UDC 004.4.05(043.2)

A.O. Lavreniuk
R.O. Savelko
Y.O. Sharko
H.O. Chernovolyk
V.P. Maidaniuk

DETECTING ANOMALIES IN PROGRAM BEHAVIOR USING STATISTICAL METHODS

Vinnytsia National Technical University

Анотація

У тезах розглянуто статистичні методи виявлення аномалій у поведінці програм. Розглянуто три класи методів: статистичний контроль процесів (SPC), регресійні моделі та підходи на основі щільності (LOF). Проаналізовано їхні принципи, переваги, недоліки та сфери застосування.

Ключові слова: виявлення аномалій, виявлення викидів, поведінка програм, статистичні методи, кібербезпека, виявлення вторгнень, статистичний контроль процесів (SPC), CUSUM, EWMA, регресія часових рядів, локальний фактор аномальності (LOF).

Abstract

The theses analyzes statistical methods for anomaly detection in program behavior. It examines three classes of methods: Statistical Process Control (SPC), regression-based models, and density-based approaches (LOF). The principles, advantages, disadvantages, and application areas for each method are analyzed.

Keywords: Anomaly Detection, Outlier Detection, Program Behavior, Statistical Methods, Cybersecurity, Intrusion Detection, Statistical Process Control (SPC), CUSUM, EWMA, Time-Series Regression, Local Outlier Factor (LOF).

Introduction

In modern computing, ensuring software reliability and security requires proactive strategies. Central to this is anomaly detection, the process of identifying events or patterns that deviate significantly from a system's normal behavior. The relevance of this field is twofold. In cybersecurity, traditional signature-based tools fail against “zero-day” threats. Anomaly detection systems, however, build a model of normal behavior (like network traffic or user activity) and flag any significant deviation as a potential, previously unknown attack [1-2].

In system reliability, this approach provides an early warning system. Post-mortems of service disruptions, such as those at eBay, often reveal that key performance metrics (like CPU load or memory usage) showed unusual behavior before a failure became critical. Statistical methods are foundational to this field, offering advantages over more complex models by being computationally cheaper and, crucially, more interpretable, which is vital for analysts investigating an alert [3-4].

Fundamental Prerequisite: Feature Engineering

Before any statistical model can be applied, a critical step is feature engineering. “Program behavior” is an abstract concept; it only becomes measurable through specific, derived features. Raw data, such as individual CPU logs, network packets, or log messages, is often too noisy and granular to be useful. Feature engineering is the process of transforming this raw data into meaningful metrics, such as “the number of new IP addresses connecting per minute” or “the 5-minute average API error rate”. The quality of these features is often more important than the complexity of the statistical algorithm itself. It is also essential that the data used to train the “normal” model is clean; if anomalies are present in the training set, the model will incorrectly learn them as normal, severely reducing its effectiveness [4-6].

Statistical Process Control (SPC)

One of the most established families of statistical methods is Statistical Process Control (SPC), which originated in manufacturing to monitor if a process is stable or “in control.” While basic SPC charts (like

Shewhart charts) are good at detecting large, abrupt spikes in data , they are poorly suited for many program anomalies, such as slow memory leaks or low-level attacks, which manifest as small, persistent drifts. For this, more advanced charts like CUSUM (Cumulative Sum) and EWMA (Exponentially Weighted Moving Average) are used. Both methods incorporate past data, making them highly sensitive to small shifts. CUSUM accumulates the sum of all past deviations from a target, weighting them equally, while EWMA calculates a weighted average that gives exponentially decreasing weight to older data, making it more responsive to recent changes [7-8].

The primary advantage of these methods is their superior sensitivity to minor, sustained shifts in a process mean that other charts would miss, allowing for the early detection of issues. They provide objective, data-driven insights into process performance in real-time [8].

However, SPC methods are not without drawbacks. Their effectiveness is highly dependent on the correct setting of their statistical parameters. If control limits are not set appropriately, they can be prone to a high rate of false alarms, which leads to "alert fatigue" and unnecessary disruptions. Furthermore, SPC may be less effective for highly complex processes with multiple interacting variables and can be more complex to implement and interpret than simpler charts [9].

In practice, SPC is widely used for monitoring program behavior. It is applied in intrusion detection systems (IDS) to identify attacks like Denial of Service (DoS) or Remote-to-Local (R2L) by monitoring network traffic statistics. It is also highly effective for system reliability monitoring, such as tracking API latency, packet retransmission rates, or even the residuals from other predictive models to detect gradual performance degradation before it impacts users [10].

Regression-Based Models

Another class of methods uses statistical regression to predict normal behavior. It is important to first distinguish this from "regression testing," which is a software engineering quality assurance practice of re-running old tests to ensure new code changes have not broken existing functionality. Statistical regression, in this context, involves building a predictive model based on historical data that is assumed to be "normal." For program metrics that have time-based dependencies, time-series models like ARIMA (Autoregressive Integrated Moving Average) are often used to capture trends and seasonality [11].

This model is trained on past data to forecast future values. An anomaly is then identified when the actual observed value is significantly different from the predicted value. This difference between the observation and the forecast is known as the residual or prediction error. If a residual is statistically significant (e.g., exceeds a predefined threshold), the data point is flagged as an anomaly [12].

The main strength of this approach is its ability to model and account for complex temporal dependencies, patterns, and seasonalities that simpler methods cannot capture. This makes it highly effective for systems with natural, predictable cycles. Some regression techniques can also be designed to be "robust," meaning their model of "normal" is not overly skewed by a few anomalous data points [11].

These models also have significant limitations. Many, including ARIMA, rely on strong statistical assumptions, such as the data being stationary (its statistical properties don't change over time), which is not always true for chaotic program metrics. A major operational risk is "concept drift," where a slowly developing anomaly (like a gradual memory leak) is mistakenly absorbed by the model as part of the "new normal," effectively hiding the problem. The model's performance can also be sensitive to parameter settings and the underlying distribution of the data [13].

Regression models are frequently used in cybersecurity to forecast "normal" network traffic volumes or user activity, with large residuals signaling potential intrusions or DoS attacks. They are also applied in software engineering for defect prediction, where models estimate the number of expected faults based on code metrics or testing phase data [14].

Density-Based Methods (Local Outlier Factor)

A third approach, density-based methods, operates on the simple premise: normal data points tend to group together in dense clusters, while anomalies are isolated points in sparse regions. The Local Outlier Factor (LOF) is a premier algorithm in this category. It is an unsupervised method, meaning it does not require pre-labeled "normal" data for training. LOF's unique strength is its ability to find local anomalies–points that are outliers only in relation to their immediate neighborhood, even if they are not the most extreme points in the entire dataset [15].

LOF works by assigning an anomaly score to each data point by comparing its local density to the local

densities of its neighbors. It calculates this score based on a ratio: it compares the local reachability density (LRD) of a point to the average LRD of its “k” nearest neighbors (where 'k' is a user-defined parameter). A score of approximately 1 means the point is “normal” and shares a similar density with its neighbors. A score significantly greater than 1 indicates the point is in a much sparser region than its neighbors, marking it as an anomaly [16].

The primary advantage of LOF is this ability to identify local outliers that global methods, which average over the entire dataset, would miss. Because it is unsupervised, it is ideal for real-world scenarios where labeled anomaly data is rare or non-existent. It also makes no assumptions about the shape of the data clusters and can identify anomalies in arbitrarily shaped groups [16].

The main drawbacks are computational. The algorithm's complexity can be high, making it slow for very large datasets. Like many distance-based methods, its performance degrades significantly in high-dimensional spaces (a problem known as the “curse of dimensionality”), as the concepts of “density” and “nearest neighbor” become less meaningful. Its results are also sensitive to the user's choice of the 'k' parameter, and the unbounded nature of the LOF score can make it more difficult to interpret than a simple probability [17].

LOF is highly effective in unsupervised intrusion detection, as it can identify new, previously unseen attacks without prior training on them. Studies have shown it is particularly adept at finding subtle attack types, like User-to-Root (U2R) attacks, that other classifiers often struggle with. It is also applied in dynamic environments like cloud computing to detect contextual anomalies in system behavior [17].

Conclusions

No single statistical method is universally superior for detecting anomalies in program behavior. The optimal choice is highly dependent on the context: SPC methods like CUSUM are ideal for detecting small, persistent drifts in stable metrics; regression models are powerful for forecasting complex, seasonal data; and density-based methods like LOF excel in unsupervised environments where local context is critical (table 1).

The most significant practical challenges in this field remain the management of false positives, which leads to “alert fatigue” , and “concept drift,” where models become outdated as a program's normal behavior evolves. Furthermore, as systems become more complex, the demand for explainable models – those that can state why an alert was triggered–is growing. Consequently, the future of anomaly detection likely lies in hybrid models that combine the strengths of multiple techniques and in adaptive, self-learning systems that can evolve with the software they monitor [18].

Table 1 – Comparison of Statistical Anomaly Detection Methods

Method Family	Key Principle	Primary Advantage	Key Disadvantage
Statistical Process Control (SPC) (e.g., CUSUM, EWMA)	Detects shifts from a stable process mean by incorporating historical data.	High sensitivity to small, persistent drifts in a single metric.	Can be prone to false alarms; less effective for complex, multivariate processes.
Regression-Based Models (e.g., ARIMA)	Forecasts normal behavior; flags large prediction errors (residuals) as anomalies.	Excellent for modeling complex systems with seasonality and trends.	Vulnerable to “concept drift” (slowly learning an anomaly as “normal”).
Density-Based Methods (e.g., LOF)	Identifies anomalies as isolated points in low-density regions relative to their local neighbors.	Unsupervised; excels at finding local anomalies that global methods miss.	High computational cost; suffers from the “curse of dimensionality” in high-dimensional data.

REFERENCES

1. Introduction to Anomaly Detection. [Online]. URL: https://www.researchgate.net/publication/390797974 [Accessed: November 18, 2025].

2. Anomaly Detection in Cybersecurity: Techniques, Challenges, and Development Best Practices. [Online]. URL: https://www.apriorit.com/dev-blog/what-is-anomaly-detection-in-cybersecurity [Accessed: November 18, 2025].

3. A Review of Anomaly Detection Strategies to Detect Threats to Cyber-Physical Systems. [Online]. URL: https://www.mdpi.com/2079-9292/12/15/3283 [Accessed: November 18, 2025].

4. The Importance of Features for Statistical Anomaly Detection. [Online]. URL:

https://www.usenix.org/system/files/conference/hotcloud15/hotcloud15-goldberg.pdf [Accessed: November 18, 2025].
5. Statistical Methods for Anomaly Detection in Cybersecurity Solutions: Balancing Cost and Efficiency. [Online]. URL: https://www.apriorit.com/dev-blog/anomaly-detection-with-statistical-methods [Accessed: November 18, 2025].
6. A Survey of Anomaly Detection in Cyber-Physical Systems. [Online]. URL: https://arxiv.org/html/2502.13256v1 [Accessed: November 18, 2025].
7. Anomaly detection beats SPC to manage quality. [Online]. URL: https://acerta.ai/blog/spc-vs-anomaly-detection-methods-to-manage-quality/ [Accessed: November 18, 2025].
8. CUSUM and EWMA Control Charts. [Online]. URL: https://www.jmp.com/en/statistics-knowledge-portal/quality-and-reliability-methods/control-charts/cusum-and-ewma-control-charts [Accessed: November 18, 2025].
9. Why Use Statistical Process Control to Reduce Defects? [Online]. URL: https://spc-software.us/why-use-statistical-process-control-to-reduce-defects [Accessed: November 18, 2025].
10. Software Reliability Modeling Incorporating Fault Detection and Fault Correction Processes with Testing Coverage and Fault Amount Dependency. [Online]. URL: https://www.mdpi.com/2227-7390/10/1/60 [Accessed: November 18, 2025].
11. ARIMA. [Online]. URL: https://www.geeksforgeeks.org/data-science/model-selection-for-arima/ [Accessed: November 18, 2025].
12. OML-AD: Online Machine Learning for Anomaly Detection in Time Series Data. [Online]. URL: https://arxiv.org/html/2409.09742v1 [Accessed: November 18, 2025].
13. Data-Driven Anomaly Detection Approach for Time-Series Streaming Data. [Online]. URL: https://www.6sigma.us/six-sigma-in-focus/cusum-charts-detecting-process-shifts/ [Accessed: November 18, 2025].
14. Everything to Know About Residual Analysis. [Online]. URL: https://www.6sigma.us/six-sigma-in-focus/residual-analysis/ [Accessed: November 18, 2025].
15. Density-based methods. [Online]. URL: https://otexts.com/weird/06-density.html [Accessed: November 18, 2025].
16. Deep Learning for Anomaly Detection in Log Data: A Survey. [Online]. URL: https://arxiv.org/pdf/2207.03820 [Accessed: November 18, 2025].
17. On the nature and types of anomalies: a review of deviations in data. [Online]. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8331998/ [Accessed: November 18, 2025].
18. A Survey on Explainable Anomaly Detection. [Online]. URL: https://arxiv.org/pdf/2210.06959 [Accessed: November 18, 2025].

Лавренюк Арсен Олександрович – студент групи 1ПІ-22б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e–mail: arsenlavreniuk@gmail.com.

Савелко Ростислав Олегович – студент групи 1ПІ-22б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e–mail: rostiksavelko@gmail.com.

Шарко Юрій Олександрович – студент групи 1ПІ-22б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e–mail: sharko.yura2601@gmail.com.

Черноволик Галина Олександрівна – кандидат технічних наук, доцент, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: lina2433@gmail.com.

Майданюк Володимир Павлович – кандидат технічних наук, доцент, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: maidaniuk2000@gmail.com.

Lavreniuk Arsen Oleksandrovich – student of group 1PI-22b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: arsenlavreniuk@gmail.com.

Savelko Rostyslav Olehovych – student of group 1PI-22b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: rostiksavelko@gmail.com.

Sharko Yurii Oleksandrovich – student of group 1PI-22b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: sharko.yura2601@gmail.com.

Chernovolyk Halyna Oleksandrivna – Ph.D., Associate Professor of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: lina2433@gmail.com.

Maidaniuk Volodymyr Pavlovych – Ph.D., Associate Professor of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: maidaniuk2000@gmail.com.