The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 2025

Volume 38, Pages 748-756

IConTES 2025: International Conference on Technology, Engineering and Science

LogSense: A Comparative Anomaly Detection Platform on Web Server Access Logs Using Multiple Algorithms

Arda Ata Turkoglu
Turkcell Payment and Electronic Money Services Inc.

Abstract: This study introduces LogSense, a platform developed for anomaly detection in large-scale web server access logs through a comparative analysis of machine learning algorithms. Three unsupervised approaches; k-Means, DBSCAN, and Isolation Forest were implemented to identify irregular patterns and abnormal traffic behaviors within server activity data. The platform establishes an end-to-end analytical workflow that includes parsing and cleaning log files, performing feature engineering, normalizing features, applying multiple algorithms, and visualizing anomalies through interpretable outputs such as SSD histograms and t-SNE projections. Experimental results show that k-Means is effective in detecting cluster-based outliers, DBSCAN identifies density-related irregularities with high clustering quality, and Isolation Forest isolates diverse anomalies with superior runtime performance. A subsequent cross-validation phase confirmed that anomalies consistently detected by multiple algorithms are highly reliable. Overall, this study underscores the value of proactive anomaly detection in web server security and highlights the complementary strengths of clustering, density, and ensemble-based learning approaches.

Keywords: Web server logs, Anomaly detection, k-Means, DBSCAN, Isolation forest

Introduction

Web servers serve as the backbone of modern digital services, facilitating e-commerce, communication, and information sharing. However, they are also prime targets for cyberattacks, including distributed denial of service (DDoS), brute-force login attempts, and data exfiltration. These attacks not only threaten data integrity but also compromise service availability and organizational reputation. While conventional intrusion detection systems (IDS) rely on known attack signatures, they fail against novel or obfuscated threats. Anomaly detection addresses this gap by flagging unusual patterns that deviate from expected behaviors. Web server access logs contain detailed request information; IP addresses, timestamps, HTTP methods, requested resources, response codes, and byte sizes that can be leveraged to detect such anomalies. Manual log inspection, however, is infeasible due to high data volume, diversity, and lack of labels. This necessitates automated approaches based on unsupervised learning.

Previous studies emphasize the relevance of clustering- and density-based approaches for intrusion detection. Chandola et al. (2009) provided a comprehensive survey highlighting their effectiveness, while Ahmed et al. (2016) evaluated DBSCAN and k-Means on noisy traffic data. More recently, Liu et al. (2008) introduced Isolation Forest, an ensemble method tailored for high-dimensional anomaly detection. Building upon these works, the present study proposes LogSense, an integrated anomaly detection and visualization platform designed for web server logs. The study further conducts a comparative evaluation of three algorithms; k-Means, DBSCAN, and Isolation Forest on real-world log data to assess their relative effectiveness in identifying irregular traffic behaviors and potential anomalies within web infrastructures.

Dataset

The dataset was sourced from the Kaggle repository: Web Server Access Logs (Zaker, 2019). It contains HTTP access logs recorded by a Nginx server of an Iranian e-commerce site (*zanbil.ir*). The raw dataset is approximately 3.3 GB in size, consisting of millions of entries. Each log entry contains multiple descriptive fields capturing the details of client–server interactions. These include the client IP address, which identifies the source of the request; the timestamp, recording the exact date and time of access; and the HTTP method (such as GET or POST) indicating the type of operation performed. The entry also specifies the requested resource or URL, the server response code (for example 200, 404, or 500) that reflects the request outcome, and the response size in bytes, representing the volume of data returned to the client. The dataset enables comprehensive traffic analysis, covering normal browsing, search engine bots, user activity, and potentially malicious behavior.

Preprocessing and Feature Engineering

To prepare the raw web server log data for machine learning–based anomaly detection, a comprehensive preprocessing and feature engineering pipeline was designed and implemented. The process began with log parsing, which was handled through a custom-developed Python module named DataPreparationModule. This component was responsible for reading each raw log entry, parsing textual records, and extracting meaningful attributes such as the client IP address, timestamp, HTTP method, requested URL, response code, and response size. The parsing operation was designed to handle large-scale text files efficiently, filtering out nonconforming records and retaining only the relevant attributes required for analysis. For instance, the function show_raw_log() within the data module retrieves a defined number of sample log entries and presents them to the preprocessing interface for inspection and verification of parsing consistency.

Following the parsing phase, the data underwent cleaning and transformation procedures to ensure consistency and integrity across all records. Incomplete or corrupted entries were removed, duplicated lines were filtered, and the timestamp column was standardized into a uniform ISO 8601 format. These transformations were essential for aligning records across different time intervals and ensuring compatibility with time-series and clustering algorithms. Additionally, all numerical attributes such as request counts and response sizes were normalized using the MinMaxScaler technique. This normalization step mapped all feature values to the range, which helped prevent scale dominance during clustering and distance-based anomaly detection.

Once the dataset was cleaned and normalized, the process continued with feature engineering, where several higher-level metrics were derived from the base attributes to enrich the model's ability to detect anomalies. The engineered features included the total number of requests per IP address, the daily mean request count, the ratio of successful requests (amount of http 200 responses), and statistical measures of the response size such as mean and standard deviation. Moreover, the average time difference between consecutive requests was calculated to capture temporal irregularities in client behavior, which often indicate automated or malicious activity. This stage resulted in a refined dataset that was both statistically consistent and semantically enriched, forming a robust foundation for the subsequent anomaly detection algorithms. The distributions of the engineered features revealed diverse behavioral patterns among client IPs, serving as an early indicator of potential outlier activities.

IP	Total Requests	Daily Mean	GET requests	Successful requests	Mean Return Size	Return Size Std	Mean Time Difference
1.43.175.52	1	1.0	0	1	133.000000	0.000000	0.000000
10.104.208.32	96	96.0	0	92	8843.968750	23508.725105	1.322917
10.105.14.186	8	8.0	0	8	15811.625000	31895.242032	116.000000
10.106.247.163	41	41.0	0	39	20207.731707	37423.013698	3.829268
10.113.251.50	76	76.0	0	60	15093.394737	30716.590386	10.144737
10.114.1.2	84	84.0	0	81	20848.321429	41419.496079	2.035714
10.114.139.15	2	2.0	0	2	94646.000000	0.000000	4680.000000
10.114.98.6	3	3.0	0	3	24592.000000	11198.807660	408.666667
10.116.12.43	37	37.0	0	36	13934.918919	25759.936819	0.189189
10.124.2.223	2	2.0	0	2	4071.000000	135.764502	1.500000

Figure 1. Sample view of engineered features.

Algorithms

k-Means Clustering

The k-Means clustering algorithm was employed as the first approach for detecting anomalies within the preprocessed dataset. This algorithm partitions the dataset into a predefined number of clusters (Chandola et al,

2009), denoted by k, with the goal of minimizing the variance within each cluster. Mathematically, k-Means aims to minimize the objective function given by:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} |x - \mu_i|^2$$

where μ_i represents the centroid of the cluster S_i. In essence, each data point is assigned to the cluster whose centroid is nearest in Euclidean space, thereby reducing intra-cluster variance.

In the context of this study, k-Means was implemented to uncover structural groupings among client IPs based on their behavioral patterns derived from the engineered features. The number of clusters, k, was determined empirically using the Elbow Method, which analyzes the trade-off between the number of clusters and the corresponding Sum of Squared Distances (SSD). The "elbow point" in the curve indicates where adding additional clusters no longer yields substantial improvement in clustering quality (Figure 2). This point was used to select the optimal k value for the dataset.

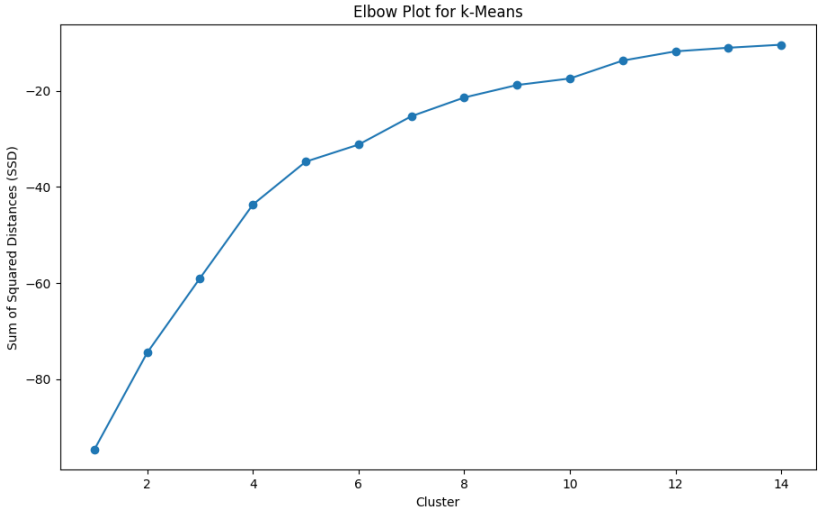

Figure 2. Elbow plot for k-Means.

Once the optimal cluster count was determined, data points with exceptionally high SSD values were labeled as outliers, since they were positioned far from any cluster centroid. The distribution of these SSD values, visualized in Figure 3, provides a clear distinction between normal and anomalous observations. Anomalies appear in the long-tail region beyond the threshold cutoff. To further illustrate the spatial relationships among data points, a t-distributed Stochastic Neighbor Embedding (t-SNE) projection was used.

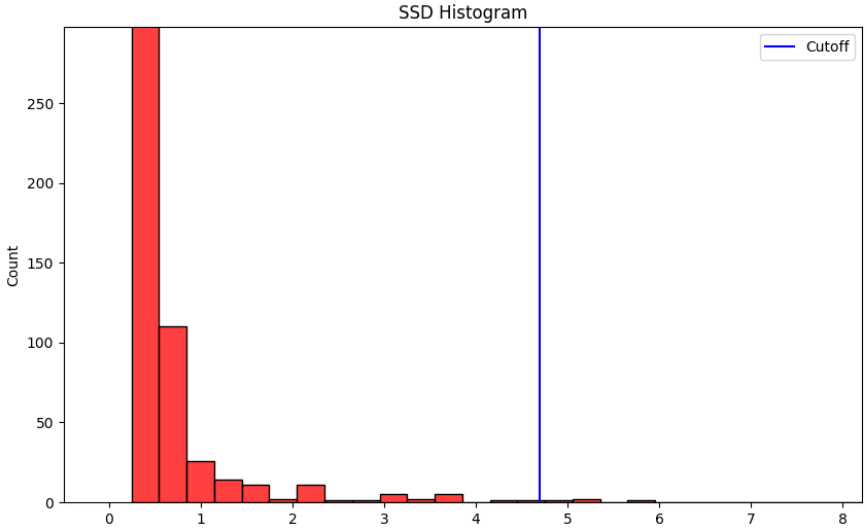

Figure 3. SSD histogram.

As depicted in Figure 4, this two-dimensional visualization of the high-dimensional feature space reveals the structure of clusters and highlights outliers located at the periphery of dense regions. The LogSense platform integrates these visualization tools; Elbow Plot, SSD Histogram, and t-SNE projection into a unified interface, offering both analytical precision and interpretability in identifying anomalous web activity.

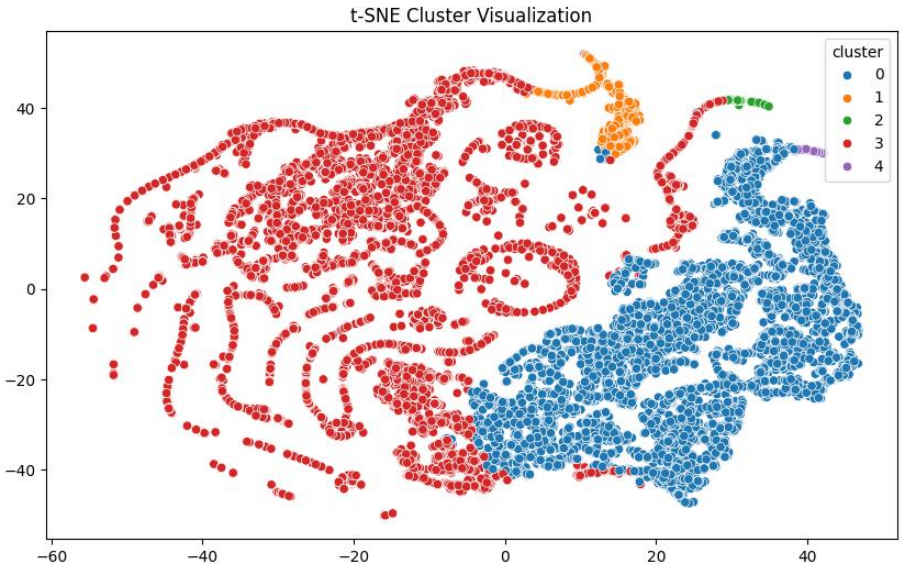

Figure 4. t-SNE cluster visualization.

DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was adopted as the second method in the study. (Ester, Kriegel, Sander, & Xu, 1996) Unlike k-Means, which requires the number of clusters to be predefined, DBSCAN is a density-based clustering algorithm that groups data points according to their spatial proximity and local density, making it particularly effective for datasets with irregular cluster shapes or varying densities. The theoretical foundation of DBSCAN is based on two key parameters: the neighborhood radius ϵ\epsilonϵ and the minimum number of points, *min_samples*, required to form a dense region. A point is defined as a *core point* if it has at least *min_samples* neighboring points within the ϵ\epsilonϵ-radius. Points that are within the neighborhood of a core point are considered *reachable points*, while those that do not meet this condition are labeled as *noise* or anomalies.

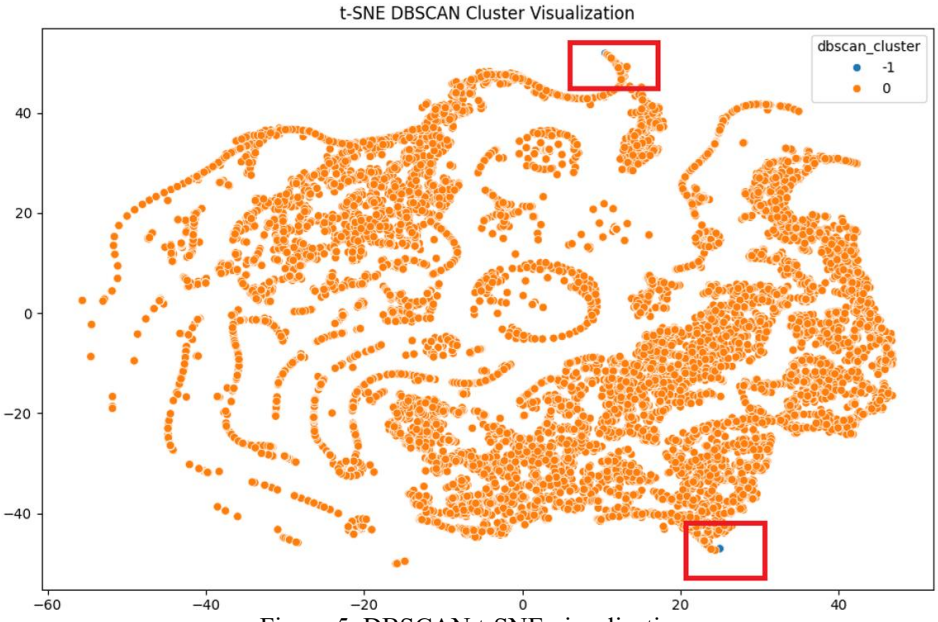

Figure 5. DBSCAN t-SNE visualization.

Formally, for a given point p, its neighborhood is defined as $N_\epsilon(p) = \{ q \in D \mid \text{dist}(p, q) \leq \epsilon \}$. If the cardinality of $N_\epsilon(p)$ is greater than or equal to *min_samples*, p is classified as a core point. In this study, the DBSCAN parameters were set to $\epsilon = 0{,}15$ and $min\ samples = 5$ after iterative experimentation, which provided the most balanced clustering quality based on silhouette scores and noise point detection rates. DBSCAN demonstrated high robustness in identifying outliers corresponding to low-density areas in the feature space, which often indicated unusual client request behaviors, such as abnormally frequent requests or extreme byte transfers. Furthermore, since DBSCAN autonomously determines the number of clusters, it eliminated the need for prior assumptions about cluster count, allowing for a more data-driven analysis of traffic irregularities.

To visualize the results, the LogSense platform employed t-SNE projections for cluster interpretation and color-coded anomaly highlighting. t-SNE visualization (Figure 5) displays the two-dimensional projection of DBSCAN clustering results, where each color represents a distinct cluster and blue points (labeled –1) denote anomalies located in sparse, low-density regions of the feature space. The regions enclosed by red rectangles highlight areas containing these anomalies to emphasize their positions and improve visual interpretability.

Isolation Forest

The third method integrated into the LogSense platform was the Isolation Forest algorithm, an ensemble-based anomaly detection technique specifically designed for high-dimensional datasets (Liu et al., 2008). Isolation Forest operates on the principle that anomalies are rare and differ significantly from normal instances, which makes them easier to isolate using random partitioning. Instead of modeling normal behavior explicitly, Isolation Forest isolates observations by recursively dividing the dataset through randomly selected attributes and split values.

The anomaly score for a given instance x is derived from the average path length $E\big(h(x)\big)$ of x in a forest of randomly generated trees and is computed using the following formula:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

$E(h(x))$ represents the average number of splits required to isolate x, and $c(n)$ denotes the average path length in a binary tree built on n samples. Instances with shorter average path lengths are considered more anomalous, as they are isolated more quickly.

For this study, the Isolation Forest implementation utilized 100 estimators (trees), and the contamination parameter was automatically inferred from the data distribution to estimate the proportion of anomalies. Each tree was trained on a random subsample of the dataset, enhancing the model's generalization capability and reducing overfitting risk. Isolation Forest proved to be both efficient and effective in detecting subtle deviations in web log patterns. Due to its linear time complexity and scalability, it processed the entire 3.3 GB dataset with an average runtime of 1.66 seconds, outperforming both k-Means and DBSCAN in terms of computational efficiency. The algorithm identified numerous anomalies characterized by irregular request intervals, inconsistent return sizes, and sudden surges in daily activity.

Results and Discussion

The experimental evaluations conducted in this study aimed to measure the performance, sensitivity, and consistency of the three anomaly detection Algorithms; k-Means, DBSCAN, and Isolation Forest when applied to the large-scale web server access log dataset. Each algorithm was executed on the same normalized feature set derived from the preprocessing pipeline, and their outcomes were analyzed both quantitatively and visually. The comparative discussion below integrates numerical results, behavioral interpretations, and cross-method validations to provide a holistic understanding of anomaly detection performance within the **LogSense** platform. Earlier comparative works on model-based network intrusion detection have demonstrated similar differences between tree-based and clustering algorithms. (Zhang, Zulkernine, & Haque, 2008)

K-Mean Results

The k-Means algorithm produced well-structured clusters that captured distinct behavioral patterns among client IP addresses. Using the Elbow method, the optimal number of clusters was determined as k = 5, corresponding to a clear inflection point where the Sum of Squared Distances (SSD) began to stabilize. The resulting model achieved a Silhouette Score of 0.618, indicating moderately well-separated and compact clusters, which is acceptable given the heterogeneous nature of web traffic data.

Within this configuration, seven anomalies were identified as data points exhibiting exceptionally large distances from their nearest centroids. The anomalous instances represented approximately 0.46% of the total dataset. Detailed examination revealed that these anomalies were characterized by an average daily request count of approximately 10,060, which was nearly eight times higher than the population mean. Similarly, their mean return size was around 113 KB, compared to a dataset-wide average of 37 KB, suggesting unusually heavy data transfers. Moreover, the mean time difference between consecutive requests (≈9,165 ms) was significantly longer than typical client behavior, potentially indicating irregular communication intervals caused by automated scripts or intermittent scraping activities.

Visual analysis using the SSD histogram provided an intuitive separation between normal and anomalous instances, where anomalous observations formed a long-tail distribution at the far end of the SSD axis. The t-SNE projection further demonstrated that anomalous points were spatially distant from any dense cluster region, confirming their deviation from the typical access patterns.

Table 1. Statistical summary of k-Means anomalies

Variables	Mean	Std
Total Requests	10060	14352
Successful Requests	7325	11265
Mean Return Size	113268	165742
Return Size Std	88951	107480
Mean Time Difference	9165	8387

DBSCAN Results

The DBSCAN algorithm yielded the highest clustering quality among the three approaches, achieving a Silhouette Score of 0.951, which reflects a highly coherent separation of dense and sparse regions within the feature space. Unlike k-Means, DBSCAN automatically discovered the number of clusters, which in this dataset resulted in six distinct dense regions representing normal user behaviors and one sparse region corresponding to anomalous patterns. The algorithm identified seven anomalous instances, matching the number detected by k-Means, but with slight differences in composition. The detected anomalies corresponded to clients whose activity was either excessively frequent or characterized by extreme data volume transfers. Specifically, these instances showed an average request count of 11,396 per day, a mean return size of 146 KB, and a mean response time of approximately 4,267 ms. These values exceeded the median of the overall dataset by factors of 9.4, 3.9, and 4.8, respectively.

DBSCAN's density-based framework allowed it to effectively distinguish normal clustered behavior from outlying low-density points. The resulting t-SNE projection highlighted how these anomalies were isolated from the main population, appearing as scattered singletons with no local neighborhood support. This observation indicates that DBSCAN can capture anomalies that are density-independent such as sporadic high-volume access attempts without the need to predefine the number of clusters. The comparative visualization between k-Means and DBSCAN confirmed that four of the anomalies were common to both algorithms, suggesting high confidence in their abnormality. The remaining three anomalies detected exclusively by DBSCAN represented borderline cases, which were slightly distant but not far enough to be isolated by centroid-based distance measures.

Table 2. Statistical summary of DBSCAN anomalies

Variables	Mean	Std
Total Requests	11396	11749
Successful Requests	8817	9347
Mean Return Size	146735	193675
Return Size Std	98169	100636
Mean Time Difference	4267	7288

Isolation Forest Results

The Isolation Forest algorithm exhibited the highest sensitivity among the tested models, identifying a much larger number of anomalies compared to both k-Means and DBSCAN. The method isolated 229 anomalous instances, corresponding to approximately 15.1% of the dataset, while maintaining an average runtime of 1.66 seconds, making it the fastest of the three. The anomalies detected by Isolation Forest displayed highly variable statistical behavior. The mean request count per day for these instances was approximately 1,171, but with an exceptionally high standard deviation of 3,240, indicating that the anomalies were not uniform in frequency. Similarly, their average return size was 35 KB with a large dispersion (std. dev. ≈ 39 KB), and their average response time was 354 seconds, suggesting irregular or stalled connections. These characteristics collectively signify that Isolation Forest captured both extreme outliers and subtler irregularities missed by clustering-based methods.

An important observation was that many of the anomalies flagged by Isolation Forest corresponded to clients with fluctuating activity patterns; users who alternated between normal and abnormal behavior within short time windows. This sensitivity demonstrates the strength of ensemble-based isolation mechanisms but also introduces a trade-off: while the algorithm can detect a broader spectrum of anomalies, it also yields a higher number of potential false positives, particularly in non-stationary data. Visual inspection of anomaly scores revealed a bimodal distribution, with a clear separation between the majority of normal samples and a smaller group of high-score anomalies. The t-SNE visualization confirmed this separation, where outliers formed diffuse, isolated clusters at the periphery of the feature space.

Table 3. Statistical summary of Isolation Forest anomalies

Variables	Mean	Std
Total Requests	11171	27779
Successful Requests	10041	22197
Mean Return Size	35674	77261
Return Size Std	39262	59199
Mean Time Difference	354	1664

Comparative Analysis

A comparative evaluation across the three algorithms provided deeper insights into their relative strengths and complementary behavior. K-Means and DBSCAN demonstrated high consistency, both detecting seven anomalies and sharing four identical results. This overlap indicates that both algorithms are effective at identifying extreme, well-defined deviations instances that are simultaneously distant from cluster centers and located in low-density regions. These anomalies can be considered “high-confidence” irregularities.

By contrast, Isolation Forest exhibited far greater sensitivity, capturing 229 anomalies. Although this broader detection scope encompasses more false positives, it also highlights its suitability for early-stage or exploratory anomaly detection, where coverage is prioritized over precision. When the anomalies detected by all three methods were cross-compared, approximately 82% of Isolation Forest’s anomalies were unique, reflecting its ability to identify subtle patterns overlooked by clustering-based methods.

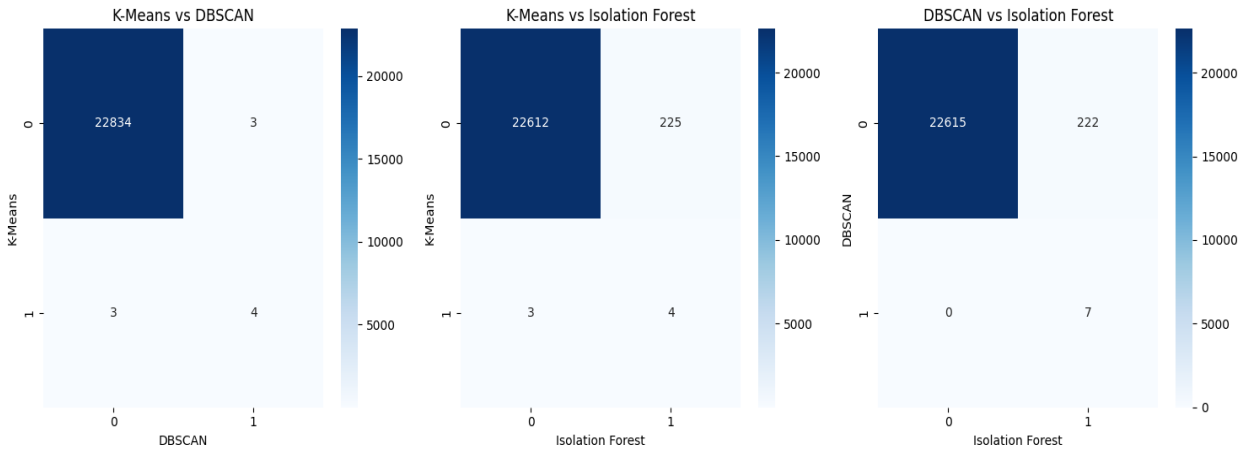

Figure 6. Pairwise comparative analysis of anomaly detection results

From a computational perspective, Isolation Forest also proved superior in terms of efficiency, completing its analysis in under two seconds, compared to approximately 3.4 seconds for k-Means and 5.1 seconds for DBSCAN. This runtime advantage stems from its logarithmic time complexity, making it particularly scalable for future real-time detection integration. An aggregated visualization comparing the anomalies detected by each algorithm reinforced these conclusions (Figure 6). The bar chart summarizing anomaly counts and overlaps demonstrated that k-Means and DBSCAN were highly aligned, while Isolation Forest contributed additional anomalies not captured by the other two.

Overall, these findings suggest that while clustering and density-based algorithms excel at precision-oriented anomaly detection, ensemble-based models like Isolation Forest are better suited for comprehensive anomaly screening. The combination of these methods integrated within the LogSense platform provides a balanced framework, enabling both high-confidence identification and broad-spectrum anomaly discovery.

Conclusion

This study presented LogSense, an integrated anomaly detection platform developed to identify and interpret irregular behaviors in large-scale web server access logs. By combining clustering-based algorithms (k-Means and DBSCAN) with an ensemble approach (Isolation Forest), the system demonstrated that unsupervised learning methods can effectively reveal both prominent and subtle anomalies in real-world traffic data. The experimental findings showed that k-Means excels at identifying well-separated statistical outliers, while DBSCAN detects anomalies in sparse or irregularly distributed regions of the feature space. In contrast, Isolation Forest isolates a broader range of anomalies efficiently, offering faster runtime performance due to its ensemble-based design. The overlapping detection patterns between these algorithms confirmed that a hybrid combination improves both precision and coverage.

Methodologically, LogSense established a complete and interpretable pipeline from preprocessing raw HTTP logs to visualizing anomalies through SSD histograms and t-SNE projections. This workflow demonstrated how structured log features such as request frequency, response size variability, and inter-request timing can meaningfully represent behavioral deviations, even in unlabeled and heterogeneous data. The platform therefore bridges the gap between machine learning theory and operational log analysis, providing a scalable foundation for data-driven web security monitoring.

Recommendations

While LogSense achieved notable accuracy and interpretability, several opportunities remain for enhancement. A key future direction involves extending the platform toward real-time anomaly detection, enabling the identification of threats and performance bottlenecks as they occur. Incorporating stream processing frameworks such as Apache Kafka or Spark Streaming would allow the system to handle continuous log flows effectively. Furthermore, integrating deep learning techniques including autoencoders, LSTM-based sequence models, and Graph Neural Networks (GNNs) could enable the platform to capture temporal dependencies and relational interactions among log entities, improving its understanding of complex attack patterns.

Another promising area is the hybridization of rule-based and machine learning approaches, where algorithmic outputs are validated by contextual rules such as frequency thresholds or response code logic. This integration would reduce false positives and align detection results more closely with operational definitions of abnormal behavior. Finally, expanding LogSense's application scope beyond cybersecurity could yield broader analytical value. The same framework can be adapted for performance optimization, user behavior modeling, and resource allocation analysis, transforming it into a general-purpose platform for log intelligence and observability in large-scale web systems.

Scientific Ethics Declaration

* The author declares that the scientific ethical and legal responsibility of this article published in EPSTEM journal belongs to the author.

Conflict of Interest

* The author declares that they have no conflicts of interest

Funding

* This research received no internal or external funding.

Acknowledgements or Notes

* This article was presented as an oral presentation at the International Conference on Technology, Engineering and Science (www.icontes.net) held in Antalya/Türkiye on November 12-15, 2025.

References

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications, 60*, 19-31.

Bridges, S. M., & Vaughn, R. B. (2000). Intrusion detection via fuzzy data mining. In *Proceedings of the 12th Annual Canadian Information Technology Security Symposium*.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 1-58.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (pp. 226-231). AAAI Press.

García, S., Zunino, A., & Campo, M. (2019). Survey on network-based intrusion detection data sets. *Computers & Security, 86*, 147-167.

Kim, Y., & Kim, J. (2019). Web traffic anomaly detection using LSTM autoencoder. *IEEE Access, 7*, 160040-160052.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM)* (pp. 413-422). IEEE.

Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38*(5), 649-659.

Author(s) Information

Arda Ata Turkoglu
Turkcell Payment and Electronic Money Services Inc., (Paycell R&D Center)
Istanbul, Türkiye
Contact e-mail: *arda.turkoglu@turkcell.com.tr*

To cite this article:

Turkoglu, A. A. (2025). LogSense: A comparative anomaly detection platform on web server access logs using multiple algorithms. *The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 38,* 748-756.