlier detection using testing data was based on finding a significant difference between the true and predicted NetFlow feature values. The calculated threshold was used to perform outlier detection using a sample of predicted data.

The quantiles and the interquartile range were computed using a list of sorted differences to establish a threshold for the number of packets and bytes. During the attack, the forward flow feature values are more likely to be unusually high than unusually low; therefore, the upper threshold is considered while creating the list of outliers to minimize the false positives. However, the backward flow feature values can be either unusually high, or low, or within a normal range; therefore, both thresholds are used to create the outlier list. The backward flow data sample was considered as an outlier if the difference value exceeded the upper threshold or did not exceed the lower threshold. The forward flow data sample was detected as an outlier if the difference value exceeded the upper threshold. Outlier detection method using quantiles assumes that the data has a symmetrical distribution, which is not present in the feature difference values. However, this method provided fewer false positives than the previously used threshold finding methods, for example, average value calculation.

The findings of the second design cycle provided valuable information to refine the initial SoF for both ML methods presented in this subsection. Additionally, ML anomaly detection methods were designed during this design cycle that can be improved in future work.

2.3 Model reliability assessment

ML models for NetFlow anomaly detection developed in the second design cycle use different data structures. This indicates that multiple data representations (time-series sequences and aggregated statistical features) can both contribute to NetFlow

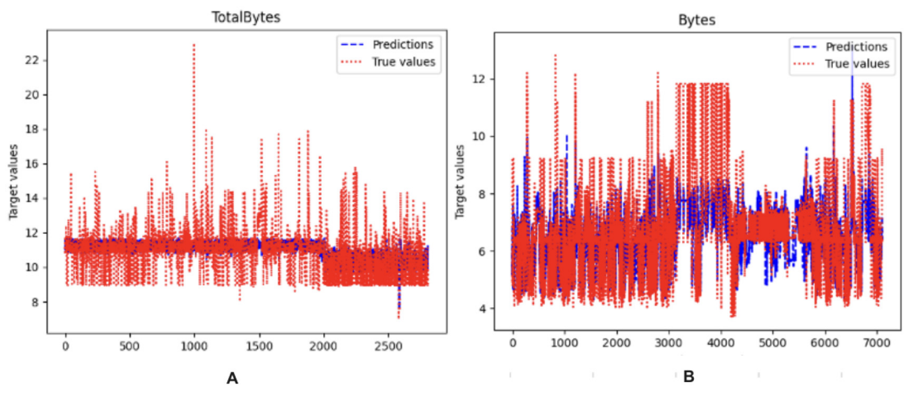


FIGURE 2
Forecasted and true values by LSTM model: **(A)** aggregated feature TotalBytes, **(B)** non-aggregated feature Bytes.

anomaly detection. However, this necessitates the integration of model outputs to mitigate the limitations of the individual models and enhance the reliability of anomaly detection results. This creates an opportunity for ensemble strategy with cross-dataset learning; however, given the complex nature of ensemble methods, it was not implemented in this study.

Statistical calculation of output reliability, or in other words, how trustworthy the result is, was decided as a preferable method in this research study. Bayes' theorem, which deals with conditional probabilities, is applicable in the context of NetFlow anomaly detection by enabling the calculation of the likelihood of an anomaly occurring, given that an ML model has previously observed anomalies. Both model outputs are presented as binary values: 0 when an anomaly is not identified and 1 when an anomaly is observed. Thus, two cases are possible: both models have returned the same result, or the results are opposite. Given that the probability of the right prediction P_A, P_B by both models $A = clustering$ and $B = LSTM$ is known, the reliability of a correct answer $P(A,B)$ can be calculated using Bayes' theorem assuming independence of the ML models predictions. In the case of both models agreeing on the same data label $P(A = 1, B = 1)$ or $P(A = 0, B = 0)$, the Equation 1 can be used. Equation 2 represents the case of disagreement of the models: $P(A = 1, B = 0)$ or $P(A = 0, B = 1)$. The Equation 2 shows the case when the clustering model identifies an anomaly but LSTM not, in the equation for the opposite outcome P_A is replaced with P_B and vice versa.

$$P((A=1,B=1)\vee(A=0,B=0)) = \frac{\mathrm{P}_A\mathrm{P}_B}{\mathrm{P}_A\mathrm{P}_B+(1-\mathrm{P}_A)(1-\mathrm{P}_B)} \quad (1)$$

$$P(A=1,B=0) = \frac{\mathrm{P}_B(1-\mathrm{P}_A)}{\mathrm{P}_B(1-\mathrm{P}_A)+(\mathrm{P}_A)(1-\mathrm{P}_B)} \quad (2)$$

3 Results

After the second design cycle, it was possible to conclude that two different SoF are required for the clustering (SoF1) and time-series forecasting (SoF2). The SoF1 is presented in Table 2. The clustering features were calculated from traffic originating from the source address and directed to the web service, while time-series features were obtained for both flow directions. The final SoF1 consisted of 27 different features, of which 24 were used for actual clustering (excluding *SrcAddr*, *TimeFlowEnd*, and *WeekDay*). The SoF2 is presented in Table 3. The final SoF2 consisted of 12 input features and two output features.

The NetFlow anomaly detection methods were refined during the third cycle. The method for NetFlow anomaly detection using the K-means algorithm was developed, consisting of 12 steps (Figure 3). After the data were prepared according to the SoF1 and split, dimensionality reduction and K-means clustering were performed for each data subset separately. The Euclidean distance is obtained for each datapoint within each cluster of the data subset to compute the maximum distance threshold. This threshold enables outlier detection by identifying data points that exceed it. Thus, providing the list of outliers for further analysis. The clustering revealed that, firstly, network traffic is less on weekends than on workdays. Secondly, the proposed algorithm labels the data point as an outlier when it is located far from the cluster center. Third, the number of clusters should be adjusted for each data subset. The proposed method was able to correctly identify, on average, 83.3% of the anomalous flows. This value is treated as an empirical probability, representing the likelihood of correctly detecting an anomaly.

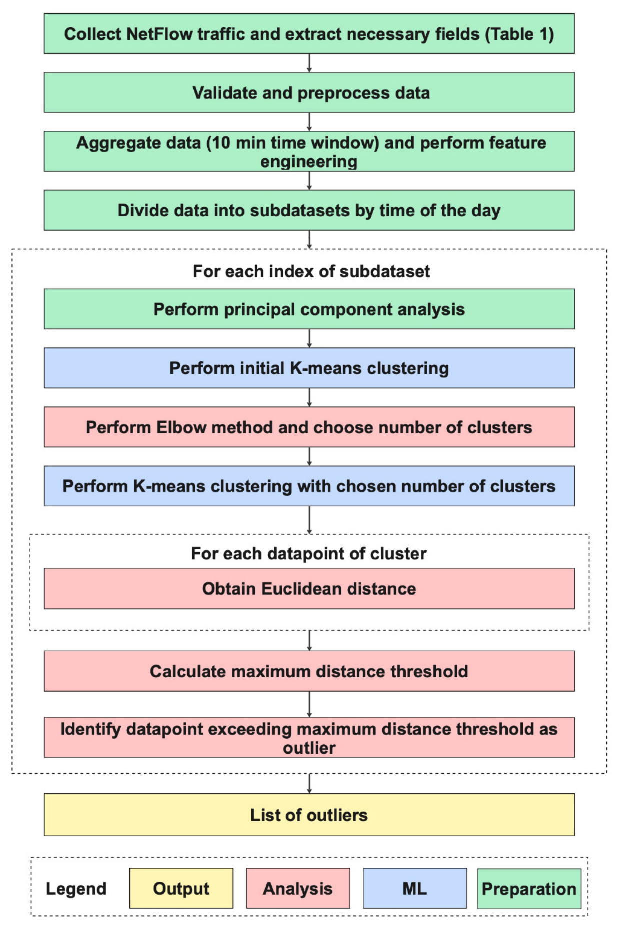
The method for detecting NetFlow anomalies using time-series was designed and refined for the LSTM algorithm. The method consisted of 12 steps (Figure 4). The LSTM model's objective is to analyze NetFlow flow to and from the ICT device. The model uses the previous 10 steps of the data sample to forecast the next step. The forecasted value is then compared to the actual value to compute the difference. Thresholds are chosen depending on the flow direction. The outliers were detected for the number of packets, the number of bytes, and both features, with the results shown in Table 4. The percentages provided in the table are used as an empirical detection probability. True outlier identification probability in the forward flow provided satisfactory results; however, the backward flow analysis provided average results. A significantly lower true positive rate in backward flow analysis was achieved using only the upper threshold. The performance of the

TABLE 2 Set of features for NetFlow clustering (SoF1).

Feature group	Aggregated feature	Description
Preprocessing features	TimeFlowEnd	Time range within 10 min
	SrcAddr	Source IP address
	WeekDay	Weekday when flow has ended
Descriptive byte features	TotalBytes	Total number of bytes
	AvarageBytes	Average number of bytes
	MinBytes	Minimal number of bytes
	MaxBytes	Maximal number of bytes
	BytesVariance	Byte variance
Descriptive packet features	TotalPackets	Total number of packets
	AvaragePackets	Average number of packets
	MinPackets	Minimal number of packets
	MaxPackets	Maximal number of packets
	PacketsVariance	Packet variance
Descriptive flow features	TotalFlowTime	Total flow duration
	AverageFlowTime	Average flow duration
	MinFlowTime	Minimal flow duration
	MaxFlowTime	Maximal flow duration
	TotalFlowNr	Total number of flows
Connections	UniqueIPfromSource	Number of unique connections
TCP flag count	FinCount	FIN flag count
	SYNCount	SYN flag count
	RSTCount	RST flag count
	PSHCount	PSH flag count
	ACKCount	ACK flag count
	URGCount	URG flag count
	ECECount	ECE flag count
	CWRCount	CWR flag count

TABLE 3 Set of features for NetFlow time-series (SoF2).

Feature group	NetFlow field	Field description
Output features	Packets	Number of packets
	Bytes	Number of bytes
Descriptive flow features	Duration	Total flow duration
	WeekDay	Weekday when flow has ended
TCP flag	FIN	Presence of FIN flag in a flow
	SYN	Presence of SYN flag in a flow
	RST	Presence of RST flag in a flow
	PSH	Presence of PSH flag in a flow
	ACK	Presence of ACK flag in a flow
	URG	Presence of URG flag in a flow
	ECE	Presence of ECE flag in a flow
	CWR	Presence of CWR flag in a flow



Collect NetFlow traffic and extract necessary fields (Table 1)
Validate and preprocess data
Aggregate data (10 min time window) and perform feature engineering
Divide data into subdatasets by time of the day
For each index of subdataset
Perform principal component analysis
Perform initial K-means clustering
Perform Elbow method and choose number of clusters
Perform K-means clustering with chosen number of clusters
For each datapoint of cluster
Obtain Euclidean distance
Calculate maximum distance threshold
Identify datapoint exceeding maximum distance threshold as outlier
List of outliers
Legend
Output
Analysis
ML
Preparation

FIGURE 3
K-means clustering steps for anomaly detection in Netflow data.

LSTM model using both features was also evaluated using standard regression task performance metrics Mean Squared Error (MSE), Mean Absolute Error (MAE), R^2, and Mean Absolute Percentage Error (MAPE).

Both MSE and MAE provide complementary perspectives on prediction accuracy: MSE emphasizes larger errors and is sensitive to outliers, while MAE reflects the average error magnitude without disproportionately weighting extreme deviations. Backward flows show lower errors compared to forward flows, suggesting more accurate predictions for backward flows. Similarly, MAPE scores indicate lower relative error for backward flows. R^2 values for both flows are relatively low, implying limited predictive power. Although the standard prediction metrics (MSE, MAE, MAPE, R^2) indicate better forecasting performance for backward flows, this does not improve anomaly detection. Forward flows contain more pronounced anomalies, which increase prediction error. These larger deviations are reflected in higher error metrics, yet they also make anomalies more visible and easier to detect. Therefore, the poorer forecasting performance for forward flows

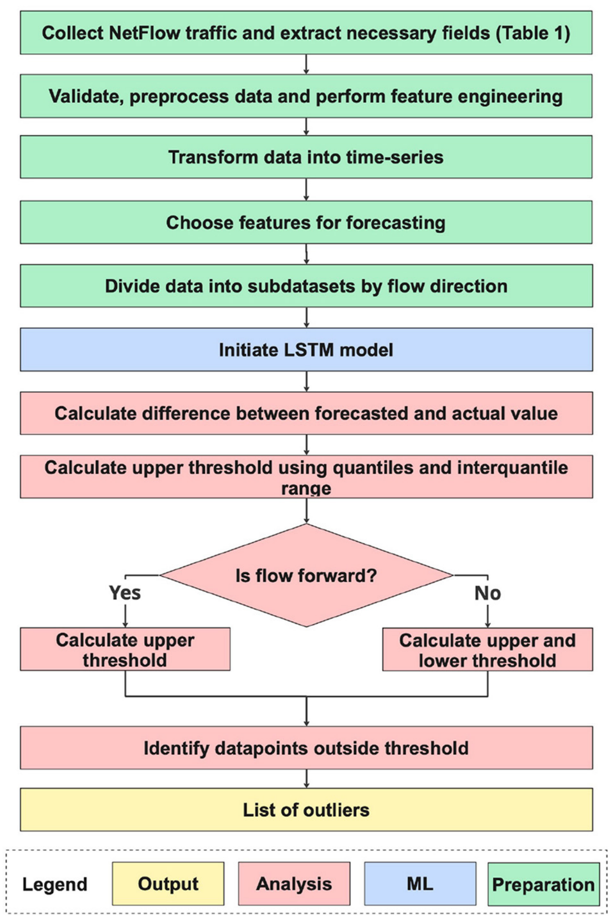

Collect NetFlow traffic and extract necessary fields (Table 1)
Validate, preprocess data and perform feature engineering
Transform data into time-series
Choose features for forecasting
Divide data into subdatasets by flow direction
Initiate LSTM model
Calculate difference between forecasted and actual value
Calculate upper threshold using quantiles and interquantile range
Is flow forward?
Yes
No
Calculate upper threshold
Calculate upper and lower threshold
Identify datapoints outside threshold
List of outliers
Legend
Output
Analysis
ML
Preparation

FIGURE 4
Forecasting steps with LSTM for anomaly detection in Netflow data.

TABLE 4 LSTM performance evaluation.

Performance metric	Forward flow	Backward flow
MSE	2.95	0.72
MAPE	2.9E+14	1.36E+14
MAE	0.99	0.47
R^2	0.29	0.32
Estimation of true outliers in Bytes (%)	95.32	59.58
Estimation of true outliers in Packets (%)	93.51	30.48
Estimation of true outliers for Both (%)	96.42	55.57

actually corresponds to improved anomaly detection capability, demonstrating that standard accuracy metrics must be interpreted in the context of the target task.

Both proposed NetFlow anomaly detection methods return the lists of identified outliers. The lists are transformed back to the original NetFlow traffic sample to include values the models did not use for analysis, such as IP source or destination addresses. Making it possible to use ML results with other cybersecurity methods, to have the results analyzed by security specialists, and to assess the output reliability.

The output reliability assessment requires to collect NetFlow data in 10 min time-window which matches with the time-window used for aggregation in clustering NetFlow dataset. The models return a list of outliers for further analysis and save other data records in a temporary list for a previously defined time for a potential assessment in the models' disagreement case (Figure 5).

The outlier identification reliability is determined using the equations provided in Section 2.3, and the probabilities of true answer are obtained from the ML models. In case of an agreement between both models, the Equation 1 is used. If only the clustering model detected an anomaly, then the Equation 2 is used, but otherwise the Equation 2 is mirrored with swapped arguments. The estimated values from the clustering $P_A = 0.83$ model, and the LSTM model: forward $P_B = 0.96$, backward $P_B = 0.56$, are used to calculate the reliability. The result is shown in Table 5 where A is a clustering model and B is an LSTM model, and 1 refers to an identified anomaly.

Output reliability, or the probability of the identified outlier being an actual outlier, highly depends on the previous model performance. As shown in the forward NetFlow case, the output of the LSTM model is more significant than the clustering model because of the higher probability, but in the backward flow, the outcome is opposite. However, when both models agree, there is a high possibility that the identification is correct.

4 Discussion

The design science research methodology was used to create the main artifacts of this research, allowing for the improvement of findings during each design cycle. As a result, four main artifacts of this research were created:

- NetFlow anomaly detection set of features for the clustering algorithm (SoF1). It can be used as a complete set of features or only include the necessary features. Furthermore, the SoF1 can be used with other clustering algorithms or data analysis methods, such as correlation analysis.
- NetFlow anomaly detection set of features for time-series forecasting (SoF2). It allows us to use a small portion of data to forecast NetFlow values one or more steps ahead using a specified number of previous records. This enables observation of sudden changes in the NetFlow traffic that are different from its baseline used in a training phase.
- The defined NetFlow anomaly detection methods. These methods enable validation and improvement of the SoF1 and SoF2. Additionally, proving that the presented SoF can be used in an ML task.
- ML models output combination method and result reliability assessment method. The methods show how the probability of correct identification can be assessed using two ML models trained on the same dataset with different preprocessing steps.

Both SoF are the main contributions of this research. Therefore, to emphasize its novelty, a comparison with the existing datasets used in network anomaly detection is performed (Table 6). The

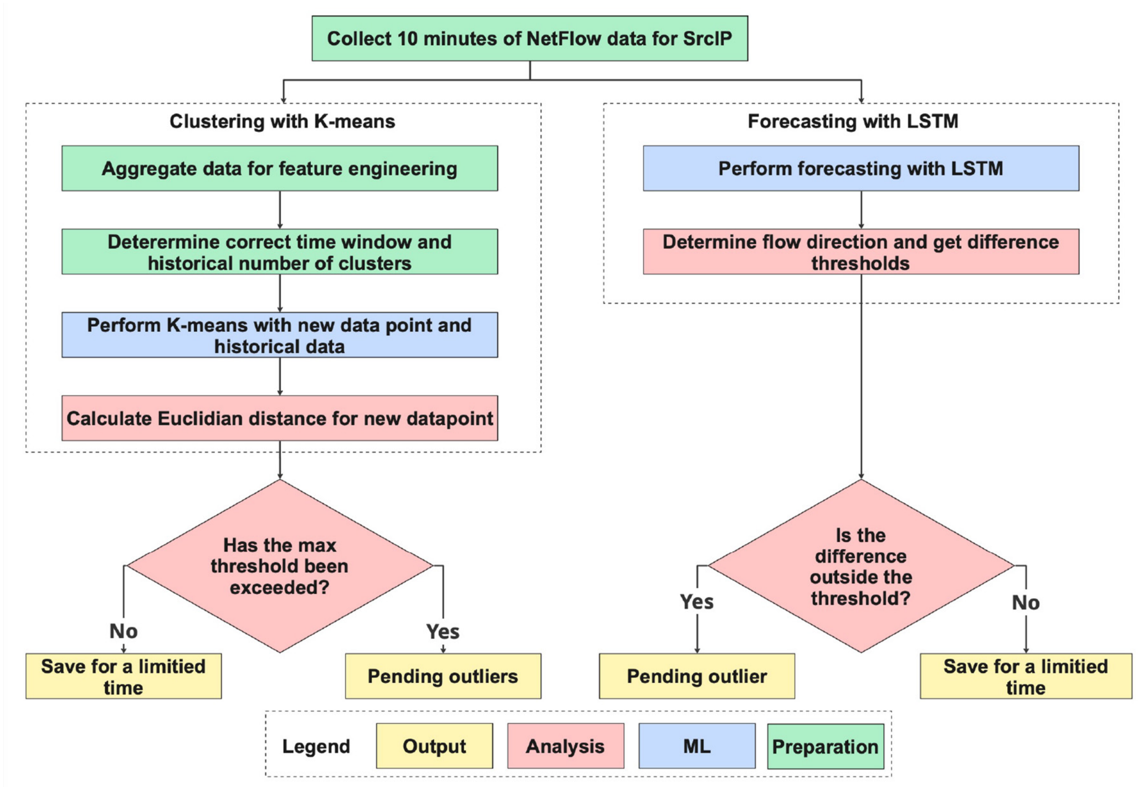

Collect 10 minutes of NetFlow data for SrcIP
Clustering with K-means
Aggregate data for feature engineering
Deterermine correct time window and historical number of clusters
Perform K-means with new data point and historical data
Calculate Euclidian distance for new datapoint
Has the max threshold been exceeded?
No
Yes
Save for a limitied time
Pending outliers
Forecasting with LSTM
Perform forecasting with LSTM
Determine flow direction and get difference thresholds
Is the difference outside the threshold?
Yes
No
Pending outlier
Save for a limitied time
Legend
Output
Analysis
ML
Preparation

FIGURE 5
Outlier detection with clustering and forecasting ML models.

TABLE 5 ML model output reliability assessment.

Outlier identification result	Forward flow	Backward flow
$P(A=1,B=1)$ or $P(A=0,B=0)$	0.99	0.86
$P(A=0,B=1)$	0.83	0.21
$P(A=1,B=0)$	0.17	0.79

TABLE 6 SoF comparison with other studies.

Set of features	Features	Temporal features	Aggregation	Testbed
SoF1	24	Yes	Yes	No
SoF2	14	Yes	No	No
Moustafa and Slay (2015)	49	Yes	No	Yes
Sharafaldin et al. (2018)	75	Yes	Yes	Yes
Sarhan et al. (2021)	12	Yes	No	Yes
Komisarek et al. (2023)	12	Yes	Yes	No
Fraihat et al. (2023)	7	No	Yes	Yes
Minkevics and Kampars (2021)	39	No	Yes	No

comparison considers the number of features, presence of temporal features, use of aggregation, labeled data, and testbed origin. These aspects directly influence a dataset's suitability for ML. The number of features reflects its descriptive richness and the possible presence of unnecessary features. Temporal features capture time-dependent behaviors essential for detecting dynamic anomalies. Aggregation indicates statistical analysis of data, and the testbed environment reflects whether the NetFlow was collected in a constrained environment or real traffic. Overall, the proposed datasets stand out for their inclusion of temporal and aggregated features, as well as their non-testbed origin, offering a practical foundation for anomaly detection research.

Evaluation confirmed that the proposed ML approach can detect anomalies in NetFlow data using the features presented in this research. However, there are several possibilities for improvement. Fine-tuning of the ML models can improve the given results, although it can also lead to overfitted models that are unable to work with new data. Analysis of NetFlow features by excluding certain features and comparing results can potentially reduce the SoF and, consequently, the dataset, thereby improving the speed of the ML models. Furthermore, current NetFlow data included only anomalies reflecting high-load conditions. Expanding the variety of anomaly classes in future iterations would further enhance the data robustness and applicability for evaluating anomaly detection methods and proposed features.

The reliability score not only indicates the probability of the correct answer but also serves as a support element for the decision selection mechanism. Despite the argument in various research on whether a human is a solution or a threat in the cybersecurity process (Zimmermann and Renaud, 2019), we argue that a human is a valuable asset during cybersecurity operations, which, unlike artificial intelligence, is able to adapt to new events. Reliability score aligned with predefined score categories can trigger an action response mechanism by a cybersecurity system. For example, a high reliability score leads to autonomous action by a system, such as blacklisting a suspicious IP address before further analysis by notified specialists. However, medium scores and low scores trigger specialist notification with different urgency levels.

Future work includes further improvement of the provided methods, supplemented with Human-in-the-Loop methods, enabling incremental learning of ML models and real NetFlow traffic labeling for the training phase. Furthermore, even though ensemble learning was considered during this research, it was not implemented in this study. The outputs of ML models can be combined to gain additional identification results by implementing ensemble learning supplemented with the reliability score. In addition, it is planned to increase the number of ML models, which would change the equations used in this research but provide more reliable assessment results. The artifacts presented in this study are validated using data obtained from a higher education institution. Although the methods and the SoF are universal and usable in other use cases, scalability may pose challenges when extending the current preprocessing and aggregation methods to significantly higher data rates and flow diversity.

Data availability statement

The datasets presented in this article are not readily available because data includes personal data. Requests to access the datasets should be directed to evita.roponena@rtu.lv.

Author contributions

ER: Visualization, Data curation, Methodology, Writing – review & editing, Conceptualization, Writing – original draft. IP: Methodology, Writing – review & editing. JG: Methodology, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

References

Barai, A., and Dey, L. (2017). Outlier detection and removal algorithm in k-means and hierarchical clustering. *World J. Comput. Appl. Technol.* 5, 24–29. doi: 10.13189/wjcat.2017.050202

CERT.LV (2025). *2025. gada 1. Ceturksnis Latvijas Kibertelpā*. Technical report, CERT.LV (Riga). Available online at: https://cert.lv/en/

Chiou, T. W., Tsai, S. C., and Lin, Y. B. (2014). Network security management with traffic pattern clustering. *Soft Comput.* 18, 1757–1770. doi: 10.1007/s00500-013-1218-0

Chollet, F (2015). *Keras*. Available online at: https://keras.io

Cisco Systems (2011). *Netflow Version 9 Flow-Record Format* (San Jose, CA).

Dias, L., Valente, S., and Correia, M. (2020). "Go with the flow: clustering dynamically-defined netflow features for network intrusion detection with dynids," in *2020 IEEE 19th International Symposium on Network Computing and Applications, NCA 2020* (Piscataway, NJ). doi: 10.1109/NCA51143.2020.9306732

Doshi-Velez, F., and Kim, B. (2017). *Towards a Rigorous Science of Interpretable Machine Learning* (Ithaca, NY).

Fragoso, T. M., Bertoli, W., and Louzada, F. (2018). Bayesian model averaging: a systematic review and conceptual classification. *Int. Stat. Rev.* 86, 1–28. doi: 10.1111/insr.12243

Fraihat, S., Makhadmeh, S., Awad, M., Al-Betar, M. A., and Al-Redhaei, A. (2023). Intrusion detection system for large-scale iot netflow networks using machine learning with modified arithmetic optimization algorithm. *Internet Things* 22:100819. doi: 10.1016/j.iot.2023.100819

Gu, G., Perdisci, R., Zhang, J., and Lee, W. (2008). "Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th USENIX Security Symposium* (Berkeley, CA), 139–154.

Han, J., Kamber, M., and Pei, J. (2012). *2 - Getting to Know Your Data*, 3rd Edn. Burlington, MA: Morgan Kaufmann, 39–82. doi: 10.1016/B978-0-12-381479-1.00002-2

Hinne, M., Gronau, Q. F., van den Bergh, D., and Wagenmakers, E.-J. (2020). A conceptual introduction to bayesian model averaging. *Adv. Methods Pract Psychol. Sci.* 3, 200–215. doi: 10.1177/2515245919898657

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognit. Lett.* 31, 651–666. doi: 10.1016/j.patrec.2009.09.011

Jun, S. (2016). Frequentist and bayesian learning approaches to artificial intelligence. *Int. J. Fuzzy Logic Intell. Syst.* 16, 111–118. doi: 10.5391/IJFIS.2016.16.2.111

Komisarek, M., Pawlicki, M., Simič, T., Kavčnik, D., Kozik, R., Choraś, M., et al. (2023). "Modern netflow network dataset with labeled attacks and detection methods," in *ACM International Conference Proceeding Series* (New York, NY: Association for Computing Machinery). doi: 10.1145/3600160.3605094

Minkevics, V., and Kampars, J. (2021). "Artificial intelligence and big data driven is security management solution with applications in higher education organizations," in *Proceedings of the 2021 17th International Conference on Network and Service Management: Smart Management for Future Networks and Services, CNSM 2021* (Izmir: IEEE), 340–344. doi: 10.23919/CNSM52442.2021.9615575

Mohammed, A., and Kora, R. (2023). A comprehensive review on ensemble deep learning: opportunities and challenges. *J. King Saud Univ. Comput. Inf. Sci.* 35, 757–774. doi: 10.1016/j.jksuci.2023.01.014

Moustafa, N., and Slay, J. (2015). "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)* (Canberra, ACT: IEEE), 1–6. doi: 10.1109/MilCIS.2015.7348942

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Comput. Secur.* 86, 147–167. doi: 10.1016/j.cose.2019.06.005

Roponena, E., Kampars, J., Gailitis, A., and Strods, J. (2021). "A literature review of machine learning techniques for cybersecurity in data centers," in *ITMS 2021 - 2021 62nd International Scientific Conference on Information Technology and Management Science of Riga Technical University, Proceedings* (Riga: IEEE). doi: 10.1109/ITMS52826.2021.9615321

Roponena, E., and Polaka, I. (2022). "Classifier selection for an ensemble of network traffic analysis machine learning models," *2022 63rd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)* (Riga: IEEE), 1–6. doi: 10.1109/ITMS56974.2022.9937116

Roponena, E., Polaka, I., and Grabis, J. (2024). "Netflow anomaly detection dataset creation for traffic analysis," in *2024 IEEE 65th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)* (Riga: IEEE), 1–6. doi: 10.1109/ITMS64072.2024.10741602

Rossum, G. V., and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2021). "Netflow datasets for machine learning-based network intrusion detection systems," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, volume 371 LNICST* (Cham: Springer Science and Business Media Deutschland GmbH), 117–135. doi: 10.1007/978-3-030-72802-1_9

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy* (Setubal). doi: 10.5220/0006639801080116

The Apache Software Foundation (2024). *Sparkr: R Front End for 'Apache Spark'*. Wilmington, DE.

Theodoridis, S. (2015). *Chapter 2 - Probability and Stochastic Processes*. Cambridge, MA: Academic Press, 9–51. doi: 10.1016/B978-0-12-801522-3.00002-1

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nat. Methods* 17, 261–272. doi: 10.1038/s41592-019-0686-2

vom Brocke, J., Hevner, A., Maedche, A. (2020). *Introduction to Design Science Research*. Cham: Springer International Publishing, 1–13. doi: 10.1007/978-3-030-46781-4_1

Zimmermann, V., and Renaud, K. (2019). Moving from a 'human-as-problem' to a 'human-as-solution' cybersecurity mindset. *Int. J. Hum.-Comput. Stud.* 131, 169–187. doi: 10.1016/j.ijhcs.2019.05.005