

Anomaly Detection in MariaDB/MySQL Database Activity Using Isolation Forest Based on Binary Log Extraction and Temporal Aggregation

Bagas Triaji

Department of Information Technology, Universitas Teknologi Digital, Indonesia

ABSTRACT

This study proposes a behavior-based framework for detecting anomalous activity in MariaDB/MySQL databases using unsupervised machine learning. Database activity was extracted from ROW-format binary logs using the pymysqlreplication library, then transformed through One-Hot Encoding and one-minute temporal aggregation into a 23-feature representation covering 82 active time slots. Isolation Forest was implemented with PyCaret using a contamination value of 0.08. The model achieved a ROC AUC of 0.9679 and 100% recall, detecting all four anomalous intervals with three false positives among 78 normal observations. Feature analysis identified transaction payload size as the strongest anomaly indicator. A controlled resampling experiment reduced anomaly prevalence from 4.9% to 0.07% while preserving the original empirical characteristics. Under this condition, precision fell to 0.87%, whereas recall remained at 100% and ROC AUC reached 0.9759. Recurring benign structural events, particularly schema initialization, could receive higher anomaly scores than genuine transaction spikes under extreme class imbalance. These findings show that the framework is effective for short-window detection but requires adaptive contamination tuning, dynamic thresholding, or explicit handling of recurring benign events for long-term deployment. An interactive Streamlit dashboard was also developed for operational monitoring.

Keyword : Anomaly Detection; Binary Log; Database Security; Isolation Forest; Temporal Aggregation; Unsupervised Learning

 This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Corresponding Author:

Bagas Triaji,
Department of Information Technology
Universitas Teknologi Digital, Indonesia
Email: bagastriaji@utdi.ac.id

Article history:

Received Jun 12, 2026
Revised Aug 17, 2026
Accepted Sept 18, 2026

1. INTRODUCTION

Database security is a fundamental component of modern information systems because databases store critical operational and sensitive information. Conventional rule- and signature-based mechanisms, however, are increasingly challenged by changing workloads and evolving attack patterns. Static rules may fail to recognize previously unseen threats, while unusual but legitimate activities can generate excessive alerts and contribute to administrator fatigue (Buczak & Guven, 2016).

Anomaly detection addresses this limitation by identifying observations that deviate from expected behavior (Chandola et al., 2009; Momand et al., 2023). This principle has encouraged behavior-based Database Intrusion Detection Systems (DIDS), which assess how a database operates rather than depending only on predefined attack signatures. Security threats may also originate from legitimate users who misuse their privileges, and such insider activities are increasingly difficult to detect with conventional controls (Alzaabi & Mehmood, 2024). Behavioral indicators such as abnormal transaction volume, unusual data modifications, and unexpected schema changes are therefore increasingly relevant to database monitoring.

Machine learning has become an important approach for log-based anomaly detection. Recent studies employ architectures such as LSTM, Transformers, Autoencoders, and Graph Neural Networks (GNNs) to capture temporal and structural patterns in system logs (Momand et al., 2023). Although these approaches can achieve high accuracy, their computational requirements and processing latency may limit real-time deployment (Ferrag et al., 2020). Their complex representations can also make anomaly decisions difficult to interpret for database administrators (Mahbooba et al., 2021). In addition, many

existing evaluations rely on network-traffic datasets such as NSL-KDD, which do not adequately represent the workload semantics of relational database systems (Ferrag et al., 2020).

This study instead places the observation point at the database data tier and uses MariaDB/MySQL binary logs as the source of activity telemetry. Specifically, ROW-format binary logs are processed with `pymysqlreplication`, providing visibility into actual row-level changes, including values before and after modification (Sallam & Bertino, 2019). This approach reduces dependence on SQL statement interpretation and enables anomaly detection to focus directly on database change behavior.

Because raw binary-log records are event-level and partly categorical, they are first transformed using One-Hot Encoding and one-minute temporal aggregation. The resulting representation allows the detection model to identify broader behavioral changes, such as shifts between DML and DDL activity, sudden increases in transaction frequency, and unusual transaction sizes, rather than relying on isolated events.

The aggregated feature space is analyzed using Isolation Forest (iForest), an unsupervised method introduced by Liu et al. (2008). Its ability to detect anomalies without labeled training data, together with its relatively low memory and computational requirements, makes it suitable for the proposed setting (Liu et al., 2008). The framework is implemented with PyCaret and complemented by an interactive Streamlit dashboard to support practical monitoring and interpretation of anomaly results.

The main contributions of this study are fourfold: direct extraction of database telemetry from ROW-format binary logs using `pymysqlreplication`; temporal feature engineering that converts granular database events into a one-minute multivariate representation; empirical evaluation of Isolation Forest using MariaDB/MySQL activity with feature-importance analysis; and implementation of the complete detection pipeline in an interactive Streamlit dashboard.

The remainder of this paper is organized as follows. Section 2 reviews Isolation Forest developments, deep learning approaches for log anomaly detection, and database-specific studies. Section 3 describes the experimental procedure, including activity generation, binlog extraction, feature engineering, modeling, evaluation, and dashboard implementation. Section 4 presents and discusses the experimental results, including the controlled prevalence-shift analysis. Section 5 concludes the study and outlines future research directions.

2. LITERATURE REVIEW

Research on anomaly detection relevant to this study can be grouped into three areas: the development of Isolation Forest and its variants, deep learning for log anomaly detection, and anomaly detection specifically targeting database systems. The first focuses on improving isolation-based detection and interpretability, the second emphasizes complex temporal and structural patterns, while the third highlights the need for database-specific telemetry and representative workloads.

A. Evolution of the Isolation Forest Algorithm

Standard Isolation Forest uses random, axis-parallel partitioning to isolate observations and is attractive because of its relatively low computational cost (Liu et al., 2008). However, this partitioning strategy may be less effective for complex or nonlinear feature distributions. (Hariri et al., 2021) addressed this limitation through Extended Isolation Forest (EIF), which uses randomly oriented separating hyperplanes to reduce score bias and “ghost regions.” Xu et al. (2023) proposed Deep Isolation Forest (DIF), combining randomized neural representations with the isolation mechanism to improve detection of difficult anomalies in nonlinear spaces. Although these variants can improve detection capability, their additional computational cost may be less suitable for high-frequency transactional logs.

The comparison in Table 1 places these developments alongside other relevant approaches and shows that research on iForest has pursued different priorities, including detection capability, efficiency, and interpretability. Cao et al. (2025) found that isolation-based methods retain advantages in execution speed and memory consumption compared with distance- and density-based approaches, supporting their use in real-time applications. However, their survey does not specifically address preprocessing requirements for categorical, event-level logs.

Feature engineering is particularly important for dynamic log data. Al Farizi et al. (2021), through a systematic review of 17 iForest improvement studies, reported that conventional iForest may

struggle with dynamic anomalies when temporal behavior is not explicitly represented. This motivates the use of temporal aggregation in the present study. From an interpretability perspective, Rachwal et al. (2024) introduced a Shapley-index-based attribute exclusion strategy that estimates feature contributions through repeated model training. Although their method was not evaluated on database logs, it provides a relevant basis for interpreting the variables influencing anomaly decisions.

B. Deep Learning Approaches for Log Anomaly Detection

Deep learning has become another major direction for log anomaly detection because of its ability to model complex temporal and structural relationships. Chourasiya et al. (2025) combined LSTM, Transformer, and GNN architectures for cross-domain log analysis and reported 98.2% accuracy with strong forensic timeline reconstruction. However, the approach requires heterogeneous log integration and involves greater computational and training complexity.

Sussan et al. (2023) investigated deep learning for database intrusion detection and prevention, particularly for SQL injection and cross-site scripting. Their results demonstrate the potential of deep learning for database security, although effective model training requires substantial representative data. Li et al. (2026) proposed Logs2Graphs, which transforms groups of log messages into weighted directed graphs and applies an OCDiGCN model for graph-level anomaly detection and node-level explanation. Despite its explanatory capability, graph construction introduces additional computational overhead.

As reflected in Table 1, these approaches offer strong modeling capability but generally require more computational resources or training data than lightweight isolation-based methods. This trade-off is particularly relevant when anomaly detection must operate continuously on high-frequency database activity.

C. Database-Specific Anomaly Detection

Database anomaly detection requires telemetry that reflects actual database behavior rather than generic network or system activity. Uchenna Jeremiah et al. (2026), in a systematic review of 1,247 studies, identified the increasing limitations of rule-based and statistical approaches and highlighted the scarcity of representative benchmark datasets reflecting genuine ACID workloads. This limitation supports the use of telemetry generated directly from a database environment.

Zhang et al. (2025) proposed E-Log, which dynamically adjusts logging granularity to balance diagnostic detail and monitoring overhead. Its emphasis, however, is on adaptive logging rather than behavioral feature construction from transactional events. Tengku et al. (2025) demonstrated the feasibility of Isolation Forest for MySQL query-performance anomaly detection using Performance Schema data, but their features were aggregate performance indicators rather than row-level database changes.

This distinction is important for the present study. As indicated in Table 1, Performance Schema primarily describes engine-level performance, whereas ROW-format binary logs capture direct data-change activity, including DML or DDL type, affected rows, and changed columns. Such telemetry may reveal unusual data manipulation or schema activity that does not necessarily produce a prominent performance anomaly.

D. Synthesis and Research Novelty Position

The literature in Table 1 indicates several consistent patterns. Improved iForest variants address limitations in complex data distributions and interpretability (Hariri et al., 2021; Rachwal et al., 2024; Xu et al., 2023), while broader evaluations continue to emphasize the computational efficiency of isolation-based methods (Cao et al., 2025). Al Farizi et al. (2021) further shows that suitable feature engineering is necessary when temporal behavior is an important part of the anomaly pattern.

Deep learning methods provide strong detection capability but introduce greater computational complexity or data requirements (Chourasiya et al., 2025; Sussan et al., 2023; Uchenna Jeremiah et al., 2026). Within the database domain, existing work also reveals a shortage of representative datasets and limited use of direct row-level transactional telemetry (Tengku et al., 2025; Uchenna Jeremiah et al., n.d.; Zhang et al., 2025).

The research gap identified from Table 1 is addressed in this study through three combined elements: direct telemetry extraction from ROW-format binary logs using pymysqlreplication; transformation of DML and DDL events into a multivariate feature space through 60-second temporal aggregation; and evaluation of a lightweight Isolation Forest model with feature-importance analysis. This approach retains the efficiency emphasized by Cao et al. (2025) and Al Farizi et al. (2021), incorporates the interpretability perspective of Rachwał et al. (2024), and avoids the higher computational burden associated with the deep learning approaches of Chourasiya et al. (2025) and Li et al. (2026).

Table 1. State-of-the-Art Literature Evaluation Matrix

Author (Year)	Method/Focus	Main Result	Limitation / Gap
(Xu et al., 2023)	Deep Isolation Forest	SOTA performance on complex/non-linear anomalies	High computational overhead; not for transactional log streams
(Hariri et al., 2021)	Extended Isolation Forest (EIF)	Resolves standard iForest score bias	Extra overhead; untested on database logs
(Cao et al., 2025)	Survey of isolation mechanisms	iForest fastest & most memory-efficient	Does not address categorical log preprocessing
(Al Farizi et al., 2021)	SLR of 17 iForest-improvement studies	Confirms iForest efficiency on high-dim. data	Fails on dynamic anomalies without feature engineering
(Rachwał et al., 2024)	iForest + SHAP	Improves accuracy & interpretability	Untested on database log domain
(Chourasiya et al., 2025)	LSTM + Transformer + GNN	98.2% accuracy; strong forensic timeline	High latency; complex cross-hierarchy log fusion
(Li et al., 2026)	Logs2Graphs (OCDiGCN)	Competitive performance + node explanation	Graph transformation is computationally heavy
(Uchenna Jeremiah et al., 2026)	SLR of ML/DL DB anomaly detection	Maps large-scale ML/DL trends	Scarcity of representative ACID-based benchmark datasets
(Zhang et al., 2025)	E-Log: adaptive DBMS logging	Balances accuracy vs. overhead	Focuses on logging strategy, not behavioral features
(Tengku et al., 2025)	iForest on MySQL Performance Schema	Proves iForest feasibility for query anomalies	Aggregate metrics, not row-level binlog events
(Sussan et al., 2023)	Deep learning for SQLi/XSS on DB	DL-based intrusion detection/prevention	Requires large training data volume

3. RESEARCH METHOD

This study follows a systematic experimental procedure consisting of five stages: database activity generation and binlog extraction, feature engineering and temporal aggregation, Isolation Forest modeling, performance evaluation, and dashboard implementation. The overall workflow is presented in Fig. 1, illustrating the transition from raw database activity to anomaly detection and monitoring.

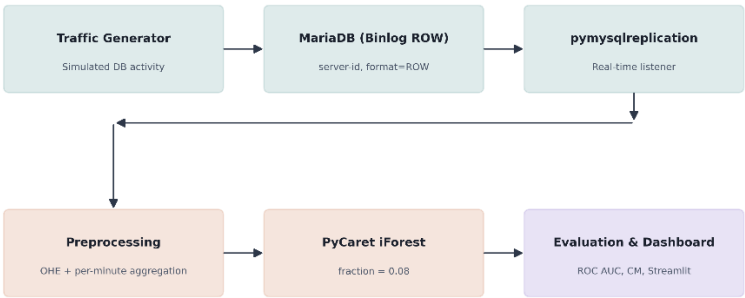


Fig 1. System architecture for database anomaly detection

A. Activity Simulation and Binlog Data Extraction

The MariaDB server was configured with ROW-format binary logging and binlog_row_image=FULL, allowing the system to record row values before and after modification (MariaDB Foundation, 2026). The experimental database, ecommerce_db, contained four related tables—users, products, orders, and order_items—linked through foreign keys to represent typical e-commerce transactions.

A Python traffic generator produced two activity classes. Normal workload consisted of `insert_order()`, `update_user()`, and `delete_order()` with a 70:20:10 probability distribution over approximately 10 minutes. Anomalous behavior was introduced through `spike_inserts()`, representing short bursts of mass INSERT operations, and `ddl_event()`, representing unscheduled schema changes such as adding a column. These events were injected at 1–2 minute intervals following normal activity cycles.

During execution, `pymysqlreplication` captured `WriteRowsEvent`, `UpdateRowsEvent`, `DeleteRowsEvent`, and DDL events such as CREATE, ALTER, and DROP. Each event was recorded with its timestamp, event type, affected-row count, event size, and query text.

The recorded period extended from 13:24:42 to 15:00:33, totaling 1 hour, 35 minutes, and 51 seconds. Of the 97 one-minute intervals, 82 contained database activity, while 15 contained no events. Fourteen consecutive inactive minutes occurred at the beginning and were treated as initialization time. During anomaly periods, approximately 2,000–2,200 INSERTs and 1,000–1,100 UPDATEs occurred per minute, compared with about 60–90 INSERTs and 30–45 UPDATEs during normal activity.

B. Feature Engineering and Temporal Aggregation

Because the extracted events were highly granular and included categorical attributes, two preprocessing steps were applied: One-Hot Encoding (OHE) and one-minute temporal aggregation. The `event_type` and `query_type` variables were converted into binary indicators, with absent categories assigned a value of zero to maintain a consistent feature structure.

The encoded events were then grouped into one-minute windows, from which count, mean, maximum, and sum statistics were calculated. This produced 82 observations with 23 features, with each row representing the database state during one active minute. The resulting variables are detailed in Table 2, covering event volume, DML and DDL operations, transaction size, row and column changes, query complexity, and table coverage.

The aggregation enables the model to capture behavioral changes across a complete time window rather than isolated events, including increased transaction frequency, shifts in DML/DDL composition, and unusual transaction payload sizes.

Table 2. Feature Variables from Per-Minute Temporal Aggregation

Category	Variable	Description
Event volume	total_events, total_queryevent	Total number of events and queries per minute
DML operations	total_insert, total_update, total_delete, total_writeroes, total_updaterows, total_deleterows	Frequency of INSERT/UPDATE/DELETE and affected rows
DDL operations	total_ddl, total_create, total_alter, total_drop	Frequency of schema-structure changes
Transaction size	sum_event_size, avg_event_size, max_event_size	Event payload size statistics (bytes)
Row/column change	sum_rows_changed, avg_rows_changed, max_rows_changed, avg_cols_changed, max_cols_changed	Statistics of rows/columns changed
Query complexity	avg_query_length, max_query_length	Query text length statistics
Table coverage	unique_tables	Number of unique tables accessed per minute

C. Anomaly Detection Modeling with Isolation Forest

Isolation Forest (iForest) operates on the principle of isolating anomalous observations from normal data without a target variable (unsupervised) (Liu et al., 2008, 2012). Anomalies are isolated faster within a random partition-tree structure (Isolation Tree/iTree) because they are “few and different.” Partitioning is performed recursively by randomly selecting a feature and a random split threshold between its minimum and maximum value. The anomaly degree of an observation x is measured by the average path length $h(x)$, defined as the number of edges traversed from the root node to the leaf node. To normalize path length against sample size n , the average path length of an unsuccessful search in a Binary Search Tree (BST) is used as the comparison constant $c(n)$ (Liu et al., 2008, 2012):

$$(1)$$

$$c(n) = 2 H(n-1) - \frac{2(n-1)}{n}, H(n-1) \approx \ln(n-1) + 0.5772156649$$

so that

$$c(n) = 2 \ln(n-1) + 1.1544313298 - \frac{2(n-1)}{n} \quad (2)$$

where 0.5772156649 is the Euler–Mascheroni constant. The anomaly score $s(x, n)$ for observation x is then computed as:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (3)$$

where $E(h(x))$ is the expected path length $h(x)$ across the Isolation Tree ensemble. Interpretation of $s(x, n)$: (1) if $E(h(x)) \rightarrow 0$, then $s \rightarrow 1$: a very short path, strongly classified as an anomaly; (2) if $E(h(x)) \rightarrow c(n)$, then $s \rightarrow 0.5$: path length comparable to a typical random sample, classified as normal; (3) if $E(h(x)) \rightarrow n-1$, then $s \rightarrow 0$, indicating the point lies at the center of the normal cluster.

Modeling was orchestrated via the PyCaret library: (a) the `setup()` function was run on feature matrix X (23 numeric columns, excluding the minute and label columns) with `normalize=True` (z-score method) and `session_id=42` for reproducibility; (b) `create_model('iforest', fraction=0.08)` trained the model with contamination ratio $\alpha = 0.08$; (c) `assign_model()` produced the prediction label (Anomaly: 1/0) and anomaly score (Anomaly_Score) for each observation. PyCaret in this study was installed directly from the GitHub development branch (not the stable PyPI release) to obtain `feature_importances_` interpretability capability for the anomaly model (PyCaret, 2026); this detail is important for reproducibility by other researchers.

D. Evaluation Procedure

Predicted anomaly labels were compared with ground-truth labels derived from the simulated activity intervals. These labels were not used during model training and served only for evaluation. Performance was assessed through the Confusion Matrix, ROC AUC, and feature-importance analysis.

The complete aggregation, normalization, and Isolation Forest pipeline was also independently reproduced from the raw dataset using scikit-learn (Pedregosa et al., 2011). This independent implementation was used to verify the consistency of the principal experimental results.

E. Monitoring Dashboard Implementation

The detection pipeline was implemented as an interactive Streamlit application to demonstrate operational feasibility. The dashboard allows users to upload aggregated CSV data, adjust the contamination parameter, execute anomaly detection, and inspect prediction results.

The interface provides anomaly-distribution, anomaly-score, and ROC-curve visualizations. The relationship between extraction, preprocessing, modeling, evaluation, and monitoring is presented in Fig. 1, while the feature structure generated during aggregation is provided in Table 2. These elements connect the experimental workflow with its intended monitoring application.

4. RESULTS AND DISCUSSION

A. Aggregated Dataset Description

The binlog extraction and aggregation process produced 82 observations with 23 features from 20,775 individual event records in `binlog_10cols.csv`. An independent reproduction generated the same 82×23 structure, confirming the consistency of the processing workflow.

The 78 normal minutes contained approximately 110–130 events per minute, following the intended workload composition of 70% INSERT, 20% UPDATE, and 10% DELETE. This relatively stable pattern provided the baseline for identifying deviations.

The four anomalous intervals—14:04, 14:20, 14:21, and 14:35—showed a substantial increase to approximately 3,000–3,300 events per minute, or about 25 times the normal level. INSERT and UPDATE operations dominated these periods at an approximate 2:1 ratio, while the 14:04 interval also

contained an ALTER TABLE operation. The difference between normal and anomalous activity is illustrated in Fig 2.

The 23-feature representation captures multiple aspects of database behavior, including event volume, DML and DDL activity, transaction size, row and column changes, query complexity, and the number of accessed tables. Thus, each one-minute observation represents a broader behavioral profile rather than a single database event, providing the feature space used in subsequent anomaly detection.

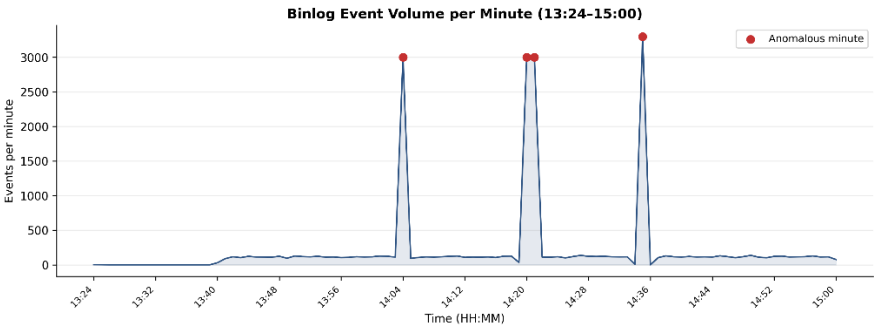


Fig 2. Binlog event volume per minute (actual data)

B. Anomaly Detection Results

The Isolation Forest model was evaluated on the 82 aggregated observations, with its results verified against the original analysis output and an independent reproduction using scikit-learn. Table 3 presents the resulting confusion matrix and evaluation metrics.

Table 3. Confusion Matrix and Evaluation Metrics		
	Predicted Normal	Predicted Anomaly
Actual Normal	75 (TN)	3 (FP)
Actual Anomaly	0 (FN)	4 (TP)

Metric	Value
Accuracy	96.34%
Precision	57.14%
Recall (Sensitivity)	100%
Specificity	96.15%
F1-Score	72.73%
ROC AUC	0.9679

The model correctly classified 75 normal observations and 4 anomalous observations, while 3 normal observations were incorrectly flagged. No actual anomaly was missed. Consequently, the model achieved 96.34% accuracy, 57.14% precision, 100% recall, 96.15% specificity, and an F1-score of 72.73%, with a ROC AUC of 0.9679.

The perfect recall indicates that all four injected anomaly intervals were detected, which is particularly important in database-security applications where missed anomalies can be more critical than false alarms. Meanwhile, the precision of 57.14% reflects the three false-positive detections among seven observations classified as anomalous.

The model's discriminative capability is further demonstrated by the ROC curve in Fig 3, where the AUC of 0.9679 indicates strong separation between normal and anomalous activity. Overall, the results show that the selected Isolation Forest configuration was effective in detecting the injected anomalies, although legitimate database states can also appear statistically unusual.

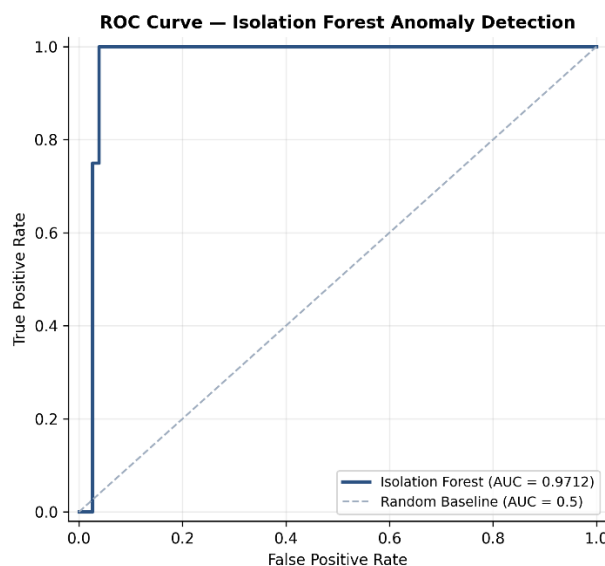


Fig 3. ROC curve for Isolation Forest anomaly detection

C. Feature Importance Analysis

The ten variables with the greatest influence on the Isolation Forest decisions are presented in Fig 4. The two dominant features were `avg_event_size` (0.1839) and `sum_event_size` (0.1690), which together accounted for more than 35% of the total importance.

The next most influential variables were `avg_cols_changed` (0.0676) and `sum_rows_changed` (0.0573). Other prominent features, including `total_writerows`, `total_updaterows`, `total_insert`, and `total_events`, contributed more evenly, with importance values generally between 0.04 and 0.05.

As shown in Fig 4, transaction-size variables had greater influence than most event-frequency measures. This suggests that anomalous behavior in the experiment was characterized not only by a higher number of database operations but also by a substantial increase in transaction payload volume, consistent with the large INSERT bursts generated by `spike_inserts()`.

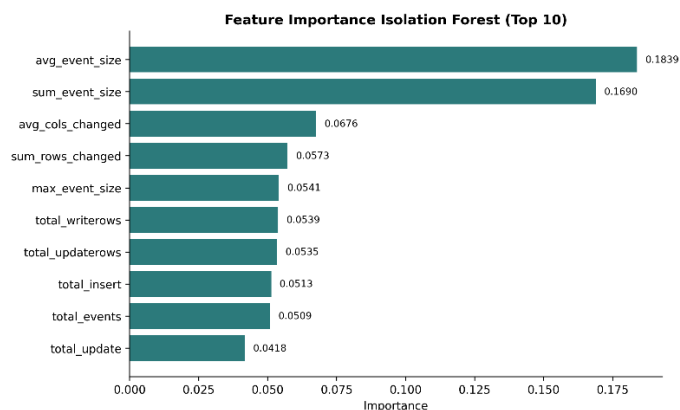


Fig 4. Isolation Forest feature importance (top 10)

The interpretation of these values requires methodological caution because standard Isolation Forest relies on random partitioning rather than information-gain-based feature selection. Consequently, scikit-learn does not provide a native `feature_importances_` attribute for `iForest`. Carletti et al. (2023) addressed this limitation through DIFFI (Depth-based Isolation Forest Feature Importance), which estimates feature contributions from isolation depth.

As an additional comparison, a Random Forest surrogate classifier (Breiman, 2001) was tested on the independently reproduced dataset. Its feature ranking differed from the PyCaret results, indicating that feature-importance interpretation depends on the method employed. Therefore, the

values in Fig 4 should be interpreted specifically within the PyCaret implementation and software environment used in this study.

D. Temporal Case Analysis

The seven observations classified as anomalous consisted of four True Positives (TP) and three False Positives (FP). The four TP cases occurred at 14:04, 14:20, 14:21, and 14:35, corresponding exactly to the intervals in which `spike_inserts()` and `ddl_event()` were introduced. Their activity levels reached approximately 3,000–3,300 events per minute, clearly exceeding the normal workload.

The three FP cases occurred at 13:24, 13:25, and 13:40. The first two corresponded to the initial CREATE TABLE operations during system initialization. Although only 2–3 events were recorded, the long DDL statements produced unusually high query-length values. The 13:40 observation occurred immediately after the approximately 14-minute initialization gap and contained only INSERT operations, unlike normal minutes that typically included INSERT, UPDATE, and DELETE activity.

These findings show that the false positives were associated with legitimate initialization and transition states rather than arbitrary model errors. The Isolation Forest therefore detected both genuine anomaly periods and statistically unusual benign events, which explains the combination of 100% recall and three false positives observed in the evaluation.

E. Comparison with Prior Studies

The performance obtained in this study is comparable to results reported in the literature summarized in Table 1. The model achieved a ROC AUC of 0.9679, approaching the 98.2% accuracy reported by Chourasiya et al. (2025), despite using the considerably simpler Isolation Forest rather than a combined LSTM, Transformer, and GNN architecture. This is consistent with the computational-efficiency advantage of isolation-based methods reported by Cao et al. (2025). Compared with Tengku et al. (2025), who used Isolation Forest on MySQL Performance Schema metrics, the present study demonstrates that ROW-format binary-log telemetry can also provide a strong detection signal, achieving 100% recall while retaining more detailed information about changed rows and columns. The distinction is important because Performance Schema represents aggregate engine-level performance, whereas ROW-format binlogs directly capture data-change activity that may reveal behavioral anomalies not apparent from performance metrics alone.

F. Dashboard-Based Visualization

The operational output of the proposed framework is illustrated in Fig 5, which presents the Streamlit dashboard using the same 82-sample dataset with a contamination value of 0.08. The left panel shows 75 normal and 7 anomalous predictions, consistent with the confusion matrix in Table 3. The right panel displays the anomaly-score trend across the observation period, separating the detected anomalous observations from the normal activity. This result confirms that the detection pipeline can be reproduced through the interactive dashboard rather than being limited to the offline analysis environment.



Fig 5. Streamlit dashboard: prediction distribution and anomaly-score trend

G. Prevalence-Shift Stress Test via Controlled Resampling

A controlled resampling experiment was conducted to examine how changes in anomaly prevalence affect detection performance. Using the empirical distributions from the original 82-minute

dataset, a synthetic 7-day timeline containing 10,080 minutes was generated while preserving the underlying activity characteristics. The anomaly prevalence was reduced from 4.9% (4/82) to approximately 0.07% (7/10,048).

A daily activity cycle with a business-hours peak around 14:00 and a $0.55\times$ weekend reduction factor was incorporated, followed by the injection of seven anomaly intervals based on the four observed anomaly minutes with small multiplicative variations. This design isolates the effect of prevalence change without introducing a new underlying data distribution.

With the original contamination = 0.08, precision dropped sharply to 0.87%, while recall remained at 100% and ROC AUC increased to 0.9759. The resulting confusion matrix contained 9,244 TN, 797 FP, 7 TP, and 0 FN. The effect of different contamination values on precision, recall, and F1-score is shown in Fig 6.

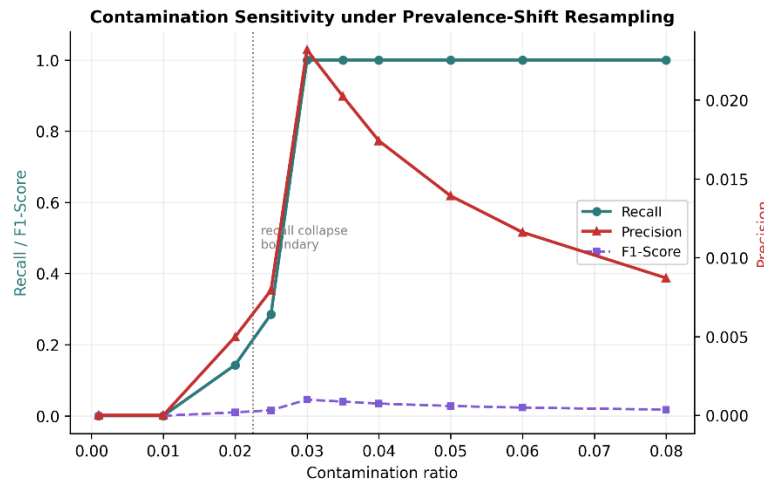


Fig 6. Precision, recall, and F1-score across contamination values under the resampled prevalence-shift dataset

The contamination sweep revealed a critical range around 0.02–0.025. Below this level, recall fell to 0%, while higher values maintained perfect recall but kept precision below 3%. Further examination showed that recurring benign structural events, particularly schema-initialization patterns, were repeatedly ranked as more anomalous than the injected transaction spikes.

These findings indicate that a fixed contamination value is unsuitable when anomaly prevalence changes substantially. Practical deployment therefore requires periodic re-estimation from recent traffic, adaptive percentile-based thresholding, or explicit whitelisting of known benign structural events. Because the experiment resampled the same original session, however, it should be regarded as a controlled prevalence-shift stress test rather than evidence of generalization to previously unseen production workloads.

5. CONCLUSION

This study developed and evaluated a MariaDB/MySQL anomaly detection framework by combining ROW-format binary-log extraction, one-minute temporal aggregation, and unsupervised Isolation Forest modeling. Based on 82 aggregated observations from 1 hour, 35 minutes, and 51 seconds of database activity, the model achieved a ROC AUC of 0.9679 and 100% recall, detecting all genuine anomalies without false negatives. Feature analysis identified `avg_event_size` and `sum_event_size` as the most influential variables, indicating that transaction payload size was a stronger anomaly indicator than event frequency or the number of changed rows and columns. The three false positives were associated with legitimate system-initialization and transition activities rather than missed anomalies.

The controlled prevalence-shift experiment demonstrated an important limitation of the fixed contamination setting. When anomaly prevalence decreased from 4.9% to 0.07%, the contamination value of 0.08 reduced precision to 0.87%, although recall remained at 100%. Reducing contamination below approximately 0.025 subsequently caused recall to fall to 0%, as recurring benign structural events could receive higher anomaly scores than genuine transaction spikes. Because the stress test was generated by resampling the same empirical session, these findings represent the effect of prevalence

shift under controlled conditions rather than evidence of long-term generalization to unseen production workloads.

The framework was further implemented as an interactive Streamlit dashboard, demonstrating its feasibility for short-window operational monitoring. Future work should validate the approach using real production logs over longer periods, develop adaptive contamination-tuning or whitelisting mechanisms, standardize feature-importance interpretation, and evaluate scalability on distributed database clusters. Further investigation should also compare Isolation Forest variants, particularly Extended Isolation Forest (Hariri et al., 2021) and Deep Isolation Forest (Xu et al., 2023), under comparable long-horizon conditions, while extending the temporal-aggregation strategy to other relational database platforms such as PostgreSQL and Oracle.

REFERENCES

- Al Farizi, W. S., Hidayah, I., & Rizal, M. N. (2021). Isolation Forest Based Anomaly Detection: A Systematic Literature Review. *2021 8th International Conference on Information Technology, Computer and Electrical Engineering, ICITACEE 2021*, 118–122. <https://doi.org/10.1109/ICITACEE53184.2021.9617498>
- Alzaabi, F. R., & Mehmood, A. (2024). A Review of Recent Advances, Challenges, and Opportunities in Malicious Insider Threat Detection Using Machine Learning Methods. *IEEE Access*, 12, 30907–30927. <https://doi.org/10.1109/ACCESS.2024.3369906>
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- Buczak, A. L., & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys and Tutorials*, 18(2), 1153–1176. <https://doi.org/10.1109/COMST.2015.2494502>
- Cao, Y., Xiang, H., Zhang, H., Zhu, Y., & Ting, K. M. (2025). Anomaly Detection Based on Isolation Mechanisms: A Survey. *Machine Intelligence Research* 2025 22:5, 22(5), 849–865. <https://doi.org/10.1007/S11633-025-1554-4>
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection. *ACM Computing Surveys*, 41(3), 1–58. <https://doi.org/10.1145/1541880.1541882>
- Chourasiya, L., Khatri, S., Lilhore, U. K., Simaiya, S., Alroobaea, R., Baqasah, A. M., Alsafyani, M., & Khan, M. (2025). Advanced system log analyzer for anomaly detection and cyber forensic investigations using LSTM and transformer networks. *Journal of Cloud Computing*, 14(1). <https://doi.org/10.1186/S13677-025-00789-Y>
- Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50, 102419. <https://doi.org/10.1016/J.JISA.2019.102419>
- Hariri, S., Kind, M. C., & Brunner, R. J. (2021). Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4), 1479–1489. <https://doi.org/10.1109/TKDE.2019.2947676>
- Li, Z., Shi, J., & van Leeuwen, M. (2026). Graph Neural Networks Based Log Anomaly Detection and Explanation. *Data Mining and Knowledge Discovery*. <https://doi.org/10.1007/s10618-026-01235-6>
- Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 413–422. <https://doi.org/10.1109/ICDM.2008.17>
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2012). Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1), 1–39. <https://doi.org/10.1145/2133360.2133363>
- Mahbooba, B., Timilsina, M., Sahal, R., & Serrano, M. (2021). Explainable Artificial Intelligence (XAI) to Enhance Trust Management in Intrusion Detection Systems Using Decision Tree Model. *Complexity*, 2021. <https://doi.org/10.1155/2021/6634811>
- MariaDB Foundation. (2026). *Binary Log Formats*. <https://mariadb.com/docs/server/server-management/server-monitoring-logs/binary-log/binary-log-formats>
- Momand, A., Jan, S. U., & Ramzan, N. (2023). A Systematic and Comprehensive Survey of Recent Advances in Intrusion Detection Systems Using Machine Learning: Deep Learning, Datasets, and Attack Taxonomy. *Journal of Sensors*, 2023(1), 6048087. <https://doi.org/10.1155/2023/6048087>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- PyCaret. (2026). *PyCaret: Low-Code Machine Learning for Python*. <https://pycaret.org/>
- Rachwal, A., Karczmarek, P., Rachwal, A., & Stegierski, R. (2024). Isolation Forest With Exclusion of Attributes Based on Shapley Index. *IEEE Access*, 12, 101797–101813. <https://doi.org/10.1109/ACCESS.2024.3432174>

- Sallam, A., & Bertino, E. (2019). Techniques and Systems for Anomaly Detection in Database Systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11550 LNCS, 113–133. https://doi.org/10.1007/978-3-030-17277-0_7
- Sussan, O., Matthias, D., & Anireh, V. (2023). A Model for Intrusion Detection and Prevention in a Database System Using Deep Learning. *Journal of Web Engineering & Technology*, 10(2), 1–9.
- Tengku, T. D. F., Buaton, R., & Syahputra, S. (2025). Implementation of the Isolation Forest Algorithm for MySQL Query Performance Anomaly Detection Based on Data Performance Schema. *Journal of Artificial Intelligence and Engineering Applications*, 5(1), 592–597.
- Uchenna Jeremiah, N., Bernice Stephen, B., Oluwatobi, O., Victor Ayomide, A., Abiodun John, O., Samuel, N., & Oluwaseyi Segun, A. (n.d.). *Machine Learning-Driven Anomaly Detection in a Large-Scale Database Systems: A Systematic Literature Review*. <https://doi.org/10.51244/IJRSI>
- Uchenna Jeremiah, N., Stephen, B. B., Oluwatobi, O., Ayomide, A. V, John, O. A., Samuel, N., & Segun, A. O. (2026). Machine Learning-Driven Anomaly Detection in a Large-Scale Database Systems: A Systematic Literature Review. *International Journal of Research and Scientific Innovation*, 13(4). <https://doi.org/10.51244/IJRSI.2026.1304000140>
- Xu, H., Pang, G., Wang, Y., & Wang, Y. (2023). Deep Isolation Forest for Anomaly Detection. *IEEE Transactions on Knowledge and Data Engineering*, 35(12), 12591–12604. <https://doi.org/10.1109/TKDE.2023.3270293>
- Zhang, L., Jia, T., Tan, X., Huang, X., Jia, M., Liu, H., Wu, Z., & Li, Y. (2025). E-Log: Fine-Grained Elastic Log-Based Anomaly Detection and Diagnosis for Databases. *IEEE Transactions on Services Computing*, 18(5), 2808–2821. <https://doi.org/10.1109/TSC.2025.3594870>