Intelligent data monitoring anomaly detection system based on statistical and machine learning approaches[⋆]

Yurii Klots[1,*,†], Vira Titova[1,†], Nataliia Petliak[1,†], Dmytro Tymoshchuk[2,†] and Nataliya Zagorodna[2,†]

[1] *Khmelnytskyi National University, 11 Instytuts'ka str., 29016 Khmelnytskyi, Ukraine*
[2] *Ternopil Ivan Puluj National Technical University, 56 Ruska str., 46001 Ternopil, Ukraine*

Abstract
One of the significant outcomes of society's digitalization is the rapid expansion of network services. This trend has heightened the demand for reliable and secure information systems administration. Ensuring system reliability and security requires regular equipment status checks, optimal software performance, and robust data protection measures—tasks typically managed through system administration and cybersecurity practices. Data analysis methods are crucial, as they process service statistics from monitoring systems to identify abnormal metric values. Anomaly detection plays a key role by pinpointing patterns or data points that deviate from normal behaviour, allowing for the early detection of issues such as fraud, security breaches, or equipment failures. This paper explores the primary performance indicators for servers, compares various computing system monitoring solutions, and configures the Zabbix monitoring system. Additionally, it considers the main types of anomalies and the methods used to detect them. An anomaly detection system was developed using an autoregressive integrated moving average (ARIMA) model, complemented by a neural network utilizing long short-term memory (LSTM) techniques.

Keywords
Anomalies, monitoring systems, time series analysis, forecasting, ARIMA model, LSTM, machine learning, cybersecurity, information security, abnormalities, abnormality detection, cyberattacks, intrusion detection, network

1. Introduction

One of the key manifestations of the digitalization of society is the extensive development of network services. Consequently, there is an increasing need for information systems administration tasks to ensure system reliability and security. This involves promptly checking equipment status, maintaining optimal software performance, and ensuring data security. These challenges are addressed within the domains of computer security and system administration. Data analysis methods play a crucial role in tackling these issues. By processing service data from statistics provided by monitoring systems, these tools can verify various metrics of the information system to detect anomalous values.

Anomaly detection identifies unusual patterns or data emissions that deviate from expected behaviour, facilitating the early identification of fraud, security threats, or equipment failures.

This work aims to develop a system for detecting anomalies in server performance indicators. To accomplish this, it is essential to examine the main types of potential anomalies, explore established methods for their detection, configure virtual server monitoring, build time series forecasting models, and create an anomaly detection program utilizing the implemented models.

[⋆] *CSDP'2025: Cyber Security and Data Protection, July 31, 2025, Lviv, Ukraine*
[*] Corresponding author.
[†] These authors contributed equally.

klots@khmnu.edu.ua (Y. Klots); titovav@khmnu.edu.ua (V. Titova); npetlyak@khmnu.edu.ua (N. Petliak); dmytro.tymoshchuk@gmail.com (D. Tymoshchuk); Zagorodna.n@gmail.com (N. Zagorodna)
0000-0002-3914-0989 (Y. Klots); 0000-0001-8668-4834 (V. Titova); 0000-0001-5971-4428 (N. Petliak); 0000-0003-0246-2236 (D. Tymoshchuk); 0000-0002-1808-835X (N. Zagorodna)

2. Analysis of existing network resource monitoring systems

The primary method for automating the management of such data involves monitoring systems like Zabbix, Nagios, and others. A comprehensive definition of monitoring encompasses the oversight of computer networks. This entails ongoing surveillance within the network, including traffic analysis and diagnostics, and assessing the load on network devices and connections. The goal is to identify slow or malfunctioning systems and inform network administrators of any failures or issues using various notification tools.

Timely detection of these failures facilitates a thorough evaluation of the system's performance by analyzing its components' and monitored entities' functioning and efficiency. A monitoring object refers to any device or service that is consistently overseen to assess its condition, analyze its processes, and identify and anticipate abnormal situations.

Server monitoring involves the collection and analysis of data regarding the current performance of computer systems. When evaluating and analyzing server performance, adopting a systems approach that considers the interdependencies of hardware components is crucial. Server load analysis entails gathering and processing statistics related to key components, including the processor, memory, disk, and network interface.

The timely detection and resolution of malfunctions are essential for the efficient operation of any computer network. Identifying failures is a complex challenge that necessitates a thorough analysis of the interaction between software and hardware components within the system, which is addressed through monitoring systems.

A monitoring system encompasses a set of technical tools designed to oversee and collect information continuously within a local computer network. This is achieved through statistical data analysis to identify faulty or improperly functioning nodes and to alert responsible personnel. Modern platforms that facilitate the stable and real-time detection of anomalies within extensive systems can be categorized based on the characteristics illustrated in Figure 1 [1, 2].

In comparing contemporary network monitoring systems according to the outlined classification, parameters such as the method of data collection and the application areas will be considered. All systems, characterized by the nature of the data analyzed, primarily employ highly accurate statistical methods for monitoring data analysis. The paper demonstrates the characteristics of network monitoring systems and their comparative analysis.

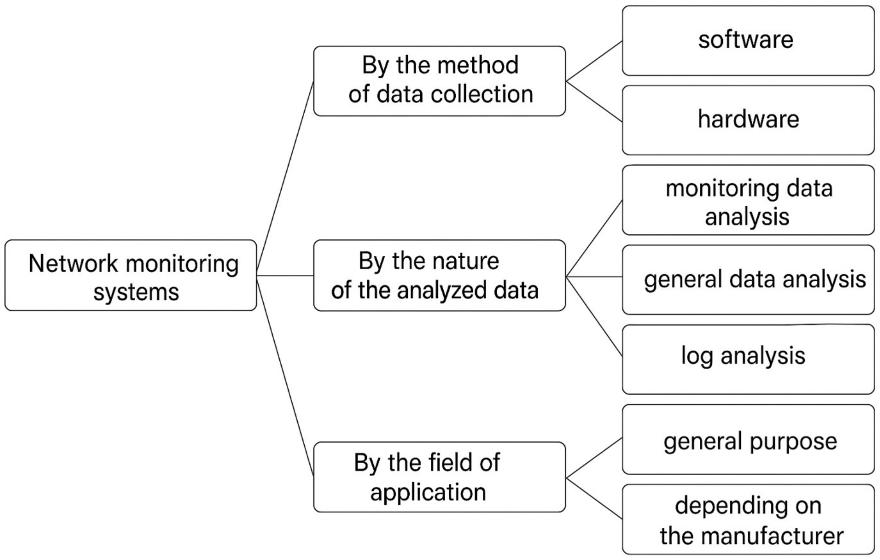

Network monitoring systems
By the method of data collection
software
hardware
By the nature of the analyzed data
monitoring data analysis
general data analysis
log analysis
By the field of application
general purpose
depending on the manufacturer

Figure 1: Classification of network monitoring systems

Let's emphasize the key parameters necessary for a point assessment of the described systems, where the presence of a function is awarded 1 point and its absence receives 0 points:

- Trend forecasting—the existence of algorithms designed to predict network statistics.
- Plugins—the availability of plugins that enhance functionality.
- Notification—the capability to detect and alert on anomalies.
- Complete control—the presence of a web interface that facilitates access to all system services.
- Distributed monitoring—the ability to utilize multiple servers.

By synthesizing the above information about the analyzed network monitoring systems, we can comprehensively assess their parameters in Table 1.

Table 1
Evaluation of the characteristics of network monitoring systems

Evaluation characteristic	Cacti	Icinga	Nagios	PRTG	Zabbix
Trend forecasting	1	0	0	1	1
Plugins	1	1	1	1	1
Notification	1	1	1	1	1
Complete control	1	1	0	0	1
Distributed monitoring	1	1	1	1	1
Final assessment	5	4	3	4	5

Based on the assessment results, Cacti and Zabbix are the most multifunctional systems, encompassing the complete range of analyzed network monitoring tools. However, it is important to highlight Zabbix's advantages, particularly its ability to manage a large number of network nodes and its support for personalized configurations of scanning systems, in contrast to the more templated approach of Cacti. Consequently, we have selected Zabbix as our preferred option.

3. Overview of Anomaly Detection Methods

An anomaly generally refers to a deviation from expected behaviour. In the context of system and network monitoring data, anomalies are viewed as changes in the behaviour of the system and its indicators over time. Anomalies can be classified in various ways, with one standard classification identifying three types: point, contextual, and collective [3, 4].

A point anomaly refers to a singular measurement that stands out as an outlier, differing significantly from the other measurements. These anomalies can be identified by evaluating the mathematical expectation and standard deviation. Any values that exceed a specified threshold based on the standard deviation are deemed abnormal.

On the other hand, a contextual anomaly occurs when a measurement is considered abnormal only within a specific context. For instance, if a server is conducting a scheduled backup at night, an increase in processor utilization during that timeframe may be regarded as usual. In contrast, the same increase during the daytime would be perceived as abnormal. In this scenario, it is essential to assess not only the behaviour of the measurement itself but also the context, particularly the range of values that defines when a measurement is considered abnormal.

In the context of collective anomalies, a sequence of related data instances (such as a time series segment) is deemed anomalous when considering the entire dataset. Within collective anomalies, we can identify two types: shift anomalies, which alter the mathematical expectation in a time series, and distribution change anomalies, which are characterized by changes in standard deviation.

As highlighted in [4], anomaly detection encompasses a range of methods and systems aimed at identifying unusual behaviour and states within systems and their observable indicators. The techniques for detecting anomalies can be broadly classified into statistical methods and machine learning approaches.

Machine learning methods are categorized into supervised (such as Decision Trees, SVM, and LSTM) and unsupervised (including K-means, hierarchical clustering, and DBSCAN). Supervised methods require a training dataset to build a model that distinguishes normal behaviour from anomalies. Machine learning methods allow for the detection of more complex, hidden, and non-obvious anomalies [5–7], which often go unnoticed when using traditional statistical approaches. Due to the ability to build flexible models that are trained on real data [8, 9], these methods can detect complex dependencies, relationships between parameters, and non-standard behavioral patterns. This makes them useful for solving a wide range of tasks, from detecting cyber threats and fraudulent activities to diagnosing technical malfunctions in complex information systems. Machine learning offers a number of methods for detecting anomalies, such as: LSTM, SVM, and K-means.

LSTM (long short-term memory) is a specific architecture of recurrent neural networks [10–12]. It predicts the value of the next time interval based on an input time series by maintaining a state. The model is trained on normal time series data, which allows it to identify anomalies. LSTM networks feature specialized compartments equipped with three gates: an input gate, a forget gate, and an output gate. These gates facilitate information flow control within the network, enabling the LSTM to determine which information to retain, discard, or transmit to the next step. The advantages of using LSTM networks for anomaly detection are twofold [13]. First, the training data does not require labeling, eliminating the need to provide the model with specific examples of anomalies. Second, the trained model can detect previously unseen anomalies. However, LSTMs also have several drawbacks. Their complex architecture, which includes multiple gates and memory cells, makes them computationally intensive, leading to increased training times and higher memory consumption.

Support Vector Machine (SVM) is a classification technique that transforms original vectors representing values into a higher-dimensional space. In this space, it identifies a separating hyperplane that maximizes the margin between normal and abnormal values [14, 15]. The advantages of SVM include the ability to derive a classification function with minimal error, the capability of applying a linear classifier to nonlinear data, and the effectiveness in handling heterogeneous datasets. However, SVM also has its drawbacks: the decision function relies heavily on pre-set parameters and is sensitive to noise present in the training set. To address these limitations, the integration of fuzzy set theory has been proposed as one potential solution, though this approach increases the algorithm's computational complexity.

The K-means method is a clustering algorithm associated with unsupervised machine learning techniques [16–18]. Its primary objective is to group data into clusters centered around predefined key points, the number of which must be specified beforehand. Elements that do not belong to any cluster are regarded as anomalous. When applied to anomaly detection, K-means presents several advantages, such as its simplicity, efficiency, and scalability, making it particularly suitable for large datasets. Additionally, since it operates unsupervised, it does not require pre-labeled data and allows for customization regarding the number of clusters. K-means can effectively identify deviations from the norm by pinpointing data points significantly distant from established cluster centroids. However, there are drawbacks: the algorithm necessitates prior knowledge to determine the number of cluster centers (the value of K), struggles with outliers and noisy data due to potential skewing of centroids, and may be inefficient with large datasets as it requires considerable computational time.

Statistical anomaly detection methods include the Holt-Winters model and the autoregressive integrated moving average (ARIMA). These statistical methods rely on the premise that typical time series data are generated by a specific statistical process, with values that deviate from this process being classified as anomalies [19–21]. A crucial aspect of these methods involves analyzing the parameters of the statistical process based on a training time series and evaluating how well a test time series aligns with the derived parameters.

When employing statistical analysis methods, it is important to recognize that time series can be decomposed into three components: the trend, which indicates the overall direction of the series in terms of increasing or decreasing values; seasonality, which represents periodic fluctuations linked to factors such as the day of the week or month; and the random component, which consists of the residuals remaining after the trend and seasonality have been accounted for, and is typically where anomalies are sought. The primary function of the ARIMA and Holt-Winters models is to forecast future values in a time series.

ARIMA models effectively analyze time series data with a pronounced trend [22–24]. ARIMA and neural networks, such as LSTM, have emerged as prominent techniques for detecting anomalies within time series data. ARIMA's key strengths lie in its interpretability and accuracy when applied to stationary datasets. In contrast, neural networks excel at modeling complex nonlinear patterns. However, a downside of ARIMA is that over-differencing can strip away meaningful patterns from the data, resulting in poor forecasting outcomes. This dependence on manual preprocessing renders ARIMA less flexible for datasets exhibiting complex non-stationary behaviour. Furthermore, ARIMA typically assumes linear relationships between past and future values.

On the other hand, the Holt-Winters model is adept at handling trends and seasonality in time series data [25]. Particularly when integrated with Brutlag's algorithm, the Holt-Winters model provides several advantages for anomaly detection, especially in seasonal contexts. It effectively captures trends and seasonal patterns, aiding in identifying deviations from expected behaviour. Its adaptability, ease of implementation, capacity to manage changing trends, and low computational complexity make it a compelling choice for various anomaly detection scenarios. Nevertheless, the Holt-Winters model does have some notable limitations. It requires initial values for level, trend, and seasonal components, which can be challenging to determine accurately. Moreover, its performance may suffer if the data exhibits non-stationary behaviour or inconsistent seasonality. Additionally, the model's reliance on smoothing constants (alpha, beta, and gamma) can complicate the selection of optimal values, and it may be sensitive to outliers or extreme values present in the data.

Recent studies demonstrate the effectiveness of combining natural language processing and machine learning techniques for real-time ransomware detection using eBPF technology, which enhances anomaly detection in cybersecurity contexts [26]. Furthermore, advances in neural network architectures have proven successful in identifying deepfake modifications in biometric images, underlining the growing importance of sophisticated machine learning models for anomaly and fraud detection [27–30]. Comparative analyses of deep learning models for speaker verification also highlight the potential for high-accuracy anomaly detection in audio data streams, expanding the applicability of AI-based monitoring systems across diverse data types [31].

Considering the advantages and disadvantages, forecasting algorithms based on ARIMA and LSTM neural network models were implemented for anomaly detection. This involved calculating sequence elements across subsequent time intervals based on the training sequence. An outlier anomaly was recorded if the predicted values diverged from the input values by a specified threshold.

To prepare the input data, we set up a virtual server using the Zabbix monitoring system, which collects data on the server's performance, including processor load, RAM capacity, network interface speed, hard drive usage, and more.

4. Data monitoring anomalies detection systems

The API offered by the Zabbix system enables users to send requests to configure the server-side aspects of the system and retrieve Zabbix configuration data. Additionally, the API allows access to historical data for specific performance indicators. Requests are made through procedures defined in the JSON-RPC protocol, with communication between clients and the API using the JSON format. To interact with this API in Python, one can utilize the pyzabbix library.

Using the Zabbix monitoring system API, the program collects system indicators for a given data element key at specified intervals and saves the obtained values in a CSV file. Let us look at the server load indicator as a percentage (Figure 2) as an example.

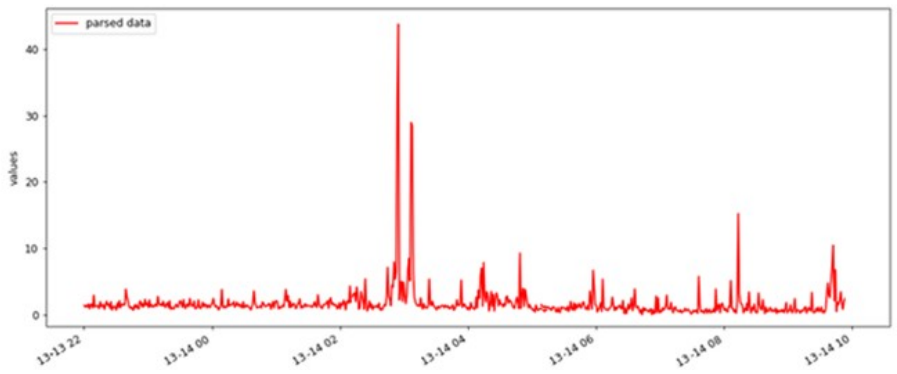


Figure 2: Active server load in percent

After that, a Pandas.Dataframe table is created from the csv-file, in which the series are divided into two time periods: the training and forecast parts. In the training part, the coefficients for ARIMA are selected, which will be used for forecasting.

The auto_arima function selects the parameters p, d, and q. With its help, the parameters that will give the lowest value of the Akaike information criterion are searched for by the enumeration method.

After training the model, forecasting is performed in batches of 10 values specified in the forecast_length variable. Then the predicted values are added to the model so that the following predictions are calculated based on them: the model is updated.

For example, the prediction for the ARIMA(4, 1, 4) model is shown in Figure 3. In this graph, the mean square error was 3.294.

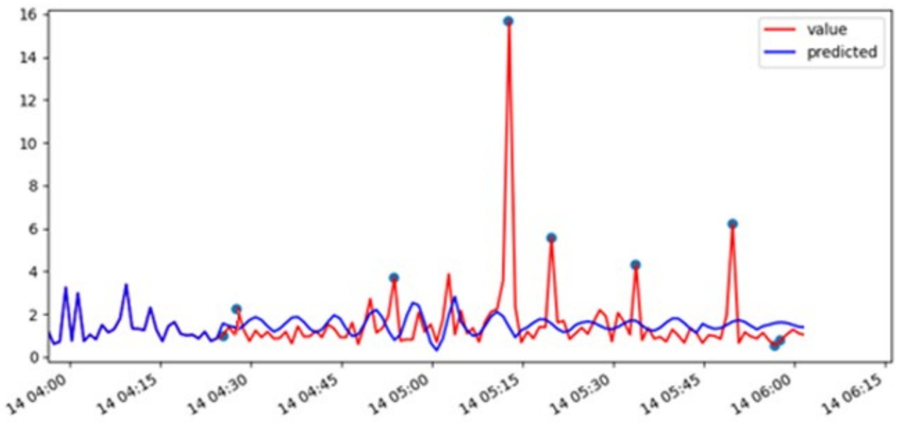


Figure 3: Model forecast

To determine the time points with anomalies, the forecast error is calculated for each value:

test_df['error'] = test_df['value'] - test_df['predicted']

For the error, the moving average and the moving standard deviation are calculated on a given number of previous values, for example, 12:

df['err_meanval'] = df['error'].rolling(window=12).mean()

df['err_deviation'] = df['error'].rolling(window=12).std()

Based on the moving average and the standard deviation, a confidence interval of two standard deviations is constructed:

df['-lim'] = df['err_meanval'] - (2 * df['err_deviation'])

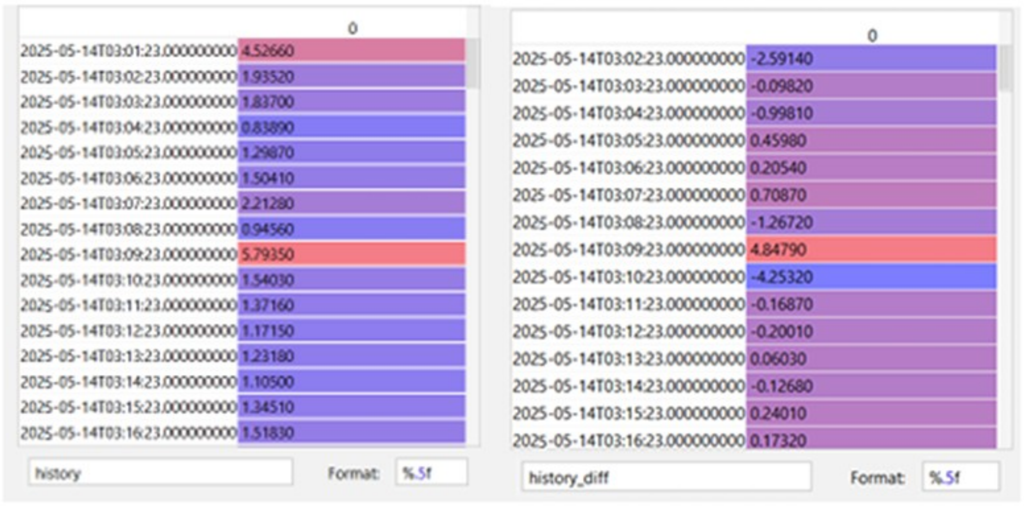
df['+lim'] = df['err_meanval'] + (2 * df['err_deviation'])

If the error value goes beyond the confidence interval, then at the point in time at which this occurred, an anomaly is noted:

df['anomaly_points'] = np.where(((df['error'] > df['+lim']) | (df['error'] < df['-lim'])), df['value'], np.nan)

Similar to the ARIMA model, the software implementation of the LSTM network works with data from a CSV file, which contains data collected from the Zabbix system.

Before starting the neural network training, the time series is processed. At the initial stage, a Pandas.Series array is created from the values. Then, the resulting series is differentiated, which in most cases allows achieving stationarity (Figure 4).



	0
2025-05-14T03:01:23.000000000	4.52660
2025-05-14T03:02:23.000000000	1.93520
2025-05-14T03:03:23.000000000	1.83700
2025-05-14T03:04:23.000000000	0.83890
2025-05-14T03:05:23.000000000	1.29870
2025-05-14T03:06:23.000000000	1.50410
2025-05-14T03:07:23.000000000	2.21280
2025-05-14T03:08:23.000000000	0.94560
2025-05-14T03:09:23.000000000	5.79350
2025-05-14T03:10:23.000000000	1.54030
2025-05-14T03:11:23.000000000	1.37160
2025-05-14T03:12:23.000000000	1.17150
2025-05-14T03:13:23.000000000	1.23180
2025-05-14T03:14:23.000000000	1.10500
2025-05-14T03:15:23.000000000	1.34510
2025-05-14T03:16:23.000000000	1.51830

history Format: %.5f

	0
2025-05-14T03:02:23.000000000	-2.59140
2025-05-14T03:03:23.000000000	-0.09820
2025-05-14T03:04:23.000000000	-0.99810
2025-05-14T03:05:23.000000000	0.45980
2025-05-14T03:06:23.000000000	0.20540
2025-05-14T03:07:23.000000000	0.70870
2025-05-14T03:08:23.000000000	-1.26720
2025-05-14T03:09:23.000000000	4.84790
2025-05-14T03:10:23.000000000	-4.25320
2025-05-14T03:11:23.000000000	-0.16870
2025-05-14T03:12:23.000000000	-0.20010
2025-05-14T03:13:23.000000000	0.06030
2025-05-14T03:14:23.000000000	-0.12680
2025-05-14T03:15:23.000000000	0.24010
2025-05-14T03:16:23.000000000	0.17320

history_diff Format: %.5f

Figure 4: Differentiation of time series

To make the data suitable for training, a matrix is created based on the array, where each row contains a fragment of the time series with a length determined by the variable time_steps. Each subsequent row in this matrix contains a shift of 1 value relative to the previous one.

Next, the values are normalized using the MinMaxScaler function of the scikit-learn package to be in the range [−1, 1]. Given a data set in the form of a shift matrix, the neural network is trained to predict the last column of the matrix based on the previous columns. Then the data is divided into training and test samples, where the test sample consists of values whose number is specified in the forecast_length variable, for example, 50. These are the last rows of the matrix in which the last column must be predicted. The Keras library was used to implement the LSTM neural network model.

The parameters were time_steps = 40, batch_size = 10, epochs = 50, neurons = 20. The parameters mean the following:

- time_steps is the length of the time interval by which the next value is predicted.
- batch_size is the size of the input sequence packet after which the weights are updated.

- epochs is the number of epochs of the neural network.
- neurons is the number of elements in the state vector, meaning the number of neurons in the expanded form of LSTM.

The network consists of an LSTM layer that saves the state for each subsequent row from the matrix. The next Dense layer is a neuron that combines 20 values (the output of the LSTM layer) into one, which is considered a forecast.

The mean square error is the loss function, and the Adam optimizer is selected to minimize it. Since the model saves the state from each row, the state must be explicitly reset after each pass through the entire matrix (the end of the epoch). Therefore, the model is trained in a cycle for one epoch. Since the input data form a sequence, we prohibit shuffling samples—matrix rows, by setting the shuffle=False parameter during training.

After training the model, it is used to obtain the last columns of the matrix on the test sample. The prediction is performed in parts with several rows equal to batch_size; then the network is trained with new values. After receiving all the values from the test sample, the inverse transformation is performed from normalized values to the 20th form, and from differentiated to differentiated-undifferentiated. Then the search for anomalies begins—those values that deviate most from the predicted ones. They are determined similarly to the ARIMA model using the moving average of the forecast error. The result of the model is shown in Figure 5. The root-mean-square error on the forecast interval is 4.810.

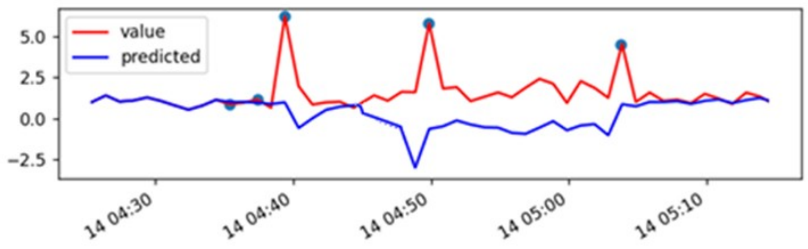


Figure 5: Anomaly Detection with LSTM

The LSTM model may demonstrate increasing deviation in forecasts when processing large input data. However, when using a smaller dataset (90 minutes of history for 20 forecasts), both models accurately capture abnormal values in the same areas. Given this observation, the ARIMA model is favoured for its stability and speed, requiring only 10 seconds of training compared to over 60 seconds for the LSTM.

Conclusions

The objective of developing an anomaly detection system is to manage and uphold the reliability and security of computing systems and networks. Primarily, identifying anomalous behaviour streamlines the analysis of server load, shortens the time required for server performance audits and incident investigations, and accelerates decision-making regarding the optimization of configurations.

The study accomplished several key tasks:

- Reviewed the key performance indicators for servers.
- Conducted an analysis of server monitoring systems.
- Examined the primary types of anomalies and methods for their detection.
- Configured server monitoring using the Zabbix system to gather initial data.
- Developed autoregressive integrated moving average (ARIMA) time series forecasting models and a long short-term memory (LSTM) neural network for anomaly detection.

- Created a program to identify anomalies within the data collected by the configured monitoring system.

Consequently, the following objectives were achieved: a system for detecting anomalies in monitoring data was established by applying machine learning and forecasting techniques.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] The Role of Network Monitoring and Analysis in Ensuring Optimal Network Performance, Int. Res. J. Mod. Eng. Technol. Sci. (2024). doi:10.56726/irjmets59269

[2] J. Alkenani, K.A. Nassar, Network Monitoring Measurements for Quality of Service: A Review, Iraqi J. Electrical Electron. Eng. 18(2) (2022) 33–42. doi:10.37917/ijeee.18.2.5

[3] M. H. Thwaini, Anomaly Detection in Network Traffic using Machine Learning for Early Threat Detection, Data Metadata 1 (2022) 34. doi:10.56294/dm202272

[4] J. Barnard, C. Stryker, What is Anomaly Detection? IBM. https://www.ibm.com/think/topics/anomaly-detection

[5] N. Petliak, Y. Klots, V. Titova, A.-B. Salem, Attack Detection System based on Network Traffic Analysis by Means of Fuzzy Inference, in: 1st Int. Workshop on Advanced Applied Information Technologie, 3899, 2024, 201–213.

[6] O. A. Alkhudaydi, M. Krichen, A. D. Alghamdi, A Deep Learning Methodology for Predicting Cybersecurity Attacks on the Internet of Things, Information 14.10 (2023) 550. doi:10.3390/info14100550

[7] D. Tymoshchuk, O. Yasniy, M. Mytnyk, N. Zagorodna, V. Tymoshchuk, Detection and Classification of DDoS Flooding Attacks by Machine Learning Methods, in: The 1st Int. Workshop on Bioinformatics and Applied Information Technologies, 3842, 2024, 184–195.

[8] M. Ramzan, et al., Distributed Denial of Service Attack Detection in Network Traffic using Deep Learning Algorithm, Sensors 23.20 (2023) 8642. doi:10.3390/s23208642

[9] B. Lypa, I. Horyn, N. Zagorodna, D. Tymoshchuk, T. Lechachenko, Comparison of Feature Extraction Tools for Network Traffic Data, in: 4th Int. Workshop on Information Technologies: Theoretical and Applied Problems, 3896, 2024, 3896, 1–11.

[10] L. Gunn, P. Smet, E. Arbon, M. D. McDonnell, Anomaly Detection in Satellite Communications Systems using LSTM Networks, in: 2018 Military Communications and Information Systems Conference (Milcis), IEEE, 2018. doi:10.1109/milcis.2018.8574109

[11] A. Duraj, P. S. Szczepaniak, A. Sadok, Detection of Anomalies in Data Streams using the LSTM-CNN Model, Sensors 25.5 (2025) 1610. doi:10.3390/s25051610

[12] N. Dash, et al., An Optimized LSTM-based Deep Learning Model for Anomaly Network Intrusion Detection, Sci. Rep. 15.1 (2025). doi:10.1038/s41598-025-85248-z

[13] Klots Y., Petliak N., Martsenko S., Tymoshchuk V., Bondarenko I. Machine Learning System for Detecting Malicious Traffic Generated by IoT Devices, in: 2nd Int. Workshop on Computer Information Technologies in Industry 4.0, 3742, 2024, 97–110.

[14] U. Yokkampon, S. Chumkamon, A. Mowshowitz, R. Fujisawa, E. Hayashi, Anomaly Detection using Support Vector Machines for Time Series Data, J. Robot., Netw. Artif. Life 8.1 (2021) 41. doi:10.2991/jrnal.k.210521.010

[15] M. Akpinar, M. F. Adak, G. Guvenc, SVM-based Anomaly Detection in Remote Working: Intelligent Software SmartRadar, Appl. Soft Comput. 109 (2021) 107457. doi:10.1016/j.asoc.2021.107457

[16] S. Gadal, et al., Machine Learning-based Anomaly Detection using k-Mean Array and Sequential Minimal Optimization, Electronics 11.14 (2022) 2158. doi:10.3390/electronics11142158

[17] R. Kumari, M. Sheetanshu, R. Singh, N. Jha, Singh, Anomaly Detection in Network Traffic using K-Mean Clustering, in: 2016 3rd Int. Conf. on Recent Advances in Information Technology (RAIT), IEEE, 2016. doi:10.1109/rait.2016.7507933

[18] A. Sarvani, B. Venugopal, N. Devarakonda, Anomaly detection using k-means approach and outliers detection technique, in: Advances in intelligent systems and computing, Springer Singapore, Singapore, 2018, 375–385. doi:10.1007/978-981-13-0589-4_35

[19] V. Titova, Y. Klots, V. Cheshun, N. Petliak, A.-B. Salem, Detection of Network Attacks in Cyber-Physical Systems using a Rule-based Logical Neural Network, in: The 1st Int. Workshop on Intelligent and CyberPhysical Systems (ICyberPhyS-2024), 3736, 2024, 255–268.

[20] A. Iqbal, R. Amin, Time Series Forecasting and Anomaly Detection using Deep Learning, Comput. & Chem. Eng. (2023) 108560. doi:10.1016/j.compchemeng.2023.108560

[21] Y. Klots, N. Petliak, V. Titova, Evaluation of the Efficiency of the System for Detecting Malicious Outgoing Traffic in Public Networks, in: 2023 13th Int. Conf. on Dependable Systems, Services and Technologies (DESSERT), IEEE, 2023. doi:10.1109/dessert61349.2023.10416502

[22] S. Xue, H. Chen, X. Zheng, Detection and Quantification of Anomalies in Communication Networks based on LSTM-ARIMA Combined Model, Int. J. Mach. Learn. Cybern. (2022). doi:10.1007/s13042-022-01586-8

[23] Q. Ai, H. Tian, H. Wang, Q. Lang, X. Huang, X. Jiang, Q. Jing, Comparative Analysis of ARIMA and LSTM Model-based Anomaly Detection for Unannotated Structural Health Monitoring Data in an Immersed Tunnel, Comput. Model. Eng. & Sci. (2023) 1–10. doi:10.32604/cmes.2023.045251

[24] W. Berriche, F. Sailhan. Predictive Anomaly Detection. In: Proc. of the 18th Int. Conf. on Information Assurance and Security (IAS 2022), 2022.

[25] S. Hansun, V. Charles, C. R. Indrati, Subanar, Revisiting the Holt-Winters' Additive Method for Better Forecasting, Int. J. Enterp. Inf. Syst. 15.2 (2019) 43–57. doi:10.4018/ijeis.2019040103

[26] D. Zhuravchak, V. Dudykevych, Real-Time Ransomware Detection by using eBPF and Natural Language Processing and Machine Learning, in: Advanced Information and Communication Technologies. Proc. of the 5th IEEE Int. Conf., 2023, 221–224.

[27] V. Dudykevych, et al., Detecting deepfake modifications of biometric images using neural networks, in: Cybersecurity Providing in Information and Telecommunication Systems, 3654 (2024) 391–397.

[28] V. Buhas, et al., Using Machine Learning Techniques to Increase the Effectiveness of Cybersecurity, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3188, no. 2 (2021) 273–281.

[29] V. Zhebka, et al., Methodology for Predicting Failures in a Smart Home based on Machine Learning Methods, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 322–332.

[30] M. Adamantis, V. Sokolov, P. Skladannyi, Evaluation of State-of-the-Art Machine Learning Smart Contract Vulnerability Detection Method, Advances in Computer Science for Engineering and Education VII, vol. 242 (2025) 53–65. doi:10.1007/978-3-031-84228-3_5

[31] V. Brydinskyi, et al., Comparison of Modern Deep Learning Models for Speaker Verification, Appl. Sci. 14(4) (2024) 1329-1–1329-12.